

libximc

2.14.31

Создано системой Doxygen 1.8.1.2

Пн 25 Авг 2025 12:22:50

# Оглавление

<b>1 Библиотека libximc</b>	<b>1</b>
1.1 Что делает контроллер 8SMC4-USB и 8SMC5-USB	1
1.2 Что умеет библиотека libximc	1
1.3 Содействие	2
<b>2 Введение</b>	<b>3</b>
2.1 О библиотеке	3
2.1.1 Поддерживаемые операционные системы и требования к окружению:	3
<b>3 Как пересобрать библиотеку</b>	<b>4</b>
3.1 Сборка для ОС Windows	4
3.2 Сборка для Linux на основе Debian	4
3.3 Сборка для MacOS X	4
3.4 Сборка для UNIX	5
3.5 Сборка на Linux на основе RedHat	5
3.6 Доступ к исходным кодам	5
<b>4 Как использовать с...</b>	<b>6</b>
4.1 Логирование в файл	9
4.2 Требуемые права доступа	9
4.3 Си-профили	9
4.4 Python-профили	9
<b>5 Работа с пользовательскими единицами</b>	<b>10</b>
5.1 Структура пересчета единиц calibration_t	10
5.2 Функции дублиеры для работы с пользовательскими единицами и структуры данных для них	10
5.3 Таблица коррекции координат для более точного позиционирования	11
<b>6 Структуры данных</b>	<b>12</b>
6.1 Структура accessories_settings_t	12
6.1.1 Подробное описание	12
6.1.2 Поля	13

6.1.2.1	LimitSwitchesSettings	13
6.1.2.2	MagneticBrakeInfo	13
6.1.2.3	MBRatedCurrent	13
6.1.2.4	MBRatedVoltage	13
6.1.2.5	MBSettings	13
6.1.2.6	MBTorque	13
6.1.2.7	TemperatureSensorInfo	13
6.1.2.8	TSGrad	13
6.1.2.9	TSMax	13
6.1.2.10	TSMin	14
6.1.2.11	TSSettings	14
6.2	Структура analog_data_t	14
6.2.1	Подробное описание	15
6.2.2	Поля	15
6.2.2.1	A1Voltage	15
6.2.2.2	A1Voltage_ADC	15
6.2.2.3	A2Voltage	15
6.2.2.4	A2Voltage_ADC	15
6.2.2.5	ACurrent	16
6.2.2.6	ACurrent_ADC	16
6.2.2.7	B1Voltage	16
6.2.2.8	B1Voltage_ADC	16
6.2.2.9	B2Voltage	16
6.2.2.10	B2Voltage_ADC	16
6.2.2.11	BCurrent	16
6.2.2.12	BCurrent_ADC	16
6.2.2.13	Enc_Check	16
6.2.2.14	Enc_Check_ADC	16
6.2.2.15	FullCurrent	16
6.2.2.16	FullCurrent_ADC	17
6.2.2.17	Joy	17
6.2.2.18	Joy_ADC	17
6.2.2.19	Pot	17
6.2.2.20	SupVoltage	17
6.2.2.21	SupVoltage_ADC	17
6.2.2.22	Temp	17
6.2.2.23	Temp_ADC	17
6.3	Структура brake_settings_t	17
6.3.1	Подробное описание	18

6.3.2	Поля	18
6.3.2.1	BrakeFlags	18
6.3.2.2	t1	18
6.3.2.3	t2	18
6.3.2.4	t3	18
6.3.2.5	t4	18
6.4	Структура calibration_settings_t	18
6.4.1	Подробное описание	19
6.4.2	Поля	19
6.4.2.1	CSS1_A	19
6.4.2.2	CSS1_B	19
6.4.2.3	CSS2_A	19
6.4.2.4	CSS2_B	19
6.4.2.5	FullCurrent_A	19
6.4.2.6	FullCurrent_B	19
6.5	Структура calibration_t	20
6.5.1	Подробное описание	20
6.5.2	Поля	21
6.5.2.1	A	21
6.5.2.2	MicrostepMode	21
6.6	Структура chart_data_t	21
6.6.1	Подробное описание	22
6.6.2	Поля	22
6.6.2.1	AveragedPowerRatio	22
6.6.2.2	Joy	22
6.6.2.3	Pot	22
6.6.2.4	WindingCurrentA	22
6.6.2.5	WindingCurrentB	22
6.6.2.6	WindingCurrentC	22
6.6.2.7	WindingVoltageA	22
6.6.2.8	WindingVoltageB	23
6.6.2.9	WindingVoltageC	23
6.7	Структура control_settings_calb_t	23
6.7.1	Подробное описание	23
6.7.2	Поля	23
6.7.2.1	Flags	23
6.7.2.2	MaxClickTime	24
6.7.2.3	MaxSpeed	24
6.7.2.4	Timeout	24

6.8	Структура <code>control_settings_t</code>	24
6.8.1	Подробное описание	24
6.8.2	Поля	25
6.8.2.1	Flags	25
6.8.2.2	MaxClickTime	25
6.8.2.3	MaxSpeed	25
6.8.2.4	Timeout	25
6.8.2.5	uDeltaPosition	25
6.8.2.6	uMaxSpeed	25
6.9	Структура <code>controller_name_t</code>	25
6.9.1	Подробное описание	26
6.9.2	Поля	26
6.9.2.1	ControllerName	26
6.9.2.2	CtrlFlags	26
6.10	Структура <code>ctr_settings_t</code>	26
6.10.1	Подробное описание	26
6.10.2	Поля	27
6.10.2.1	CTPFlags	27
6.10.2.2	CTPMinError	27
6.11	Структура <code>debug_read_t</code>	27
6.11.1	Подробное описание	27
6.11.2	Поля	27
6.11.2.1	DebugData	27
6.12	Структура <code>debug_write_t</code>	27
6.12.1	Подробное описание	28
6.12.2	Поля	28
6.12.2.1	DebugData	28
6.13	Структура <code>device_information_t</code>	28
6.13.1	Подробное описание	28
6.13.2	Поля	29
6.13.2.1	Major	29
6.13.2.2	Minor	29
6.13.2.3	Release	29
6.14	Структура <code>device_network_information_t</code>	29
6.14.1	Подробное описание	29
6.15	Структура <code>edges_settings_calb_t</code>	29
6.15.1	Подробное описание	30
6.15.2	Поля	30
6.15.2.1	BorderFlags	30

6.15.2.2 EnderFlags . . . . .	30
6.15.2.3 LeftBorder . . . . .	30
6.15.2.4 RightBorder . . . . .	30
6.16 Структура edges_settings_t . . . . .	30
6.16.1 Подробное описание . . . . .	31
6.16.2 Поля . . . . .	31
6.16.2.1 BorderFlags . . . . .	31
6.16.2.2 EnderFlags . . . . .	31
6.16.2.3 LeftBorder . . . . .	31
6.16.2.4 RightBorder . . . . .	31
6.16.2.5 uLeftBorder . . . . .	31
6.16.2.6 uRightBorder . . . . .	32
6.17 Структура emf_settings_t . . . . .	32
6.17.1 Подробное описание . . . . .	32
6.17.2 Поля . . . . .	32
6.17.2.1 BackEMFFlags . . . . .	32
6.17.2.2 Km . . . . .	32
6.17.2.3 L . . . . .	32
6.17.2.4 R . . . . .	33
6.18 Структура encoder_information_t . . . . .	33
6.18.1 Подробное описание . . . . .	33
6.18.2 Поля . . . . .	33
6.18.2.1 Manufacturer . . . . .	33
6.18.2.2 PartNumber . . . . .	33
6.19 Структура encoder_settings_t . . . . .	33
6.19.1 Подробное описание . . . . .	34
6.19.2 Поля . . . . .	34
6.19.2.1 EncoderSettings . . . . .	34
6.19.2.2 MaxCurrentConsumption . . . . .	34
6.19.2.3 MaxOperatingFrequency . . . . .	34
6.19.2.4 SupplyVoltageMax . . . . .	34
6.19.2.5 SupplyVoltageMin . . . . .	34
6.20 Структура engine_advansed_setup_t . . . . .	34
6.20.1 Подробное описание . . . . .	35
6.20.2 Поля . . . . .	35
6.20.2.1 stepcloseloop_Kp_high . . . . .	35
6.20.2.2 stepcloseloop_Kp_low . . . . .	35
6.20.2.3 stepcloseloop_Kw . . . . .	35
6.21 Структура engine_settings_calb_t . . . . .	35

6.21.1	Подробное описание	36
6.21.2	Поля	36
6.21.2.1	Antiplay	36
6.21.2.2	EngineFlags	36
6.21.2.3	MicrostepMode	36
6.21.2.4	NomCurrent	36
6.21.2.5	NomSpeed	37
6.21.2.6	NomVoltage	37
6.21.2.7	StepsPerRev	37
6.22	Структура engine_settings_t	37
6.22.1	Подробное описание	37
6.22.2	Поля	38
6.22.2.1	Antiplay	38
6.22.2.2	EngineFlags	38
6.22.2.3	MicrostepMode	38
6.22.2.4	NomCurrent	38
6.22.2.5	NomSpeed	38
6.22.2.6	NomVoltage	38
6.22.2.7	StepsPerRev	38
6.22.2.8	uNomSpeed	38
6.23	Структура entype_settings_t	39
6.23.1	Подробное описание	39
6.23.2	Поля	39
6.23.2.1	DriverType	39
6.23.2.2	EngineType	39
6.24	Структура extended_settings_t	39
6.24.1	Подробное описание	40
6.25	Структура extio_settings_t	40
6.25.1	Подробное описание	40
6.25.2	Поля	40
6.25.2.1	EXTIOModeFlags	40
6.25.2.2	EXTIOSetupFlags	40
6.26	Структура feedback_settings_t	40
6.26.1	Подробное описание	41
6.26.2	Поля	41
6.26.2.1	CountsPerTurn	41
6.26.2.2	FeedbackFlags	41
6.26.2.3	FeedbackType	41
6.26.2.4	IPS	41

6.27 Структура gear_information_t . . . . .	41
6.27.1 Подробное описание . . . . .	42
6.27.2 Поля . . . . .	42
6.27.2.1 Manufacturer . . . . .	42
6.27.2.2 PartNumber . . . . .	42
6.28 Структура gear_settings_t . . . . .	42
6.28.1 Подробное описание . . . . .	43
6.28.2 Поля . . . . .	43
6.28.2.1 Efficiency . . . . .	43
6.28.2.2 InputInertia . . . . .	43
6.28.2.3 MaxOutputBacklash . . . . .	43
6.28.2.4 RatedInputSpeed . . . . .	43
6.28.2.5 RatedInputTorque . . . . .	43
6.28.2.6 ReductionIn . . . . .	43
6.28.2.7 ReductionOut . . . . .	43
6.29 Структура get_position_calb_t . . . . .	44
6.29.1 Подробное описание . . . . .	44
6.29.2 Поля . . . . .	44
6.29.2.1 EncPosition . . . . .	44
6.29.2.2 Position . . . . .	44
6.30 Структура get_position_t . . . . .	44
6.30.1 Подробное описание . . . . .	45
6.30.2 Поля . . . . .	45
6.30.2.1 EncPosition . . . . .	45
6.30.2.2 uPosition . . . . .	45
6.31 Структура globally_unique_identifier_t . . . . .	45
6.31.1 Подробное описание . . . . .	45
6.31.2 Поля . . . . .	46
6.31.2.1 UniqueID0 . . . . .	46
6.31.2.2 UniqueID1 . . . . .	46
6.31.2.3 UniqueID2 . . . . .	46
6.31.2.4 UniqueID3 . . . . .	46
6.32 Структура hallsensor_information_t . . . . .	46
6.32.1 Подробное описание . . . . .	46
6.32.2 Поля . . . . .	46
6.32.2.1 Manufacturer . . . . .	46
6.32.2.2 PartNumber . . . . .	47
6.33 Структура hallsensor_settings_t . . . . .	47
6.33.1 Подробное описание . . . . .	47



6.33.2 Поля	47
6.33.2.1 MaxCurrentConsumption	47
6.33.2.2 MaxOperatingFrequency	47
6.33.2.3 SupplyVoltageMax	47
6.33.2.4 SupplyVoltageMin	48
6.34 Структура home_settings_calb_t	48
6.34.1 Подробное описание	48
6.34.2 Поля	48
6.34.2.1 FastHome	48
6.34.2.2 HomeDelta	48
6.34.2.3 HomeFlags	48
6.34.2.4 SlowHome	48
6.35 Структура home_settings_t	49
6.35.1 Подробное описание	49
6.35.2 Поля	49
6.35.2.1 FastHome	49
6.35.2.2 HomeDelta	49
6.35.2.3 HomeFlags	49
6.35.2.4 SlowHome	50
6.35.2.5 uFastHome	50
6.35.2.6 uHomeDelta	50
6.35.2.7 uSlowHome	50
6.36 Структура init_random_t	50
6.36.1 Подробное описание	50
6.36.2 Поля	51
6.36.2.1 key	51
6.37 Структура joystick_settings_t	51
6.37.1 Подробное описание	51
6.37.2 Поля	51
6.37.2.1 DeadZone	52
6.37.2.2 ExpFactor	52
6.37.2.3 JoyCenter	52
6.37.2.4 JoyFlags	52
6.37.2.5 JoyHighEnd	52
6.37.2.6 JoyLowEnd	52
6.38 Структура measurements_t	52
6.38.1 Подробное описание	53
6.38.2 Поля	53
6.38.2.1 Length	53

6.39 Структура motor_information_t . . . . .	53
6.39.1 Подробное описание . . . . .	53
6.39.2 Поля . . . . .	53
6.39.2.1 Manufacturer . . . . .	53
6.39.2.2 PartNumber . . . . .	54
6.40 Структура motor_settings_t . . . . .	54
6.40.1 Подробное описание . . . . .	55
6.40.2 Поля . . . . .	55
6.40.2.1 DetentTorque . . . . .	55
6.40.2.2 MaxCurrent . . . . .	55
6.40.2.3 MaxCurrentTime . . . . .	55
6.40.2.4 MaxSpeed . . . . .	55
6.40.2.5 MechanicalTimeConstant . . . . .	56
6.40.2.6 MotorType . . . . .	56
6.40.2.7 NoLoadCurrent . . . . .	56
6.40.2.8 NoLoadSpeed . . . . .	56
6.40.2.9 NominalCurrent . . . . .	56
6.40.2.10 NominalPower . . . . .	56
6.40.2.11 NominalSpeed . . . . .	56
6.40.2.12 NominalTorque . . . . .	56
6.40.2.13 NominalVoltage . . . . .	56
6.40.2.14 Phases . . . . .	57
6.40.2.15 Poles . . . . .	57
6.40.2.16 RotorInertia . . . . .	57
6.40.2.17 SpeedConstant . . . . .	57
6.40.2.18 SpeedTorqueGradient . . . . .	57
6.40.2.19 StallTorque . . . . .	57
6.40.2.20 TorqueConstant . . . . .	57
6.40.2.21 WindingInductance . . . . .	57
6.40.2.22 WindingResistance . . . . .	57
6.41 Структура move_settings_calb_t . . . . .	58
6.41.1 Подробное описание . . . . .	58
6.41.2 Поля . . . . .	58
6.41.2.1 Accel . . . . .	58
6.41.2.2 AntiplaySpeed . . . . .	58
6.41.2.3 Decel . . . . .	58
6.41.2.4 MoveFlags . . . . .	59
6.41.2.5 Speed . . . . .	59
6.42 Структура move_settings_t . . . . .	59

6.42.1	Подробное описание	59
6.42.2	Поля	59
6.42.2.1	Accel	59
6.42.2.2	AntiplaySpeed	60
6.42.2.3	Decel	60
6.42.2.4	MoveFlags	60
6.42.2.5	Speed	60
6.42.2.6	uAntiplaySpeed	60
6.42.2.7	uSpeed	60
6.43	Структура network_settings_t	60
6.43.1	Подробное описание	61
6.43.2	Поля	61
6.43.2.1	DefaultGateway	61
6.43.2.2	DHCPEnabled	61
6.43.2.3	IPv4Address	61
6.43.2.4	SubnetMask	61
6.44	Структура nonvolatile_memory_t	61
6.44.1	Подробное описание	61
6.44.2	Поля	62
6.44.2.1	UserData	62
6.45	Структура password_settings_t	62
6.45.1	Подробное описание	62
6.45.2	Поля	62
6.45.2.1	UserPassword	62
6.46	Структура pid_settings_t	62
6.46.1	Подробное описание	63
6.47	Структура power_settings_t	63
6.47.1	Подробное описание	63
6.47.2	Поля	64
6.47.2.1	CurrentSetTime	64
6.47.2.2	CurrReductDelay	64
6.47.2.3	HoldCurrent	64
6.47.2.4	PowerFlags	64
6.47.2.5	PowerOffDelay	64
6.48	Структура secure_settings_t	64
6.48.1	Подробное описание	65
6.48.2	Поля	65
6.48.2.1	Criticalpwr	65
6.48.2.2	Criticalusb	65

6.48.2.3 CriticalUpwr . . . . .	65
6.48.2.4 CriticalUusb . . . . .	65
6.48.2.5 Flags . . . . .	65
6.48.2.6 LowUpwrOff . . . . .	65
6.48.2.7 MinimumUusb . . . . .	65
6.49 Структура serial_number_t . . . . .	65
6.49.1 Подробное описание . . . . .	66
6.49.2 Поля . . . . .	66
6.49.2.1 Key . . . . .	66
6.49.2.2 Major . . . . .	66
6.49.2.3 Minor . . . . .	66
6.49.2.4 Release . . . . .	66
6.49.2.5 SN . . . . .	66
6.50 Структура set_position_calb_t . . . . .	66
6.50.1 Подробное описание . . . . .	67
6.50.2 Поля . . . . .	67
6.50.2.1 EncPosition . . . . .	67
6.50.2.2 PosFlags . . . . .	67
6.50.2.3 Position . . . . .	67
6.51 Структура set_position_t . . . . .	67
6.51.1 Подробное описание . . . . .	68
6.51.2 Поля . . . . .	68
6.51.2.1 EncPosition . . . . .	68
6.51.2.2 PosFlags . . . . .	68
6.51.2.3 uPosition . . . . .	68
6.52 Структура stage_information_t . . . . .	68
6.52.1 Подробное описание . . . . .	68
6.52.2 Поля . . . . .	69
6.52.2.1 Manufacturer . . . . .	69
6.52.2.2 PartNumber . . . . .	69
6.53 Структура stage_name_t . . . . .	69
6.53.1 Подробное описание . . . . .	69
6.53.2 Поля . . . . .	69
6.53.2.1 PositionerName . . . . .	69
6.54 Структура stage_settings_t . . . . .	69
6.54.1 Подробное описание . . . . .	70
6.54.2 Поля . . . . .	70
6.54.2.1 HorizontalLoadCapacity . . . . .	70
6.54.2.2 LeadScrewPitch . . . . .	70

6.54.2.3	MaxCurrentConsumption	70
6.54.2.4	MaxSpeed	70
6.54.2.5	SupplyVoltageMax	71
6.54.2.6	SupplyVoltageMin	71
6.54.2.7	TravelRange	71
6.54.2.8	Units	71
6.54.2.9	VerticalLoadCapacity	71
6.55	Структура status_calb_t	71
6.55.1	Подробное описание	72
6.55.2	Поля	72
6.55.2.1	CmdBufFreeSpace	72
6.55.2.2	CurPosition	72
6.55.2.3	CurSpeed	72
6.55.2.4	CurT	72
6.55.2.5	EncPosition	73
6.55.2.6	EncSts	73
6.55.2.7	Flags	73
6.55.2.8	GPIOFlags	73
6.55.2.9	lpwr	73
6.55.2.10	lusb	73
6.55.2.11	MoveSts	73
6.55.2.12	MvCmdSts	73
6.55.2.13	PWRSts	73
6.55.2.14	Upwr	73
6.55.2.15	Uusb	73
6.55.2.16	WindSts	73
6.56	Структура status_t	74
6.56.1	Подробное описание	75
6.56.2	Поля	75
6.56.2.1	CmdBufFreeSpace	75
6.56.2.2	CurPosition	75
6.56.2.3	CurSpeed	75
6.56.2.4	CurT	75
6.56.2.5	EncPosition	75
6.56.2.6	EncSts	75
6.56.2.7	Flags	75
6.56.2.8	GPIOFlags	75
6.56.2.9	lpwr	76
6.56.2.10	lusb	76

6.56.2.11 MoveSts . . . . .	76
6.56.2.12 MvCmdSts . . . . .	76
6.56.2.13 PWRSts . . . . .	76
6.56.2.14 uCurPosition . . . . .	76
6.56.2.15 uCurSpeed . . . . .	76
6.56.2.16 Upwr . . . . .	76
6.56.2.17 Uusb . . . . .	76
6.56.2.18 WindSts . . . . .	76
6.57 Структура sync_in_settings_calb_t . . . . .	77
6.57.1 Подробное описание . . . . .	77
6.57.2 Поля . . . . .	77
6.57.2.1 ClutterTime . . . . .	77
6.57.2.2 Position . . . . .	77
6.57.2.3 Speed . . . . .	77
6.57.2.4 SyncInFlags . . . . .	77
6.58 Структура sync_in_settings_t . . . . .	77
6.58.1 Подробное описание . . . . .	78
6.58.2 Поля . . . . .	78
6.58.2.1 ClutterTime . . . . .	78
6.58.2.2 Speed . . . . .	78
6.58.2.3 SyncInFlags . . . . .	78
6.58.2.4 uPosition . . . . .	78
6.58.2.5 uSpeed . . . . .	79
6.59 Структура sync_out_settings_calb_t . . . . .	79
6.59.1 Подробное описание . . . . .	79
6.59.2 Поля . . . . .	79
6.59.2.1 Accuracy . . . . .	79
6.59.2.2 SyncOutFlags . . . . .	79
6.59.2.3 SyncOutPeriod . . . . .	79
6.59.2.4 SyncOutPulseSteps . . . . .	80
6.60 Структура sync_out_settings_t . . . . .	80
6.60.1 Подробное описание . . . . .	80
6.60.2 Поля . . . . .	80
6.60.2.1 Accuracy . . . . .	80
6.60.2.2 SyncOutFlags . . . . .	80
6.60.2.3 SyncOutPeriod . . . . .	81
6.60.2.4 SyncOutPulseSteps . . . . .	81
6.60.2.5 uAccuracy . . . . .	81
6.61 Структура uart_settings_t . . . . .	81

6.61.1	Подробное описание	81
6.61.2	Поля	81
6.61.2.1	UARTSetupFlags	81
<b>7</b>	<b>Файлы</b>	<b>82</b>
7.1	Файл <code>libmc.h</code>	82
7.1.1	Подробное описание	108
7.1.2	Макросы	108
7.1.2.1	ALARM_ON_DRIVER_OVERHEATING	108
7.1.2.2	BACK_EMF_INDUCTANCE_AUTO	108
7.1.2.3	BACK_EMF_KM_AUTO	108
7.1.2.4	BACK_EMF_RESISTANCE_AUTO	108
7.1.2.5	BORDER_IS_ENCODER	108
7.1.2.6	BORDER_STOP_LEFT	108
7.1.2.7	BORDER_STOP_RIGHT	108
7.1.2.8	BORDERS_SWAP_MISSET_DETECTION	109
7.1.2.9	BRAKE_ENABLED	109
7.1.2.10	BRAKE_ENG_PWROFF	109
7.1.2.11	BRAKING_OVERVOLTAGE_PROTECTION	109
7.1.2.12	CONTROL_BTN_LEFT_PUSHED_OPEN	109
7.1.2.13	CONTROL_BTN_RIGHT_PUSHED_OPEN	109
7.1.2.14	CONTROL_MODE_BITS	109
7.1.2.15	CONTROL_MODE_JOY	109
7.1.2.16	CONTROL_MODE_LR	109
7.1.2.17	CONTROL_MODE_OFF	109
7.1.2.18	CTP_ALARM_ON_ERROR	109
7.1.2.19	CTP_BASE	110
7.1.2.20	CTP_ENABLED	110
7.1.2.21	CTP_ERROR_CORRECTION	110
7.1.2.22	DRIVER_TYPE_DISCRETE_FET	110
7.1.2.23	DRIVER_TYPE_EXTERNAL	110
7.1.2.24	DRIVER_TYPE_INTEGRATE	110
7.1.2.25	EEPROM_PRECEDENCE	110
7.1.2.26	ENC_STATE_ABSENT	110
7.1.2.27	ENC_STATE_MALFUNC	110
7.1.2.28	ENC_STATE_OK	110
7.1.2.29	ENC_STATE_REVERS	110
7.1.2.30	ENC_STATE_UNKNOWN	111
7.1.2.31	ENDER_SW1_ACTIVE_LOW	111

7.1.2.32	ENDER_SW2_ACTIVE_LOW	111
7.1.2.33	ENDER_SWAP	111
7.1.2.34	ENGINE_ACCEL_ON	111
7.1.2.35	ENGINE_ANTIPLAY	111
7.1.2.36	ENGINE_CURRENT_AS_RMS	111
7.1.2.37	ENGINE_LIMIT_CURR	111
7.1.2.38	ENGINE_LIMIT_RPM	111
7.1.2.39	ENGINE_LIMIT_VOLT	112
7.1.2.40	ENGINE_MAX_SPEED	112
7.1.2.41	ENGINE_REVERSE	112
7.1.2.42	ENGINE_TYPE_2DC	112
7.1.2.43	ENGINE_TYPE_BRUSHLESS	112
7.1.2.44	ENGINE_TYPE_DC	112
7.1.2.45	ENGINE_TYPE_NONE	112
7.1.2.46	ENGINE_TYPE_STEP	112
7.1.2.47	ENGINE_TYPE_TEST	112
7.1.2.48	ENUMERATE_PROBE	113
7.1.2.49	EXTIO_SETUP_INVERT	113
7.1.2.50	EXTIO_SETUP_MODE_IN_ALARM	113
7.1.2.51	EXTIO_SETUP_MODE_IN_BITS	113
7.1.2.52	EXTIO_SETUP_MODE_IN_HOME	113
7.1.2.53	EXTIO_SETUP_MODE_IN_MOVR	113
7.1.2.54	EXTIO_SETUP_MODE_IN_NOP	113
7.1.2.55	EXTIO_SETUP_MODE_IN_PWOF	113
7.1.2.56	EXTIO_SETUP_MODE_IN_STOP	113
7.1.2.57	EXTIO_SETUP_MODE_OUT_ALARM	113
7.1.2.58	EXTIO_SETUP_MODE_OUT_BITS	113
7.1.2.59	EXTIO_SETUP_MODE_OUT_MOTOR_ON	114
7.1.2.60	EXTIO_SETUP_MODE_OUT_MOVING	114
7.1.2.61	EXTIO_SETUP_MODE_OUT_OFF	114
7.1.2.62	EXTIO_SETUP_MODE_OUT_ON	114
7.1.2.63	EXTIO_SETUP_OUTPUT	114
7.1.2.64	FEEDBACK_EMF	114
7.1.2.65	FEEDBACK_ENC_ADAPTIVE_HOLDING	114
7.1.2.66	FEEDBACK_ENC_REVERSE	114
7.1.2.67	FEEDBACK_ENC_TYPE_AUTO	114
7.1.2.68	FEEDBACK_ENC_TYPE_BITS	114
7.1.2.69	FEEDBACK_ENC_TYPE_DIFFERENTIAL	114
7.1.2.70	FEEDBACK_ENC_TYPE_SINGLE_ENDED	114



7.1.2.71	FEEDBACK_ENCODER	115
7.1.2.72	FEEDBACK_ENCODER_MEDIATED	115
7.1.2.73	FEEDBACK_NONE	115
7.1.2.74	HOME_DIR_FIRST	115
7.1.2.75	HOME_DIR_SECOND	115
7.1.2.76	HOME_HALF_MV	115
7.1.2.77	HOME_MV_SEC_EN	115
7.1.2.78	HOME_STOP_FIRST_BITS	115
7.1.2.79	HOME_STOP_FIRST_LIM	115
7.1.2.80	HOME_STOP_FIRST_REV	115
7.1.2.81	HOME_STOP_FIRST_SYN	115
7.1.2.82	HOME_STOP_SECOND_BITS	116
7.1.2.83	HOME_STOP_SECOND_LIM	116
7.1.2.84	HOME_STOP_SECOND_REV	116
7.1.2.85	HOME_STOP_SECOND_SYN	116
7.1.2.86	HOME_USE_FAST	116
7.1.2.87	JOY_REVERSE	116
7.1.2.88	LOW_UPWR_PROTECTION	116
7.1.2.89	LS_SHORTED	116
7.1.2.90	MICROSTEP_MODE_FRAC_128	116
7.1.2.91	MICROSTEP_MODE_FRAC_16	116
7.1.2.92	MICROSTEP_MODE_FRAC_2	116
7.1.2.93	MICROSTEP_MODE_FRAC_256	116
7.1.2.94	MICROSTEP_MODE_FRAC_32	117
7.1.2.95	MICROSTEP_MODE_FRAC_4	117
7.1.2.96	MICROSTEP_MODE_FRAC_64	117
7.1.2.97	MICROSTEP_MODE_FRAC_8	117
7.1.2.98	MICROSTEP_MODE_FULL	117
7.1.2.99	MOVE_STATE_ANTIPLAY	117
7.1.2.100	MOVE_STATE_MOVING	117
7.1.2.101	MOVE_STATE_TARGET_SPEED	117
7.1.2.102	MVCMD_ERROR	117
7.1.2.103	MVCMD_HOME	117
7.1.2.104	MVCMD_LEFT	117
7.1.2.105	MVCMD_LOFT	118
7.1.2.106	MVCMD_MOVE	118
7.1.2.107	MVCMD_MOVR	118
7.1.2.108	MVCMD_NAME_BITS	118
7.1.2.109	MVCMD_RIGHT	118

7.1.2.110MVCMD_RUNNING	118
7.1.2.111MVCMD_SSTP	118
7.1.2.112MVCMD_STOP	118
7.1.2.113MVCMD_UKNWN	118
7.1.2.114POWER_OFF_ENABLED	118
7.1.2.115POWER_REDUCT_ENABLED	118
7.1.2.116POWER_SMOOTH_CURRENT	119
7.1.2.117PWR_STATE_MAX	119
7.1.2.118PWR_STATE_NORM	119
7.1.2.119PWR_STATE_OFF	119
7.1.2.120PWR_STATE_REDUCT	119
7.1.2.121PWR_STATE_UNKNOWN	119
7.1.2.122REV_SENS_INV	119
7.1.2.123RPM_DIV_1000	119
7.1.2.124SETPOS_IGNORE_ENCODER	119
7.1.2.125SETPOS_IGNORE_POSITION	119
7.1.2.126STATE_ALARM	120
7.1.2.127STATE_BORDERS_SWAP_MISSET	120
7.1.2.128STATE_BRAKE	120
7.1.2.129STATE_BUTTON_LEFT	120
7.1.2.130STATE_BUTTON_RIGHT	120
7.1.2.131STATE_CONTR	120
7.1.2.132STATE_CONTROLLER_OVERHEAT	120
7.1.2.133STATE_CTP_ERROR	120
7.1.2.134STATE_DIG_SIGNAL	120
7.1.2.135STATE_EEPROM_CONNECTED	120
7.1.2.136STATE_ENC_A	121
7.1.2.137STATE_ENC_B	121
7.1.2.138STATE_ENGINE_RESPONSE_ERROR	121
7.1.2.139STATE_ERRC	121
7.1.2.140STATE_ERRD	121
7.1.2.141STATE_ERRV	121
7.1.2.142STATE_EXTIO_ALARM	121
7.1.2.143STATE_GPIO_LEVEL	121
7.1.2.144STATE_GPIO_PINOUT	121
7.1.2.145STATE_IS_HOMED	122
7.1.2.146STATE_LEFT_EDGE	122
7.1.2.147STATE_LOW_USB_VOLTAGE	122
7.1.2.148STATE_OVERLOAD_POWER_CURRENT	122

7.1.2.149	STATE_OVERLOAD_POWER_VOLTAGE	122
7.1.2.150	STATE_OVERLOAD_USB_CURRENT	122
7.1.2.151	STATE_OVERLOAD_USB_VOLTAGE	122
7.1.2.152	STATE_POWER_OVERHEAT	122
7.1.2.153	STATE_REV_SENSOR	122
7.1.2.154	STATE_RIGHT_EDGE	123
7.1.2.155	STATE_SECUR	123
7.1.2.156	STATE_SYNC_INPUT	123
7.1.2.157	STATE_SYNC_OUTPUT	123
7.1.2.158	STATE_WINDING_RES_MISMATCH	123
7.1.2.159	SYNCIN_ENABLED	123
7.1.2.160	SYNCIN_INVERT	123
7.1.2.161	SYNCOUT_ENABLED	123
7.1.2.162	SYNCOUT_IN_STEPS	123
7.1.2.163	SYNCOUT_INVERT	123
7.1.2.164	SYNCOUT_ONPERIOD	124
7.1.2.165	SYNCOUT_ONSTART	124
7.1.2.166	SYNCOUT_ONSTOP	124
7.1.2.167	SYNCOUT_STATE	124
7.1.2.168	TS_TYPE_BITS	124
7.1.2.169	UART_PARITY_BITS	124
7.1.2.170	WIND_A_STATE_ABSENT	124
7.1.2.171	WIND_A_STATE_MALFUNC	124
7.1.2.172	WIND_A_STATE_OK	124
7.1.2.173	WIND_A_STATE_UNKNOWN	124
7.1.2.174	WIND_B_STATE_ABSENT	124
7.1.2.175	WIND_B_STATE_MALFUNC	124
7.1.2.176	WIND_B_STATE_OK	125
7.1.2.177	WIND_B_STATE_UNKNOWN	125
7.1.2.178	XIMC_API	125
7.1.3	Типы	125
7.1.3.1	calibration_t	125
7.1.3.2	logging_callback_t	126
7.1.4	Функции	126
7.1.4.1	close_device	126
7.1.4.2	command_clear_fram	126
7.1.4.3	command_eeread_settings	126
7.1.4.4	command_eesave_settings	126
7.1.4.5	command_home	127

7.1.4.6	command_homezero	127
7.1.4.7	command_left	127
7.1.4.8	command_loft	128
7.1.4.9	command_move	128
7.1.4.10	command_move_calb	128
7.1.4.11	command_movr	128
7.1.4.12	command_movr_calb	129
7.1.4.13	command_power_off	129
7.1.4.14	command_read_robust_settings	130
7.1.4.15	command_read_settings	130
7.1.4.16	command_reset	130
7.1.4.17	command_right	130
7.1.4.18	command_save_robust_settings	130
7.1.4.19	command_save_settings	131
7.1.4.20	command_sstp	131
7.1.4.21	command_start_measurements	131
7.1.4.22	command_stop	131
7.1.4.23	command_update_firmware	131
7.1.4.24	command_wait_for_stop	132
7.1.4.25	command_zero	132
7.1.4.26	enumerate_devices	132
7.1.4.27	free_enumerate_devices	134
7.1.4.28	get_accessories_settings	134
7.1.4.29	get_analog_data	134
7.1.4.30	get_bootloader_version	134
7.1.4.31	get_brake_settings	135
7.1.4.32	get_calibration_settings	135
7.1.4.33	get_chart_data	135
7.1.4.34	get_control_settings	136
7.1.4.35	get_control_settings_calb	136
7.1.4.36	get_controller_name	136
7.1.4.37	get_ctp_settings	136
7.1.4.38	get_debug_read	137
7.1.4.39	get_device_count	137
7.1.4.40	get_device_information	137
7.1.4.41	get_device_name	138
7.1.4.42	get_edges_settings	138
7.1.4.43	get_edges_settings_calb	138
7.1.4.44	get_emf_settings	138

7.1.4.45	get_encoder_information	139
7.1.4.46	get_encoder_settings	139
7.1.4.47	get_engine_advanced_setup	139
7.1.4.48	get_engine_settings	140
7.1.4.49	get_engine_settings_calb	140
7.1.4.50	get_entype_settings	140
7.1.4.51	get_enumerate_device_controller_name	140
7.1.4.52	get_enumerate_device_information	141
7.1.4.53	get_enumerate_device_network_information	141
7.1.4.54	get_enumerate_device_serial	141
7.1.4.55	get_enumerate_device_stage_name	142
7.1.4.56	get_extended_settings	142
7.1.4.57	get_extio_settings	142
7.1.4.58	get_feedback_settings	142
7.1.4.59	get_firmware_version	143
7.1.4.60	get_gear_information	143
7.1.4.61	get_gear_settings	143
7.1.4.62	get_globally_unique_identifier	143
7.1.4.63	get_hallsensor_information	144
7.1.4.64	get_hallsensor_settings	144
7.1.4.65	get_home_settings	144
7.1.4.66	get_home_settings_calb	145
7.1.4.67	get_init_random	145
7.1.4.68	get_joystick_settings	145
7.1.4.69	get_measurements	146
7.1.4.70	get_motor_information	146
7.1.4.71	get_motor_settings	146
7.1.4.72	get_move_settings	146
7.1.4.73	get_move_settings_calb	147
7.1.4.74	get_network_settings	147
7.1.4.75	get_nonvolatile_memory	147
7.1.4.76	get_password_settings	147
7.1.4.77	get_pid_settings	148
7.1.4.78	get_position	148
7.1.4.79	get_position_calb	148
7.1.4.80	get_power_settings	148
7.1.4.81	get_secure_settings	149
7.1.4.82	get_serial_number	149
7.1.4.83	get_stage_information	149

7.1.4.84	get_stage_name	149
7.1.4.85	get_stage_settings	150
7.1.4.86	get_status	150
7.1.4.87	get_status_calb	150
7.1.4.88	get_sync_in_settings	150
7.1.4.89	get_sync_in_settings_calb	151
7.1.4.90	get_sync_out_settings	151
7.1.4.91	get_sync_out_settings_calb	151
7.1.4.92	get_uart_settings	152
7.1.4.93	goto_firmware	152
7.1.4.94	has_firmware	152
7.1.4.95	load_correction_table	152
7.1.4.96	logging_callback_stderr_narrow	153
7.1.4.97	logging_callback_stderr_wide	153
7.1.4.98	msec_sleep	154
7.1.4.99	open_device	154
7.1.4.100	probe_device	154
7.1.4.101	service_command_updf	154
7.1.4.102	set_accessories_settings	155
7.1.4.103	set_bindy_key	155
7.1.4.104	set_brake_settings	155
7.1.4.105	set_calibration_settings	155
7.1.4.106	set_control_settings	156
7.1.4.107	set_control_settings_calb	156
7.1.4.108	set_controller_name	156
7.1.4.109	set_correction_table	156
7.1.4.110	set_ctp_settings	157
7.1.4.111	set_debug_write	157
7.1.4.112	set_edges_settings	158
7.1.4.113	set_edges_settings_calb	158
7.1.4.114	set_emf_settings	158
7.1.4.115	set_encoder_information	159
7.1.4.116	set_encoder_settings	159
7.1.4.117	set_engine_advansed_setup	159
7.1.4.118	set_engine_settings	159
7.1.4.119	set_engine_settings_calb	160
7.1.4.120	set_entype_settings	160
7.1.4.121	set_extended_settings	160
7.1.4.122	set_extio_settings	161

7.1.4.123set_feedback_settings . . . . .	161
7.1.4.124set_gear_information . . . . .	161
7.1.4.125set_gear_settings . . . . .	162
7.1.4.126set_hallsensor_information . . . . .	162
7.1.4.127set_hallsensor_settings . . . . .	162
7.1.4.128set_home_settings . . . . .	162
7.1.4.129set_home_settings_calb . . . . .	163
7.1.4.130set_joystick_settings . . . . .	163
7.1.4.131set_logging_callback . . . . .	163
7.1.4.132set_motor_information . . . . .	164
7.1.4.133set_motor_settings . . . . .	164
7.1.4.134set_move_settings . . . . .	164
7.1.4.135set_move_settings_calb . . . . .	164
7.1.4.136set_network_settings . . . . .	165
7.1.4.137set_nonvolatile_memory . . . . .	165
7.1.4.138set_password_settings . . . . .	165
7.1.4.139set_pid_settings . . . . .	165
7.1.4.140set_position . . . . .	166
7.1.4.141set_position_calb . . . . .	166
7.1.4.142set_power_settings . . . . .	166
7.1.4.143set_secure_settings . . . . .	166
7.1.4.144set_serial_number . . . . .	167
7.1.4.145set_stage_information . . . . .	167
7.1.4.146set_stage_name . . . . .	167
7.1.4.147set_stage_settings . . . . .	167
7.1.4.148set_sync_in_settings . . . . .	168
7.1.4.149set_sync_in_settings_calb . . . . .	168
7.1.4.150set_sync_out_settings . . . . .	168
7.1.4.151set_sync_out_settings_calb . . . . .	169
7.1.4.152set_uart_settings . . . . .	169
7.1.4.153write_key . . . . .	169
7.1.4.154ximc_fix_usbser_sys . . . . .	169
7.1.4.155ximc_version . . . . .	170

# Глава 1

## Библиотека libximc

Документация для библиотеки libximc.

Libximc - **потокобезопасная**, кросс-платформенная библиотека для работы с контроллерами 8SMC4-USB и 8SMC5-USB.

Полная документация по контроллерам доступна по [ссылке](#).

Полная документация по API libximc доступна на странице [ximc.h](#).

Библиотека libximc теперь доступна на [PyPI](#), и её можно установить напрямую через pip:

```
pip install libximc
```

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

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

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

### 1.2 Что умеет библиотека libximc

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



#### Предупреждения

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

Пожалуйста, прочитайте [Введение](#) для начала работы с библиотекой.

Для того, чтобы использовать `libximc` в проекте, ознакомьтесь со страницей [Как использовать с...](#)

## 1.3 Содействие

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

## Глава 2

# Введение

### 2.1 О библиотеке

Этот документ содержит всю необходимую информацию о библиотеке libximc. Библиотека libximc использует распространенный и проверенный интерфейс виртуального последовательного порта, поэтому вы можете работать с модулями управления моторами через эту библиотеку практически под всеми ОС: Windows, Linux, MacOS X для Intel и Apple Silicon (с использованием Rosetta 2), в том числе с 64-битными версиями. Библиотека поддерживает подключение и отключение устройств "на лету".

**С одним устройством в каждый момент может работать не более одного экземпляра управляющей программы - множественный доступ управляющих программ к одному и тому же устройству не допускается!**

#### 2.1.1 Поддерживаемые операционные системы и требования к окружению:

- MacOS X 10.6 или новее
- Windows 2000 или новее
- Linux на основе debian. DEB собирается на Debian Squeeze 7
- Linux на основе debian ARM. DEB собирается кросс-компилятором на Ubuntu 14.04
- Linux на основе rpm. RPM собирается на OpenSUSE 12

Требования сборки:

- Windows: Microsoft Visual C++ 2013, MATLAB, Code::Blocks, Delphi, Java, Python, cygwin с tar, bison, flex, curl, 7z, mingw
- UNIX: gcc 4 или новее, gmake, doxygen, LaTeX, flex 2.5.30+, bison 2.3+, autotools (autoconf, autoheader, aclocal, automake, autoreconf, libtool)
- Mac OS X: XCode 4 или новее, doxygen, mactex, autotools (autoconf, autoheader, aclocal, automake, autoreconf, libtool)
- JDK 7 - 9

## Глава 3

# Как пересобрать библиотеку

### 3.1 Сборка для ОС Windows

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

Запустите скрипт:

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

Собранные файлы располагаются в `./ximc/win32` и `./ximc/win64`.

Если вы хотите собрать отладочную версию библиотеки, то перед запуском скрипта сборки установите переменную окружения "DEBUG" в значение "true".

### 3.2 Сборка для Linux на основе Debian

Полный набор пакетов:

```
sudo apt-get install build-essential make cmake curl git ruby1.9.1 autotools-  
dev automake autoconf libtool doxygen bison flex debhelper lintian texlive texlive  
-latex-extra texlive-latex texlive-fonts-extra texlive-lang-cyrillic java-1_7_0-  
openjdk java-1_7_0-openjdk-devel default-jre-headless default-jdk openjdk-6-jdk  
rpm-build rpm-devel rpmlint pkg-config check dh-autoreconf hardening-wrapper  
libfl-dev lsb-release
```

Для кросс-компиляции ARM установите `gcc-arm-linux-gnueabi` из вашего инструментария ARM.

Необходимо соблюдать парность архитектуры библиотеки и системы: 32-битная библиотека может быть собрана только на 32-битной системе, а 64-битная - только на 64-битной. Библиотека под ARM собирается кросс-компилятором `gcc-arm-linux-gnueabi`.

Для сборки библиотеки и пакета запустите скрипт:

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

Для библиотеки ARM замените 'libdeb' на 'libdebarm'.

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

### 3.3 Сборка для MacOS X

Для сборки библиотеки и пакета запустите скрипт:

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

Собранная библиотека (классическая и фреймворк), приложения (классическая и фреймворк) и документация располагаются в `./ximc/macosx`, локально установленные файлы - в `./dist/local`.

## 3.4 Сборка для UNIX

Обобщенная версия собирается обычными autotools.

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

Собранные файлы (библиотека, заголовочные файлы, документация) устанавливаются в локальную директорию `./dist/local`. Это сборка для разработчика, при необходимости можно указать дополнительные параметры командной строки для вашей системы.

## 3.5 Сборка на Linux на основе RedHat

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

Полный набор пакетов:

```
sudo apt-get install build-essential make cmake curl git ruby1.9.1 autotools-  
dev automake autoconf libtool doxygen bison flex debhelper lintian texlive texlive  
-latex-extra texlive-latex texlive-fonts-extra texlive-lang-cyrillic java-1_7_0-  
openjdk java-1_7_0-openjdk-devel default-jre-headless default-jdk openjdk-6-jdk  
rpm-build rpm-devel rpmlint pkg-config check dh-autoreconf hardening-wrapper  
libfl-dev lsb-release
```

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 `./ximc/rpm`, локально установленные файлы - в `./dist/local`.

## 3.6 Доступ к исходным кодам

Исходные коды библиотеки libximc можно найти на [github](#).

## Глава 4

# Как использовать с...

Для приобретения первых навыков использования библиотеки создано простое тестовое приложение `testappeasy_C`.

Языки, отличные от C-подобных, поддерживаются с помощью вызовов с преобразованием аргументов типа `stdcall`.

### Заметки

Для работы с SDK требуется Microsoft Visual C++ Redistributable Package (поставляется с SDK, файлы `vcredist_x86` или `vcredist_x64`).

Для работы на Linux требуется установить оба пакета `libximc7_x.x.x` и `libximc7-dev_x.x.x` целевой архитектуры в указанном порядке. Для установки пакетов можно воспользоваться `.deb` командой:

```
sudo dpkg -i <имя_пакета>.deb
```

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

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

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

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

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

MinGW это вариант GCC для платформы win32. Требуется установка пакета MinGW.

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

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

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

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

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

---

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

Затем скомпилируйте тестовое приложение:

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

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

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

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

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

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

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

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

testapp может быть собран следующим образом с установленной библиотекой:

```
make
```

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

Затем запустите приложение с помощью:

```
make run
```

Примечание: `make run` на MacOS 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.

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

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

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

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

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

Консольное тестовое приложение размещено в директории 'test\_Delphi'. Тестировалось с Delphi 6 на 32-битной системе.

Просто скомпилируйте, разместите .dll в директории с исполняемым примером и запустите его.

В случае, если планируется использовать Ethernet-адаптер 8Eth1, в файле test\_Delphi.dpr перед сборкой нужно прописать IP адрес Ethernet-адаптера (переменная enum\_hints).

Как запустить пример на Linux. Перейдите в examples/test\_Java/compiled-winX/ и выполните

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

Как запустить пример на Windows. Перейдите в examples/test\_Java/compiled-winX/. Запустите:

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

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

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

Затем пересоберите исходные тексты:

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

или для Windows или MacOS X:

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

Затем соберите jar:

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

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

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

Для запуска:

```
python xxxx.py
```

В случае, если планируется использовать Ethernet-адаптер 8Eth1, в файле test\_Python.py перед запуском нужно прописать IP адрес Ethernet-адаптера (переменная enum\_hints).

Тестовая программа на MATLAB testximc.m располагается в директории examples/test\_MATLAB.

Перед запуском:

На MacOS X: скопируйте ximc/macosx/libximc.framework, ximc/macosx/wrappers/ximcm.h, ximc/ximc.h в директорию examples/test\_MATLAB. Установите XCode, совместимый с Matlab

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

Для проверки совместимых XCode и gcc проверьте документы [https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/support/sysreq/files/SystemRequirements--Release2014a\\_SupportedCompilers.pdf](https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/support/sysreq/files/SystemRequirements--Release2014a_SupportedCompilers.pdf) или похожие.

На Windows: перед запуском ничего делать не нужно

Измените текущую директорию в MATLAB на examples/test\_MATLAB. Затем запустите в MATLAB:

```
testximc
```

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

## 4.1 Логирование в файл

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

## 4.2 Требуемые права доступа

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

## 4.3 Си-профили

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

## 4.4 Python-профили

Python-профили - это набор конфигурационных функций, распространяемых вместе с библиотекой libximc. Они позволяют в программе на языке Python загрузить в контроллер настройки одной из поддерживаемых подвижек вызовом всего одной функции.

Пример использования python-профилей вы можете посмотреть в директории примеров "examples/test\_Python/profiletest/testpythonprofile.py".



## Глава 5

# Работа с пользовательскими единицами

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

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

### 5.1 Структура пересчета единиц `calibration_t`

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

Формулы пересчета:

- Пересчет в пользовательские единицы.

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

- Пересчет из пользовательских единиц.

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

### 5.2 Функции дублиры для работы с пользовательскими единицами и структуры данных для них

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

## 5.3 Таблица коррекции координат для более точного позиционирования

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

### Заметки

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

### Формат файла:

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

### Пример файла:

X	dX
0	0
5.0	0.005
10.0	-0.01

## Глава 6

# Структуры данных

### 6.1 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.*

#### 6.1.1 Подробное описание

Информация о дополнительных аксессуарах.

См. также

[set\\_accessories\\_settings](#)  
[get\\_accessories\\_settings](#)  
[get\\_accessories\\_settings](#), [set\\_accessories\\_settings](#)

### 6.1.2 Поля

6.1.2.1 `unsigned int LimitSwitchesSettings`

[Флаги настроек температурного датчика.](#)

6.1.2.2 `char MagneticBrakeInfo[25]`

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

6.1.2.3 `float MBRatedCurrent`

Номинальный ток для управления магнитным тормозом (А).

Тип данных: `float`.

6.1.2.4 `float MBRatedVoltage`

Номинальное напряжение для управления магнитным тормозом (В).

Тип данных: `float`.

6.1.2.5 `unsigned int MBSettings`

[Флаги настроек энкодера.](#)

6.1.2.6 `float MBTorque`

Удерживающий момент (мН \* м).

Тип данных: `float`.

6.1.2.7 `char TemperatureSensorInfo[25]`

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

6.1.2.8 `float TSGrad`

Температурный градиент (В/град Цельсия).

Тип данных: `float`.

6.1.2.9 `float TSMax`

Максимальная измеряемая температура (град Цельсия) Тип данных: `float`.

6.1.2.10 `float` TSMIn

Минимальная измеряемая температура (град Цельсия).

Тип данных: `float`.

6.1.2.11 `unsigned int` TSSettings

Флаги настроек температурного датчика.

6.2 Структура `analog_data_t`

Аналоговые данные.

Поля данных

- `unsigned int` [A1Voltage\\_ADC](#)  
*"Выходное напряжение на 1 выводе обмотки A" необработанные данные с АЦП.*
- `unsigned int` [A2Voltage\\_ADC](#)  
*"Выходное напряжение на 2 выводе обмотки A" необработанные данные с АЦП.*
- `unsigned int` [B1Voltage\\_ADC](#)  
*"Выходное напряжение на 1 выводе обмотки B" необработанные данные с АЦП.*
- `unsigned int` [B2Voltage\\_ADC](#)  
*"Выходное напряжение на 2 выводе обмотки B" необработанные данные с АЦП.*
- `unsigned int` [SupVoltage\\_ADC](#)  
*"Напряжение питания ключей H-моста" необработанные данные с АЦП.*
- `unsigned int` [ACurrent\\_ADC](#)  
*"Ток через обмотку A" необработанные данные с АЦП.*
- `unsigned int` [BCurrent\\_ADC](#)  
*"Ток через обмотку B" необработанные данные с АЦП.*
- `unsigned int` [FullCurrent\\_ADC](#)  
*"Полный ток" необработанные данные с АЦП.*
- `unsigned int` [Temp\\_ADC](#)  
*Напряжение с датчика температуры, необработанные данные с АЦП.*
- `unsigned int` [Joy\\_ADC](#)  
*Джойстик, необработанные данные с АЦП.*
- `unsigned int` [Pot\\_ADC](#)  
*Напряжение на аналоговом входе, необработанные данные с АЦП.*
- `unsigned int` [Enc\\_Check\\_ADC](#)  
*Напряжение на линии проверки типа энкодера, необработанные данные АЦП.*
- `unsigned int` **deprecated0**
- `int` [A1Voltage](#)  
*"Выходное напряжение на 1 выводе обмотки A" откалиброванные данные (в десятках мВ).*
- `int` [A2Voltage](#)  
*"Выходное напряжение на 2 выводе обмотки A" откалиброванные данные (в десятках мВ).*
- `int` [B1Voltage](#)  
*"Выходное напряжение на 1 выводе обмотки B" откалиброванные данные (в десятках мВ).*
- `int` [B2Voltage](#)  
*"Выходное напряжение на 2 выводе обмотки B" откалиброванные данные (в десятках мВ).*

- `int SupVoltage`  
*"Напряжение питания ключей H-моста" откалиброванные данные (в десятках мВ).*
- `int ACurrent`  
*"Ток через обмотку A" откалиброванные данные (в мА).*
- `int BCurrent`  
*"Ток через обмотку B" откалиброванные данные (в мА).*
- `int FullCurrent`  
*"Полный ток" откалиброванные данные (в мА).*
- `int Temp`  
*Температура, откалиброванные данные (в десятых долях градуса Цельсия).*
- `int Joy`  
*Джойстик во внутренних единицах.*
- `int Pot`  
*Аналоговый вход во внутренних единицах.*
- `int Enc_Check`  
*Напряжение на линии проверки типа энкодера, экспоненциально сглаженные данные АЦП.*
- `unsigned int deprecated1 [2]`
- `int R`  
*Сопротивление обмоток двигателя(для шагового двигателя), в мОм*
- `int L`  
*Псевдоиндуктивность обмоток двигателя(для шагового двигателя), в мкГн*

### 6.2.1 Подробное описание

Аналоговые данные.

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

См. также

```
get_analog_data  
get_analog_data
```

### 6.2.2 Поля

#### 6.2.2.1 `int A1Voltage`

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

#### 6.2.2.2 `unsigned int A1Voltage_ADC`

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

#### 6.2.2.3 `int A2Voltage`

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

#### 6.2.2.4 `unsigned int A2Voltage_ADC`

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

6.2.2.5 `int ACurrent`

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

6.2.2.6 `unsigned int ACurrent_ADC`

"Ток через обмотку A" необработанные данные с АЦП.

6.2.2.7 `int B1Voltage`

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

6.2.2.8 `unsigned int B1Voltage_ADC`

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

6.2.2.9 `int B2Voltage`

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

6.2.2.10 `unsigned int B2Voltage_ADC`

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

6.2.2.11 `int BCurrent`

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

6.2.2.12 `unsigned int BCurrent_ADC`

"Ток через обмотку B" необработанные данные с АЦП.

6.2.2.13 `int Enc_Check`

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

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

6.2.2.14 `unsigned int Enc_Check_ADC`

Напряжение на линии проверки типа энкодера, необработанные данные АЦП.

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

6.2.2.15 `int FullCurrent`

"Полный ток" откалиброванные данные (в мА).

6.2.2.16 `unsigned int FullCurrent_ADC`

"Полный ток" необработанные данные с АЦП.

6.2.2.17 `int Joy`

Джойстик во внутренних единицах.

Диапазон: 0..10000

6.2.2.18 `unsigned int Joy_ADC`

Джойстик, необработанные данные с АЦП.

6.2.2.19 `int Pot`

Аналоговый вход во внутренних единицах.

Диапазон: 0..10000

6.2.2.20 `int SupVoltage`

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

6.2.2.21 `unsigned int SupVoltage_ADC`

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

6.2.2.22 `int Temp`

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

6.2.2.23 `unsigned int Temp_ADC`

Напряжение с датчика температуры, необработанные данные с АЦП.

## 6.3 Структура `brake_settings_t`

Настройки тормоза.

Поля данных

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



*Время в мс между включением тормоза и отключением питания мотора.*

- unsigned int `BrakeFlags`  
*Флаги настроек тормоза.*

### 6.3.1 Подробное описание

Настройки тормоза.

Эта структура содержит параметры управления тормозом.

См. также

`set_brake_settings`  
`get_brake_settings`  
`get_brake_settings, set_brake_settings`

### 6.3.2 Поля

#### 6.3.2.1 unsigned int `BrakeFlags`

*Флаги настроек тормоза.*

#### 6.3.2.2 unsigned int `t1`

*Время в мс между включением питания мотора и отключением тормоза.*

#### 6.3.2.3 unsigned int `t2`

*Время в мс между отключением тормоза и готовностью к движению.*

*Все команды движения начинают выполняться только по истечении этого времени.*

#### 6.3.2.4 unsigned int `t3`

*Время в мс между остановкой мотора и включением тормоза.*

#### 6.3.2.5 unsigned int `t4`

*Время в мс между включением тормоза и отключением питания мотора.*

## 6.4 Структура `calibration_settings_t`

Калибровочные коэффициенты.

Поля данных

- float `CSS1_A`  
*Коэффициент масштабирования для аналоговых измерений тока в обмотке A.*
- float `CSS1_B`  
*Коэффициент сдвига для аналоговых измерений тока в обмотке A.*

- float `CSS2_A`  
*Коэффициент масштабирования для аналоговых измерений тока в обмотке B.*
- float `CSS2_B`  
*Коэффициент сдвига для аналоговых измерений тока в обмотке B.*
- float `FullCurrent_A`  
*Коэффициент масштабирования для аналоговых измерений полного тока.*
- float `FullCurrent_B`  
*Коэффициент сдвига для аналоговых измерений полного тока.*

#### 6.4.1 Подробное описание

Калибровочные коэффициенты.

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

См. также

`get_calibration_settings`  
`set_calibration_settings`  
`get_calibration_settings, set_calibration_settings`

#### 6.4.2 Поля

##### 6.4.2.1 float `CSS1_A`

Коэффициент масштабирования для аналоговых измерений тока в обмотке A.

##### 6.4.2.2 float `CSS1_B`

Коэффициент сдвига для аналоговых измерений тока в обмотке A.

##### 6.4.2.3 float `CSS2_A`

Коэффициент масштабирования для аналоговых измерений тока в обмотке B.

##### 6.4.2.4 float `CSS2_B`

Коэффициент сдвига для аналоговых измерений тока в обмотке B.

##### 6.4.2.5 float `FullCurrent_A`

Коэффициент масштабирования для аналоговых измерений полного тока.

##### 6.4.2.6 float `FullCurrent_B`

Коэффициент сдвига для аналоговых измерений полного тока.

## 6.5 Структура calibration\_t

Структура калибровок

Поля данных

- double [A](#)  
*Коэффициент преобразования, равный количеству миллиметров (или других пользовательских единиц) на один шаг.*
- unsigned int [MicrostepMode](#)  
*Настройка контроллера, определяющая режим пошагового деления.*

### 6.5.1 Подробное описание

Структура калибровок

Где найти все значения для расчета?

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

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

1. Используя XILab, загрузите профиль для вашего позиционера. Профиль должен соответствовать полному названию вашего позиционера. Например: 8MT173-25-MEn1.cfg
2. Включите режим Feedback encoder, если он не был включен ранее
3. Вводите скорость в пользовательских единицах в поле Working speed
4. Во вкладке User units выключите флаг User units. Это позволит видеть значение в поле Working speed в RPM
5. Умножьте значение из поля Working speed (в RPM) на коэффициент B. Например:  $480 * 0.000009375 = 0.00045$ . Значение 0.00045 для позиционера 8MT173-25-MEn1 в режиме Encoder будет равно скорости 2 мм/сек

Ускорение, замедление и скорость в режиме антилюфта считаются аналогично

### 6.5.2 Поля

#### 6.5.2.1 `double A`

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

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

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

#### 6.5.2.2 `unsigned int MicrostepMode`

Настройка контроллера, определяющая режим пошагового деления.

## 6.6 Структура `chart_data_t`

Дополнительное состояние устройства.

### Поля данных

- `int WindingVoltageA`  
*В случае ШД, напряжение на обмотке A (в десятках мВ); в случае бесщеточного, напряжение на первой обмотке; в случае DC - на единственной.*
- `int WindingVoltageB`  
*В случае ШД, напряжение на обмотке B (в десятках мВ); в случае бесщеточного, напряжение на второй обмотке; в случае DC не используется.*
- `int WindingVoltageC`  
*В случае бесщеточного, напряжение на третьей обмотке (в десятках мВ); в случае ШД и DC не используется.*
- `int WindingCurrentA`  
*В случае ШД, ток в обмотке A (в мА); в случае бесщеточного, ток в первой обмотке; в случае DC в единственной.*
- `int WindingCurrentB`  
*В случае ШД, ток в обмотке B (в мА); в случае бесщеточного, ток в второй обмотке; в случае DC не используется.*
- `int WindingCurrentC`  
*В случае бесщеточного, ток в третьей обмотке (в мА); в случае ШД и DC не используется.*
- `unsigned int Pot`  
*Значение на аналоговом входе.*

- unsigned int Joy

*Положение джойстика в десятичных долях.*

- int AveragedPowerRatio

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

### 6.6.1 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](#)

### 6.6.2 Поля

#### 6.6.2.1 int AveragedPowerRatio

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

#### 6.6.2.2 unsigned int Joy

Положение джойстика в десятичных долях.

Диапазон: 0..10000

#### 6.6.2.3 unsigned int Pot

Значение на аналоговом входе.

Диапазон: 0..10000

#### 6.6.2.4 int WindingCurrentA

В случае ШД, ток в обмотке А (в мА); в случае бесщеточного, ток в первой обмотке; в случае DC в единственной.

#### 6.6.2.5 int WindingCurrentB

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

#### 6.6.2.6 int WindingCurrentC

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

#### 6.6.2.7 int WindingVoltageA

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

## 6.6.2.8 int WindingVoltageB

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

## 6.6.2.9 int WindingVoltageC

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

## 6.7 Структура control\_settings\_calb\_t

Настройки управления с использованием пользовательских единиц.

## Поля данных

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

## 6.7.1 Подробное описание

Настройки управления с использованием пользовательских единиц.

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

См. также

[set\\_control\\_settings\\_calb](#)  
[get\\_control\\_settings\\_calb](#)  
[get\\_control\\_settings](#), [set\\_control\\_settings](#)

## 6.7.2 Поля

## 6.7.2.1 unsigned int Flags

[Флаги управления.](#)

## 6.7.2.2 unsigned int MaxClickTime

Максимальное время клика (в мс).

До истечения этого времени первая скорость не включается.

## 6.7.2.3 float MaxSpeed[10]

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

## 6.7.2.4 unsigned int Timeout[9]

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

## 6.8 Структура control\_settings\_t

Настройки управления.

Поля данных

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

### 6.8.1 Подробное описание

Настройки управления.

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

См. также

[set\\_control\\_settings](#)  
[get\\_control\\_settings](#)  
[get\\_control\\_settings, set\\_control\\_settings](#)

## 6.8.2 Поля

### 6.8.2.1 unsigned int Flags

[Флаги управления.](#)

### 6.8.2.2 unsigned int MaxClickTime

Максимальное время клика (в мс).

До истечения этого времени первая скорость не включается.

### 6.8.2.3 unsigned int MaxSpeed[10]

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

Диапазон: 0..100000.

### 6.8.2.4 unsigned int Timeout[9]

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

### 6.8.2.5 int uDeltaPosition

Дробная часть смещения в микрошагах.

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

### 6.8.2.6 unsigned int uMaxSpeed[10]

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

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

## 6.9 Структура `controller_name_t`

Пользовательское имя контроллера и флаги настройки.



Поля данных

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

### 6.9.1 Подробное описание

Пользовательское имя контроллера и флаги настройки.

См. также

[get\\_controller\\_name](#), [set\\_controller\\_name](#)

### 6.9.2 Поля

#### 6.9.2.1 `char ControllerName[17]`

Пользовательское имя контроллера.

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

#### 6.9.2.2 `unsigned int CtrlFlags`

*Флаги настроек контроллера.*

## 6.10 Структура `ctp_settings_t`

Настройки контроля позиции(для шагового двигателя).

Поля данных

- `unsigned int CTPMinError`  
*Минимальное отличие шагов ШД от положения энкодера, устанавливающее флаг `STATE_RT_ERROR`.*
- `unsigned int CTPFlags`  
*Флаги контроля позиции.*

### 6.10.1 Подробное описание

Настройки контроля позиции(для шагового двигателя).

При управлении ШД с энкодером (`CTP_BASE 0`) появляется возможность обнаруживать потерю шагов. Контроллер знает 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](#)

### 6.10.2 Поля

#### 6.10.2.1 unsigned int CTPFlags

Флаги контроля позиции.

#### 6.10.2.2 unsigned int CTPMinError

Минимальное отличие шагов ШД от положения энкодера, устанавливающее флаг STATE\_RT\_ERROR.

Измеряется в шагах ШД.

## 6.11 Структура `debug_read_t`

Отладочные данные.

Поля данных

- `uint8_t DebugData [128]`  
*Отладочные данные.*

### 6.11.1 Подробное описание

Отладочные данные.

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

См. также

[get\\_debug\\_read](#)

### 6.11.2 Поля

#### 6.11.2.1 uint8\_t DebugData[128]

Отладочные данные.

## 6.12 Структура `debug_write_t`

Отладочные данные.

Поля данных

- `uint8_t` [DebugData](#) [128]

*Отладочные данные.*

#### 6.12.1 Подробное описание

Отладочные данные.

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

См. также

[set\\_debug\\_write](#)

#### 6.12.2 Поля

##### 6.12.2.1 `uint8_t` `DebugData`[128]

Отладочные данные.

### 6.13 Структура `device_information_t`

Информации о контроллере.

Поля данных

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

#### 6.13.1 Подробное описание

Информации о контроллере.

См. также

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

### 6.13.2 Поля

#### 6.13.2.1 unsigned int Major

Основной номер версии железа.

#### 6.13.2.2 unsigned int Minor

Второстепенный номер версии железа.

#### 6.13.2.3 unsigned int Release

Номер правок этой версии железа.

## 6.14 Структура `device__network__information__t`

Структура данных с информацией о сетевом устройстве.

Поля данных

- uint32\_t `ipv4`  
*IPv4-адрес, передаваемый в сетевом байтовом порядке (big-endian byte order)*
- char `nodename` [16]  
*имя узла Bindu, на котором размещено устройство*
- uint32\_t `axis_state`  
*флаги, представляющие состояние устройства*
- char `locker_username` [16]  
*имя пользователя, заблокировавшего устройство (если таковое имеется)*
- char `locker_nodename` [16]  
*имя узла Bindu, которое использовалось для блокировки устройства (если таковое имеется)*
- time\_t `locked_time`  
*время, в которое была получена блокировка (UTC, микросекунды с момента начала эпохи)*

### 6.14.1 Подробное описание

Структура данных с информацией о сетевом устройстве.

## 6.15 Структура `edges__settings__calb__t`

Настройки границ с использованием пользовательских единиц.

Поля данных

- unsigned int `BorderFlags`  
*Флаги границ.*
- unsigned int `EnderFlags`  
*Флаги концевых выключателей.*

- float [LeftBorder](#)

*Позиция левой границы, используется если установлен флаг `BORDER_IS_ENCODER`.*

- float [RightBorder](#)

*Позиция правой границы, используется если установлен флаг `BORDER_IS_ENCODER`.*

### 6.15.1 Подробное описание

Настройки границ с использованием пользовательских единиц.

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

См. также

[set\\_edges\\_settings\\_calb](#)  
[get\\_edges\\_settings\\_calb](#)  
[get\\_edges\\_settings](#), [set\\_edges\\_settings](#)

### 6.15.2 Поля

#### 6.15.2.1 unsigned int BorderFlags

[Флаги границ.](#)

#### 6.15.2.2 unsigned int EnderFlags

[Флаги концевых выключателей.](#)

#### 6.15.2.3 float LeftBorder

Позиция левой границы, используется если установлен флаг `BORDER_IS_ENCODER`.

Корректируется таблицей.

#### 6.15.2.4 float RightBorder

Позиция правой границы, используется если установлен флаг `BORDER_IS_ENCODER`.

Корректируется таблицей.

## 6.16 Структура `edges_settings_t`

Настройки границ.

Поля данных

- unsigned int [BorderFlags](#)

*[Флаги границ.](#)*

- unsigned int [EnderFlags](#)

*[Флаги концевых выключателей.](#)*

- `int LeftBorder`  
*Позиция левой границы, используется если установлен флаг `BORDER_IS_ENCODER`.*
- `int uLeftBorder`  
*Позиция левой границы в микрошагах (используется только с шаговым двигателем).*
- `int RightBorder`  
*Позиция правой границы, используется если установлен флаг `BORDER_IS_ENCODER`.*
- `int uRightBorder`  
*Позиция правой границы в микрошагах (используется только с шаговым двигателем).*

### 6.16.1 Подробное описание

Настройки границ.

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

См. также

```
set_edges_settings  
get_edges_settings  
get_edges_settings, set_edges_settings
```

### 6.16.2 Поля

#### 6.16.2.1 `unsigned int BorderFlags`

Флаги границ.

#### 6.16.2.2 `unsigned int EnderFlags`

Флаги концевых выключателей.

#### 6.16.2.3 `int LeftBorder`

Позиция левой границы, используется если установлен флаг `BORDER_IS_ENCODER`.

#### 6.16.2.4 `int RightBorder`

Позиция правой границы, используется если установлен флаг `BORDER_IS_ENCODER`.

#### 6.16.2.5 `int uLeftBorder`

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

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

6.16.2.6 `int uRightBorder`

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

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

6.17 Структура `emf_settings_t`

Настройки EMF.

Поля данных

- float `L`  
*Индуктивность обмоток двигателя.*
- float `R`  
*Сопротивление обмоток двигателя.*
- float `Km`  
*Электромеханический коэффициент двигателя.*
- unsigned int `BackEMFFlags`  
*Флаги автоопределения характеристик обмоток двигателя.*

## 6.17.1 Подробное описание

Настройки EMF.

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

См. также

```
set_emf_settings  
get_emf_settings  
get_emf_settings, set_emf_settings
```

## 6.17.2 Поля

6.17.2.1 unsigned int `BackEMFFlags`

*Флаги автоопределения характеристик обмоток двигателя.*

6.17.2.2 float `Km`

*Электромеханический коэффициент двигателя.*

6.17.2.3 float `L`

*Индуктивность обмоток двигателя.*

## 6.17.2.4 float R

Сопротивление обмоток двигателя.

6.18 Структура `encoder_information_t`

Информация об энкодере.

Поля данных

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

## 6.18.1 Подробное описание

Информация об энкодере.

См. также

[set\\_encoder\\_information](#)  
[get\\_encoder\\_information](#)  
[get\\_encoder\\_information](#), [set\\_encoder\\_information](#)

## 6.18.2 Поля

6.18.2.1 char `Manufacturer`[17]

Производитель.

Максимальная длина строки: 16 символов.

6.18.2.2 char `PartNumber`[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

6.19 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.*

### 6.19.1 Подробное описание

Настройки энкодера.

См. также

[set\\_encoder\\_settings](#)  
[get\\_encoder\\_settings](#)  
[get\\_encoder\\_settings](#), [set\\_encoder\\_settings](#)

### 6.19.2 Поля

#### 6.19.2.1 unsigned int EncoderSettings

[Флаги настроек энкодера.](#)

#### 6.19.2.2 float MaxCurrentConsumption

Максимальное потребление тока (мА).

Тип данных: float.

#### 6.19.2.3 float MaxOperatingFrequency

Максимальная частота (кГц).

Тип данных: float.

#### 6.19.2.4 float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

#### 6.19.2.5 float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

## 6.20 Структура engine\_advanced\_setup\_t

Настройки EAS.

## Поля данных

- unsigned int `stepcloseloop_Kw`  
*Коэффициент смещения реальной и заданной скорости, диапазон [0, 100], значение по умолчанию 50.*
- unsigned int `stepcloseloop_Kp_low`  
*Обратная связь по позиции в зоне малых скоростей, диапазон [0, 65535], значение по умолчанию 1000.*
- unsigned int `stepcloseloop_Kp_high`  
*Обратная связь по позиции в зоне больших скоростей, диапазон [0, 65535], значение по умолчанию 33.*

## 6.20.1 Подробное описание

Настройки EAS.

Эта структура предназначена для настройки параметров алгоритмов, которые невозможно отнести к стандартным  $K_p$ ,  $K_i$ ,  $K_d$  и  $L$ ,  $R$ ,  $K_m$ . Эти данные хранятся во flash памяти контроллера.

См. также

`set_engine_advansed_setup`  
`get_engine_advansed_setup`  
`get_engine_advansed_setup`, `set_engine_advansed_setup`

## 6.20.2 Поля

6.20.2.1 unsigned int `stepcloseloop_Kp_high`

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

6.20.2.2 unsigned int `stepcloseloop_Kp_low`

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

6.20.2.3 unsigned int `stepcloseloop_Kw`

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

6.21 Структура `engine_settings_calb_t`

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

## Поля данных

- unsigned int `NomVoltage`  
*Номинальное напряжение мотора в десятках мВ.*

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

### 6.21.1 Подробное описание

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

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

См. также

[set\\_engine\\_settings\\_calb](#)  
[get\\_engine\\_settings\\_calb](#)  
[get\\_engine\\_settings, set\\_engine\\_settings](#)

### 6.21.2 Поля

#### 6.21.2.1 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.

#### 6.21.2.2 unsigned int EngineFlags

[Флаги параметров мотора.](#)

#### 6.21.2.3 unsigned int MicrostepMode

[Флаги параметров микрошагового режима.](#)

#### 6.21.2.4 unsigned int NomCurrent

Номинальный ток через мотор (в мА).

Ток стабилизируется для шаговых и может быть ограничен для DC(если установлен флаг ENGINE\_ LIMIT\_CURR). Диапазон: 15..8000

## 6.21.2.5 float NomSpeed

Номинальная скорость.

Контроллер будет сохранять скорость мотора не выше номинальной, если установлен флаг ENGINE\_LIMIT\_RPM.

## 6.21.2.6 unsigned int NomVoltage

Номинальное напряжение мотора в десятках мВ.

Контроллер будет сохранять напряжение на моторе не выше номинального, если установлен флаг ENGINE\_LIMIT\_VOLT (используется только с DC двигателем).

## 6.21.2.7 unsigned int StepsPerRev

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

Диапазон: 1..65535.

## 6.22 Структура engine\_settings\_t

Ограничения и настройки движения, связанные с двигателем.

## Поля данных

- unsigned int [NomVoltage](#)  
*Номинальное напряжение мотора в десятках мВ.*
- unsigned int [NomCurrent](#)  
*Номинальный ток через мотор (в мА).*
- unsigned int [NomSpeed](#)  
*Номинальная (максимальная) скорость (в целых шагах/с или грт 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).*

## 6.22.1 Подробное описание

Ограничения и настройки движения, связанные с двигателем.

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

См. также

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

## 6.22.2 Поля

### 6.22.2.1 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.

### 6.22.2.2 unsigned int EngineFlags

[Флаги параметров мотора.](#)

### 6.22.2.3 unsigned int MicrostepMode

[Флаги параметров микрошагового режима.](#)

### 6.22.2.4 unsigned int NomCurrent

Номинальный ток через мотор (в мА).

Ток стабилизируется для шаговых и может быть ограничен для DC(если установлен флаг ENGINE\_LIMIT\_CURR). Диапазон: 15..8000

### 6.22.2.5 unsigned int NomSpeed

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

Контроллер будет сохранять скорость мотора не выше номинальной, если установлен флаг ENGINE\_LIMIT\_RPM. Диапазон: 1..100000.

### 6.22.2.6 unsigned int NomVoltage

Номинальное напряжение мотора в десятках мВ.

Контроллер будет сохранять напряжение на моторе не выше номинального, если установлен флаг ENGINE\_LIMIT\_VOLT (используется только с DC двигателем).

### 6.22.2.7 unsigned int StepsPerRev

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

Диапазон: 1..65535.

### 6.22.2.8 unsigned int uNomSpeed

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

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

## 6.23 Структура `entype_settings_t`

Настройки типа мотора и типа силового драйвера.

Поля данных

- unsigned int `EngineType`  
*Флаги, определяющие тип мотора.*
- unsigned int `DriverType`  
*Флаги, определяющие тип силового драйвера.*

### 6.23.1 Подробное описание

Настройки типа мотора и типа силового драйвера.

Эта структура содержит настройки типа мотора и типа силового драйвера.

Аргументы

<i>id</i>	идентификатор устройства
<i>EngineType</i>	тип мотора
<i>DriverType</i>	тип силового драйвера

См. также

`get_entype_settings`, `set_entype_settings`

### 6.23.2 Поля

#### 6.23.2.1 unsigned int `DriverType`

*Флаги, определяющие тип силового драйвера.*

#### 6.23.2.2 unsigned int `EngineType`

*Флаги, определяющие тип мотора.*

## 6.24 Структура `extended_settings_t`

Настройки EST.

Поля данных

- unsigned int **`Param1`**

### 6.24.1 Подробное описание

Настройки EST.

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

См. также

[set\\_extended\\_settings](#)  
[get\\_extended\\_settings](#)  
[get\\_extended\\_settings](#), [set\\_extended\\_settings](#)

## 6.25 Структура extio\_settings\_t

Настройки EXTIO.

Поля данных

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

### 6.25.1 Подробное описание

Настройки EXTIO.

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

См. также

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

### 6.25.2 Поля

#### 6.25.2.1 unsigned int EXTIOModeFlags

*Флаги настройки режимов внешнего ввода/вывода.*

#### 6.25.2.2 unsigned int EXTIOSetupFlags

*Флаги настройки работы внешнего ввода/вывода.*

## 6.26 Структура feedback\_settings\_t

Настройки обратной связи.

Поля данных

- unsigned int `IPS`  
*Количество отсчётов энкодера на оборот вала.*
- unsigned int `FeedbackType`  
*Тип обратной связи.*
- unsigned int `FeedbackFlags`  
*Флаги обратной связи.*
- unsigned int `CountsPerTurn`  
*Количество отсчётов энкодера на оборот вала.*

### 6.26.1 Подробное описание

Настройки обратной связи.

Эта структура содержит настройки обратной связи.

См. также

[get\\_feedback\\_settings](#), [set\\_feedback\\_settings](#)

### 6.26.2 Поля

#### 6.26.2.1 unsigned int `CountsPerTurn`

Количество отсчётов энкодера на оборот вала.

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

#### 6.26.2.2 unsigned int `FeedbackFlags`

[Флаги обратной связи.](#)

#### 6.26.2.3 unsigned int `FeedbackType`

[Тип обратной связи.](#)

#### 6.26.2.4 unsigned int `IPS`

Количество отсчётов энкодера на оборот вала.

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

## 6.27 Структура `gear_information_t`

Информация о редукторе.



Поля данных

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

### 6.27.1 Подробное описание

Информация о редукторе.

См. также

[set\\_gear\\_information](#)  
[get\\_gear\\_information](#)  
[get\\_gear\\_information](#), [set\\_gear\\_information](#)

### 6.27.2 Поля

#### 6.27.2.1 char Manufacturer[17]

Производитель.

Максимальная длина строки: 16 символов.

#### 6.27.2.2 char PartNumber[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

## 6.28 Структура gear\_settings\_t

Настройки редуктора.

Поля данных

- float [ReductionIn](#)  
*Входной коэффициент редуктора.*
- float [ReductionOut](#)  
*Выходной коэффициент редуктора.*
- float [RatedInputTorque](#)  
*Максимальный крутящий момент ( $H * м$ ).*
- float [RatedInputSpeed](#)  
*Максимальная скорость на входном валу редуктора (об/мин).*
- float [MaxOutputBacklash](#)  
*Выходной люфт редуктора (градус).*
- float [InputInertia](#)  
*Эквивалентная входная инерция редуктора ( $г * см^2$ ).*
- float [Efficiency](#)  
*КПД редуктора (%).*

## 6.28.1 Подробное описание

Настройки редуктора.

См. также

[set\\_gear\\_settings](#)  
[get\\_gear\\_settings](#)  
[get\\_gear\\_settings, set\\_gear\\_settings](#)

## 6.28.2 Поля

## 6.28.2.1 float Efficiency

КПД редуктора (%).

Тип данных: float.

## 6.28.2.2 float InputInertia

Эквивалентная входная инерция редуктора(г \* см<sup>2</sup>).

Тип данных: float.

## 6.28.2.3 float MaxOutputBacklash

Выходной люфт редуктора (градус).

Тип данных: float.

## 6.28.2.4 float RatedInputSpeed

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

Тип данных: float.

## 6.28.2.5 float RatedInputTorque

Максимальный крутящий момент (Н \* м).

Тип данных: float.

## 6.28.2.6 float ReductionIn

Входной коэффициент редуктора.

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

## 6.28.2.7 float ReductionOut

Выходной коэффициент редуктора.

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

## 6.29 Структура `get_position_calb_t`

Данные о позиции.

Поля данных

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

### 6.29.1 Подробное описание

Данные о позиции.

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

См. также

[get\\_position](#)

### 6.29.2 Поля

#### 6.29.2.1 long\_t EncPosition

Позиция энкодера.

#### 6.29.2.2 float Position

Позиция двигателя.

Корректируется таблицей.

## 6.30 Структура `get_position_t`

Данные о позиции.

Поля данных

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

### 6.30.1 Подробное описание

Данные о позиции.

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

См. также

[get\\_position](#)

### 6.30.2 Поля

#### 6.30.2.1 `long_t EncPosition`

Позиция энкодера.

#### 6.30.2.2 `int uPosition`

Позиция в микрошагах (используется только с шаговыми двигателями).

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

## 6.31 Структура `globally_unique_identifier_t`

Глобальный уникальный идентификатор.

Поля данных

- unsigned int [UniqueID0](#)  
*Уникальный ID 0.*
- unsigned int [UniqueID1](#)  
*Уникальный ID 1.*
- unsigned int [UniqueID2](#)  
*Уникальный ID 2.*
- unsigned int [UniqueID3](#)  
*Уникальный ID 3.*

### 6.31.1 Подробное описание

Глобальный уникальный идентификатор.

Только для производителя.

См. также

[get\\_globally\\_unique\\_identifier](#)

## 6.31.2 Поля

6.31.2.1 `unsigned int UniqueID0`

Уникальный ID 0.

6.31.2.2 `unsigned int UniqueID1`

Уникальный ID 1.

6.31.2.3 `unsigned int UniqueID2`

Уникальный ID 2.

6.31.2.4 `unsigned int UniqueID3`

Уникальный ID 3.

6.32 Структура `hallsensor_information_t`

Информация о датчиках Холла.

Поля данных

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

## 6.32.1 Подробное описание

Информация о датчиках Холла.

См. также

`set_hallsensor_information`  
`get_hallsensor_information`  
`get_hallsensor_information, set_hallsensor_information`

## 6.32.2 Поля

6.32.2.1 `char Manufacturer`[17]

Производитель.

Максимальная длина строки: 16 символов.

6.32.2.2 `char PartNumber[25]`

Серия и номер модели.

Максимальная длина строки: 24 символа.

## 6.33 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*

### 6.33.1 Подробное описание

Настройки датчиков Холла.

См. также

[set\\_hallsensor\\_settings](#)  
[get\\_hallsensor\\_settings](#)  
[get\\_hallsensor\\_settings](#), [set\\_hallsensor\\_settings](#)

### 6.33.2 Поля

6.33.2.1 `float MaxCurrentConsumption`

Максимальное потребление тока (мА).

Тип данных: `float`.

6.33.2.2 `float MaxOperatingFrequency`

Максимальная частота (кГц).

Тип данных: `float`.

6.33.2.3 `float SupplyVoltageMax`

Максимальное напряжение питания (В).

Тип данных: `float`.

## 6.33.2.4 float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

6.34 Структура `home_settings_calb_t`

Настройки калибровки позиции с использованием пользовательских единиц.

Поля данных

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

## 6.34.1 Подробное описание

Настройки калибровки позиции с использованием пользовательских единиц.

Эта структура содержит настройки, используемые при калибровке позиции.

См. также

```
get_home_settings_calb
set_home_settings_calb
command_home
get_home_settings, set_home_settings
```

## 6.34.2 Поля

## 6.34.2.1 float FastHome

Скорость первого движения.

## 6.34.2.2 float HomeDelta

Расстояние отхода от точки останова.

## 6.34.2.3 unsigned int HomeFlags

[Флаги настроек команды home.](#)

## 6.34.2.4 float SlowHome

Скорость второго движения.

## 6.35 Структура `home_settings_t`

Настройки калибровки позиции.

Поля данных

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

### 6.35.1 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`

### 6.35.2 Поля

#### 6.35.2.1 unsigned int `FastHome`

Скорость первого движения (в полных шагах).

Диапазон: 0..100000

#### 6.35.2.2 int `HomeDelta`

Расстояние отхода от точки останова (в полных шагах).

#### 6.35.2.3 unsigned int `HomeFlags`

Флаги настроек команды `home`.



6.35.2.4 `unsigned int SlowHome`

Скорость второго движения (в полных шагах).

Диапазон: 0..100000.

6.35.2.5 `unsigned int uFastHome`

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

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

6.35.2.6 `int uHomeDelta`

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

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

6.35.2.7 `unsigned int uSlowHome`

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

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

## 6.36 Структура `init_random_t`

Случайный ключ.

Поля данных

- `uint8_t key [16]`  
*Случайный ключ.*

### 6.36.1 Подробное описание

Случайный ключ.

Только для производителя. Структура которая содержит случайный ключ, использующийся для шифрования содержимого команд `WKEY` и `SSER`.

См. также

[get\\_init\\_random](#)

## 6.36.2 Поля

## 6.36.2.1 uint8\_t key[16]

Случайный ключ.

## 6.37 Структура joystick\_settings\_t

Настройки джойстика.

## Поля данных

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

## 6.37.1 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 (при этом постоянно выполняется команда "soft stop"), а 100% отклонения соответствует MaxSpeed  $i$ , где  $i=0$ , если предыдущим использованием этого режима не было выбрано другое  $i$ . Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. Первая скорость в списке не должна быть нулевой. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Зависимость между отклонением и скоростью экспоненциальная, что позволяет без переключения режимов скорости сочетать высокую подвижность и точность.

См. также

[set\\_joystick\\_settings](#)  
[get\\_joystick\\_settings](#)  
[get\\_joystick\\_settings](#), [set\\_joystick\\_settings](#)

## 6.37.2 Поля

## 6.37.2.1 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.

## 6.37.2.2 unsigned int ExpFactor

Фактор экспоненциальной нелинейности отклика джойстика.

## 6.37.2.3 unsigned int JoyCenter

Значение в шагах джойстика, соответствующее неотклонённому устройству.

Должно лежать в пределах. Диапазон: 0..10000.

## 6.37.2.4 unsigned int JoyFlags

Флаги джойстика.

## 6.37.2.5 unsigned int JoyHighEnd

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

Должно лежать в пределах. Диапазон: 0..10000.

## 6.37.2.6 unsigned int JoyLowEnd

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

Должно лежать в пределах. Диапазон: 0..10000.

## 6.38 Структура measurements\_t

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

Поля данных

- int [Speed](#) [25]

*Последовательность измеренных скоростей (в отсчётах энкодера/сек или микрошагах/сек, в зависимости от типа двигателя и режима управления)*

- int [Error](#) [25]

*Последовательность измеренных ошибок позиции (в отсчётах энкодера или микрошагах, в зависимости от типа двигателя и режима управления)*

- unsigned int [Length](#)

*Фактическая длина последовательности.*

### 6.38.1 Подробное описание

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

Шаг по времени между двумя последовательными измерениями - 1 мс.

См. также

`measurements`  
[get\\_measurements](#)

### 6.38.2 Поля

#### 6.38.2.1 unsigned int Length

Фактическая длина последовательности.

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

## 6.39 Структура `motor_information_t`

Информация о двигателе.

Поля данных

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

### 6.39.1 Подробное описание

Информация о двигателе.

См. также

[set\\_motor\\_information](#)  
[get\\_motor\\_information](#)  
[get\\_motor\\_information, set\\_motor\\_information](#)

### 6.39.2 Поля

#### 6.39.2.1 char Manufacturer[17]

Производитель.

Максимальная длина строки: 16 символов.

## 6.39.2.2 char PartNumber[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

## 6.40 Структура motor\_settings\_t

Физический характеристики и ограничения мотора.

Поля данных

- unsigned int [MotorType](#)  
*Флаги 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](#)  
*Не используется.*
- float [NominalTorque](#)  
*Номинальный крутящий момент (мН \* м).*
- float [NominalPower](#)  
*Номинальная мощность(Вт).*
- float [WindingResistance](#)  
*Сопротивление обмотки DC двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC двигателя (Ом).*
- float [WindingInductance](#)  
*Индуктивность обмотки DC двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC двигателя (мГн).*
- float [RotorInertia](#)  
*Инерция ротора (г см<sup>2</sup>).*
- 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).*

#### 6.40.1 Подробное описание

Физические характеристики и ограничения мотора.

См. также

`set_motor_settings`  
`get_motor_settings`  
`get_motor_settings, set_motor_settings`

#### 6.40.2 Поля

##### 6.40.2.1 float `DetentTorque`

Момент удержания позиции с незапитанными обмотками (мН м).

Тип данных: float.

##### 6.40.2.2 float `MaxCurrent`

Максимальный ток в обмотке (А).

Тип данных: float.

##### 6.40.2.3 float `MaxCurrentTime`

Безопасная длительность максимального тока в обмотке (мс).

Тип данных: float.

##### 6.40.2.4 float `MaxSpeed`

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

Тип данных: float.

6.40.2.5 float MechanicalTimeConstant

Механическая постоянная времени (мс).

Тип данных: float.

6.40.2.6 unsigned int MotorType

Флаги типа двигателя.

6.40.2.7 float NoLoadCurrent

Ток потребления в холостом режиме (А).

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

6.40.2.8 float NoLoadSpeed

Скорость в холостом режиме (об/мин).

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

6.40.2.9 float NominalCurrent

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

Тип данных: float.

6.40.2.10 float NominalPower

Номинальная мощность(Вт).

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

6.40.2.11 float NominalSpeed

Не используется.

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

6.40.2.12 float NominalTorque

Номинальный крутящий момент (мН \* м).

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

6.40.2.13 float NominalVoltage

Номинальное напряжение на обмотке (В).

Тип данных: float.

## 6.40.2.14 unsigned int Phases

Кол-во фаз у BLDC двигателя.

## 6.40.2.15 unsigned int Poles

Кол-во пар полюсов у DC или BLDC двигателя или кол-во шагов на оборот для шагового двигателя.

## 6.40.2.16 float RotorInertia

Инерция ротора (г см<sup>2</sup>).

Тип данных: float.

## 6.40.2.17 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.

## 6.40.2.18 float SpeedTorqueGradient

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

Тип данных: float.

## 6.40.2.19 float StallTorque

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

Тип данных: float.

## 6.40.2.20 float TorqueConstant

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

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

## 6.40.2.21 float WindingInductance

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

Тип данных: float.

## 6.40.2.22 float WindingResistance

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

Тип данных: float.



## 6.41 Структура `move_settings_calb_t`

Настройки движения с использованием пользовательских единиц.

Поля данных

- float [Speed](#)  
*Скорость*
- float [Accel](#)  
*Ускорение*
- float [Decel](#)  
*Торможение*
- float [AntiplaySpeed](#)  
*Скорость в режиме антилюфта*
- unsigned int [MoveFlags](#)  
*Флаги параметров движения.*

### 6.41.1 Подробное описание

Настройки движения с использованием пользовательских единиц.

См. также

[set\\_move\\_settings\\_calb](#)  
[get\\_move\\_settings\\_calb](#)  
[get\\_move\\_settings](#), [set\\_move\\_settings](#)

### 6.41.2 Поля

#### 6.41.2.1 float Accel

Ускорение

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

#### 6.41.2.2 float AntiplaySpeed

Скорость в режиме антилюфта

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

#### 6.41.2.3 float Decel

Торможение

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

## 6.41.2.4 unsigned int MoveFlags

Флаги параметров движения.

## 6.41.2.5 float Speed

Скорость

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

6.42 Структура `move_settings_t`

Настройки движения.

Поля данных

- unsigned int `Speed`  
*Заданная скорость (для ШД: шагов/с, для DC: rpm).*
- unsigned int `uSpeed`  
*Заданная скорость в единицах деления микрошага в секунду.*
- unsigned int `Accel`  
*Ускорение, заданное в шагах в секунду<sup>2</sup> (ШД) или в оборотах в минуту за секунду (DC).*
- unsigned int `Decel`  
*Торможение, заданное в шагах в секунду<sup>2</sup> (ШД) или в оборотах в минуту за секунду (DC).*
- unsigned int `AntiplaySpeed`  
*Скорость в режиме антилюфта, заданная в целых шагах/с (ШД) или в оборотах/с (DC).*
- unsigned int `uAntiplaySpeed`  
*Скорость в режиме антилюфта, выраженная в микрошагах в секунду.*
- unsigned int `MoveFlags`  
*Флаги параметров движения.*

## 6.42.1 Подробное описание

Настройки движения.

См. также

`set_move_settings`  
`get_move_settings`  
`get_move_settings, set_move_settings`

## 6.42.2 Поля

## 6.42.2.1 unsigned int Accel

Ускорение, заданное в шагах в секунду<sup>2</sup> (ШД) или в оборотах в минуту за секунду (DC).

Диапазон: 1..65535.

## 6.42.2.2 unsigned int AntiplaySpeed

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

Диапазон: 0..100000.

## 6.42.2.3 unsigned int Decel

Торможение, заданное в шагах в секунду<sup>2</sup> (ШД) или в оборотах в минуту за секунду (DC).

Диапазон: 1..65535.

## 6.42.2.4 unsigned int MoveFlags

Флаги параметров движения.

## 6.42.2.5 unsigned int Speed

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

Диапазон: 0..100000.

## 6.42.2.6 unsigned int uAntiplaySpeed

Скорость в режиме антилюфта, выраженная в микрошагах в секунду.

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

## 6.42.2.7 unsigned int uSpeed

Заданная скорость в единицах деления микрошага в секунду.

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

6.43 Структура `network_settings_t`

Настройки сети.

Поля данных

- unsigned int [DHCPEnabled](#)  
*Определяет способ получения IP-адреса каналов.*
- unsigned int [IPv4Address](#) [4]  
*IP-адрес устройства в формате x.x.x.x.*
- unsigned int [SubnetMask](#) [4]  
*Маска подсети в формате x.x.x.x.*
- unsigned int [DefaultGateway](#) [4]  
*Шлюз сети по умолчанию в формате x.x.x.x.*

### 6.43.1 Подробное описание

Настройки сети.

Только для производителя. Эта структура содержит настройки сети.

См. также

[get\\_network\\_settings](#)  
[set\\_network\\_settings](#)  
[get\\_network\\_settings](#), [set\\_network\\_settings](#)

### 6.43.2 Поля

#### 6.43.2.1 unsigned int DefaultGateway[4]

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

#### 6.43.2.2 unsigned int DHCPEnabled

Определяет способ получения IP-адреса каналов.

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

#### 6.43.2.3 unsigned int IPv4Address[4]

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

#### 6.43.2.4 unsigned int SubnetMask[4]

Маска подсети в формате x.x.x.x.

## 6.44 Структура `nonvolatile_memory_t`

Пользовательские данные для сохранения во FRAM.

Поля данных

- unsigned int [UserData](#) [7]  
*Пользовательские данные.*

### 6.44.1 Подробное описание

Пользовательские данные для сохранения во FRAM.

См. также

[get\\_nonvolatile\\_memory](#), [set\\_nonvolatile\\_memory](#)

## 6.44.2 Поля

## 6.44.2.1 unsigned int UserData[7]

Пользовательские данные.

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

## 6.45 Структура password\_settings\_t

Пароль.

Поля данных

- char [UserPassword](#) [21]

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

## 6.45.1 Подробное описание

Пароль.

Только для производителя. Эта структура содержит пароль к веб-странице.

См. также

[get\\_password\\_settings](#)  
[set\\_password\\_settings](#)  
[get\\_password\\_settings](#), [set\\_password\\_settings](#)

## 6.45.2 Поля

## 6.45.2.1 char UserPassword[21]

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

## 6.46 Структура pid\_settings\_t

Настройки ПИД.

Поля данных

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

- Дифференциальный коэффициент ПИД контура по напряжению
  - float [Kpf](#)
    - Пропорциональный коэффициент ПИД контура по позиции для BLDC.
- float [Kif](#)
  - Интегральный коэффициент ПИД контура по позиции для BLDC.
- float [Kdf](#)
  - Дифференциальный коэффициент ПИД контура по позиции для BLDC.

#### 6.46.1 Подробное описание

Настройки ПИД.

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

См. также

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

## 6.47 Структура power\_settings\_t

Настройки питания шагового мотора.

Поля данных

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

#### 6.47.1 Подробное описание

Настройки питания шагового мотора.

См. также

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

## 6.47.2 Поля

## 6.47.2.1 unsigned int CurrentSetTime

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

## 6.47.2.2 unsigned int CurrReductDelay

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

## 6.47.2.3 unsigned int HoldCurrent

Ток мотора в режиме удержания, в процентах от номинального.

Диапазон: 0..100.

## 6.47.2.4 unsigned int PowerFlags

Флаги параметров питания шагового мотора.

## 6.47.2.5 unsigned int PowerOffDelay

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

6.48 Структура `secure_settings_t`

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

## Поля данных

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

## 6.48.1 Подробное описание

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

См. также

[get\\_secure\\_settings](#)  
[set\\_secure\\_settings](#)  
[get\\_secure\\_settings, set\\_secure\\_settings](#)

## 6.48.2 Поля

6.48.2.1 `unsigned int Criticalpwr`

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

6.48.2.2 `unsigned int Criticalusb`

Устарело.

Максимальный ток USB, вызывающий состояние ALARM, в мА.

6.48.2.3 `unsigned int CriticalUpwr`

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

6.48.2.4 `unsigned int CriticalUusb`

Устарело.

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

6.48.2.5 `unsigned int Flags`

[Флаги критических параметров.](#)

6.48.2.6 `unsigned int LowUpwrOff`

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

6.48.2.7 `unsigned int MinimumUusb`

Устарело.

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

6.49 Структура `serial_number_t`

Структура с серийным номером и версией железа.



Поля данных

- unsigned int `SN`  
*Новый серийный номер платы.*
- uint8\_t `Key` [32]  
*Ключ защиты для установки серийного номера (256 бит).*
- unsigned int `Major`  
*Основной номер версии железа.*
- unsigned int `Minor`  
*Второстепенный номер версии железа.*
- unsigned int `Release`  
*Номер правок этой версии железа.*

#### 6.49.1 Подробное описание

Структура с серийным номером и версией железа.

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

См. также

[set\\_serial\\_number](#)

#### 6.49.2 Поля

##### 6.49.2.1 uint8\_t Key[32]

Ключ защиты для установки серийного номера (256 бит).

##### 6.49.2.2 unsigned int Major

Основной номер версии железа.

##### 6.49.2.3 unsigned int Minor

Второстепенный номер версии железа.

##### 6.49.2.4 unsigned int Release

Номер правок этой версии железа.

##### 6.49.2.5 unsigned int SN

Новый серийный номер платы.

## 6.50 Структура `set_position_calb_t`

Данные о позиции с использованием пользовательских единиц.

Поля данных

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

#### 6.50.1 Подробное описание

Данные о позиции с использованием пользовательских единиц.

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

См. также

`set_position`

#### 6.50.2 Поля

##### 6.50.2.1 long\_t EncPosition

Позиция энкодера.

##### 6.50.2.2 unsigned int PosFlags

*Флаги установки положения.*

##### 6.50.2.3 float Position

Позиция двигателя.

### 6.51 Структура `set_position_t`

Данные о позиции.

Поля данных

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

### 6.51.1 Подробное описание

Данные о позиции.

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

См. также

[set\\_position](#)

### 6.51.2 Поля

#### 6.51.2.1 `long_t EncPosition`

Позиция энкодера.

#### 6.51.2.2 `unsigned int PosFlags`

[Флаги установки положения.](#)

#### 6.51.2.3 `int uPosition`

Позиция в микрошагах (используется только с шаговыми двигателями).

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

## 6.52 Структура `stage_information_t`

Информация о позиционере.

Поля данных

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

### 6.52.1 Подробное описание

Информация о позиционере.

См. также

[set\\_stage\\_information](#)  
[get\\_stage\\_information](#)  
[get\\_stage\\_information, set\\_stage\\_information](#)

## 6.52.2 Поля

6.52.2.1 `char Manufacturer[17]`

Производитель.

Максимальная длина строки: 16 символов.

6.52.2.2 `char PartNumber[25]`

Серия и номер модели.

Максимальная длина строки: 24 символа.

6.53 Структура `stage_name_t`

Пользовательское имя подвижки.

Поля данных

- `char PositionerName [17]`

*Пользовательское имя подвижки.*

## 6.53.1 Подробное описание

Пользовательское имя подвижки.

См. также

[get\\_stage\\_name](#), [set\\_stage\\_name](#)

## 6.53.2 Поля

6.53.2.1 `char PositionerName[17]`

Пользовательское имя подвижки.

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

6.54 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 (V).*
- float `SupplyVoltageMax`  
*Максимальное напряжение питания (V).*
- float `MaxCurrentConsumption`  
*Максимальный ток потребления (A).*
- float `HorizontalLoadCapacity`  
*Горизонтальная грузоподъемность (кг).*
- float `VerticalLoadCapacity`  
*Вертикальная грузоподъемность (кг).*

#### 6.54.1 Подробное описание

Настройки позиционера.

См. также

[set\\_stage\\_settings](#)  
[get\\_stage\\_settings](#)  
[get\\_stage\\_settings, set\\_stage\\_settings](#)

#### 6.54.2 Поля

##### 6.54.2.1 float `HorizontalLoadCapacity`

Горизонтальная грузоподъемность (кг).

Тип данных: float.

##### 6.54.2.2 float `LeadScrewPitch`

Шаг ходового винта в мм.

Тип данных: float.

##### 6.54.2.3 float `MaxCurrentConsumption`

Максимальный ток потребления (A).

Тип данных: float.

##### 6.54.2.4 float `MaxSpeed`

Максимальная скорость (Units/c).

Тип данных: float.

## 6.54.2.5 float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

## 6.54.2.6 float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

## 6.54.2.7 float TravelRange

Диапазон перемещения (Units).

Тип данных: float.

## 6.54.2.8 char Units[9]

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

## 6.54.2.9 float VerticalLoadCapacity

Вертикальная грузоподъемность (кг).

Тип данных: float.

## 6.55 Структура status\_calb\_t

Состояние устройства с использованием пользовательских единиц.

Поля данных

- unsigned int [MoveSts](#)  
*Флаги состояния движения.*
- unsigned int [MvCmdSts](#)  
*Состояние команды движения.*
- unsigned int [PWRSts](#)  
*Флаги состояния питания шагового мотора.*
- unsigned int [EncSts](#)  
*Состояние энкодера.*
- unsigned int [WindSts](#)  
*Состояние обмоток.*
- float [CurPosition](#)  
*Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь.*
- long\_t [EncPosition](#)  
*Текущая позиция по данным с энкодера в импульсах энкодера, используется только если энкодер установлен, активизирован и не является основным датчиком положения, например при использовании энкодера совместно с шаговым двигателем для контроля проскальзывания.*

- float `CurSpeed`  
*Текущая скорость.*
- int `lpwr`  
*Ток потребления силовой части, мА.*
- int `Upwr`  
*Напряжение на силовой части, десятки мВ.*
- int `lusb`  
*Ток потребления по USB, мА.*
- int `Uusb`  
*Напряжение на USB, десятки мВ.*
- int `CurT`  
*Температура процессора в десятых долях градусов Цельсия.*
- unsigned int `Flags`  
*Флаги состояния.*
- unsigned int `GPIOFlags`  
*Флаги состояния GPIO входов.*
- unsigned int `CmdBufFreeSpace`  
*Данное поле служебное.*

#### 6.55.1 Подробное описание

Состояние устройства с использованием пользовательских единиц.

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

См. также

`get_status_impl`

#### 6.55.2 Поля

##### 6.55.2.1 unsigned int `CmdBufFreeSpace`

Данное поле служебное.

Оно показывает количество свободных ячеек буфера цепочки синхронизации.

##### 6.55.2.2 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. Корректируется таблицей.

##### 6.55.2.3 float `CurSpeed`

Текущая скорость.

##### 6.55.2.4 int `CurT`

Температура процессора в десятых долях градусов Цельсия.

6.55.2.5 `long_t` `EncPosition`

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

6.55.2.6 `unsigned int` `EncSts`

[Состояние энкодера.](#)

6.55.2.7 `unsigned int` `Flags`

[Флаги состояния.](#)

6.55.2.8 `unsigned int` `GPIOFlags`

[Флаги состояния GPIO входов.](#)

6.55.2.9 `int` `Ipwr`

Ток потребления силовой части, мА.

6.55.2.10 `int` `Iusb`

Ток потребления по USB, мА.

6.55.2.11 `unsigned int` `MoveSts`

[Флаги состояния движения.](#)

6.55.2.12 `unsigned int` `MvCmdSts`

[Состояние команды движения.](#)

6.55.2.13 `unsigned int` `PWRSts`

[Флаги состояния питания шагового мотора.](#)

6.55.2.14 `int` `Upwr`

Напряжение на силовой части, десятки мВ.

6.55.2.15 `int` `Uusb`

Напряжение на USB, десятки мВ.

6.55.2.16 `unsigned int` `WindSts`

[Состояние обмоток.](#)



## 6.56 Структура status\_t

Состояние устройства.

Поля данных

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

### 6.56.1 Подробное описание

Состояние устройства.

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

См. также

`get_status_impl`

### 6.56.2 Поля

#### 6.56.2.1 `unsigned int CmdBufFreeSpace`

Данное поле служебное.

Оно показывает количество свободных ячеек буфера цепочки синхронизации.

#### 6.56.2.2 `int CurPosition`

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

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

#### 6.56.2.3 `int CurSpeed`

Текущая скорость.

#### 6.56.2.4 `int CurT`

Температура процессора в десятых долях градусов Цельсия.

#### 6.56.2.5 `long_t EncPosition`

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

#### 6.56.2.6 `unsigned int EncSts`

Состояние энкодера.

#### 6.56.2.7 `unsigned int Flags`

Флаги состояния.

#### 6.56.2.8 `unsigned int GPIOFlags`

Флаги состояния GPIO входов.

6.56.2.9 int lpwr

Ток потребления силовой части, мА.

6.56.2.10 int lusb

Ток потребления по USB, мА.

6.56.2.11 unsigned int MoveSts

Флаги состояния движения.

6.56.2.12 unsigned int MvCmdSts

Состояние команды движения.

6.56.2.13 unsigned int PWRSts

Флаги состояния питания шагового мотора.

6.56.2.14 int uCurPosition

Дробная часть текущей позиции в микрошагах.

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

6.56.2.15 int uCurSpeed

Дробная часть текущей скорости в микрошагах.

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

6.56.2.16 int Upwr

Напряжение на силовой части, десятки мВ.

6.56.2.17 int Uusb

Напряжение на USB, десятки мВ.

6.56.2.18 unsigned int WindSts

Состояние обмоток.

## 6.57 Структура `sync_in_settings_calb_t`

Настройки входной синхронизации с использованием пользовательских единиц.

Поля данных

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

### 6.57.1 Подробное описание

Настройки входной синхронизации с использованием пользовательских единиц.

Эта структура содержит все настройки, определяющие поведение входа синхронизации.

См. также

[get\\_sync\\_in\\_settings\\_calb](#)  
[set\\_sync\\_in\\_settings\\_calb](#)  
[get\\_sync\\_in\\_settings](#), [set\\_sync\\_in\\_settings](#)

### 6.57.2 Поля

#### 6.57.2.1 unsigned int `ClutterTime`

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

#### 6.57.2.2 float `Position`

Желаемая позиция или смещение.

#### 6.57.2.3 float `Speed`

Заданная скорость.

#### 6.57.2.4 unsigned int `SyncInFlags`

[Флаги настроек синхронизации входа.](#)

## 6.58 Структура `sync_in_settings_t`

Настройки входной синхронизации.

Поля данных

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

### 6.58.1 Подробное описание

Настройки входной синхронизации.

Эта структура содержит все настройки, определяющие поведение входа синхронизации.

См. также

`get_sync_in_settings`  
`set_sync_in_settings`  
`get_sync_in_settings`, `set_sync_in_settings`

### 6.58.2 Поля

#### 6.58.2.1 unsigned int `ClutterTime`

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

#### 6.58.2.2 unsigned int `Speed`

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

Диапазон: 0..100000.

#### 6.58.2.3 unsigned int `SyncInFlags`

Флаги настроек синхронизации входа.

#### 6.58.2.4 int `uPosition`

Дробная часть позиции или смещения в микрошагах.

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

## 6.58.2.5 unsigned int uSpeed

Заданная скорость в микрошагах в секунду.

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

6.59 Структура `sync_out_settings_calb_t`

Настройки выходной синхронизации с использованием пользовательских единиц.

Поля данных

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

## 6.59.1 Подробное описание

Настройки выходной синхронизации с использованием пользовательских единиц.

Эта структура содержит все настройки, определяющие поведение выхода синхронизации.

См. также

```
get_sync_out_settings_calb
set_sync_out_settings_calb
get_sync_out_settings, set_sync_out_settings
```

## 6.59.2 Поля

## 6.59.2.1 float Accuracy

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

## 6.59.2.2 unsigned int SyncOutFlags

[Флаги настроек синхронизации выхода.](#)

## 6.59.2.3 unsigned int SyncOutPeriod

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

## 6.59.2.4 unsigned int SyncOutPulseSteps

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

6.60 Структура `sync_out_settings_t`

Настройки выходной синхронизации.

Поля данных

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

## 6.60.1 Подробное описание

Настройки выходной синхронизации.

Эта структура содержит все настройки, определяющие поведение выхода синхронизации.

См. также

[get\\_sync\\_out\\_settings](#)  
[set\\_sync\\_out\\_settings](#)  
[get\\_sync\\_out\\_settings](#), [set\\_sync\\_out\\_settings](#)

## 6.60.2 Поля

## 6.60.2.1 unsigned int Accuracy

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

## 6.60.2.2 unsigned int SyncOutFlags

[Флаги настроек синхронизации выхода.](#)

6.60.2.3 `unsigned int SyncOutPeriod`

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

6.60.2.4 `unsigned int SyncOutPulseSteps`

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

6.60.2.5 `unsigned int uAccuracy`

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

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

## 6.61 Структура `uart_settings_t`

Настройки UART.

Поля данных

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

### 6.61.1 Подробное описание

Настройки UART.

Эта структура содержит настройки UART.

См. также

`get_uart_settings`  
`set_uart_settings`  
`get_uart_settings, set_uart_settings`

### 6.61.2 Поля

6.61.2.1 `unsigned int UARTSetupFlags`

*Флаги настроек четности команды UART.*



## Глава 7

# Файлы

### 7.1 Файл ximc.h

Заголовочный файл для библиотеки libximc.

#### Структуры данных

- struct [calibration\\_t](#)  
*Структура калибровок*
- struct [device\\_network\\_information\\_t](#)  
*Структура данных с информацией о сетевом устройстве.*
- struct [feedback\\_settings\\_t](#)  
*Настройки обратной связи.*
- struct [home\\_settings\\_t](#)  
*Настройки калибровки позиции.*
- struct [home\\_settings\\_calb\\_t](#)  
*Настройки калибровки позиции с использованием пользовательских единиц.*
- struct [move\\_settings\\_t](#)  
*Настройки движения.*
- struct [move\\_settings\\_calb\\_t](#)  
*Настройки движения с использованием пользовательских единиц.*
- struct [engine\\_settings\\_t](#)  
*Ограничения и настройки движения, связанные с двигателем.*
- struct [engine\\_settings\\_calb\\_t](#)  
*Ограничения и настройки движения, связанные с двигателем, с использованием пользовательских единиц.*
- struct [entype\\_settings\\_t](#)  
*Настройки типа мотора и типа силового драйвера.*
- struct [power\\_settings\\_t](#)  
*Настройки питания шагового мотора.*
- struct [secure\\_settings\\_t](#)  
*Структура содержит параметры защиты: критические значения электрических характеристик, флаги управления алгоритмами защиты.*
- struct [edges\\_settings\\_t](#)  
*Настройки границ.*
- struct [edges\\_settings\\_calb\\_t](#)

- `struct pid_settings_t`  
*Настройки границ с использованием пользовательских единиц.*
- `struct sync_in_settings_t`  
*Настройки ПИД.*
- `struct sync_in_settings_calb_t`  
*Настройки входной синхронизации.*
- `struct sync_out_settings_t`  
*Настройки выходной синхронизации с использованием пользовательских единиц.*
- `struct sync_out_settings_calb_t`  
*Настройки выходной синхронизации с использованием пользовательских единиц.*
- `struct extio_settings_t`  
*Настройки EXTIO.*
- `struct brake_settings_t`  
*Настройки тормоза.*
- `struct control_settings_t`  
*Настройки управления.*
- `struct control_settings_calb_t`  
*Настройки управления с использованием пользовательских единиц.*
- `struct joystick_settings_t`  
*Настройки джойстика.*
- `struct ctp_settings_t`  
*Настройки контроля позиции(для шагового двигателя).*
- `struct uart_settings_t`  
*Настройки UART.*
- `struct network_settings_t`  
*Настройки сети.*
- `struct password_settings_t`  
*Пароль.*
- `struct calibration_settings_t`  
*Калибровочные коэффициенты.*
- `struct controller_name_t`  
*Пользовательское имя контроллера и флаги настройки.*
- `struct nonvolatile_memory_t`  
*Пользовательские данные для сохранения во FRAM.*
- `struct emf_settings_t`  
*Настройки EMF.*
- `struct engine_advansed_setup_t`  
*Настройки EAS.*
- `struct extended_settings_t`  
*Настройки EST.*
- `struct get_position_t`  
*Данные о позиции.*
- `struct get_position_calb_t`  
*Данные о позиции.*
- `struct set_position_t`  
*Данные о позиции.*
- `struct set_position_calb_t`  
*Данные о позиции с использованием пользовательских единиц.*

- struct `status_t`  
*Состояние устройства.*
- struct `status_calb_t`  
*Состояние устройства с использованием пользовательских единиц.*
- struct `measurements_t`  
*Структура содержит последовательность измеренных параметров движения оси – скоростей и ошибок по позиции.*
- struct `chart_data_t`  
*Дополнительное состояние устройства.*
- struct `device_information_t`  
*Информации о контроллере.*
- struct `serial_number_t`  
*Структура с серийным номером и версией железа.*
- struct `analog_data_t`  
*Аналоговые данные.*
- struct `debug_read_t`  
*Отладочные данные.*
- struct `debug_write_t`  
*Отладочные данные.*
- struct `stage_name_t`  
*Пользовательское имя подвижки.*
- struct `stage_information_t`  
*Информация о позиционере.*
- struct `stage_settings_t`  
*Настройки позиционера.*
- struct `motor_information_t`  
*Информация о двигателе.*
- struct `motor_settings_t`  
*Физический характеристики и ограничения мотора.*
- struct `encoder_information_t`  
*Информация об энкодере.*
- struct `encoder_settings_t`  
*Настройки энкодера.*
- struct `hallsensor_information_t`  
*Информация о 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.*
- struct `init_random_t`  
*Случайный ключ.*
- struct `globally_unique_identifier_t`  
*Глобальный уникальный идентификатор.*

## Макросы

- #define `XIMC_API`  
*Макрос импорта библиотеки.*
- #define `XIMC_CALLCONV`  
*Библиотека вызывающая условные макросы.*
- #define `XIMC_RETTYPE` void\*  
*Возвращаемый тип потока.*
- #define `device_undefined` -1  
*Макрос, означающий неопределенное устройство*

## Результаты выполнения команд

- #define `result_ok` 0  
*выполнено успешно*
- #define `result_error` -1  
*общая ошибка*
- #define `result_not_implemented` -2  
*функция не определена*
- #define `result_value_error` -3  
*ошибка записи значения*
- #define `result_nodvice` -4  
*устройство не подключено*

## Уровень логирования

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

## Флаги поиска устройств

Это битовая маска для побитовых операций.

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

## Флаги состояния движения

Это битовая маска для побитовых операций. Возвращаются командой `get_status`.

См. также

`get_status`  
`status_t::MoveSts, get_status_impl`

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

### Флаги настроек контроллера

Это битовая маска для побитовых операций.

См. также

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

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

### Флаги состояния питания шагового мотора

Это битовая маска для побитовых операций. Возвращаются командой `get_status`.

См. также

`get_status`  
`status_t::PWRSts, get_status_impl`

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

### Флаги состояния

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

См. также

`get_status`  
`status_t::Flags, get_status_impl`

- `#define STATE_CONTR 0x0000003F`  
*Флаги состояния контроллера.*
- `#define STATE_ERRC 0x00000001`  
*Недопустимая команда.*
- `#define STATE_ERRD 0x00000002`

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

### Флаги состояния GPIO входов

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

См. также

`get_status`  
`status_t::GPIOFlags, get_status_impl`

- `#define STATE_DIG_SIGNAL 0xFFFF`
- Флаги цифровых сигналов.
- `#define STATE_RIGHT_EDGE 0x0001`
- Достижение правой границы.
- `#define STATE_LEFT_EDGE 0x0002`
- Достижение левой границы.

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

### Состояние энкодера

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

#### Заметки

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

См. также

`get_status`  
`status_t::EncSts, get_status_impl`

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

### Состояние обмоток

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

См. также

`get_status`  
`status_t::WindSts, get_status_impl`

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

### Состояние команды движения

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

См. также

`get_status`  
`status_t::MvCmdSts, get_status_impl`

- #define `MVCMD_NAME_BITS` 0x3F  
*Битовая маска активной команды.*
- #define `MVCMD_UKNWN` 0x00  
*Неизвестная команда.*
- #define `MVCMD_MOVE` 0x01  
*Команда move.*
- #define `MVCMD_MOVR` 0x02  
*Команда movr.*
- #define `MVCMD_LEFT` 0x03  
*Команда left.*
- #define `MVCMD_RIGHT` 0x04  
*Команда 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 движения**

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

См. также

`set_move_settings`  
`get_move_settings`  
`move_settings_t::MoveFlags, get_move_settings, set_move_settings`

- `#define RPM_DIV_1000 0x01`  
 Флаг указывает на то что рабочая скорость указанная в команде задана в милли rpm.

**Флаги параметров мотора**

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

См. также

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

- `#define ENGINE_REVERSE 0x01`  
 Флаг реверса.
- `#define ENGINE_CURRENT_AS_RMS 0x02`  
 Флаг интерпретации значения тока.
- `#define ENGINE_MAX_SPEED 0x04`  
 Флаг максимальной скорости.
- `#define ENGINE_ANTIPLAY 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**

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

См. также

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

- `#define MICROSTEP_MODE_FULL 0x01`  
 Полношаговый режим.
- `#define MICROSTEP_MODE_FRAC_2 0x02`  
 Деление шага 1/2.
- `#define MICROSTEP_MODE_FRAC_4 0x03`  
 Деление шага 1/4.

- `#define MICROSTEP_MODE_FRAC_8 0x04`  
*Деление шага 1/8.*
- `#define MICROSTEP_MODE_FRAC_16 0x05`  
*Деление шага 1/16.*
- `#define MICROSTEP_MODE_FRAC_32 0x06`  
*Деление шага 1/32.*
- `#define MICROSTEP_MODE_FRAC_64 0x07`  
*Деление шага 1/64.*
- `#define MICROSTEP_MODE_FRAC_128 0x08`  
*Деление шага 1/128.*
- `#define MICROSTEP_MODE_FRAC_256 0x09`  
*Деление шага 1/256.*

### Флаги, определяющие тип мотора

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

См. также

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

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

### Флаги, определяющие тип силового драйвера

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

См. также

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

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

### Флаги параметров питания шагового мотора

Это битовая маска для побитовых операций. Возвращаются командой `get_power_settings`.

См. также

`get_power_settings`  
`set_power_settings`  
`power_settings_t::PowerFlags, get_power_settings, set_power_settings`

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

### Флаги критических параметров.

Это битовая маска для побитовых операций. Возвращаются командой `get_secure_settings`.

См. также

`get_secure_settings`  
`set_secure_settings`  
`secure_settings_t::Flags, get_secure_settings, set_secure_settings`

- `#define ALARM_ON_DRIVER_OVERHEATING 0x01`  
*Если флаг установлен, то войти в состояние Alarm при получении сигнала подступающего перегрева с драйвера.*
- `#define LOW_UPWR_PROTECTION 0x02`  
*Если 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 концевого выключателя.*
- `#define ALARM_FLAGS_STICKING 0x10`  
*Если флаг установлен, то только по команде STOP возможен сброс всех флагов ALARM.*
- `#define BRAKING_OVERVOLTAGE_PROTECTION 0x20`  
*Если флаг установлен, то микропрограмма контроллера будет замыкать нижние ключи H-моста, отсоединяя мотор от цепи питания, при перенапряжении.*
- `#define ALARM_WINDING_MISMATCH 0x40`  
*Если флаг установлен, то войти в состояние Alarm при получении сигнала рассогласования обмоток*
- `#define ALARM_ENGINE_RESPONSE 0x80`  
*Если флаг установлен, то войти в состояние Alarm при получении сигнала ошибки реакции двигателя на управляющее воздействие*

### Флаги установки положения

Это битовая маска для побитовых операций. Возвращаются командой `get_position`.

См. также

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

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

*Если установлен, то счётчик энкодера не обновляется.*

#### Тип обратной связи.

*Это битовая маска для побитовых операций.*

См. также

[set\\_feedback\\_settings](#)  
[get\\_feedback\\_settings](#)  
[feedback\\_settings\\_t::FeedbackType, get\\_feedback\\_settings, set\\_feedback\\_settings](#)

- #define **FEEDBACK\_ENCODER** 0x01  
*Обратная связь с помощью энкодера.*
- #define **FEEDBACK\_EMF** 0x04  
*Обратная связь по ЭДС.*
- #define **FEEDBACK\_NONE** 0x05  
*Обратная связь отсутствует.*
- #define **FEEDBACK\_ENCODER\_MEDIATED** 0x06  
*Обратная связь по энкодеру, опосредованному относительно двигателя механической передачей (например, винтовой передачей).*

#### Флаги обратной связи.

*Это битовая маска для побитовых операций.*

См. также

[set\\_feedback\\_settings](#)  
[get\\_feedback\\_settings](#)  
[feedback\\_settings\\_t::FeedbackFlags, get\\_feedback\\_settings, set\\_feedback\\_settings](#)

- #define **FEEDBACK\_ENC\_REVERSE** 0x01  
*Обратный счет у энкодера.*
- #define **FEEDBACK\_ENC\_ADAPTIVE\_HOLDING** 0x02  
*Включает алгоритм адаптивного удержания.*
- #define **FEEDBACK\_ENC\_TYPE\_BITS** 0xC0  
*Биты, отвечающие за тип энкодера.*
- #define **FEEDBACK\_ENC\_TYPE\_AUTO** 0x00  
*Определяет тип энкодера автоматически.*
- #define **FEEDBACK\_ENC\_TYPE\_SINGLE\_ENDED** 0x40  
*Недифференциальный энкодер.*
- #define **FEEDBACK\_ENC\_TYPE\_DIFFERENTIAL** 0x80  
*Дифференциальный энкодер.*

#### Флаги настроек синхронизации входа

*Это битовая маска для побитовых операций.*

См. также

[sync\\_in\\_settings\\_t::SyncInFlags, get\\_sync\\_in\\_settings, set\\_sync\\_in\\_settings](#)

- #define **SYNCIN\_ENABLED** 0x01  
*Включение необходимости импульса синхронизации для начала движения.*
- #define **SYNCIN\_INVERT** 0x02  
*Если установлен - срабатывает 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

*Это битовая маска для побитовых операций.*

См. также

[sync\\_out\\_settings\\_t::SyncOutFlags](#), [get\\_sync\\_out\\_settings](#), [set\\_sync\\_out\\_settings](#)

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

### Флаги настройки работы внешнего ввода/вывода

Это битовая маска для побитовых операций.

См. также

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

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

[extio\\_settings\\_t::EXTIOSetupFlags](#), [get\\_extio\\_settings](#), [set\\_extio\\_settings](#)

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

### Флаги настройки режимов внешнего ввода/вывода

Это битовая маска для побитовых операций.

См. также

[extio\\_settings\\_t::extio\\_mode\\_flags](#)

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

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

[extio\\_settings\\_t::EXTIOModeFlags](#), [get\\_extio\\_settings](#), [set\\_extio\\_settings](#)

- `#define EXTIO_SETUP_MODE_IN_BITS 0x0F`  
*Биты, отвечающие за поведение при переходе сигнала в активное состояние.*
- `#define EXTIO_SETUP_MODE_IN_NOP 0x00`  
*Ничего не делать.*
- `#define EXTIO_SETUP_MODE_IN_STOP 0x01`  
*По переднему фронту входного сигнала делается остановка двигателя (эквивалент команды STOP).*
- `#define EXTIO_SETUP_MODE_IN_PWOF 0x02`  
*Выполняет команду PWOF, обесточивая обмотки двигателя.*
- `#define EXTIO_SETUP_MODE_IN_MOVR 0x03`  
*Выполняется команда MOVR с последними настройками.*
- `#define EXTIO_SETUP_MODE_IN_HOME 0x04`  
*Выполняется команда HOME.*

- `#define EXTIO_SETUP_MODE_IN_ALARM 0x05`  
*Войти в состояние ALARM при переходе сигнала в активное состояние.*
- `#define EXTIO_SETUP_MODE_OUT_BITS 0xF0`  
*Биты выбора поведения на выходе.*
- `#define EXTIO_SETUP_MODE_OUT_OFF 0x00`  
*Ножка всегда в неактивном состоянии.*
- `#define EXTIO_SETUP_MODE_OUT_ON 0x10`  
*Ножка всегда в активном состоянии.*
- `#define EXTIO_SETUP_MODE_OUT_MOVING 0x20`  
*Ножка находится в активном состоянии при движении.*
- `#define EXTIO_SETUP_MODE_OUT_ALARM 0x30`  
*Ножка находится в активном состоянии при нахождении в состоянии ALARM.*
- `#define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40`  
*Ножка находится в активном состоянии при подаче питания на обмотки.*

### Флаги границ

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

См. также

[get\\_edges\\_settings](#)  
[set\\_edges\\_settings](#)  
[edges\\_settings\\_t::BorderFlags, get\\_edges\\_settings, set\\_edges\\_settings](#)

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

### Флаги конечных выключателей

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

См. также

[get\\_edges\\_settings](#)  
[set\\_edges\\_settings](#)  
[edges\\_settings\\_t::EnderFlags, get\\_edges\\_settings, set\\_edges\\_settings](#)

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

### Флаги настроек тормоза

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

См. также

[get\\_brake\\_settings](#)  
[set\\_brake\\_settings](#)  
[brake\\_settings\\_t::BrakeFlags, get\\_brake\\_settings, set\\_brake\\_settings](#)

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

### Флаги управления

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

См. также

[get\\_control\\_settings](#)  
[set\\_control\\_settings](#)  
[control\\_settings\\_t::Flags, get\\_control\\_settings, set\\_control\\_settings](#)

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

### Флаги джойстика

Это битовая маска для побитовых операций. Управляют состояниями джойстика.

См. также

[set\\_joystick\\_settings](#)  
[get\\_joystick\\_settings](#)  
[joystick\\_settings\\_t::JoyFlags, get\\_joystick\\_settings, set\\_joystick\\_settings](#)

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

### Флаги контроля позиции

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

См. также

[get\\_ctp\\_settings](#)  
[set\\_ctp\\_settings](#)  
[ctp\\_settings\\_t::CTPFlags, get\\_ctp\\_settings, set\\_ctp\\_settings](#)

- `#define CTP_ENABLED 0x01`  
*Контроль позиции включен, если флаг установлен.*
- `#define CTP_BASE 0x02`

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

- `#define CTP_ALARM_ON_ERROR 0x04`  
Войти в состояние *ALARM* при расхождении позиции, если флаг установлен.
- `#define REV_SENS_INV 0x08`  
Сенсор считается активным, когда на нём 0, инвертирование делает активным уровень 1.
- `#define CTP_ERROR_CORRECTION 0x10`  
Корректировать ошибки, возникающие при проскальзывании, если флаг установлен.

### Флаги настроек команды `home`

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

См. также

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

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

### Флаги настроек четности команды `UART`

Это битовая маска для побитовых операций.



См. также

[`uart\_settings\_t::UARTSetupFlags`](#), [`get\_uart\_settings`](#), [`set\_uart\_settings`](#)

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

### Флаги типа двигателя

Это битовая маска для побитовых операций.

См. также

[`motor\_settings\_t::MotorType`](#), [`get\_motor\_settings`](#), [`set\_motor\_settings`](#)

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

### Флаги настроек энкодера

Это битовая маска для побитовых операций.

См. также

[`accessories\_settings\_t::MBSettings`](#), [`get\_accessories\_settings`](#), [`set\_accessories\_settings`](#)

- `#define ENCSET_DIFFERENTIAL_OUTPUT 0x001`  
*Если флаг установлен, то энкодер имеет дифференциальный выход, иначе - несимметричный выход*
- `#define ENCSET_PUSHPULL_OUTPUT 0x004`  
*Если флаг установлен, то энкодер имеет двухтактный выход, иначе - выход с открытым коллектором*
- `#define ENCSET_INDEXCHANNEL_PRESENT 0x010`  
*Если флаг установлен, то энкодер имеет дополнительный индексный канал, иначе - он отсутствует*
- `#define ENCSET_REVOLUTIONSENSOR_PRESENT 0x040`  
*Если флаг установлен, то энкодер имеет датчик оборотов, иначе - он отсутствует*
- `#define ENCSET_REVOLUTIONSENSOR_ACTIVE_HIGH 0x100`  
*Если флаг установлен, то активное состояние датчика оборотов соответствует логической 1, иначе - логическому 0.*
- `#define MB_AVAILABLE 0x01`  
*Если флаг установлен, то магнитный тормоз доступен*

- `#define MB_POWERED_HOLD 0x02`

*Если флаг установлен, то магнитный тормоз находится в режиме удержания (активен) при подаче питания*

### Флаги настроек температурного датчика

Это битовая маска для побитовых операций.

См. также

`accessories_settings_t::LimitSwitchesSettings, get_accessories_settings, set_accessories_settings`

- `#define TS_TYPE_BITS 0x07`

*Биты, отвечающие за тип температурного датчика.*

- `#define TS_TYPE_UNKNOWN 0x00`

*Неизвестный сенсор*

- `#define TS_TYPE_THERMOCOUPLE 0x01`

*Термопара*

- `#define TS_TYPE_SEMICONDUCTOR 0x02`

*Полупроводниковый температурный датчик*

- `#define TS_AVAILABLE 0x08`

*Если флаг установлен, то датчик температуры доступен*

- `#define LS_ON_SW1_AVAILABLE 0x01`

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

- `#define LS_ON_SW2_AVAILABLE 0x02`

*Если флаг установлен, то концевой переключатель, подключенный к ножке SW2, доступен*

- `#define LS_SW1_ACTIVE_LOW 0x04`

*Если флаг установлен, то концевой переключатель, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте*

- `#define LS_SW2_ACTIVE_LOW 0x08`

*Если флаг установлен, то концевой переключатель, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте*

- `#define LS_SHORTED 0x10`

*Если флаг установлен, то концевые переключатели замкнуты.*

### Флаги автоопределения характеристик обмоток двигателя.

Это битовая маска для побитовых операций.

См. также

`set_emf_settings, get_emf_settings, emf_settings_t::BackEMFFlags, get_emf_settings, set_emf_settings`

- `#define BACK_EMF_INDUCTANCE_AUTO 0x01`

*Флаг автоопределения индуктивности обмоток двигателя.*

- `#define BACK_EMF_RESISTANCE_AUTO 0x02`

*Флаг автоопределения сопротивления обмоток двигателя.*

- `#define BACK_EMF_KM_AUTO 0x04`

*Флаг автоопределения электромеханического коэффициента двигателя.*

### Определения типов

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

Тип идентификатора устройства

- typedef int `result_t`

Тип, определяющий результат выполнения команды.

- typedef uint32\_t `device_enumeration_t`

Тип, определяющий структуру данных о всех контроллерах, обнаруженных при опросе устройств.

- typedef struct `calibration_t` `calibration_t`

Структура калибровок

- typedef struct

`device_network_information_t` `device_network_information_t`

Структура данных с информацией о сетевом устройстве.

## Функции

### Группа команд настройки контроллера

Функции для чтения/записи большинства настроек контроллера.

- `result_t XIMC_API set_feedback_settings(device_t id, const feedback_settings_t *feedback_settings)`  
Запись настроек обратной связи.
- `result_t XIMC_API get_feedback_settings(device_t id, feedback_settings_t *feedback_settings)`  
Чтение настроек обратной связи
- `result_t XIMC_API set_home_settings(device_t id, const home_settings_t *home_settings)`  
Команда записи настроек для подхода в home position.
- `result_t XIMC_API set_home_settings_calb(device_t id, const home_settings_calb_t *home_settings_calb, const calibration_t *calibration)`  
Команда записи настроек для подхода в home position с использованием пользовательских единиц.
- `result_t XIMC_API get_home_settings(device_t id, home_settings_t *home_settings)`  
Команда чтения настроек для подхода в home position.
- `result_t XIMC_API get_home_settings_calb(device_t id, home_settings_calb_t *home_settings_calb, const calibration_t *calibration)`  
Команда чтения настроек для подхода в home position с использованием пользовательских единиц.
- `result_t XIMC_API set_move_settings(device_t id, const move_settings_t *move_settings)`  
Команда записи настроек перемещения (скорость, ускорение, 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)`  
Команда записи настроек перемещения, с использованием пользовательских единиц (скорость, ускорение, threshold и скорость в режиме антилюфта).
- `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)`  
Команда чтения настроек перемещения с использованием пользовательских единиц (скорость, ускорение, threshold и скорость в режиме антилюфта).
- `result_t XIMC_API set_engine_settings(device_t id, const engine_settings_t *engine_settings)`  
Запись настроек мотора.
- `result_t XIMC_API set_engine_settings_calb(device_t id, const engine_settings_calb_t *engine_settings_calb, const calibration_t *calibration)`  
Запись настроек мотора с использованием пользовательских единиц.
- `result_t XIMC_API get_engine_settings(device_t id, engine_settings_t *engine_settings)`  
Чтение настроек мотора.

- `result_t XIMC_API get_engine_settings_calb (device_t id, engine_settings_calb_t *engine_settings_calb, const calibration_t *calibration)`  
Чтение настроек мотора с использованием пользовательских единиц.
- `result_t XIMC_API set_entype_settings (device_t id, const entype_settings_t *entype_settings)`  
Запись информации о типе мотора и типе силового драйвера.
- `result_t XIMC_API get_entype_settings (device_t id, entype_settings_t *entype_settings)`  
Возвращает информацию о типе мотора и силового драйвера.
- `result_t XIMC_API set_power_settings (device_t id, const power_settings_t *power_settings)`  
Команда записи параметров питания мотора.
- `result_t XIMC_API get_power_settings (device_t id, power_settings_t *power_settings)`  
Команда чтения параметров питания мотора.
- `result_t XIMC_API set_secure_settings (device_t id, const secure_settings_t *secure_settings)`  
Команда записи установок защит.
- `result_t XIMC_API get_secure_settings (device_t id, secure_settings_t *secure_settings)`  
Команда записи установок защит.
- `result_t XIMC_API set_edges_settings (device_t id, const edges_settings_t *edges_settings)`  
Запись настроек границ и концевых выключателей.
- `result_t XIMC_API set_edges_settings_calb (device_t id, const edges_settings_calb_t *edges_settings_calb, const calibration_t *calibration)`  
Запись настроек границ и концевых выключателей с использованием пользовательских единиц.
- `result_t XIMC_API get_edges_settings (device_t id, edges_settings_t *edges_settings)`  
Чтение настроек границ и концевых выключателей.
- `result_t XIMC_API get_edges_settings_calb (device_t id, edges_settings_calb_t *edges_settings_calb, const calibration_t *calibration)`  
Чтение настроек границ и концевых выключателей с использованием пользовательских единиц.
- `result_t XIMC_API set_pid_settings (device_t id, const pid_settings_t *pid_settings)`  
Запись ПИД коэффициентов.
- `result_t XIMC_API get_pid_settings (device_t id, pid_settings_t *pid_settings)`  
Чтение ПИД коэффициентов.
- `result_t XIMC_API set_sync_in_settings (device_t id, const sync_in_settings_t *sync_in_settings)`  
Запись настроек для входного импульса синхронизации.
- `result_t XIMC_API set_sync_in_settings_calb (device_t id, const sync_in_settings_calb_t *sync_in_settings_calb, const calibration_t *calibration)`  
Запись настроек для входного импульса синхронизации с использованием пользовательских единиц.
- `result_t XIMC_API get_sync_in_settings (device_t id, sync_in_settings_t *sync_in_settings)`  
Чтение настроек для входного импульса синхронизации.
- `result_t XIMC_API get_sync_in_settings_calb (device_t id, sync_in_settings_calb_t *sync_in_settings_calb, const calibration_t *calibration)`  
Чтение настроек для входного импульса синхронизации с использованием пользовательских единиц.
- `result_t XIMC_API set_sync_out_settings (device_t id, const sync_out_settings_t *sync_out_settings)`  
Запись настроек для выходного импульса синхронизации.
- `result_t XIMC_API set_sync_out_settings_calb (device_t id, const sync_out_settings_calb_t *sync_out_settings_calb, const calibration_t *calibration)`  
Запись настроек для выходного импульса синхронизации с использованием пользовательских единиц.
- `result_t XIMC_API get_sync_out_settings (device_t id, sync_out_settings_t *sync_out_settings)`  
Чтение настроек для выходного импульса синхронизации.

- `result_t XIMC_API get_sync_out_settings_calb (device_t id, sync_out_settings_calb_t *sync_out_settings_calb, const calibration_t *calibration)`  
Чтение настроек для выходного импульса синхронизации с использованием пользовательских единиц.
- `result_t XIMC_API set_extio_settings (device_t id, const extio_settings_t *extio_settings)`  
Команда записи параметров настройки режимов внешнего ввода/вывода.
- `result_t XIMC_API get_extio_settings (device_t id, extio_settings_t *extio_settings)`  
Команда чтения параметров настройки режимов внешнего ввода/вывода.
- `result_t XIMC_API set_brake_settings (device_t id, const brake_settings_t *brake_settings)`  
Запись настроек управления тормозом.
- `result_t XIMC_API get_brake_settings (device_t id, brake_settings_t *brake_settings)`  
Чтение настроек управления тормозом.
- `result_t XIMC_API set_control_settings (device_t id, const control_settings_t *control_settings)`  
Запись настроек управления мотором.
- `result_t XIMC_API set_control_settings_calb (device_t id, const control_settings_calb_t *control_settings_calb, const calibration_t *calibration)`  
Запись настроек управления мотором с использованием пользовательских единиц.
- `result_t XIMC_API get_control_settings (device_t id, control_settings_t *control_settings)`  
Чтение настроек управления мотором.
- `result_t XIMC_API get_control_settings_calb (device_t id, control_settings_calb_t *control_settings_calb, const calibration_t *calibration)`  
Чтение настроек управления мотором с использованием пользовательских единиц.
- `result_t XIMC_API set_joystick_settings (device_t id, const joystick_settings_t *joystick_settings)`  
Запись настроек джойстика.
- `result_t XIMC_API get_joystick_settings (device_t id, joystick_settings_t *joystick_settings)`  
Чтение настроек джойстика.
- `result_t XIMC_API set_ctp_settings (device_t id, const ctp_settings_t *ctp_settings)`  
Запись настроек контроля позиции(для шагового двигателя).
- `result_t XIMC_API get_ctp_settings (device_t id, ctp_settings_t *ctp_settings)`  
Чтение настроек контроля позиции(для шагового двигателя).
- `result_t XIMC_API set_uart_settings (device_t id, const uart_settings_t *uart_settings)`  
Команда записи настроек UART.
- `result_t XIMC_API get_uart_settings (device_t id, uart_settings_t *uart_settings)`  
Команда чтения настроек UART.
- `result_t XIMC_API set_network_settings (device_t id, const network_settings_t *network_settings)`  
Команда записи сетевых настроек.
- `result_t XIMC_API get_network_settings (device_t id, network_settings_t *network_settings)`  
Команда чтения сетевых настроек.
- `result_t XIMC_API set_password_settings (device_t id, const password_settings_t *password_settings)`  
Команда записи пароля к веб-странице.
- `result_t XIMC_API get_password_settings (device_t id, password_settings_t *password_settings)`  
Команда чтения пароля к веб-странице.
- `result_t XIMC_API set_calibration_settings (device_t id, const calibration_settings_t *calibration_settings)`  
Команда записи калибровочных коэффициентов.
- `result_t XIMC_API get_calibration_settings (device_t id, calibration_settings_t *calibration_settings)`  
Команда чтения калибровочных коэффициентов.
- `result_t XIMC_API set_controller_name (device_t id, const controller_name_t *controller_name)`

- Запись пользовательского имени контроллера и настроек в FRAM.*

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

*Чтение пользовательского имени контроллера и настроек из FRAM.*

  - `result_t XIMC_API set_nonvolatile_memory (device_t id, const nonvolatile_memory_t *nonvolatile_memory)`

*Запись пользовательских данных во FRAM.*

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

*Чтение пользовательских данных из FRAM.*

  - `result_t XIMC_API set_emf_settings (device_t id, const emf_settings_t *emf_settings)`

*Запись электромеханических настроек шагового двигателя.*

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

*Чтение электромеханических настроек шагового двигателя.*

  - `result_t XIMC_API set_engine_advansed_setup (device_t id, const engine_advansed_setup_t *engine_advansed_setup)`

*Запись расширенных настроек.*

  - `result_t XIMC_API get_engine_advansed_setup (device_t id, engine_advansed_setup_t *engine_advansed_setup)`

*Чтение расширенных настроек.*

  - `result_t XIMC_API set_extended_settings (device_t id, const extended_settings_t *extended_settings)`

*Запись расширенных настроек.*

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

*Чтение расширенных настроек.*

#### Группа команд управления движением

- `result_t XIMC_API command_stop (device_t id)`

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

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

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

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

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

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

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

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

*При получении команды "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 Antiplay, заданное в настройках мотора (engine\_settings), затем двигается в ту же точку.*

- `result_t XIMC_API command_sstp (device_t id)`  
*Плавная остановка.*
- `result_t XIMC_API get_position (device_t id, get_position_t *the_get_position)`  
*Считывает значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.*
- `result_t XIMC_API get_position_calb (device_t id, get_position_calb_t *the_get_position_calb, const calibration_t *calibration)`  
*Считывает значение положения в пользовательских единицах для шагового двигателя и в шагах энкодера всех двигателей.*
- `result_t XIMC_API set_position (device_t id, const set_position_t *the_set_position)`  
*Устанавливает произвольное значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера для всех двигателей.*
- `result_t XIMC_API set_position_calb (device_t id, const set_position_calb_t *the_set_position_calb, const calibration_t *calibration)`  
*Устанавливает произвольное значение положения и значение энкодера всех двигателей с использованием пользовательских единиц.*
- `result_t XIMC_API command_zero (device_t id)`  
*Устанавливает текущую позицию равной 0.*

### Группа команд сохранения и загрузки настроек

- `result_t XIMC_API command_save_settings (device_t id)`  
*При получении команды контроллер выполняет операцию сохранения текущих настроек во встроенную энергонезависимую память контроллера.*
- `result_t XIMC_API command_read_settings (device_t id)`  
*Чтение всех настроек контроллера из flash памяти в оперативную, заменяя текущие настройки.*
- `result_t XIMC_API command_save_robust_settings (device_t id)`  
*При получении команды контроллер выполняет операцию сохранения важных настроек (калибровочные коэффициенты и т.*
- `result_t XIMC_API command_read_robust_settings (device_t id)`  
*Чтение важных настроек (калибровочные коэффициенты и т.*
- `result_t XIMC_API command_eesave_settings (device_t id)`  
*Запись настроек контроллера в EEPROM память позиционера. Функция должна использоваться только производителем.*
- `result_t XIMC_API command_eeread_settings (device_t id)`  
*Чтение настроек контроллера из EEPROM памяти позиционера.*
- `result_t XIMC_API command_start_measurements (device_t id)`  
*Начать измерения и буферизацию скорости, ошибки следования.*
- `result_t XIMC_API get_measurements (device_t id, measurements_t *measurements)`  
*Команда чтения буфера данных для построения графиков скорости и ошибки следования.*
- `result_t XIMC_API get_chart_data (device_t id, chart_data_t *chart_data)`  
*Команда чтения состояния обмоток и других 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 и версии железа во flash память контроллера.*
- `result_t XIMC_API get_analog_data (device_t id, analog_data_t *analog_data)`  
*Чтение аналоговых данных, содержащих данные с АЦП и нормированные значения величин.*
- `result_t XIMC_API get_debug_read (device_t id, debug_read_t *debug_read)`

- Чтение данных из прошивки для отладки и поиска неисправностей.  
`result_t XIMC_API set_debug_write (device_t id, const debug_write_t *debug_write)`  
 Запись данных в прошивку для отладки и поиска неисправностей.

### Группа команд работы с EEPROM подвижки

- `result_t XIMC_API set_stage_name (device_t id, const stage_name_t *stage_name)`  
 Запись пользовательского имени подвижки в EEPROM.
- `result_t XIMC_API get_stage_name (device_t id, stage_name_t *stage_name)`  
 Чтение пользовательского имени подвижки из EEPROM.
- `result_t XIMC_API set_stage_information (device_t id, const stage_information_t *stage_information)`  
 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 о датчиках Холла в EEPROM.
- `result_t XIMC_API get_hallsensor_information (device_t id, hallsensor_information_t *hallsensor_information)`  
 Чтение информации о 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 загрузчика контроллера.*
- `result_t XIMC_API get_init_random (device_t id, init_random_t *init_random)`
- Чтение случайного числа из контроллера.*
- `result_t XIMC_API get_globally_unique_identifier (device_t id, globally_unique_identifier_t *globally_unique_identifier)`
- Считывает уникальный идентификатор каждого чипа, это значение не является случайным.*
- `result_t XIMC_API goto_firmware (device_t id, uint8_t *ret)`
- Перезагрузка в прошивку в контроллере*
- `result_t XIMC_API has_firmware (const char *uri, uint8_t *ret)`
- Проверка наличия прошивки в контроллере*
- `result_t XIMC_API command_update_firmware (const char *uri, const uint8_t *data, uint32_t data_size)`
- Обновление прошивки.*
- `result_t XIMC_API write_key (const char *uri, uint8_t *key)`
- Запись ключа защиты. Функция используется только производителем.*
- `result_t XIMC_API command_reset (device_t id)`
- Перезагрузка контроллера.*
- `result_t XIMC_API command_clear_fram (device_t id)`
- Очистка FRAM памяти контроллера.*

## Управление устройством

### Функции поиска и открытия/закрытия устройств

- `typedef char * pchar`
- Не обращайтесь на меня внимание*
- `typedef void (XIMC_CALLCONV * logging_callback_t)(int loglevel, const wchar_t *message, void *user_data)`
- Прототип функции обратного вызова для логирования*
- `device_t XIMC_API open_device (const char *uri)`
- Открывает устройство по имени uri и возвращает идентификатор, который будет использоваться для обращения к устройству.*
- `result_t XIMC_API close_device (device_t *id)`
- Закрывает устройство*
- `result_t XIMC_API load_correction_table (device_t *id, const char *namefile)`
- Команда загрузки корректирующей таблицы из текстового файла (данная функция устарела).*
- `result_t XIMC_API set_correction_table (device_t id, const char *namefile)`
- Команда загрузки корректирующей таблицы из текстового файла.*

- `result_t XIMC_API probe_device` (const char \*uri)  
*Проверяет, является ли устройство с уникальным идентификатором uri XIMC-совместимым.*
- `result_t XIMC_API set_bindy_key` (const char \*keyfilepath)  
*Устарело.*
- `device_enumeration_t XIMC_API enumerate_devices` (int enumerate\_flags, const char \*hints)  
*Поиск и составление списка доступных устройств.*
- `result_t XIMC_API free_enumerate_devices` (device\_enumeration\_t device\_enumeration)  
*Освобождает память, выделенную enumerate\_devices.*
- `int XIMC_API get_device_count` (device\_enumeration\_t device\_enumeration)  
*Возвращает количество подключенных устройств.*
- `pchar XIMC_API get_device_name` (device\_enumeration\_t device\_enumeration, int device\_index)  
*Возвращает имя подключенного устройства из перечисления устройств.*
- `result_t XIMC_API get_enumerate_device_serial` (device\_enumeration\_t device\_enumeration, int device\_index, uint32\_t \*serial)  
*Возвращает серийный номер подключенного устройства из перечисления устройств.*
- `result_t XIMC_API get_enumerate_device_information` (device\_enumeration\_t device\_enumeration, int device\_index, device\_information\_t \*device\_information)  
*Возвращает информацию о подключенном устройстве из перечисления устройств.*
- `result_t XIMC_API get_enumerate_device_controller_name` (device\_enumeration\_t device\_enumeration, int device\_index, controller\_name\_t \*controller\_name)  
*Возвращает имя подключенного устройства из перечисления устройств.*
- `result_t XIMC_API get_enumerate_device_stage_name` (device\_enumeration\_t device\_enumeration, int device\_index, stage\_name\_t \*stage\_name)  
*Возвращает имя подвижки для подключенного устройства из перечисления устройств.*
- `result_t XIMC_API get_enumerate_device_network_information` (device\_enumeration\_t device\_enumeration, int device\_index, device\_network\_information\_t \*device\_network\_information)  
*Возвращает сетевую информацию о подключенном устройстве из перечисления устройств.*
- `result_t XIMC_API reset_locks` ()  
*Сбрасывает ошибку неправильной передачи данных.*
- `result_t XIMC_API ximc_fix_usbser_sys` (const char \*device\_uri)  
*(Устарело) 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, void \*user\_data)  
*Простая функция логирования на stderr в широких символах*
- `void XIMC_API logging_callback_stderr_narrow` (int loglevel, const wchar\_t \*message, void \*user\_data)  
*Простая функция логирования на stderr в узких (однобайтных) символах*
- `void XIMC_API set_logging_callback` (logging\_callback\_t logging\_callback, void \*user\_data)  
*Устанавливает функцию обратного вызова для логирования.*
- `result_t XIMC_API get_status` (device\_t id, status\_t \*status)  
*Возвращает информацию о текущем состоянии устройства.*
- `result_t XIMC_API get_status_calb` (device\_t id, status\_calb\_t \*status, const calibration\_t \*calibration)  
*Возвращает информацию о текущем состоянии устройства.*
- `result_t XIMC_API get_device_information` (device\_t id, device\_information\_t \*device\_information)

*Возвращает информацию об устройстве.*

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

*Ожидание остановки контроллера*

- `result_t XIMC_API command_homezero (device_t id)`

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

### 7.1.1 Подробное описание

Заголовочный файл для библиотеки `libximc`.

### 7.1.2 Макросы

#### 7.1.2.1 `#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.

#### 7.1.2.2 `#define BACK_EMF_INDUCTANCE_AUTO 0x01`

Флаг автоопределения индуктивности обмоток двигателя.

#### 7.1.2.3 `#define BACK_EMF_KM_AUTO 0x04`

Флаг автоопределения электромеханического коэффициента двигателя.

#### 7.1.2.4 `#define BACK_EMF_RESISTANCE_AUTO 0x02`

Флаг автоопределения сопротивления обмоток двигателя.

#### 7.1.2.5 `#define BORDER_IS_ENCODER 0x01`

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

Если флаг сброшен, границы определяются концевыми выключателями.

#### 7.1.2.6 `#define BORDER_STOP_LEFT 0x02`

Если флаг установлен, мотор останавливается при достижении левой границы.

#### 7.1.2.7 `#define BORDER_STOP_RIGHT 0x04`

Если флаг установлен, мотор останавливается при достижении правой границы.

7.1.2.8 `#define BORDERS_SWAP_MISSET_DETECTION 0x08`

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

Нужен для предотвращения поломки двигателя при неправильных настройках концевых выключателей

7.1.2.9 `#define BRAKE_ENABLED 0x01`

Управление тормозом включено, если флаг установлен.

7.1.2.10 `#define BRAKE_ENG_PWROFF 0x02`

Тормоз отключает питание шагового мотора, если флаг установлен.

7.1.2.11 `#define BRAKING_OVERVOLTAGE_PROTECTION 0x20`

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

7.1.2.12 `#define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04`

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

7.1.2.13 `#define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08`

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

7.1.2.14 `#define CONTROL_MODE_BITS 0x03`

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

7.1.2.15 `#define CONTROL_MODE_JOY 0x01`

Управление с помощью джойстика.

7.1.2.16 `#define CONTROL_MODE_LR 0x02`

Управление с помощью кнопок влево/вправо.

7.1.2.17 `#define CONTROL_MODE_OFF 0x00`

Управление отключено.

7.1.2.18 `#define CTP_ALARM_ON_ERROR 0x04`

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

7.1.2.19 `#define CTP_BASE 0x02`

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

7.1.2.20 `#define CTP_ENABLED 0x01`

Контроль позиции включен, если флаг установлен.

7.1.2.21 `#define CTP_ERROR_CORRECTION 0x10`

Корректировать ошибки, возникающие при проскальзывании, если флаг установлен.

Работает только с энкодером. Несовместимо с флагом `CTP_ALARM_ON_ERROR`.

7.1.2.22 `#define DRIVER_TYPE_DISCRETE_FET 0x01`

Силовой драйвер на дискретных мосфет-ключах.

Используется по умолчанию.

7.1.2.23 `#define DRIVER_TYPE_EXTERNAL 0x03`

Внешний силовой драйвер.

7.1.2.24 `#define DRIVER_TYPE_INTEGRATE 0x02`

Силовой драйвер с использованием ключей, интегрированных в микросхему.

7.1.2.25 `#define EEPROM_PRECEDENCE 0x01`

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

7.1.2.26 `#define ENC_STATE_ABSENT 0x00`

Энкодер не подключен.

7.1.2.27 `#define ENC_STATE_MALFUNC 0x02`

Энкодер подключен и неисправен.

7.1.2.28 `#define ENC_STATE_OK 0x04`

Энкодер подключен и работает должным образом.

7.1.2.29 `#define ENC_STATE_REVERS 0x03`

Энкодер подключен и исправен, но считает в другую сторону.

7.1.2.30 `#define ENC_STATE_UNKNOWN 0x01`

Состояние энкодера неизвестно.

7.1.2.31 `#define ENDER_SW1_ACTIVE_LOW 0x02`

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

7.1.2.32 `#define ENDER_SW2_ACTIVE_LOW 0x04`

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

7.1.2.33 `#define ENDER_SWAP 0x01`

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

7.1.2.34 `#define ENGINE_ACCEL_ON 0x10`

Ускорение.

Если флаг установлен, движение происходит с ускорением.

7.1.2.35 `#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.

7.1.2.36 `#define ENGINE_CURRENT_AS_RMS 0x02`

Флаг интерпретации значения тока.

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

7.1.2.37 `#define ENGINE_LIMIT_CURR 0x40`

Номинальный ток мотора.

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

7.1.2.38 `#define ENGINE_LIMIT_RPM 0x80`

Номинальная частота вращения мотора.

Если флаг установлен, частота вращения ограничивается заданным номинальным значением.

7.1.2.39 `#define ENGINE_LIMIT_VOLT 0x20`

Номинальное напряжение мотора.

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

7.1.2.40 `#define ENGINE_MAX_SPEED 0x04`

Флаг максимальной скорости.

Если флаг установлен, движение происходит на максимальной скорости.

7.1.2.41 `#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.

7.1.2.42 `#define ENGINE_TYPE_2DC 0x02`

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

7.1.2.43 `#define ENGINE_TYPE_BRUSHLESS 0x05`

Бесщеточный мотор.

7.1.2.44 `#define ENGINE_TYPE_DC 0x01`

Мотор постоянного тока.

7.1.2.45 `#define ENGINE_TYPE_NONE 0x00`

Это значение не нужно использовать.

7.1.2.46 `#define ENGINE_TYPE_STEP 0x03`

Шаговый мотор.

7.1.2.47 `#define ENGINE_TYPE_TEST 0x04`

Продолжительность включения фиксирована.

Используется только производителем.

7.1.2.48 `#define ENUMERATE_PROBE 0x01`

Проверять, является ли устройство XIMC-совместимым.

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

7.1.2.49 `#define EXTIO_SETUP_INVERT 0x02`

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

7.1.2.50 `#define EXTIO_SETUP_MODE_IN_ALARM 0x05`

Войти в состояние ALARM при переходе сигнала в активное состояние.

7.1.2.51 `#define EXTIO_SETUP_MODE_IN_BITS 0x0F`

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

7.1.2.52 `#define EXTIO_SETUP_MODE_IN_HOME 0x04`

Выполняется команда HOME.

7.1.2.53 `#define EXTIO_SETUP_MODE_IN_MOVR 0x03`

Выполняется команда MOVR с последними настройками.

7.1.2.54 `#define EXTIO_SETUP_MODE_IN_NOP 0x00`

Ничего не делать.

7.1.2.55 `#define EXTIO_SETUP_MODE_IN_PWOF 0x02`

Выполняет команду PWOF, обесточивая обмотки двигателя.

7.1.2.56 `#define EXTIO_SETUP_MODE_IN_STOP 0x01`

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

7.1.2.57 `#define EXTIO_SETUP_MODE_OUT_ALARM 0x30`

Ножка находится в активном состоянии при нахождении в состоянии ALARM.

7.1.2.58 `#define EXTIO_SETUP_MODE_OUT_BITS 0xF0`

Биты выбора поведения на выходе.



7.1.2.59 `#define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40`

Ножка находится в активном состоянии при подаче питания на обмотки.

7.1.2.60 `#define EXTIO_SETUP_MODE_OUT_MOVING 0x20`

Ножка находится в активном состоянии при движении.

7.1.2.61 `#define EXTIO_SETUP_MODE_OUT_OFF 0x00`

Ножка всегда в неактивном состоянии.

7.1.2.62 `#define EXTIO_SETUP_MODE_OUT_ON 0x10`

Ножка всегда в активном состоянии.

7.1.2.63 `#define EXTIO_SETUP_OUTPUT 0x01`

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

7.1.2.64 `#define FEEDBACK_EMF 0x04`

Обратная связь по ЭДС.

7.1.2.65 `#define FEEDBACK_ENC_ADAPTIVE_HOLDING 0x02`

Включает алгоритм адаптивного удержания.

7.1.2.66 `#define FEEDBACK_ENC_REVERSE 0x01`

Обратный счет у энкодера.

7.1.2.67 `#define FEEDBACK_ENC_TYPE_AUTO 0x00`

Определяет тип энкодера автоматически.

7.1.2.68 `#define FEEDBACK_ENC_TYPE_BITS 0xC0`

Биты, отвечающие за тип энкодера.

7.1.2.69 `#define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80`

Дифференциальный энкодер.

7.1.2.70 `#define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40`

Недифференциальный энкодер.

7.1.2.71 `#define FEEDBACK_ENCODER 0x01`

Обратная связь с помощью энкодера.

7.1.2.72 `#define FEEDBACK_ENCODER_MEDIATED 0x06`

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

7.1.2.73 `#define FEEDBACK_NONE 0x05`

Обратная связь отсутствует.

7.1.2.74 `#define HOME_DIR_FIRST 0x001`

Определяет направление первоначального движения мотора после поступления команды HOME. Если флаг установлен - вправо; иначе - влево.

7.1.2.75 `#define HOME_DIR_SECOND 0x002`

Определяет направление второго движения мотора. Если флаг установлен - вправо; иначе - влево.

7.1.2.76 `#define HOME_HALF_MV 0x008`

Если флаг установлен, в начале второго движения первые пол оборота сигналы завершения движения игнорируются.

7.1.2.77 `#define HOME_MV_SEC_EN 0x004`

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

7.1.2.78 `#define HOME_STOP_FIRST_BITS 0x030`

Биты, отвечающие за выбор сигнала завершения первого движения.

7.1.2.79 `#define HOME_STOP_FIRST_LIM 0x030`

Первое движение завершается по сигналу с концевого переключателя.

7.1.2.80 `#define HOME_STOP_FIRST_REV 0x010`

Первое движение завершается по сигналу с Revolution sensor.

7.1.2.81 `#define HOME_STOP_FIRST_SYN 0x020`

Первое движение завершается по сигналу со входа синхронизации.

7.1.2.82 `#define HOME_STOP_SECOND_BITS 0x0C0`

Биты, отвечающие за выбор сигнала завершения второго движения.

7.1.2.83 `#define HOME_STOP_SECOND_LIM 0x0C0`

Второе движение завершается по сигналу с концевого переключателя.

7.1.2.84 `#define HOME_STOP_SECOND_REV 0x040`

Второе движение завершается по сигналу с Revolution sensor.

7.1.2.85 `#define HOME_STOP_SECOND_SYN 0x080`

Второе движение завершается по сигналу со входа синхронизации.

7.1.2.86 `#define HOME_USE_FAST 0x100`

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

7.1.2.87 `#define JOY_REVERSE 0x01`

Реверс воздействия джойстика.

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

7.1.2.88 `#define LOW_UPWR_PROTECTION 0x02`

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

7.1.2.89 `#define LS_SHORTED 0x10`

Если флаг установлен, то концевые переключатели замкнуты.

7.1.2.90 `#define MICROSTEP_MODE_FRAC_128 0x08`

Деление шага 1/128.

7.1.2.91 `#define MICROSTEP_MODE_FRAC_16 0x05`

Деление шага 1/16.

7.1.2.92 `#define MICROSTEP_MODE_FRAC_2 0x02`

Деление шага 1/2.

7.1.2.93 `#define MICROSTEP_MODE_FRAC_256 0x09`

Деление шага 1/256.

7.1.2.94 `#define MICROSTEP_MODE_FRAC_32 0x06`

Деление шага 1/32.

7.1.2.95 `#define MICROSTEP_MODE_FRAC_4 0x03`

Деление шага 1/4.

7.1.2.96 `#define MICROSTEP_MODE_FRAC_64 0x07`

Деление шага 1/64.

7.1.2.97 `#define MICROSTEP_MODE_FRAC_8 0x04`

Деление шага 1/8.

7.1.2.98 `#define MICROSTEP_MODE_FULL 0x01`

Полношаговый режим.

7.1.2.99 `#define MOVE_STATE_ANTIPLAY 0x04`

Выполняется компенсация люфта, если флаг установлен.

7.1.2.100 `#define MOVE_STATE_MOVING 0x01`

Если флаг установлен, то контроллер пытается вращать двигателем.

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

7.1.2.101 `#define MOVE_STATE_TARGET_SPEED 0x02`

Флаг устанавливается при достижении заданной скорости.

7.1.2.102 `#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.

7.1.2.103 `#define MVCMD_HOME 0x06`

Команда home.

7.1.2.104 `#define MVCMD_LEFT 0x03`

Команда left.

7.1.2.105 `#define MVCMD_LOFT 0x07`

Команда `loft`.

7.1.2.106 `#define MVCMD_MOVE 0x01`

Команда `move`.

7.1.2.107 `#define MVCMD_MOVR 0x02`

Команда `movr`.

7.1.2.108 `#define MVCMD_NAME_BITS 0x3F`

Битовая маска активной команды.

7.1.2.109 `#define MVCMD_RIGHT 0x04`

Команда `rigt`.

7.1.2.110 `#define MVCMD_RUNNING 0x80`

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

7.1.2.111 `#define MVCMD_SSTP 0x08`

Команда плавной остановки(SSTP).

7.1.2.112 `#define MVCMD_STOP 0x05`

Команда `stop`.

7.1.2.113 `#define MVCMD_UKNWN 0x00`

Неизвестная команда.

7.1.2.114 `#define POWER_OFF_ENABLED 0x02`

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

Иначе - не снимать.

7.1.2.115 `#define POWER_REDUCT_ENABLED 0x01`

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

Иначе - не уменьшать.

7.1.2.116 `#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.

7.1.2.117 `#define PWR_STATE_MAX 0x05`

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

7.1.2.118 `#define PWR_STATE_NORM 0x03`

Обмотки запитаны номинальным током.

7.1.2.119 `#define PWR_STATE_OFF 0x01`

Обмотки мотора разомкнуты и не управляются драйвером.

7.1.2.120 `#define PWR_STATE_REDUCT 0x04`

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

7.1.2.121 `#define PWR_STATE_UNKNOWN 0x00`

Неизвестное состояние, которое не должно никогда реализовываться.

7.1.2.122 `#define REV_SENS_INV 0x08`

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

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

7.1.2.123 `#define RPM_DIV_1000 0x01`

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

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

7.1.2.124 `#define SETPOS_IGNORE_ENCODER 0x02`

Если установлен, то счётчик энкодера не обновляется.

7.1.2.125 `#define SETPOS_IGNORE_POSITION 0x01`

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

7.1.2.126 `#define STATE_ALARM 0x0000040`

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

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

7.1.2.127 `#define STATE_BORDERS_SWAP_MISSET 0x0008000`

Достижение неверной границы.

7.1.2.128 `#define STATE_BRAKE 0x0200`

Состояние вывода управления тормозом.

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

7.1.2.129 `#define STATE_BUTTON_LEFT 0x0008`

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

7.1.2.130 `#define STATE_BUTTON_RIGHT 0x0004`

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

7.1.2.131 `#define STATE_CONTR 0x000003F`

Флаги состояния контроллера.

7.1.2.132 `#define STATE_CONTROLLER_OVERHEAT 0x0000200`

Перегрелась микросхема контроллера.

7.1.2.133 `#define STATE_CTP_ERROR 0x0000080`

Контроль позиции нарушен(используется только с шаговым двигателем).

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

7.1.2.134 `#define STATE_DIG_SIGNAL 0xFFFF`

Флаги цифровых сигналов.

7.1.2.135 `#define STATE_EEPROM_CONNECTED 0x0000010`

Подключена память EEPROM с настройками.

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

7.1.2.136 `#define STATE_ENC_A 0x2000`

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

7.1.2.137 `#define STATE_ENC_B 0x4000`

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

7.1.2.138 `#define STATE_ENGINE_RESPONSE_ERROR 0x0800000`

Ошибка реакции двигателя на управляющее воздействие.

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

7.1.2.139 `#define STATE_ERRC 0x0000001`

Недопустимая команда.

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

7.1.2.140 `#define STATE_ERRD 0x0000002`

Обнаружена ошибка целостности данных.

Данные внутри команды и ее CRC-код не соответствуют, поэтому данные не могут считаться действительными. Эта ошибка может быть вызвана электромагнитными помехами в интерфейсе UART/RS232.

7.1.2.141 `#define STATE_ERRV 0x0000004`

Недопустимое значение данных.

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

7.1.2.142 `#define STATE_EXTIO_ALARM 0x1000000`

Ошибка вызвана внешним входным сигналом EXTIO.

7.1.2.143 `#define STATE_GPIO_LEVEL 0x0020`

Состояние ввода/вывода общего назначения.

7.1.2.144 `#define STATE_GPIO_PINOUT 0x0010`

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



7.1.2.145 `#define STATE_IS_HOMED 0x0000020`

Калибровка выполнена.

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

7.1.2.146 `#define STATE_LEFT_EDGE 0x0002`

Достижение левой границы.

7.1.2.147 `#define STATE_LOW_USB_VOLTAGE 0x0002000`

Устарело.

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

7.1.2.148 `#define STATE_OVERLOAD_POWER_CURRENT 0x0000800`

Превышен максимальный ток потребления силовой части.

7.1.2.149 `#define STATE_OVERLOAD_POWER_VOLTAGE 0x0000400`

Превышено напряжение на силовой части.

7.1.2.150 `#define STATE_OVERLOAD_USB_CURRENT 0x0004000`

Устарело.

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

7.1.2.151 `#define STATE_OVERLOAD_USB_VOLTAGE 0x0001000`

Устарело.

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

7.1.2.152 `#define STATE_POWER_OVERHEAT 0x0000100`

Перегрев силового драйвера.

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

7.1.2.153 `#define STATE_REV_SENSOR 0x0400`

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

7.1.2.154 `#define STATE_RIGHT_EDGE 0x0001`

Достижение правой границы.

7.1.2.155 `#define STATE_SECUR 0x1B3FFC0`

Флаги опасности.

7.1.2.156 `#define STATE_SYNC_INPUT 0x0800`

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

7.1.2.157 `#define STATE_SYNC_OUTPUT 0x1000`

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

7.1.2.158 `#define STATE_WINDING_RES_MISMATCH 0x0100000`

Сопротивления обмоток слишком сильно отличаются друг от друга.

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

7.1.2.159 `#define SYNCIN_ENABLED 0x01`

Включение необходимости импульса синхронизации для начала движения.

7.1.2.160 `#define SYNCIN_INVERT 0x02`

Если установлен - срабатывает по переходу из 1 в 0.

Иначе - из 0 в 1.

7.1.2.161 `#define SYNCOUT_ENABLED 0x01`

Синхронизация выхода работает согласно настройкам, если флаг установлен.

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

7.1.2.162 `#define SYNCOUT_IN_STEPS 0x08`

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

7.1.2.163 `#define SYNCOUT_INVERT 0x04`

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

7.1.2.164 `#define SYNCOUT_ONPERIOD 0x40`

Выдает импульс синхронизации после прохождения SyncOutPeriod отсчётов.

7.1.2.165 `#define SYNCOUT_ONSTART 0x10`

Генерация синхронизирующего импульса при начале движения.

7.1.2.166 `#define SYNCOUT_ONSTOP 0x20`

Генерация синхронизирующего импульса при остановке.

7.1.2.167 `#define SYNCOUT_STATE 0x02`

Когда значение выхода управляется напрямую (см. флаг `SYNCOUT_ENABLED`), значение на выходе соответствует значению этого флага.

7.1.2.168 `#define TS_TYPE_BITS 0x07`

Биты, отвечающие за тип температурного датчика.

7.1.2.169 `#define UART_PARITY_BITS 0x03`

Биты, отвечающие за выбор четности.

7.1.2.170 `#define WIND_A_STATE_ABSENT 0x00`

Обмотка А не подключена.

7.1.2.171 `#define WIND_A_STATE_MALFUNC 0x02`

Короткое замыкание на обмотке А.

7.1.2.172 `#define WIND_A_STATE_OK 0x03`

Обмотка А работает адекватно.

7.1.2.173 `#define WIND_A_STATE_UNKNOWN 0x01`

Состояние обмотки А неизвестно.

7.1.2.174 `#define WIND_B_STATE_ABSENT 0x00`

Обмотка В не подключена.

7.1.2.175 `#define WIND_B_STATE_MALFUNC 0x20`

Короткое замыкание на обмотке В.

7.1.2.176 `#define WIND_B_STATE_OK 0x30`

Обмотка В работает адекватно.

7.1.2.177 `#define WIND_B_STATE_UNKNOWN 0x10`

Состояние обмотки В неизвестно.

7.1.2.178 `#define XIMC_API`

Макрос импорта библиотеки.

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

### 7.1.3 Типы

7.1.3.1 `typedef struct calibration_t calibration_t`

Структура калибровок

Где найти все значения для расчета?

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

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

1. Используя XILab, загрузите профиль для вашего позиционера. Профиль должен соответствовать полному названию вашего позиционера. Например: 8MT173-25-MEn1.cfg
2. Включите режим Feedback encoder, если он не был включен ранее
3. Вводите скорость в пользовательских единицах в поле Working speed
4. Во вкладке User units выключите флаг User units. Это позволит видеть значение в поле Working speed в RPM
5. Умножьте значение из поля Working speed (в RPM) на коэффициент B. Например:  $480 * 0.000009375 = 0.00045$ . Значение 0.00045 для позиционера 8MT173-25-MEn1 в режиме Encoder будет равно скорости 2 мм/сек

Ускорение, замедление и скорость в режиме антилюфта считаются аналогично

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

Прототип функции обратного вызова для логирования

Аргументы

<i>loglevel</i>	уровень логирования
<i>message</i>	сообщение

#### 7.1.4 Функции

7.1.4.1 `result_t XIMC_API close_device ( device_t * id )`

Закрывает устройство

Аргументы

<i>id</i>	- идентификатор устройства
-----------	----------------------------

Заметки

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

7.1.4.2 `result_t XIMC_API command_clear_fram ( device_t id )`

Очистка FRAM памяти контроллера.

Функция используется только производителем.

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.3 `result_t XIMC_API command_eeread_settings ( device_t id )`

Чтение настроек контроллера из EEPROM памяти позиционера.

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

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.4 `result_t XIMC_API command_eesave_settings ( device_t id )`

Запись настроек контроллера в EEPROM память позиционера Функция должна использоваться только производителем.

## Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.5 `result_t XIMC_API command_home ( device_t id )`

Движение в домашнюю позицию.

Алгоритм движения:

- 1) Двигает мотор согласно скоростям `FastHome`, `uFastHome` и флагу `HOME_DIR_FAST` до достижения конечного выключателя, если флаг `HOME_STOP_ENDS` установлен. Или двигает до достижения сигнала с входа синхронизации, если установлен флаг `HOME_STOP_SYNC`. Или до поступления сигнала с датчика оборотов, если установлен флаг `HOME_STOP_REV_SN`
- 2) далее двигает согласно скоростям `SlowHome`, `uSlowHome` и флагу `HOME_DIR_SLOW` до достижения сигнала с входа синхронизации, если установлен флаг `HOME_MV_SEC`. Если флаг `HOME_MV_SEC` сброшен, пропускаем этот пункт.
- 3) далее двигает мотор согласно скоростям `FastHome`, `uFastHome` и флагу `HOME_DIR_SLOW` на расстояние `HomeDelta`, `uHomeDelta`.

Описание флагов и переменных см. описание команд `GHOM/SHOM`

## Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

См. также

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

7.1.4.6 `result_t XIMC_API command_homezero ( device_t id )`

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

Это удобный путь для калибровки нулевой позиции.

## Аргументы

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

7.1.4.7 `result_t XIMC_API command_left ( device_t id )`

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

## Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.8 **result\_t XIMC\_API** `command_loft ( device_t id )`

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

## Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.9 **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

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

7.1.4.10 **result\_t XIMC\_API** `command_move_calb ( device_t id, float Position, const calibration_t * calibration )`

Перемещение в позицию с использованием пользовательских единиц.

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

## Аргументы

<i>id</i>	идентификатор устройства
<i>Position</i>	позиция для перемещения
<i>calibration</i>	настройки пользовательских единиц

## Заметки

Параметр `Position` корректируется таблицей коррекции.

7.1.4.11 **result\_t XIMC\_API** `command_movr ( device_t id, int DeltaPosition, int uDeltaPosition )`

Перемещение на заданное смещение.

При получении команды "movr" двигатель начинает смещаться (если не используется режим "ТТЛСинхроВхода"), 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 ДС мотора это поле не используется.

Аргументы

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

7.1.4.12 **result\_t XIMC\_API** `command_movr_calb ( device_t id, float DeltaPosition, const calibration_t * calibration )`

Перемещение на заданное смещение с использованием пользовательских единиц.

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

Аргументы

<i>DeltaPosition</i>	смещение.
<i>id</i>	идентификатор устройства
<i>calibration</i>	настройки пользовательских единиц

Заметки

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

7.1.4.13 **result\_t XIMC\_API** `command_power_off ( device_t id )`

Немедленное отключение питания двигателя вне зависимости от его состояния.

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

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

См. также

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



7.1.4.14 **result\_t** **XIMC\_API** `command_read_robust_settings ( device_t id )`

Чтение важных настроек (калибровочные коэффициенты и т. п.) контроллера из flash памяти в оперативную, заменяя текущие настройки. Только для производителя.

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.15 **result\_t** **XIMC\_API** `command_read_settings ( device_t id )`

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

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.16 **result\_t** **XIMC\_API** `command_reset ( device_t id )`

Перезагрузка контроллера.

Функция используется только производителем.

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.17 **result\_t** **XIMC\_API** `command_right ( device_t id )`

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

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.18 **result\_t** **XIMC\_API** `command_save_robust_settings ( device_t id )`

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

п.) во встроенную энергонезависимую память контроллера. Только для производителя.

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.19 `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

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.20 `result_t XIMC_API command_sstp ( device_t id )`

Плавная остановка.

Двигатель останавливается с ускорением замедления.

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.21 `result_t XIMC_API command_start_measurements ( device_t id )`

Начать измерения и буферизацию скорости, ошибки следования.

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

7.1.4.22 `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).

При вызове этой команды сбрасывается флаг ALARM.

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

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

Обновление прошивки.

Команда только для производителя.

Аргументы

<i>uri</i>	идентификатор устройства
<i>data</i>	указатель на массив байтов прошивки
<i>data_size</i>	размер массива в байтах

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

Ожидание остановки контроллера

Аргументы

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

7.1.4.25 `result_t XIMC_API command_zero ( device_t id )`

Устанавливает текущую позицию равной 0.

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

Аргументы

<i>id</i>	идентификатор устройства
-----------	--------------------------

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

Поиск и составление списка доступных устройств.

По умолчанию формирует список устройств, подключенных к данному компьютеру и представленных в виде COM-портов. Дополнительно можно включить поиск сетевых устройств. Найденные в локальной сети устройства попадут в тот же список.

Для получения информации из собранного списка воспользуйтесь соответствующими функциями с префиксом `get_enumerate_` и полученным идентификатором `device_enumeration`.

После завершения работы со списком найденных устройств следует освободить память с помощью функции `free_enumerate_devices()`.

## Аргументы

in	<i>enumerate_ - flags</i>	<p>набор флагов, задающих режимы поиска. Флаги могут применяться совместно через побитовое "ИЛИ".</p> <ul style="list-style-type: none"> <li>• <code>ENUMERATE_NETWORK</code> - включает поиск сетевых устройств. Если флаг установлен, сетевые устройства будут добавлены в общий список. Если флаг не установлен, в списке будут только устройства, подключенные к данному компьютеру.</li> <li>• <code>ENUMERATE_ALL_COM</code> - при включенной опции опрашивает все устройства типа COM-порт в системе. При отключенной опции опрашивает только устройства, имена которых соответствуют маске устройств XIMC ("XIMC Motor Controller" в Windows, /dev/ximc/ и /dev/ttyACM/ на Linux/Mac).</li> <li>• <code>ENUMERATE_PROBE</code> - включает проверку устройств и сбор дополнительной информации (серийный номер, версию, модель, имя...). Если данный флаг установлен, в список будут добавлены только устройства, которые гарантированно можно открыть, но могут не попадать устройства, подключенные через RS232-преобразователи. Если флаг не установлен, то в списке будет больше устройств (в частности, попадут устройства, явным образом перечисленные в <code>hints</code>), но доступность и совместимость с данной библиотекой не гарантируется.</li> </ul>
in	<i>hints</i>	<p>дополнительная информация для повышения эффективности поиска. Имеет смысл использовать в случае сложной сетевой конфигурации, когда автоматический поиск может находить не всё. Формат - строка "ключ1=значение1\ключ2=значение2". Неизвестные ключи игнорируются. Один ключ может иметь несколько значений, которые перечисляются через запятую: ключ=значение1, значение2, значение3. Допустимые ключи:</p> <ul style="list-style-type: none"> <li>• <code>addr</code> - список URЛОВ сетевых контроллеров или серверов с подключенными контроллерами. Поле применяется совместно с флагом <code>ENUMERATE_NETWORK</code>. Протоколы и форматы адресов: <ul style="list-style-type: none"> <li>– <code>xi-tcp://&lt;ip-адрес&gt;</code> - сетевые контроллеры и контроллеры, подключенные через Ethernet-RS232 преобразователи. Если флаг <code>ENUMERATE_PROBE</code> не установлен, все перечисленные устройства попадут в список.</li> <li>– <code>xi-net://&lt;ip-адрес&gt;</code> - сетевые многоосевые системы, <code>xi-net</code> сервера. Запрос информации о наличии контроллеров по этим адресам будет сделан независимо от результатов автоматической процедуры сетевого поиска.</li> </ul> </li> <li>• <code>adapter_addr</code> - список IP-адресов локальных сетевых адаптеров, через который должен осуществляться поиск. Если ключ отсутствует или ни одного адаптера не указано, то поиск производится на всех адаптерах.</li> </ul>

## Возвращает

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

## Примеры использования:

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

7.1.4.27 **result\_t XIMC\_API** free\_enumerate\_devices ( **device\_enumeration\_t** device\_enumeration )

Освобождает память, выделенную *enumerate\_devices*.

Аргументы

in	<i>device_enumeration</i>	закрытый указатель на данные о перечисленных устройствах
----	---------------------------	--

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

Чтение информации о дополнительных аксессуарах из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>accessories_settings</i>	структура, содержащая информацию о дополнительных аксессуарах

7.1.4.29 **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

	<i>id</i>	идентификатор устройства
out	<i>analog_data</i>	аналоговые данные

7.1.4.30 **result\_t XIMC\_API** get\_bootloader\_version ( **device\_t** id, unsigned int \* Major, unsigned int \* Minor, unsigned int \* Release )

Чтение номера версии загрузчика контроллера.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>Major</i>	номер основной версии
out	<i>Minor</i>	номер дополнительной версии
out	<i>Release</i>	номер релиза

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

Чтение настроек управления тормозом.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>brake_settings</i>	структура, содержащая настройки управления тормозом

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

Команда чтения калибровочных коэффициентов.

Команда только для производителя. Эта функция заполняет структуру калибровочных коэффициентов. Эти коэффициенты используются для пересчёта кодов АЦП в токи обмоток и полный ток потребления. Коэффициенты сгруппированы в пары, XXX\_A и XXX\_B; пары представляют собой коэффициенты линейного уравнения. Первый коэффициент - тангенс угла наклона, второй - постоянное смещение. Таким образом,  $XXX\_Current[mA] = XXX\_A[mA/ADC] * XXX\_ADC\_CODE[ADC] + XXX\_B[mA]$ .

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>calibration_settings</i>	калибровочные коэффициенты

```
7.1.4.33 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

	<i>id</i>	идентификатор устройства
out	<i>chart_data</i>	структура chart_data.

7.1.4.34 `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

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

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

Чтение настроек управления мотором с использованием пользовательских единиц.

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

Аргументы

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

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

Чтение пользовательского имени контроллера и настроек из FRAM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>controller_name</i>	структура, содержащая установленное пользовательское имя контроллера и флаги настроек

7.1.4.37 `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

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

	<i>id</i>	идентификатор устройства
out	<i>ctp_settings</i>	структура, содержащая настройки контроля позиции

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

Чтение данных из прошивки для отладки и поиска неисправностей.

Команда только для производителя. Получаемые данные зависят от версии прошивки, истории и контекста использования.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>debug_read</i>	Данные для отладки.

7.1.4.39 `int XIMC_API get_device_count ( device_enumeration_t device_enumeration )`

Возвращает количество подключенных устройств.

Аргументы

in	<i>device_enumeration</i>	закрытый указатель на данные о перечисленных устройствах
----	---------------------------	--

7.1.4.40 `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

	<i>id</i>	идентификатор устройства.
out	<i>device_information</i>	информация об устройстве Информация об устройстве.

См. также

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



7.1.4.41 **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	<code>device_enumeration</code>	закрытый указатель на данные о перечисленных устройствах
in	<code>device_index</code>	номер устройства

7.1.4.42 **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

	<code>id</code>	идентификатор устройства
out	<code>edges_settings</code>	настройки, определяющие тип границ, поведение мотора при их достижении и параметры концевых выключателей

7.1.4.43 **result\_t XIMC\_API** `get_edges_settings_calb ( device_t id, edges_settings_calb_t * edges_settings_calb, const calibration_t * calibration )`

Чтение настроек границ и концевых выключателей с использованием пользовательских единиц.

См. также

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

Аргументы

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

Заметки

Внимание! Некоторые параметры структуры `edges_settings_calb` корректируются таблицей коррекции координат.

7.1.4.44 **result\_t XIMC\_API** `get_emf_settings ( device_t id, emf_settings_t * emf_settings )`

Чтение электромеханических настроек шагового двигателя.

Настройки различны для разных двигателей.

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>emf_settings</i>	настройки EMF

7.1.4.45 **result\_t XIMC\_API** get\_encoder\_information ( **device\_t** id, **encoder\_information\_t** \* encoder\_information )

Чтение информации об энкодере из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>encoder_information</i>	структура, содержащая информацию об энкодере

7.1.4.46 **result\_t XIMC\_API** get\_encoder\_settings ( **device\_t** id, **encoder\_settings\_t** \* encoder\_settings )

Чтение настроек энкодера из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>encoder_settings</i>	структура, содержащая настройки энкодера

7.1.4.47 **result\_t XIMC\_API** get\_engine\_advanced\_setup ( **device\_t** id, **engine\_advanced\_setup\_t** \* engine\_advanced\_setup )

Чтение расширенных настроек.

См. также

[set\\_engine\\_advanced\\_setup](#)

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>engine_advanced_setup</i>	настройки EAS

7.1.4.48 `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

	<i>id</i>	идентификатор устройства
out	<i>engine_settings</i>	структура с настройками мотора

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

Чтение настроек мотора с использованием пользовательских единиц.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>engine_settings_calb</i>	структура с настройками мотора
	<i>calibration</i>	настройки пользовательских единиц

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

Возвращает информацию о типе мотора и силового драйвера.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>entype_settings</i>	структура, содержащая настройки типа мотора и типа силового драйвера

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

Возвращает имя подключенного устройства из перечисления устройств.

Возвращает имя устройства с номером `device_index`.

Аргументы

in	<code>device_ - enumeration</code>	закрытый указатель на данные о перечисленных устройствах
in	<code>device_index</code>	номер устройства
out	<code>controller</code>	память имя устройства

7.1.4.52 **result\_t XIMC\_API** `get_enumerate_device_information ( device_enumeration_t device_enumeration, int device_index, device_information_t * device_information )`

Возвращает информацию о подключенном устройстве из перечисления устройств.

Возвращает информацию о устройстве с номером `device_index`.

Аргументы

in	<code>device_ - enumeration</code>	закрытый указатель на данные о перечисленных устройствах
in	<code>device_index</code>	номер устройства
out	<code>device_ - information</code>	информация об устройстве

7.1.4.53 **result\_t XIMC\_API** `get_enumerate_device_network_information ( device_enumeration_t device_enumeration, int device_index, device_network_information_t * device_network_information )`

Возвращает сетевую информацию о подключенном устройстве из перечисления устройств.

Возвращает сетевую информацию о устройстве с номером `device_index`.

Аргументы

in	<code>device_ - enumeration</code>	закрытый указатель на данные о перечисленных устройствах
in	<code>device_index</code>	номер устройства
out	<code>device_ - network_ - information</code>	сетевая информация об устройстве

7.1.4.54 **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	<code>device_ - enumeration</code>	закрытый указатель на данные о перечисленных устройствах
in	<code>device_index</code>	номер устройства
in	<code>serial</code>	серийный номер устройства

7.1.4.55 **result\_t XIMC\_API** get\_enumerate\_device\_stage\_name ( **device\_enumeration\_t** device\_enumeration, int device\_index, **stage\_name\_t** \* stage\_name )

Возвращает имя подвижки для подключенного устройства из перечисления устройств.

Возвращает имя подвижки устройства с номером *device\_index*.

Аргументы

in	<i>device_enumeration</i>	закрытый указатель на данные о перечисленных устройствах
in	<i>device_index</i>	номер устройства
out	<i>stage</i>	наименование подвижки

7.1.4.56 **result\_t XIMC\_API** get\_extended\_settings ( **device\_t** id, **extended\_settings\_t** \* extended\_settings )

Чтение расширенных настроек.

В настоящее время не используется.

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>extended_settings</i>	настройки EST

7.1.4.57 **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

	<i>id</i>	идентификатор устройства
out	<i>extio_settings</i>	настройки EXTIO

7.1.4.58 **result\_t XIMC\_API** get\_feedback\_settings ( **device\_t** id, **feedback\_settings\_t** \* feedback\_settings )

Чтение настроек обратной связи

Аргументы

	<i>id</i>	идентификатор устройства
--	-----------	--------------------------

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

7.1.4.59 **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

	<i>id</i>	идентификатор устройства
out	<i>Major</i>	номер основной версии
out	<i>Minor</i>	номер дополнительной версии
out	<i>Release</i>	номер релиза

7.1.4.60 **result\_t XIMC\_API** get\_gear\_information ( **device\_t** id, **gear\_information\_t** \* gear\_information )

Чтение информации о редукторе из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>gear_information</i>	структура, содержащая информацию о редукторе

7.1.4.61 **result\_t XIMC\_API** get\_gear\_settings ( **device\_t** id, **gear\_settings\_t** \* gear\_settings )

Чтение настроек редуктора из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>gear_settings</i>	структура, содержащая настройки редуктора

7.1.4.62 **result\_t XIMC\_API** get\_globally\_unique\_identifier ( **device\_t** id, **globally\_unique\_identifier\_t** \* globally\_unique\_identifier )

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

Только для производителя. Уникальный идентификатор может быть использован в качестве инициализационного вектора для операций шифрования бутлоадера или в качестве серийного номера для

USB и других применений.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>globally_ - unique_ - identifier</i>	результат полей 0-3 определяет уникальный 128-битный идентификатор.

7.1.4.63 **result\_t XIMC\_API** `get_hallsensor_information ( device_t id, hallsensor_information_t * hallsensor_information )`

Чтение информации о датчиках Холла из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>hallsensor_ - information</i>	структура, содержащая информацию о датчиках Холла

7.1.4.64 **result\_t XIMC\_API** `get_hallsensor_settings ( device_t id, hallsensor_settings_t * hallsensor_settings )`

Чтение настроек датчиков Холла из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>hallsensor_ - settings</i>	структура, содержащая настройки датчиков Холла

7.1.4.65 **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, используемых для калибровки позиции, в память контроллера.

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>home_settings</i>	настройки калибровки позиции

7.1.4.66 **result\_t XIMC\_API** get\_home\_settings\_calb ( **device\_t** id, **home\_settings\_calb\_t** \* home\_settings\_calb, const **calibration\_t** \* calibration )

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

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>home_settings_calb</i>	настройки калибровки позиции
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.67 **result\_t XIMC\_API** get\_init\_random ( **device\_t** id, **init\_random\_t** \* init\_random )

Чтение случайного числа из контроллера.

Только для производителя.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>init_random</i>	случайная последовательность, сгенерированная контроллером

7.1.4.68 **result\_t XIMC\_API** get\_joystick\_settings ( **device\_t** id, **joystick\_settings\_t** \* joystick\_settings )

Чтение настроек джойстика.

При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость (при этом постоянно выполняется команда "soft stop"), а 100% отклонения соответствует MaxSpeed *i*, где *i*=0, если предыдущим использованием этого режима не было выбрано другое *i*. Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. Первая скорость в списке не должна быть нулевой. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Подробнее см. раздел "Управление с помощью джойстика" на сайте [https://doc.xisupport.com/ru/8smc5-usb/8SMCn-USB/Technical\\_specification/Additional\\_features/Joystick\\_control.html](https://doc.xisupport.com/ru/8smc5-usb/8SMCn-USB/Technical_specification/Additional_features/Joystick_control.html).

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>joystick_settings</i>	структура, содержащая настройки джойстика



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

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

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

См. также

[measurements\\_t](#)

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>measurements</i>	структура с буфером и его длиной.

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

Чтение информации о двигателе из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>motor_information</i>	структура, содержащая информацию о двигателе

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

Чтение настроек двигателя из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>motor_settings</i>	структура, содержащая настройки двигателя

7.1.4.72 `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 анти-люфта).

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>move_settings</i>	структура, содержащая настройки движения: скорость, ускорение, и т.д.

7.1.4.73 **result\_t XIMC\_API** get\_move\_settings\_calb ( **device\_t** id, **move\_settings\_calb\_t** \* move\_settings\_calb, const **calibration\_t** \* calibration )

Команда чтения настроек перемещения с использованием пользовательских единиц(скорость, ускорение, threshold и скорость в режиме антилюфта).

Аргументы

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

7.1.4.74 **result\_t XIMC\_API** get\_network\_settings ( **device\_t** id, **network\_settings\_t** \* network\_settings )

Команда чтения сетевых настроек.

Только для производителя. Эта функция возвращает текущие сетевые настройки.

Аргументы

<i>DHCPEnabled</i>	DHCP включен (1) или нет (0)
<i>IPv4Address[4]</i>	Массив[4] с IP-адресом
<i>SubnetMask[4]</i>	Массив[4] с маской подсети
<i>Default-Gateway[4]</i>	Массив[4] со шлюзом сети

7.1.4.75 **result\_t XIMC\_API** get\_nonvolatile\_memory ( **device\_t** id, **nonvolatile\_memory\_t** \* nonvolatile\_memory )

Чтение пользовательских данных из FRAM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>nonvolatile_memory</i>	структура, содержащая установленные пользовательские данные

7.1.4.76 **result\_t XIMC\_API** get\_password\_settings ( **device\_t** id, **password\_settings\_t** \* password\_settings )

Команда чтения пароля к веб-странице.

Только для производителя. Эта функция пользователя прочитает пользовательский пароль к веб-странице из памяти контроллера.

Аргументы

<i>User-Password[20]</i>	Строчка-пароль для доступа к веб-странице
--------------------------	---

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

Чтение ПИД коэффициентов.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>pid_settings</i>	настройки ПИД

7.1.4.78 `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

	<i>id</i>	идентификатор устройства
out	<i>the_get_position</i>	структура, содержащая позицию мотора.

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

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>the_get_position_calb</i>	структура, содержащая позицию мотора.
	<i>calibration</i>	настройки пользовательских единиц

Заметки

Внимание! Некоторые параметры структуры `get_position_calb` корректируются таблицей коррекции координат.

7.1.4.80 `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

	<i>id</i>	идентификатор устройства
out	<i>power_settings</i>	структура, содержащая настройки питания шагового мотора

7.1.4.81 **result\_t XIMC\_API** get\_secure\_settings ( **device\_t** id, **secure\_settings\_t** \* secure\_settings )

Команда записи установок защит.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>secure_settings</i>	настройки, определяющие максимально допустимые параметры, для защиты оборудования

См. также

status\_t::flags

7.1.4.82 **result\_t XIMC\_API** get\_serial\_number ( **device\_t** id, unsigned int \* SerialNumber )

Чтение серийного номера контроллера.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>SerialNumber</i>	серийный номер контроллера

7.1.4.83 **result\_t XIMC\_API** get\_stage\_information ( **device\_t** id, **stage\_information\_t** \* stage\_information )

Чтение информации о позиционере из EEPROM.

Не поддерживается.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>stage_information</i>	структура, содержащая информацию о позиционере

7.1.4.84 **result\_t XIMC\_API** get\_stage\_name ( **device\_t** id, **stage\_name\_t** \* stage\_name )

Чтение пользовательского имени подвижки из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>stage_name</i>	структура, содержащая установленное пользовательское имя позиционера

7.1.4.85 **result\_t** XIMC\_API get\_stage\_settings ( **device\_t** id, **stage\_settings\_t** \* stage\_settings )

Чтение настроек позиционера из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>stage_settings</i>	структура, содержащая настройки позиционера

7.1.4.86 **result\_t** XIMC\_API get\_status ( **device\_t** id, **status\_t** \* status )

Возвращает информацию о текущем состоянии устройства.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>status</i>	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](#)

7.1.4.87 **result\_t** XIMC\_API get\_status\_calb ( **device\_t** id, **status\_calb\_t** \* status, const **calibration\_t** \* calibration )

Возвращает информацию о текущем состоянии устройства.

Аргументы

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

См. также

[get\\_status](#)

7.1.4.88 **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

	<i>id</i>	идентификатор устройства
out	<i>sync_in_settings</i>	настройки синхронизации

7.1.4.89 **result\_t XIMC\_API** get\_sync\_in\_settings\_calb ( **device\_t** id, **sync\_in\_settings\_calb\_t** \* sync\_in\_settings\_calb, const **calibration\_t** \* calibration )

Чтение настроек для входного импульса синхронизации с использованием пользовательских единиц.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>sync_in_settings_calb</i>	настройки синхронизации
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.90 **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.

7.1.4.91 **result\_t XIMC\_API** get\_sync\_out\_settings\_calb ( **device\_t** id, **sync\_out\_settings\_calb\_t** \* sync\_out\_settings\_calb, const **calibration\_t** \* calibration )

Чтение настроек для выходного импульса синхронизации с использованием пользовательских единиц.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
in	<i>sync_out_settings_calb</i>	настройки синхронизации
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.92 **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

	<i>Speed</i>	Скорость UART
out	<i>uart_settings</i>	настройки UART

7.1.4.93 **result\_t** XIMC\_API goto\_firmware ( **device\_t** id, uint8\_t \* ret )

Перезагрузка в прошивку в контроллере

Аргументы

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

7.1.4.94 **result\_t** XIMC\_API has\_firmware ( const char \* uri, uint8\_t \* ret )

Проверка наличия прошивки в контроллере

Аргументы

	<i>uri</i>	уникальный идентификатор ресурса устройства
out	<i>ret</i>	ноль, если прошивка присутствует

7.1.4.95 **result\_t** XIMC\_API load\_correction\_table ( **device\_t** \* id, const char \* namefile )

Команда загрузки корректирующей таблицы из текстового файла (данная функция устарела).

Используйте функцию [set\\_correction\\_table\(device\\_t id, const char\\* namefile\)](#). Таблица используется для коррекции положения в случае механических неточностей. Работает для некоторых параметров в `_calb` командах.

## Аргументы

	<i>id</i>	- идентификатор устройства
<i>in</i>	<i>namefile</i>	- имя файла должно быть полным. Если используется короткое имя, файл должен находиться в директории приложения. Если имя файла равно NULL таблица коррекции будет очищена. Формат файла: два столбца разделенных табуляцией. Заголовки столбцов строковые. Данные действительные разделитель точка. Первый столбец координата. Второй - отклонение вызванное ошибкой механики. Между координатами отклонение рассчитывается линейно. За диапазоном константа равная отклонению на границе. Максимальная длина таблицы 100 строк.

## Заметки

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

## См. также

[command\\_move](#)  
[command\\_movr](#)  
[get\\_position\\_calb](#)  
[get\\_position\\_calb\\_t](#)  
[get\\_status\\_calb](#)  
[status\\_calb\\_t](#)  
[get\\_edges\\_settings\\_calb](#)  
[set\\_edges\\_settings\\_calb](#)  
[edges\\_settings\\_calb\\_t](#)

7.1.4.96 void **XIMC\_API** logging\_callback\_stderr\_narrow ( int loglevel, const wchar\_t \* message, void \* user\_data )

Простая функция логирования на stderr в узких (однобайтных) символах

## Аргументы

<i>loglevel</i>	уровень логирования
<i>message</i>	сообщение

7.1.4.97 void **XIMC\_API** logging\_callback\_stderr\_wide ( int loglevel, const wchar\_t \* message, void \* user\_data )

Простая функция логирования на stderr в широких символах

## Аргументы

<i>loglevel</i>	уровень логирования
<i>message</i>	сообщение



7.1.4.98 `void XIMC_API msec_sleep ( unsigned int msec )`

Приостанавливает работу на указанное время

Аргументы

<code>msec</code>	время в миллисекундах
-------------------	-----------------------

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

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

Аргументы

<code>in</code>	<i>uri</i>	- уникальный идентификатор устройства. URI устройства имеет вид "xi-com:port" или "xi-net://host/serial" или "xi-emu:///abs_path_to_file". На POSIX системах допускается пропуск "рутовского" слэша; например, "xi-emu:///home/user/virt_controller.bin". Для USB--COM устройства "port" это URI устройства в ОС. Например, "xi-com:\\\\.\\COM3" в Windows (с учётом экранирования двойные обратные слэши преобразуются в одинарные) или "xi-com:///dev/tty-ACM0" в Linux/Mac. Для сетевого устройства "host" это IPv4 адрес или полностью определённое имя домена, "serial" это серийный номер устройства в шестнадцатеричной системе. Например, "xi-net://192.168.0.1/00001234" или "xi-net://hostname.com/89ABCDEF". Для работы по TCP протоколу используйте "xi-tcp://<ip/host>:<port>". Например, "xi-tcp://192.168.0.1:1818". Для виртуального устройства "abs_file_to_file" это путь к файлу с сохранённым состоянием устройства. Если файл не существует, он будет создан и инициализирован значениями по умолчанию. Например, "xi-emu:///C:/dir/file.bin" в Windows или "xi-emu:///home/user/file.bin" в Linux/-Mac.
-----------------	------------	---

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

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

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

Аргументы

<code>in</code>	<i>uri</i>	- уникальный идентификатор устройства
-----------------	------------	---------------------------------------

7.1.4.101 `result_t XIMC_API service_command_updf ( device_t id )`

Команда переводит контроллер в режим обновления прошивки.

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

7.1.4.102 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>accessories_settings</i>	структура, содержащая информацию о дополнительных аксессуарах

7.1.4.103 **result\_t XIMC\_API** set\_bindy\_key ( const char \* keyfilepath )

Устарело.

Оставлено для совместимости. Ничего не делает.

7.1.4.104 **result\_t XIMC\_API** set\_brake\_settings ( **device\_t** id, const **brake\_settings\_t** \* brake\_settings )

Запись настроек управления тормозом.

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>brake_settings</i>	структура, содержащая настройки управления тормозом

7.1.4.105 **result\_t XIMC\_API** set\_calibration\_settings ( **device\_t** id, const **calibration\_settings\_t** \* calibration\_settings )

Команда записи калибровочных коэффициентов.

Команда только для производителя. Эта функция записывает структуру калибровочных коэффициентов в память контроллера. Эти коэффициенты используются для пересчёта кодов АЦП в токи обмоток и полный ток потребления. Коэффициенты сгруппированы в пары, XXX\_A и XXX\_B; пары представляют собой коэффициенты линейного уравнения. Первый коэффициент - тангенс угла наклона, второй - постоянное смещение. Таким образом,  $XXX\_Current[mA] = XXX\_A[mA/ADC] * XX\_ADC\_CODE[ADC] + XXX\_B[mA]$ .

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>calibration_settings</i>	калибровочные коэффициенты

7.1.4.106 **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

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

7.1.4.107 **result\_t** **XIMC\_API** `set_control_settings_calb ( device_t id, const control_settings_calb_t * control_settings_calb, const calibration_t * calibration )`

Запись настроек управления мотором с использованием пользовательских единиц.

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>control_settings_calb</i>	структура, содержащая настройки управления мотором с помощью джойстика или кнопок влево/вправо.
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.108 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>controller_information</i>	структура, содержащая информацию о контроллере

7.1.4.109 **result\_t** **XIMC\_API** `set_correction_table ( device_t id, const char * namefile )`

Команда загрузки корректирующей таблицы из текстового файла.

Таблица используется для коррекции положения в случае механических неточностей. Работает для некоторых параметров в `_calb` командах.

Аргументы

	<i>id</i>	- идентификатор устройства
<i>in</i>	<i>namefile</i>	- путь до файла должен быть полным или относительным. Если параметр равен NULL, таблица коррекции будет очищена. Формат файла: два столбца, разделенных табуляцией. Заголовки столбцов строковые. Данные действительные, разделитель точка. Первый столбец - координата. Второй - отклонение, вызванное ошибкой механики. Максимальная длина таблицы 100 строк. Координаты должны быть отсортированы по возрастанию.

См. также

```

command_move
command_movr
get_position_calb
get_position_calb_t
get_status_calb
status_calb_t
get_edges_settings_calb
set_edges_settings_calb
edges_settings_calb_t

```

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>ctp_settings</i>	структура, содержащая настройки контроля позиции

```
7.1.4.111 result_t XIMC_API set_debug_write ( device_t id, const debug_write_t * debug_write )
```

Запись данных в прошивку для отладки и поиска неисправностей.

Команда только для производителя.

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>debug_write</i>	Данные для отладки.

7.1.4.112 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>edges_settings</i>	настройки, определяющие тип границ, поведение мотора при их достижении и параметры концевых выключателей

7.1.4.113 **result\_t** XIMC\_API set\_edges\_settings\_calb ( **device\_t** id, const **edges\_settings\_calb\_t** \* edges\_settings\_calb, const **calibration\_t** \* calibration )

Запись настроек границ и концевых выключателей с использованием пользовательских единиц.

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>edges_settings_calb</i>	настройки, определяющие тип границ, поведение мотора при их достижении и параметры концевых выключателей
	<i>calibration</i>	настройки пользовательских единиц

Заметки

Внимание! Некоторые параметры структуры `edges_settings_calb` корректируются таблицей коррекции координат.

7.1.4.114 **result\_t** XIMC\_API set\_emf\_settings ( **device\_t** id, const **emf\_settings\_t** \* emf\_settings )

Запись электромеханических настроек шагового двигателя.

Настройки различны для разных двигателей. Пожалуйста, загружайте новые настройки, когда вы меняете мотор.

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>emf_settings</i>	настройки EMF

7.1.4.115 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>encoder_information</i>	структура, содержащая информацию об энкодере

7.1.4.116 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>encoder_settings</i>	структура, содержащая настройки энкодера

7.1.4.117 **result\_t** **XIMC\_API** set\_engine\_advanced\_setup ( **device\_t** id, const **engine\_advanced\_setup\_t** \* engine\_advanced\_setup )

Запись расширенных настроек.

См. также

[get\\_engine\\_advanced\\_setup](#)

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>engine_advanced_setup</i>	настройки EAS

7.1.4.118 **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 движе-

ния 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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>engine_settings</i>	структура с настройками мотора

7.1.4.119 **result\_t XIMC\_API** set\_engine\_settings\_calb ( **device\_t** id, const **engine\_settings\_calb\_t** \* engine\_settings\_calb, const **calibration\_t** \* calibration )

Запись настроек мотора с использованием пользовательских единиц.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>engine_settings_calb</i>	структура с настройками мотора
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.120 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>entype_settings</i>	структура, содержащая настройки типа мотора и типа силового драйвера

7.1.4.121 **result\_t XIMC\_API** set\_extended\_settings ( **device\_t** id, const **extended\_settings\_t** \* extended\_settings )

Запись расширенных настроек.

В настоящее время не используется.

См. также

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

## Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>extended_ - settings</i>	настройки EST

7.1.4.122 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>extio_settings</i>	настройки EXTIO

7.1.4.123 **result\_t XIMC\_API** set\_feedback\_settings ( **device\_t** id, const **feedback\_settings\_t** \* feedback\_settings )

Запись настроек обратной связи.

## Аргументы

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

7.1.4.124 **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

	<i>id</i>	идентификатор устройства
--	-----------	--------------------------



<code>in</code>	<code>gear_ - information</code>	структура, содержащая информацию о редукторе
-----------------	--------------------------------------	--

7.1.4.125 `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

	<code>id</code>	идентификатор устройства
<code>in</code>	<code>gear_settings</code>	структура, содержащая настройки редуктора

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

Запись информации о датчиках Холла в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	<code>id</code>	идентификатор устройства
<code>in</code>	<code>hallsensor_ - information</code>	структура, содержащая информацию о датчиках Холла

7.1.4.127 `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

	<code>id</code>	идентификатор устройства
<code>in</code>	<code>hallsensor_ - settings</code>	структура, содержащая настройки датчиков Холла

7.1.4.128 `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

	<i>id</i>	идентификатор устройства
out	<i>home_settings</i>	настройки калибровки позиции

7.1.4.129 **result\_t XIMC\_API** set\_home\_settings\_calb ( **device\_t** id, const **home\_settings\_calb\_t** \* home\_settings\_calb, const **calibration\_t** \* calibration )

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

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
in	<i>home_settings_calb</i>	настройки калибровки позиции
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.130 **result\_t XIMC\_API** set\_joystick\_settings ( **device\_t** id, const **joystick\_settings\_t** \* joystick\_settings )

Запись настроек джойстика.

При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость (при этом постоянно выполняется команда "soft stop"), а 100% отклонения соответствует MaxSpeed *i*, где *i*=0, если предыдущим использованием этого режима не было выбрано другое *i*. Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. Первая скорость в списке не должна быть нулевой. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Подробнее см. раздел "Управление с помощью джойстика" на сайте [https://doc.xisupport.com/ru/8smc5-usb/8SMCn-USB/Technical\\_specification/Additional\\_features/Joystick\\_control.html](https://doc.xisupport.com/ru/8smc5-usb/8SMCn-USB/Technical_specification/Additional_features/Joystick_control.html).

Аргументы

	<i>id</i>	идентификатор устройства
in	<i>joystick_settings</i>	структура, содержащая настройки джойстика

7.1.4.131 **void XIMC\_API** set\_logging\_callback ( **logging\_callback\_t** logging\_callback, void \* user\_data )

Устанавливает функцию обратного вызова для логирования.

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

Аргументы

<code>logging_callback</code>	указатель на функцию обратного вызова
-------------------------------	---------------------------------------

7.1.4.132 **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

	<code>id</code>	идентификатор устройства
<code>in</code>	<code>motor_information</code>	структура, содержащая информацию о двигателе

7.1.4.133 **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

	<code>id</code>	идентификатор устройства
<code>in</code>	<code>motor_settings</code>	структура, содержащая настройки двигателя

7.1.4.134 **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).

Аргументы

	<code>id</code>	идентификатор устройства
<code>in</code>	<code>move_settings</code>	структура, содержащая настройки движения: скорость, ускорение, и т.д.

7.1.4.135 **result\_t** **XIMC\_API** set\_move\_settings\_calb ( **device\_t** id, const **move\_settings\_calb\_t** \* move\_settings\_calb, const **calibration\_t** \* calibration )

Команда записи настроек перемещения, с использованием пользовательских единиц (скорость, ускорение, threshold и скорость в режиме антилюфта).

Аргументы

	<code>id</code>	идентификатор устройства
<code>in</code>	<code>move_settings_calb</code>	структура, содержащая настройки движения: скорость, ускорение, и т.д.

	<i>calibration</i>	настройки пользовательских единиц
--	--------------------	-----------------------------------

7.1.4.136 **result\_t XIMC\_API** `set_network_settings ( device_t id, const network_settings_t * network_settings )`

Команда записи сетевых настроек.

Только для производителя. Эта функция меняет сетевые настройки на заданные.

Аргументы

<i>DHCPEnabled</i>	DHCP включен (1) или нет (0)
<i>IPv4Address[4]</i>	Массив[4] с IP-адресом
<i>SubnetMask[4]</i>	Массив[4] с маской подсети
<i>Default-Gateway[4]</i>	Массив[4] со шлюзом сети

7.1.4.137 **result\_t XIMC\_API** `set_nonvolatile_memory ( device_t id, const nonvolatile_memory_t * nonvolatile_memory )`

Запись пользовательских данных во FRAM.

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>nonvolatile_memory</i>	структура, содержащая установленные пользовательские данные

7.1.4.138 **result\_t XIMC\_API** `set_password_settings ( device_t id, const password_settings_t * password_settings )`

Команда записи пароля к веб-странице.

Только для производителя. Эта функция меняет пользовательский пароль к веб-странице.

Аргументы

<i>User-Password[20]</i>	Строчка-пароль для доступа к веб-странице
--------------------------	---

7.1.4.139 **result\_t XIMC\_API** `set_pid_settings ( device_t id, const pid_settings_t * pid_settings )`

Запись ПИД коэффициентов.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>pid_settings</i>	настройки ПИД

7.1.4.140 **result\_t** XIMC\_API set\_position ( **device\_t** id, const **set\_position\_t** \* the\_set\_position )

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>out</i>	<i>the_set_position</i>	структура, содержащая позицию мотора.

7.1.4.141 **result\_t** XIMC\_API set\_position\_calb ( **device\_t** id, const **set\_position\_calb\_t** \* the\_set\_position\_calb, const **calibration\_t** \* calibration )

Устанавливает произвольное значение положения и значение энкодера всех двигателей с использованием пользовательских единиц.

Аргументы

	<i>id</i>	идентификатор устройства
<i>out</i>	<i>the_set_position_calb</i>	структура, содержащая позицию мотора.
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.142 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>power_settings</i>	структура, содержащая настройки питания шагового мотора

7.1.4.143 **result\_t** XIMC\_API set\_secure\_settings ( **device\_t** id, const **secure\_settings\_t** \* secure\_settings )

Команда записи установок защит.

Аргументы

	<i>id</i>	идентификатор устройства
<i>secure_settings</i>		структура с настройками критических значений

См. также

`status_t::flags`

7.1.4.144 **result\_t XIMC\_API** `set_serial_number ( device_t id, const serial_number_t * serial_number )`

Запись серийного номера и версии железа во flash память контроллера.

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>serial_number</i>	структура, содержащая серийный номер, версию железа и ключ.

7.1.4.145 **result\_t XIMC\_API** `set_stage_information ( device_t id, const stage_information_t * stage_information )`

Запись информации о позиционере в EEPROM.

Не поддерживается. Функция должна использоваться только производителем.

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>stage_information</i>	структура, содержащая информацию о позиционере

7.1.4.146 **result\_t XIMC\_API** `set_stage_name ( device_t id, const stage_name_t * stage_name )`

Запись пользовательского имени подвижки в EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>stage_name</i>	структура, содержащая установленное пользовательское имя позиционера

7.1.4.147 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>stage_settings</i>	структура, содержащая настройки позиционера

7.1.4.148 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>sync_in_settings</i>	настройки синхронизации

7.1.4.149 **result\_t XIMC\_API** `set_sync_in_settings_calb ( device_t id, const sync_in_settings_calb_t * sync_in_settings_calb, const calibration_t * calibration )`

Запись настроек для входного импульса синхронизации с использованием пользовательских единиц.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>sync_in_settings_calb</i>	настройки синхронизации
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.150 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>sync_out_settings</i>	настройки синхронизации

7.1.4.151 **result\_t** XIMC\_API set\_sync\_out\_settings\_calb ( **device\_t** id, const **sync\_out\_settings\_calb\_t** \* sync\_out\_settings\_calb, const **calibration\_t** \* calibration )

Запись настроек для выходного импульса синхронизации с использованием пользовательских единиц.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>sync_out_settings_calb</i>	настройки синхронизации
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.152 **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

	<i>Speed</i>	Скорость UART
<i>in</i>	<i>uart_settings</i>	настройки UART

7.1.4.153 **result\_t** XIMC\_API write\_key ( const char \* uri, uint8\_t \* key )

Запись ключа защиты. Функция используется только производителем.

Аргументы

	<i>uri</i>	идентификатор устройства
<i>in</i>	<i>key</i>	ключ защиты. Диапазон: 0..4294967295

7.1.4.154 **result\_t** XIMC\_API ximc\_fix\_usbser\_sys ( const char \* device\_uri )

(Устарела) Исправление ошибки драйвера USB в Windows.

Подсистема USB-COM на ОС Windows не всегда работает корректно. При работе возможны следующие неисправности: Все попытки открыть устройство заканчиваются неудачно. Устройство можно открыть и отправить в него данные, но ответные данные не приходят. Эти проблемы исправляются переподключением устройства или его переинициализацией в диспетчере устройств. Функция [ximc\\_fix\\_usbser\\_sys\(\)](#) автоматизирует процесс удаления-обнаружения.



7.1.4.155 `void XIMC_API ximc_version ( char * version )`

Возвращает версию библиотеки

Аргументы

<i>version</i>	буфер для строки с версией, 32 байт достаточно
----------------	--

# Предметный указатель

## A

- calibration\_t, [21](#)
- A1Voltage
  - analog\_data\_t, [15](#)
- A1Voltage\_ADC
  - analog\_data\_t, [15](#)
- A2Voltage
  - analog\_data\_t, [15](#)
- A2Voltage\_ADC
  - analog\_data\_t, [15](#)
- ACurrent
  - analog\_data\_t, [15](#)
- ACurrent\_ADC
  - analog\_data\_t, [16](#)
- Accel
  - move\_settings\_calb\_t, [58](#)
  - move\_settings\_t, [59](#)
- accessories\_settings\_t, [12](#)
- LimitSwitchesSettings, [13](#)
- MBRatedCurrent, [13](#)
- MBRatedVoltage, [13](#)
- MBSettings, [13](#)
- MBTorque, [13](#)
- MagneticBrakeInfo, [13](#)
- TSGrad, [13](#)
- TSMaх, [13](#)
- TSMin, [13](#)
- TSSettings, [14](#)
- TemperatureSensorInfo, [13](#)
- Accuracy
  - sync\_out\_settings\_calb\_t, [79](#)
  - sync\_out\_settings\_t, [80](#)
- analog\_data\_t, [14](#)
- A1Voltage, [15](#)
- A1Voltage\_ADC, [15](#)
- A2Voltage, [15](#)
- A2Voltage\_ADC, [15](#)
- ACurrent, [15](#)
- ACurrent\_ADC, [16](#)
- B1Voltage, [16](#)
- B1Voltage\_ADC, [16](#)
- B2Voltage, [16](#)
- B2Voltage\_ADC, [16](#)
- BCurrent, [16](#)
- BCurrent\_ADC, [16](#)
- Enc\_Check, [16](#)
- Enc\_Check\_ADC, [16](#)
- FullCurrent, [16](#)
- FullCurrent\_ADC, [16](#)
- Joy, [17](#)
- Joy\_ADC, [17](#)
- Pot, [17](#)
- SupVoltage, [17](#)
- SupVoltage\_ADC, [17](#)
- Temp, [17](#)
- Temp\_ADC, [17](#)
- Antiplay
  - engine\_settings\_calb\_t, [36](#)
  - engine\_settings\_t, [38](#)
- AntiplaySpeed
  - move\_settings\_calb\_t, [58](#)
  - move\_settings\_t, [59](#)
- AveragedPowerRatio
  - chart\_data\_t, [22](#)
- B1Voltage
  - analog\_data\_t, [16](#)
- B1Voltage\_ADC
  - analog\_data\_t, [16](#)
- B2Voltage
  - analog\_data\_t, [16](#)
- B2Voltage\_ADC
  - analog\_data\_t, [16](#)
- BACK\_EMF\_KM\_AUTO
  - ximc.h, [108](#)
- BCurrent
  - analog\_data\_t, [16](#)
- BCurrent\_ADC
  - analog\_data\_t, [16](#)
- BORDER\_IS\_ENCODER
  - ximc.h, [108](#)
- BORDER\_STOP\_LEFT
  - ximc.h, [108](#)
- BORDER\_STOP\_RIGHT
  - ximc.h, [108](#)
- BRAKE\_ENABLED
  - ximc.h, [109](#)
- BRAKE\_ENG\_PWROFF
  - ximc.h, [109](#)
- BackEMFFlags
  - emf\_settings\_t, [32](#)
- BorderFlags
  - edges\_settings\_calb\_t, [30](#)
  - edges\_settings\_t, [31](#)

- brake\_settings\_t, 17
  - BrakeFlags, 18
  - t1, 18
  - t2, 18
  - t3, 18
  - t4, 18
- BrakeFlags
  - brake\_settings\_t, 18
- CONTROL\_MODE\_BITS
  - ximc.h, 109
- CONTROL\_MODE\_JOY
  - ximc.h, 109
- CONTROL\_MODE\_LR
  - ximc.h, 109
- CONTROL\_MODE\_OFF
  - ximc.h, 109
- CSS1\_A
  - calibration\_settings\_t, 19
- CSS1\_B
  - calibration\_settings\_t, 19
- CSS2\_A
  - calibration\_settings\_t, 19
- CSS2\_B
  - calibration\_settings\_t, 19
- CTP\_ALARM\_ON\_ERROR
  - ximc.h, 109
- CTP\_BASE
  - ximc.h, 109
- CTP\_ENABLED
  - ximc.h, 110
- CTP\_ERROR\_CORRECTION
  - ximc.h, 110
- CTPFlags
  - ctp\_settings\_t, 27
- CTPMinError
  - ctp\_settings\_t, 27
- calibration\_settings\_t, 18
  - CSS1\_A, 19
  - CSS1\_B, 19
  - CSS2\_A, 19
  - CSS2\_B, 19
  - FullCurrent\_A, 19
  - FullCurrent\_B, 19
- calibration\_t, 20
  - A, 21
  - MicrostepMode, 21
  - ximc.h, 125
- chart\_data\_t, 21
  - AveragedPowerRatio, 22
  - Joy, 22
  - Pot, 22
  - WindingCurrentA, 22
  - WindingCurrentB, 22
  - WindingCurrentC, 22
  - WindingVoltageA, 22
  - WindingVoltageB, 22
  - WindingVoltageC, 23
- close\_device
  - ximc.h, 126
- ClutterTime
  - sync\_in\_settings\_calb\_t, 77
  - sync\_in\_settings\_t, 78
- CmdBufFreeSpace
  - status\_calb\_t, 72
  - status\_t, 75
- command\_clear\_fram
  - ximc.h, 126
- command\_eeread\_settings
  - ximc.h, 126
- command\_eesave\_settings
  - ximc.h, 126
- command\_home
  - ximc.h, 127
- command\_homezero
  - ximc.h, 127
- command\_left
  - ximc.h, 127
- command\_loft
  - ximc.h, 128
- command\_move
  - ximc.h, 128
- command\_move\_calb
  - ximc.h, 128
- command\_movr
  - ximc.h, 128
- command\_movr\_calb
  - ximc.h, 129
- command\_power\_off
  - ximc.h, 129
- command\_read\_robust\_settings
  - ximc.h, 129
- command\_read\_settings
  - ximc.h, 130
- command\_reset
  - ximc.h, 130
- command\_right
  - ximc.h, 130
- command\_save\_robust\_settings
  - ximc.h, 130
- command\_save\_settings
  - ximc.h, 130
- command\_sstp
  - ximc.h, 131
- command\_start\_measurements
  - ximc.h, 131
- command\_stop
  - ximc.h, 131
- command\_update\_firmware
  - ximc.h, 131
- command\_wait\_for\_stop
  - ximc.h, 132

- command\_zero
  - ximc.h, [132](#)
- control\_settings\_calb\_t, [23](#)
  - Flags, [23](#)
  - MaxClickTime, [23](#)
  - MaxSpeed, [24](#)
  - Timeout, [24](#)
- control\_settings\_t, [24](#)
  - Flags, [25](#)
  - MaxClickTime, [25](#)
  - MaxSpeed, [25](#)
  - Timeout, [25](#)
  - uDeltaPosition, [25](#)
  - uMaxSpeed, [25](#)
- controller\_name\_t, [25](#)
  - ControllerName, [26](#)
  - CtrlFlags, [26](#)
- ControllerName
  - controller\_name\_t, [26](#)
- CountsPerTurn
  - feedback\_settings\_t, [41](#)
- Criticalpwr
  - secure\_settings\_t, [65](#)
- Criticalusb
  - secure\_settings\_t, [65](#)
- CriticalUpwr
  - secure\_settings\_t, [65](#)
- CriticalUusb
  - secure\_settings\_t, [65](#)
- ctp\_settings\_t, [26](#)
  - CTPFlags, [27](#)
  - CTPMinError, [27](#)
- CtrlFlags
  - controller\_name\_t, [26](#)
- CurPosition
  - status\_calb\_t, [72](#)
  - status\_t, [75](#)
- CurSpeed
  - status\_calb\_t, [72](#)
  - status\_t, [75](#)
- CurT
  - status\_calb\_t, [72](#)
  - status\_t, [75](#)
- CurrReductDelay
  - power\_settings\_t, [64](#)
- CurrentSetTime
  - power\_settings\_t, [64](#)
- DHCPEnabled
  - network\_settings\_t, [61](#)
- DRIVER\_TYPE\_EXTERNAL
  - ximc.h, [110](#)
- DeadZone
  - joystick\_settings\_t, [51](#)
- debug\_read\_t, [27](#)
  - DebugData, [27](#)
- debug\_write\_t, [27](#)
  - DebugData, [28](#)
- DebugData
  - debug\_read\_t, [27](#)
  - debug\_write\_t, [28](#)
- Decel
  - move\_settings\_calb\_t, [58](#)
  - move\_settings\_t, [60](#)
- DefaultGateway
  - network\_settings\_t, [61](#)
- DetentTorque
  - motor\_settings\_t, [55](#)
- device\_information\_t, [28](#)
  - Major, [29](#)
  - Minor, [29](#)
  - Release, [29](#)
- device\_network\_information\_t, [29](#)
- DriverType
  - entype\_settings\_t, [39](#)
- EEPROM\_PRECEDENCE
  - ximc.h, [110](#)
- ENC\_STATE\_ABSENT
  - ximc.h, [110](#)
- ENC\_STATE\_MALFUNC
  - ximc.h, [110](#)
- ENC\_STATE\_OK
  - ximc.h, [110](#)
- ENC\_STATE\_REVERS
  - ximc.h, [110](#)
- ENC\_STATE\_UNKNOWN
  - ximc.h, [110](#)
- ENDER\_SW1\_ACTIVE\_LOW
  - ximc.h, [111](#)
- ENDER\_SW2\_ACTIVE\_LOW
  - ximc.h, [111](#)
- ENDER\_SWAP
  - ximc.h, [111](#)
- ENGINE\_ACCEL\_ON
  - ximc.h, [111](#)
- ENGINE\_ANTIPLAY
  - ximc.h, [111](#)
- ENGINE\_LIMIT\_CURR
  - ximc.h, [111](#)
- ENGINE\_LIMIT\_RPM
  - ximc.h, [111](#)
- ENGINE\_LIMIT\_VOLT
  - ximc.h, [111](#)
- ENGINE\_MAX\_SPEED
  - ximc.h, [112](#)
- ENGINE\_REVERSE
  - ximc.h, [112](#)
- ENGINE\_TYPE\_2DC
  - ximc.h, [112](#)
- ENGINE\_TYPE\_DC
  - ximc.h, [112](#)

ENGINE\_TYPE\_NONE  
     ximc.h, [112](#)  
 ENGINE\_TYPE\_STEP  
     ximc.h, [112](#)  
 ENGINE\_TYPE\_TEST  
     ximc.h, [112](#)  
 ENUMERATE\_PROBE  
     ximc.h, [112](#)  
 EXTIO\_SETUP\_INVERT  
     ximc.h, [113](#)  
 EXTIO\_SETUP\_OUTPUT  
     ximc.h, [114](#)  
 EXTIOModeFlags  
     extio\_settings\_t, [40](#)  
 EXTIOSetupFlags  
     extio\_settings\_t, [40](#)  
 edges\_settings\_calb\_t, [29](#)  
     BorderFlags, [30](#)  
     EnderFlags, [30](#)  
     LeftBorder, [30](#)  
     RightBorder, [30](#)  
 edges\_settings\_t, [30](#)  
     BorderFlags, [31](#)  
     EnderFlags, [31](#)  
     LeftBorder, [31](#)  
     RightBorder, [31](#)  
     uLeftBorder, [31](#)  
     uRightBorder, [31](#)  
 Efficiency  
     gear\_settings\_t, [43](#)  
 emf\_settings\_t, [32](#)  
     BackEMFFlags, [32](#)  
     Km, [32](#)  
     L, [32](#)  
     R, [32](#)  
 Enc\_Check  
     analog\_data\_t, [16](#)  
 Enc\_Check\_ADC  
     analog\_data\_t, [16](#)  
 EncPosition  
     get\_position\_calb\_t, [44](#)  
     get\_position\_t, [45](#)  
     set\_position\_calb\_t, [67](#)  
     set\_position\_t, [68](#)  
     status\_calb\_t, [72](#)  
     status\_t, [75](#)  
 EncSts  
     status\_calb\_t, [73](#)  
     status\_t, [75](#)  
 encoder\_information\_t, [33](#)  
     Manufacturer, [33](#)  
     PartNumber, [33](#)  
 encoder\_settings\_t, [33](#)  
     EncoderSettings, [34](#)  
     MaxCurrentConsumption, [34](#)  
     MaxOperatingFrequency, [34](#)  
     SupplyVoltageMax, [34](#)  
     SupplyVoltageMin, [34](#)  
 EncoderSettings  
     encoder\_settings\_t, [34](#)  
 EnderFlags  
     edges\_settings\_calb\_t, [30](#)  
     edges\_settings\_t, [31](#)  
 engine\_advansed\_setup\_t, [34](#)  
     stepcloseloop\_Kp\_high, [35](#)  
     stepcloseloop\_Kp\_low, [35](#)  
     stepcloseloop\_Kw, [35](#)  
 engine\_settings\_calb\_t, [35](#)  
     Antiplay, [36](#)  
     EngineFlags, [36](#)  
     MicrostepMode, [36](#)  
     NomCurrent, [36](#)  
     NomSpeed, [36](#)  
     NomVoltage, [37](#)  
     StepsPerRev, [37](#)  
 engine\_settings\_t, [37](#)  
     Antiplay, [38](#)  
     EngineFlags, [38](#)  
     MicrostepMode, [38](#)  
     NomCurrent, [38](#)  
     NomSpeed, [38](#)  
     NomVoltage, [38](#)  
     StepsPerRev, [38](#)  
     uNomSpeed, [38](#)  
 EngineFlags  
     engine\_settings\_calb\_t, [36](#)  
     engine\_settings\_t, [38](#)  
 EngineType  
     entype\_settings\_t, [39](#)  
 entype\_settings\_t, [39](#)  
     DriverType, [39](#)  
     EngineType, [39](#)  
 enumerate\_devices  
     ximc.h, [132](#)  
 ExpFactor  
     joystick\_settings\_t, [52](#)  
 extended\_settings\_t, [39](#)  
 extio\_settings\_t, [40](#)  
     EXTIOModeFlags, [40](#)  
     EXTIOSetupFlags, [40](#)  
 FEEDBACK\_EMF  
     ximc.h, [114](#)  
 FEEDBACK\_ENC\_REVERSE  
     ximc.h, [114](#)  
 FEEDBACK\_ENCODER  
     ximc.h, [114](#)  
 FEEDBACK\_NONE  
     ximc.h, [115](#)  
 FastHome  
     home\_settings\_calb\_t, [48](#)  
     home\_settings\_t, [49](#)

- feedback\_settings\_t, [40](#)
  - CountsPerTurn, [41](#)
  - FeedbackFlags, [41](#)
  - FeedbackType, [41](#)
  - IPS, [41](#)
- FeedbackFlags
  - feedback\_settings\_t, [41](#)
- FeedbackType
  - feedback\_settings\_t, [41](#)
- Flags
  - control\_settings\_calb\_t, [23](#)
  - control\_settings\_t, [25](#)
  - secure\_settings\_t, [65](#)
  - status\_calb\_t, [73](#)
  - status\_t, [75](#)
- free\_enumerate\_devices
  - ximc.h, [134](#)
- FullCurrent
  - analog\_data\_t, [16](#)
- FullCurrent\_A
  - calibration\_settings\_t, [19](#)
- FullCurrent\_ADC
  - analog\_data\_t, [16](#)
- FullCurrent\_B
  - calibration\_settings\_t, [19](#)
- GPIOFlags
  - status\_calb\_t, [73](#)
  - status\_t, [75](#)
- gear\_information\_t, [41](#)
  - Manufacturer, [42](#)
  - PartNumber, [42](#)
- gear\_settings\_t, [42](#)
  - Efficiency, [43](#)
  - InputInertia, [43](#)
  - MaxOutputBacklash, [43](#)
  - RatedInputSpeed, [43](#)
  - RatedInputTorque, [43](#)
  - ReductionIn, [43](#)
  - ReductionOut, [43](#)
- get\_accessories\_settings
  - ximc.h, [134](#)
- get\_analog\_data
  - ximc.h, [134](#)
- get\_bootloader\_version
  - ximc.h, [134](#)
- get\_brake\_settings
  - ximc.h, [135](#)
- get\_calibration\_settings
  - ximc.h, [135](#)
- get\_chart\_data
  - ximc.h, [135](#)
- get\_control\_settings
  - ximc.h, [136](#)
- get\_control\_settings\_calb
  - ximc.h, [136](#)
- get\_controller\_name
  - ximc.h, [136](#)
- get\_ctp\_settings
  - ximc.h, [136](#)
- get\_debug\_read
  - ximc.h, [137](#)
- get\_device\_count
  - ximc.h, [137](#)
- get\_device\_information
  - ximc.h, [137](#)
- get\_device\_name
  - ximc.h, [137](#)
- get\_edges\_settings
  - ximc.h, [138](#)
- get\_edges\_settings\_calb
  - ximc.h, [138](#)
- get\_emf\_settings
  - ximc.h, [138](#)
- get\_encoder\_information
  - ximc.h, [139](#)
- get\_encoder\_settings
  - ximc.h, [139](#)
- get\_engine\_advansed\_setup
  - ximc.h, [139](#)
- get\_engine\_settings
  - ximc.h, [139](#)
- get\_engine\_settings\_calb
  - ximc.h, [140](#)
- get\_entype\_settings
  - ximc.h, [140](#)
- get\_enumerate\_device\_controller\_name
  - ximc.h, [140](#)
- get\_enumerate\_device\_information
  - ximc.h, [141](#)
- get\_enumerate\_device\_network\_information
  - ximc.h, [141](#)
- get\_enumerate\_device\_serial
  - ximc.h, [141](#)
- get\_enumerate\_device\_stage\_name
  - ximc.h, [142](#)
- get\_extended\_settings
  - ximc.h, [142](#)
- get\_extio\_settings
  - ximc.h, [142](#)
- get\_feedback\_settings
  - ximc.h, [142](#)
- get\_firmware\_version
  - ximc.h, [143](#)
- get\_gear\_information
  - ximc.h, [143](#)
- get\_gear\_settings
  - ximc.h, [143](#)
- get\_globally\_unique\_identifier
  - ximc.h, [143](#)
- get\_hallsensor\_information
  - ximc.h, [144](#)

get\_hallsensor\_settings  
     ximc.h, 144  
 get\_home\_settings  
     ximc.h, 144  
 get\_home\_settings\_calb  
     ximc.h, 144  
 get\_init\_random  
     ximc.h, 145  
 get\_joystick\_settings  
     ximc.h, 145  
 get\_measurements  
     ximc.h, 145  
 get\_motor\_information  
     ximc.h, 146  
 get\_motor\_settings  
     ximc.h, 146  
 get\_move\_settings  
     ximc.h, 146  
 get\_move\_settings\_calb  
     ximc.h, 146  
 get\_network\_settings  
     ximc.h, 147  
 get\_nonvolatile\_memory  
     ximc.h, 147  
 get\_password\_settings  
     ximc.h, 147  
 get\_pid\_settings  
     ximc.h, 147  
 get\_position  
     ximc.h, 148  
 get\_position\_calb  
     ximc.h, 148  
 get\_position\_calb\_t, 44  
     EncPosition, 44  
     Position, 44  
 get\_position\_t, 44  
     EncPosition, 45  
     uPosition, 45  
 get\_power\_settings  
     ximc.h, 148  
 get\_secure\_settings  
     ximc.h, 149  
 get\_serial\_number  
     ximc.h, 149  
 get\_stage\_information  
     ximc.h, 149  
 get\_stage\_name  
     ximc.h, 149  
 get\_stage\_settings  
     ximc.h, 149  
 get\_status  
     ximc.h, 150  
 get\_status\_calb  
     ximc.h, 150  
 get\_sync\_in\_settings  
     ximc.h, 150  
 get\_sync\_in\_settings\_calb  
     ximc.h, 151  
 get\_sync\_out\_settings  
     ximc.h, 151  
 get\_sync\_out\_settings\_calb  
     ximc.h, 151  
 get\_uart\_settings  
     ximc.h, 152  
 globally\_unique\_identifier\_t, 45  
     UniquelD0, 46  
     UniquelD1, 46  
     UniquelD2, 46  
     UniquelD3, 46  
 goto\_firmware  
     ximc.h, 152  
  
 HOME\_DIR\_FIRST  
     ximc.h, 115  
 HOME\_DIR\_SECOND  
     ximc.h, 115  
 HOME\_HALF\_MV  
     ximc.h, 115  
 HOME\_MV\_SEC\_EN  
     ximc.h, 115  
 HOME\_STOP\_FIRST\_LIM  
     ximc.h, 115  
 HOME\_STOP\_FIRST\_REV  
     ximc.h, 115  
 HOME\_STOP\_FIRST\_SYN  
     ximc.h, 115  
 HOME\_USE\_FAST  
     ximc.h, 116  
 hallsensor\_information\_t, 46  
     Manufacturer, 46  
     PartNumber, 46  
 hallsensor\_settings\_t, 47  
     MaxCurrentConsumption, 47  
     MaxOperatingFrequency, 47  
     SupplyVoltageMax, 47  
     SupplyVoltageMin, 47  
 has\_firmware  
     ximc.h, 152  
 HoldCurrent  
     power\_settings\_t, 64  
 home\_settings\_calb\_t, 48  
     FastHome, 48  
     HomeDelta, 48  
     HomeFlags, 48  
     SlowHome, 48  
 home\_settings\_t, 49  
     FastHome, 49  
     HomeDelta, 49  
     HomeFlags, 49  
     SlowHome, 49  
     uFastHome, 50  
     uHomeDelta, 50

- uSlowHome, [50](#)
- HomeDelta
  - home\_settings\_calb\_t, [48](#)
  - home\_settings\_t, [49](#)
- HomeFlags
  - home\_settings\_calb\_t, [48](#)
  - home\_settings\_t, [49](#)
- HorizontalLoadCapacity
  - stage\_settings\_t, [70](#)
- IPS
  - feedback\_settings\_t, [41](#)
- IPv4Address
  - network\_settings\_t, [61](#)
- init\_random\_t, [50](#)
  - key, [51](#)
- InputInertia
  - gear\_settings\_t, [43](#)
- lpwr
  - status\_calb\_t, [73](#)
  - status\_t, [75](#)
- lusb
  - status\_calb\_t, [73](#)
  - status\_t, [76](#)
- JOY\_REVERSE
  - ximc.h, [116](#)
- Joy
  - analog\_data\_t, [17](#)
  - chart\_data\_t, [22](#)
- Joy\_ADC
  - analog\_data\_t, [17](#)
- JoyCenter
  - joystick\_settings\_t, [52](#)
- JoyFlags
  - joystick\_settings\_t, [52](#)
- JoyHighEnd
  - joystick\_settings\_t, [52](#)
- JoyLowEnd
  - joystick\_settings\_t, [52](#)
- joystick\_settings\_t, [51](#)
  - DeadZone, [51](#)
  - ExpFactor, [52](#)
  - JoyCenter, [52](#)
  - JoyFlags, [52](#)
  - JoyHighEnd, [52](#)
  - JoyLowEnd, [52](#)
- Key
  - serial\_number\_t, [66](#)
- key
  - init\_random\_t, [51](#)
- Km
  - emf\_settings\_t, [32](#)
- L
  - emf\_settings\_t, [32](#)
- LOW\_UPWR\_PROTECTION
  - ximc.h, [116](#)
- LS\_SHORTED
  - ximc.h, [116](#)
- LeadScrewPitch
  - stage\_settings\_t, [70](#)
- LeftBorder
  - edges\_settings\_calb\_t, [30](#)
  - edges\_settings\_t, [31](#)
- Length
  - measurements\_t, [53](#)
- LimitSwitchesSettings
  - accessories\_settings\_t, [13](#)
- load\_correction\_table
  - ximc.h, [152](#)
- logging\_callback\_stderr\_narrow
  - ximc.h, [153](#)
- logging\_callback\_stderr\_wide
  - ximc.h, [153](#)
- logging\_callback\_t
  - ximc.h, [125](#)
- LowUpwrOff
  - secure\_settings\_t, [65](#)
- MBRatedCurrent
  - accessories\_settings\_t, [13](#)
- MBRatedVoltage
  - accessories\_settings\_t, [13](#)
- MBSSettings
  - accessories\_settings\_t, [13](#)
- MBTorque
  - accessories\_settings\_t, [13](#)
- MICROSTEP\_MODE\_FULL
  - ximc.h, [117](#)
- MOVE\_STATE\_ANTIPLAY
  - ximc.h, [117](#)
- MOVE\_STATE\_MOVING
  - ximc.h, [117](#)
- MVCMD\_ERROR
  - ximc.h, [117](#)
- MVCMD\_HOME
  - ximc.h, [117](#)
- MVCMD\_LEFT
  - ximc.h, [117](#)
- MVCMD\_LOFT
  - ximc.h, [117](#)
- MVCMD\_MOVE
  - ximc.h, [118](#)
- MVCMD\_MOVR
  - ximc.h, [118](#)
- MVCMD\_NAME\_BITS
  - ximc.h, [118](#)
- MVCMD\_RIGHT
  - ximc.h, [118](#)
- MVCMD\_RUNNING
  - ximc.h, [118](#)



- MVCMD\_SSTP
  - ximc.h, [118](#)
- MVCMD\_STOP
  - ximc.h, [118](#)
- MVCMD\_UKNWN
  - ximc.h, [118](#)
- MagneticBrakeInfo
  - accessories\_settings\_t, [13](#)
- Major
  - device\_information\_t, [29](#)
  - serial\_number\_t, [66](#)
- Manufacturer
  - encoder\_information\_t, [33](#)
  - gear\_information\_t, [42](#)
  - hallsensor\_information\_t, [46](#)
  - motor\_information\_t, [53](#)
  - stage\_information\_t, [69](#)
- MaxClickTime
  - control\_settings\_calb\_t, [23](#)
  - control\_settings\_t, [25](#)
- MaxCurrent
  - motor\_settings\_t, [55](#)
- MaxCurrentConsumption
  - encoder\_settings\_t, [34](#)
  - hallsensor\_settings\_t, [47](#)
  - stage\_settings\_t, [70](#)
- MaxCurrentTime
  - motor\_settings\_t, [55](#)
- MaxOperatingFrequency
  - encoder\_settings\_t, [34](#)
  - hallsensor\_settings\_t, [47](#)
- MaxOutputBacklash
  - gear\_settings\_t, [43](#)
- MaxSpeed
  - control\_settings\_calb\_t, [24](#)
  - control\_settings\_t, [25](#)
  - motor\_settings\_t, [55](#)
  - stage\_settings\_t, [70](#)
- measurements\_t, [52](#)
  - Length, [53](#)
- MechanicalTimeConstant
  - motor\_settings\_t, [55](#)
- MicrostepMode
  - calibration\_t, [21](#)
  - engine\_settings\_calb\_t, [36](#)
  - engine\_settings\_t, [38](#)
- MinimumUusb
  - secure\_settings\_t, [65](#)
- Minor
  - device\_information\_t, [29](#)
  - serial\_number\_t, [66](#)
- motor\_information\_t, [53](#)
  - Manufacturer, [53](#)
  - PartNumber, [53](#)
- motor\_settings\_t, [54](#)
  - DetentTorque, [55](#)
  - MaxCurrent, [55](#)
  - MaxCurrentTime, [55](#)
  - MaxSpeed, [55](#)
  - MechanicalTimeConstant, [55](#)
  - MotorType, [56](#)
  - NoLoadCurrent, [56](#)
  - NoLoadSpeed, [56](#)
  - NominalCurrent, [56](#)
  - NominalPower, [56](#)
  - NominalSpeed, [56](#)
  - NominalTorque, [56](#)
  - NominalVoltage, [56](#)
  - Phases, [56](#)
  - Poles, [57](#)
  - RotorInertia, [57](#)
  - SpeedConstant, [57](#)
  - SpeedTorqueGradient, [57](#)
  - StallTorque, [57](#)
  - TorqueConstant, [57](#)
  - WindingInductance, [57](#)
  - WindingResistance, [57](#)
- MotorType
  - motor\_settings\_t, [56](#)
- move\_settings\_calb\_t, [58](#)
  - Accel, [58](#)
  - AntiplaySpeed, [58](#)
  - Decel, [58](#)
  - MoveFlags, [58](#)
  - Speed, [59](#)
- move\_settings\_t, [59](#)
  - Accel, [59](#)
  - AntiplaySpeed, [59](#)
  - Decel, [60](#)
  - MoveFlags, [60](#)
  - Speed, [60](#)
  - uAntiplaySpeed, [60](#)
  - uSpeed, [60](#)
- MoveFlags
  - move\_settings\_calb\_t, [58](#)
  - move\_settings\_t, [60](#)
- MoveSts
  - status\_calb\_t, [73](#)
  - status\_t, [76](#)
- msec\_sleep
  - ximc.h, [153](#)
- MvCmdSts
  - status\_calb\_t, [73](#)
  - status\_t, [76](#)
- network\_settings\_t, [60](#)
  - DHCPEnabled, [61](#)
  - DefaultGateway, [61](#)
  - IPv4Address, [61](#)
  - SubnetMask, [61](#)
- NoLoadCurrent
  - motor\_settings\_t, [56](#)

- NoLoadSpeed
  - motor\_settings\_t, 56
- NomCurrent
  - engine\_settings\_calb\_t, 36
  - engine\_settings\_t, 38
- NomSpeed
  - engine\_settings\_calb\_t, 36
  - engine\_settings\_t, 38
- NomVoltage
  - engine\_settings\_calb\_t, 37
  - engine\_settings\_t, 38
- NominalCurrent
  - motor\_settings\_t, 56
- NominalPower
  - motor\_settings\_t, 56
- NominalSpeed
  - motor\_settings\_t, 56
- NominalTorque
  - motor\_settings\_t, 56
- NominalVoltage
  - motor\_settings\_t, 56
- nonvolatile\_memory\_t, 61
  - UserData, 62
- open\_device
  - ximc.h, 154
- POWER\_OFF\_ENABLED
  - ximc.h, 118
- POWER\_REDUCE\_ENABLED
  - ximc.h, 118
- POWER\_SMOOTH\_CURRENT
  - ximc.h, 118
- PWR\_STATE\_MAX
  - ximc.h, 119
- PWR\_STATE\_NORM
  - ximc.h, 119
- PWR\_STATE\_OFF
  - ximc.h, 119
- PWR\_STATE\_REDUCE
  - ximc.h, 119
- PWR\_STATE\_UNKNOWN
  - ximc.h, 119
- PWRSts
  - status\_calb\_t, 73
  - status\_t, 76
- PartNumber
  - encoder\_information\_t, 33
  - gear\_information\_t, 42
  - hallsensor\_information\_t, 46
  - motor\_information\_t, 53
  - stage\_information\_t, 69
- password\_settings\_t, 62
  - UserPassword, 62
- Phases
  - motor\_settings\_t, 56
- pid\_settings\_t, 62
- Poles
  - motor\_settings\_t, 57
- PosFlags
  - set\_position\_calb\_t, 67
  - set\_position\_t, 68
- Position
  - get\_position\_calb\_t, 44
  - set\_position\_calb\_t, 67
  - sync\_in\_settings\_calb\_t, 77
- PositionerName
  - stage\_name\_t, 69
- Pot
  - analog\_data\_t, 17
  - chart\_data\_t, 22
- power\_settings\_t, 63
  - CurrReductDelay, 64
  - CurrentSetTime, 64
  - HoldCurrent, 64
  - PowerFlags, 64
  - PowerOffDelay, 64
- PowerFlags
  - power\_settings\_t, 64
- PowerOffDelay
  - power\_settings\_t, 64
- probe\_device
  - ximc.h, 154
- R
  - emf\_settings\_t, 32
- REV\_SENS\_INV
  - ximc.h, 119
- RPM\_DIV\_1000
  - ximc.h, 119
- RatedInputSpeed
  - gear\_settings\_t, 43
- RatedInputTorque
  - gear\_settings\_t, 43
- ReductionIn
  - gear\_settings\_t, 43
- ReductionOut
  - gear\_settings\_t, 43
- Release
  - device\_information\_t, 29
  - serial\_number\_t, 66
- RightBorder
  - edges\_settings\_calb\_t, 30
  - edges\_settings\_t, 31
- RotorInertia
  - motor\_settings\_t, 57
- SN
  - serial\_number\_t, 66
- STATE\_ALARM
  - ximc.h, 119
- STATE\_BRAKE
  - ximc.h, 120
- STATE\_BUTTON\_LEFT

ximc.h, [120](#)  
STATE\_BUTTON\_RIGHT  
ximc.h, [120](#)  
STATE\_CONTR  
ximc.h, [120](#)  
STATE\_CTP\_ERROR  
ximc.h, [120](#)  
STATE\_DIG\_SIGNAL  
ximc.h, [120](#)  
STATE\_ENC\_A  
ximc.h, [120](#)  
STATE\_ENC\_B  
ximc.h, [121](#)  
STATE\_ERRC  
ximc.h, [121](#)  
STATE\_ERRD  
ximc.h, [121](#)  
STATE\_ERRV  
ximc.h, [121](#)  
STATE\_EXTIO\_ALARM  
ximc.h, [121](#)  
STATE\_GPIO\_LEVEL  
ximc.h, [121](#)  
STATE\_GPIO\_PINOUT  
ximc.h, [121](#)  
STATE\_IS\_HOMED  
ximc.h, [121](#)  
STATE\_LEFT\_EDGE  
ximc.h, [122](#)  
STATE\_POWER\_OVERHEAT  
ximc.h, [122](#)  
STATE\_REV\_SENSOR  
ximc.h, [122](#)  
STATE\_RIGHT\_EDGE  
ximc.h, [122](#)  
STATE\_SECUR  
ximc.h, [123](#)  
STATE\_SYNC\_INPUT  
ximc.h, [123](#)  
STATE\_SYNC\_OUTPUT  
ximc.h, [123](#)  
SYNCIN\_ENABLED  
ximc.h, [123](#)  
SYNCIN\_INVERT  
ximc.h, [123](#)  
SYNCOUT\_ENABLED  
ximc.h, [123](#)  
SYNCOUT\_IN\_STEPS  
ximc.h, [123](#)  
SYNCOUT\_INVERT  
ximc.h, [123](#)  
SYNCOUT\_ONPERIOD  
ximc.h, [123](#)  
SYNCOUT\_ONSTART  
ximc.h, [124](#)  
SYNCOUT\_ONSTOP  
ximc.h, [124](#)  
SYNCOUT\_STATE  
ximc.h, [124](#)  
secure\_settings\_t, [64](#)  
Criticalpwr, [65](#)  
Criticalusb, [65](#)  
CriticalUpwr, [65](#)  
CriticalUusb, [65](#)  
Flags, [65](#)  
LowUpwrOff, [65](#)  
MinimumUusb, [65](#)  
serial\_number\_t, [65](#)  
Key, [66](#)  
Major, [66](#)  
Minor, [66](#)  
Release, [66](#)  
SN, [66](#)  
service\_command\_updf  
ximc.h, [154](#)  
set\_accessories\_settings  
ximc.h, [154](#)  
set\_bindy\_key  
ximc.h, [155](#)  
set\_brake\_settings  
ximc.h, [155](#)  
set\_calibration\_settings  
ximc.h, [155](#)  
set\_control\_settings  
ximc.h, [155](#)  
set\_control\_settings\_calb  
ximc.h, [156](#)  
set\_controller\_name  
ximc.h, [156](#)  
set\_correction\_table  
ximc.h, [156](#)  
set\_ctp\_settings  
ximc.h, [157](#)  
set\_debug\_write  
ximc.h, [157](#)  
set\_edges\_settings  
ximc.h, [158](#)  
set\_edges\_settings\_calb  
ximc.h, [158](#)  
set\_emf\_settings  
ximc.h, [158](#)  
set\_encoder\_information  
ximc.h, [159](#)  
set\_encoder\_settings  
ximc.h, [159](#)  
set\_engine\_advansed\_setup  
ximc.h, [159](#)  
set\_engine\_settings  
ximc.h, [159](#)  
set\_engine\_settings\_calb  
ximc.h, [160](#)  
set\_entype\_settings

- ximc.h, [160](#)
- set\_extended\_settings
  - ximc.h, [160](#)
- set\_extio\_settings
  - ximc.h, [161](#)
- set\_feedback\_settings
  - ximc.h, [161](#)
- set\_gear\_information
  - ximc.h, [161](#)
- set\_gear\_settings
  - ximc.h, [162](#)
- set\_hallsensor\_information
  - ximc.h, [162](#)
- set\_hallsensor\_settings
  - ximc.h, [162](#)
- set\_home\_settings
  - ximc.h, [162](#)
- set\_home\_settings\_calb
  - ximc.h, [163](#)
- set\_joystick\_settings
  - ximc.h, [163](#)
- set\_logging\_callback
  - ximc.h, [163](#)
- set\_motor\_information
  - ximc.h, [164](#)
- set\_motor\_settings
  - ximc.h, [164](#)
- set\_move\_settings
  - ximc.h, [164](#)
- set\_move\_settings\_calb
  - ximc.h, [164](#)
- set\_network\_settings
  - ximc.h, [165](#)
- set\_nonvolatile\_memory
  - ximc.h, [165](#)
- set\_password\_settings
  - ximc.h, [165](#)
- set\_pid\_settings
  - ximc.h, [165](#)
- set\_position
  - ximc.h, [166](#)
- set\_position\_calb
  - ximc.h, [166](#)
- set\_position\_calb\_t, [66](#)
  - EncPosition, [67](#)
  - PosFlags, [67](#)
  - Position, [67](#)
- set\_position\_t, [67](#)
  - EncPosition, [68](#)
  - PosFlags, [68](#)
  - uPosition, [68](#)
- set\_power\_settings
  - ximc.h, [166](#)
- set\_secure\_settings
  - ximc.h, [166](#)
- set\_serial\_number
  - ximc.h, [167](#)
- set\_stage\_information
  - ximc.h, [167](#)
- set\_stage\_name
  - ximc.h, [167](#)
- set\_stage\_settings
  - ximc.h, [167](#)
- set\_sync\_in\_settings
  - ximc.h, [167](#)
- set\_sync\_in\_settings\_calb
  - ximc.h, [168](#)
- set\_sync\_out\_settings
  - ximc.h, [168](#)
- set\_sync\_out\_settings\_calb
  - ximc.h, [168](#)
- set\_uart\_settings
  - ximc.h, [169](#)
- SlowHome
  - home\_settings\_calb\_t, [48](#)
  - home\_settings\_t, [49](#)
- Speed
  - move\_settings\_calb\_t, [59](#)
  - move\_settings\_t, [60](#)
  - sync\_in\_settings\_calb\_t, [77](#)
  - sync\_in\_settings\_t, [78](#)
- SpeedConstant
  - motor\_settings\_t, [57](#)
- SpeedTorqueGradient
  - motor\_settings\_t, [57](#)
- stage\_information\_t, [68](#)
  - Manufacturer, [69](#)
  - PartNumber, [69](#)
- stage\_name\_t, [69](#)
  - PositionerName, [69](#)
- stage\_settings\_t, [69](#)
  - HorizontalLoadCapacity, [70](#)
  - LeadScrewPitch, [70](#)
  - MaxCurrentConsumption, [70](#)
  - MaxSpeed, [70](#)
  - SupplyVoltageMax, [70](#)
  - SupplyVoltageMin, [71](#)
  - TravelRange, [71](#)
  - Units, [71](#)
  - VerticalLoadCapacity, [71](#)
- StallTorque
  - motor\_settings\_t, [57](#)
- status\_calb\_t, [71](#)
  - CmdBufFreeSpace, [72](#)
  - CurPosition, [72](#)
  - CurSpeed, [72](#)
  - CurT, [72](#)
  - EncPosition, [72](#)
  - EncSts, [73](#)
  - Flags, [73](#)
  - GPIOFlags, [73](#)
  - lpwr, [73](#)

- lusb, [73](#)
  - MoveSts, [73](#)
  - MvCmdSts, [73](#)
  - PWRSts, [73](#)
  - Upwr, [73](#)
  - Uusb, [73](#)
  - WindSts, [73](#)
- status\_t, [74](#)
  - CmdBufFreeSpace, [75](#)
  - CurPosition, [75](#)
  - CurSpeed, [75](#)
  - CurT, [75](#)
  - EncPosition, [75](#)
  - EncSts, [75](#)
  - Flags, [75](#)
  - GPIOFlags, [75](#)
  - lpwr, [75](#)
  - lusb, [76](#)
  - MoveSts, [76](#)
  - MvCmdSts, [76](#)
  - PWRSts, [76](#)
  - uCurPosition, [76](#)
  - uCurSpeed, [76](#)
  - Upwr, [76](#)
  - Uusb, [76](#)
  - WindSts, [76](#)
- stepcloseloop\_Kp\_high
  - engine\_advansed\_setup\_t, [35](#)
- stepcloseloop\_Kp\_low
  - engine\_advansed\_setup\_t, [35](#)
- stepcloseloop\_Kw
  - engine\_advansed\_setup\_t, [35](#)
- StepsPerRev
  - engine\_settings\_calb\_t, [37](#)
  - engine\_settings\_t, [38](#)
- SubnetMask
  - network\_settings\_t, [61](#)
- SupVoltage
  - analog\_data\_t, [17](#)
- SupVoltage\_ADC
  - analog\_data\_t, [17](#)
- SupplyVoltageMax
  - encoder\_settings\_t, [34](#)
  - hallsensor\_settings\_t, [47](#)
  - stage\_settings\_t, [70](#)
- SupplyVoltageMin
  - encoder\_settings\_t, [34](#)
  - hallsensor\_settings\_t, [47](#)
  - stage\_settings\_t, [71](#)
- sync\_in\_settings\_calb\_t, [77](#)
  - ClutterTime, [77](#)
  - Position, [77](#)
  - Speed, [77](#)
  - SyncInFlags, [77](#)
- sync\_in\_settings\_t, [77](#)
  - ClutterTime, [78](#)
  - Speed, [78](#)
  - SyncInFlags, [78](#)
  - uPosition, [78](#)
  - uSpeed, [78](#)
- sync\_out\_settings\_calb\_t, [79](#)
  - Accuracy, [79](#)
  - SyncOutFlags, [79](#)
  - SyncOutPeriod, [79](#)
  - SyncOutPulseSteps, [79](#)
- sync\_out\_settings\_t, [80](#)
  - Accuracy, [80](#)
  - SyncOutFlags, [80](#)
  - SyncOutPeriod, [80](#)
  - SyncOutPulseSteps, [81](#)
  - uAccuracy, [81](#)
- SyncInFlags
  - sync\_in\_settings\_calb\_t, [77](#)
  - sync\_in\_settings\_t, [78](#)
- SyncOutFlags
  - sync\_out\_settings\_calb\_t, [79](#)
  - sync\_out\_settings\_t, [80](#)
- SyncOutPeriod
  - sync\_out\_settings\_calb\_t, [79](#)
  - sync\_out\_settings\_t, [80](#)
- SyncOutPulseSteps
  - sync\_out\_settings\_calb\_t, [79](#)
  - sync\_out\_settings\_t, [81](#)
- t1
  - brake\_settings\_t, [18](#)
- t2
  - brake\_settings\_t, [18](#)
- t3
  - brake\_settings\_t, [18](#)
- t4
  - brake\_settings\_t, [18](#)
- TS\_TYPE\_BITS
  - ximc.h, [124](#)
- TSGrad
  - accessories\_settings\_t, [13](#)
- TSMMax
  - accessories\_settings\_t, [13](#)
- TSMIn
  - accessories\_settings\_t, [13](#)
- TSSettings
  - accessories\_settings\_t, [14](#)
- Temp
  - analog\_data\_t, [17](#)
- Temp\_ADC
  - analog\_data\_t, [17](#)
- TemperatureSensorInfo
  - accessories\_settings\_t, [13](#)
- Timeout
  - control\_settings\_calb\_t, [24](#)
  - control\_settings\_t, [25](#)
- TorqueConstant

- motor\_settings\_t, 57
- TravelRange
  - stage\_settings\_t, 71
- UART\_PARITY\_BITS
  - ximc.h, 124
- UARTSetupFlags
  - uart\_settings\_t, 81
- uAccuracy
  - sync\_out\_settings\_t, 81
- uAntiplaySpeed
  - move\_settings\_t, 60
- uCurPosition
  - status\_t, 76
- uCurSpeed
  - status\_t, 76
- uDeltaPosition
  - control\_settings\_t, 25
- uFastHome
  - home\_settings\_t, 50
- uHomeDelta
  - home\_settings\_t, 50
- uLeftBorder
  - edges\_settings\_t, 31
- uMaxSpeed
  - control\_settings\_t, 25
- uNomSpeed
  - engine\_settings\_t, 38
- uPosition
  - get\_position\_t, 45
  - set\_position\_t, 68
  - sync\_in\_settings\_t, 78
- uRightBorder
  - edges\_settings\_t, 31
- uSlowHome
  - home\_settings\_t, 50
- uSpeed
  - move\_settings\_t, 60
  - sync\_in\_settings\_t, 78
- uart\_settings\_t, 81
  - UARTSetupFlags, 81
- UniquelD0
  - globally\_unique\_identifier\_t, 46
- UniquelD1
  - globally\_unique\_identifier\_t, 46
- UniquelD2
  - globally\_unique\_identifier\_t, 46
- UniquelD3
  - globally\_unique\_identifier\_t, 46
- Units
  - stage\_settings\_t, 71
- Upwr
  - status\_calb\_t, 73
  - status\_t, 76
- UserData
  - nonvolatile\_memory\_t, 62
- UserPassword
  - password\_settings\_t, 62
- Uusb
  - status\_calb\_t, 73
  - status\_t, 76
- VerticalLoadCapacity
  - stage\_settings\_t, 71
- WIND\_A\_STATE\_ABSENT
  - ximc.h, 124
- WIND\_A\_STATE\_OK
  - ximc.h, 124
- WIND\_B\_STATE\_ABSENT
  - ximc.h, 124
- WIND\_B\_STATE\_OK
  - ximc.h, 124
- WindSts
  - status\_calb\_t, 73
  - status\_t, 76
- WindingCurrentA
  - chart\_data\_t, 22
- WindingCurrentB
  - chart\_data\_t, 22
- WindingCurrentC
  - chart\_data\_t, 22
- WindingInductance
  - motor\_settings\_t, 57
- WindingResistance
  - motor\_settings\_t, 57
- WindingVoltageA
  - chart\_data\_t, 22
- WindingVoltageB
  - chart\_data\_t, 22
- WindingVoltageC
  - chart\_data\_t, 23
- write\_key
  - ximc.h, 169
- XIMC\_API
  - ximc.h, 125
- ximc.h, 82
  - BACK\_EMF\_KM\_AUTO, 108
  - BORDER\_IS\_ENCODER, 108
  - BORDER\_STOP\_LEFT, 108
  - BORDER\_STOP\_RIGHT, 108
  - BRAKE\_ENABLED, 109
  - BRAKE\_ENG\_PWROFF, 109
  - CONTROL\_MODE\_BITS, 109
  - CONTROL\_MODE\_JOY, 109
  - CONTROL\_MODE\_LR, 109
  - CONTROL\_MODE\_OFF, 109
  - CTP\_ALARM\_ON\_ERROR, 109
  - CTP\_BASE, 109
  - CTP\_ENABLED, 110
  - calibration\_t, 125
  - close\_device, 126

- command\_clear\_fram, 126
- command\_eeread\_settings, 126
- command\_eesave\_settings, 126
- command\_home, 127
- command\_homezero, 127
- command\_left, 127
- command\_loft, 128
- command\_move, 128
- command\_move\_calb, 128
- command\_movr, 128
- command\_movr\_calb, 129
- command\_power\_off, 129
- command\_read\_robust\_settings, 129
- command\_read\_settings, 130
- command\_reset, 130
- command\_right, 130
- command\_save\_robust\_settings, 130
- command\_save\_settings, 130
- command\_sstp, 131
- command\_start\_measurements, 131
- command\_stop, 131
- command\_update\_firmware, 131
- command\_wait\_for\_stop, 132
- command\_zero, 132
- EEPROM\_PRECEDENCE, 110
- ENC\_STATE\_ABSENT, 110
- ENC\_STATE\_MALFUNC, 110
- ENC\_STATE\_OK, 110
- ENC\_STATE\_REVERS, 110
- ENC\_STATE\_UNKNOWN, 110
- ENDER\_SWAP, 111
- ENGINE\_ACCEL\_ON, 111
- ENGINE\_ANTIPLAY, 111
- ENGINE\_LIMIT\_CURR, 111
- ENGINE\_LIMIT\_RPM, 111
- ENGINE\_LIMIT\_VOLT, 111
- ENGINE\_MAX\_SPEED, 112
- ENGINE\_REVERSE, 112
- ENGINE\_TYPE\_2DC, 112
- ENGINE\_TYPE\_DC, 112
- ENGINE\_TYPE\_NONE, 112
- ENGINE\_TYPE\_STEP, 112
- ENGINE\_TYPE\_TEST, 112
- ENUMERATE\_PROBE, 112
- EXTIO\_SETUP\_INVERT, 113
- EXTIO\_SETUP\_OUTPUT, 114
- enumerate\_devices, 132
- FEEDBACK\_EMF, 114
- FEEDBACK\_ENCODER, 114
- FEEDBACK\_NONE, 115
- free\_enumerate\_devices, 134
- get\_accessories\_settings, 134
- get\_analog\_data, 134
- get\_bootloader\_version, 134
- get\_brake\_settings, 135
- get\_calibration\_settings, 135
- get\_chart\_data, 135
- get\_control\_settings, 136
- get\_control\_settings\_calb, 136
- get\_controller\_name, 136
- get\_ctp\_settings, 136
- get\_debug\_read, 137
- get\_device\_count, 137
- get\_device\_information, 137
- get\_device\_name, 137
- get\_edges\_settings, 138
- get\_edges\_settings\_calb, 138
- get\_emf\_settings, 138
- get\_encoder\_information, 139
- get\_encoder\_settings, 139
- get\_engine\_advanced\_setup, 139
- get\_engine\_settings, 139
- get\_engine\_settings\_calb, 140
- get\_entype\_settings, 140
- get\_enumerate\_device\_controller\_name, 140
- get\_enumerate\_device\_information, 141
- get\_enumerate\_device\_network\_information, 141
- get\_enumerate\_device\_serial, 141
- get\_enumerate\_device\_stage\_name, 142
- get\_extended\_settings, 142
- get\_extio\_settings, 142
- get\_feedback\_settings, 142
- get\_firmware\_version, 143
- get\_gear\_information, 143
- get\_gear\_settings, 143
- get\_globally\_unique\_identifier, 143
- get\_hallsensor\_information, 144
- get\_hallsensor\_settings, 144
- get\_home\_settings, 144
- get\_home\_settings\_calb, 144
- get\_init\_random, 145
- get\_joystick\_settings, 145
- get\_measurements, 145
- get\_motor\_information, 146
- get\_motor\_settings, 146
- get\_move\_settings, 146
- get\_move\_settings\_calb, 146
- get\_network\_settings, 147
- get\_nonvolatile\_memory, 147
- get\_password\_settings, 147
- get\_pid\_settings, 147
- get\_position, 148
- get\_position\_calb, 148
- get\_power\_settings, 148
- get\_secure\_settings, 149
- get\_serial\_number, 149
- get\_stage\_information, 149
- get\_stage\_name, 149
- get\_stage\_settings, 149
- get\_status, 150
- get\_status\_calb, 150

- get\_sync\_in\_settings, 150
- get\_sync\_in\_settings\_calb, 151
- get\_sync\_out\_settings, 151
- get\_sync\_out\_settings\_calb, 151
- get\_uart\_settings, 152
- goto\_firmware, 152
- HOME\_DIR\_FIRST, 115
- HOME\_DIR\_SECOND, 115
- HOME\_HALF\_MV, 115
- HOME\_MV\_SEC\_EN, 115
- HOME\_USE\_FAST, 116
- has\_firmware, 152
- JOY\_REVERSE, 116
- LOW\_UPWR\_PROTECTION, 116
- LS\_SHORTED, 116
- load\_correction\_table, 152
- logging\_callback\_stderr\_narrow, 153
- logging\_callback\_stderr\_wide, 153
- logging\_callback\_t, 125
- MICROSTEP\_MODE\_FULL, 117
- MOVE\_STATE\_ANTIPLAY, 117
- MOVE\_STATE\_MOVING, 117
- MVCMD\_ERROR, 117
- MVCMD\_HOME, 117
- MVCMD\_LEFT, 117
- MVCMD\_LOFT, 117
- MVCMD\_MOVE, 118
- MVCMD\_MOVR, 118
- MVCMD\_NAME\_BITS, 118
- MVCMD\_RIGHT, 118
- MVCMD\_RUNNING, 118
- MVCMD\_SSTP, 118
- MVCMD\_STOP, 118
- MVCMD\_UKNWN, 118
- msec\_sleep, 153
- open\_device, 154
- POWER\_OFF\_ENABLED, 118
- PWR\_STATE\_MAX, 119
- PWR\_STATE\_NORM, 119
- PWR\_STATE\_OFF, 119
- PWR\_STATE\_REDUCT, 119
- PWR\_STATE\_UNKNOWN, 119
- probe\_device, 154
- REV\_SENS\_INV, 119
- RPM\_DIV\_1000, 119
- STATE\_ALARM, 119
- STATE\_BRAKE, 120
- STATE\_BUTTON\_LEFT, 120
- STATE\_BUTTON\_RIGHT, 120
- STATE\_CONTR, 120
- STATE\_CTP\_ERROR, 120
- STATE\_DIG\_SIGNAL, 120
- STATE\_ENC\_A, 120
- STATE\_ENC\_B, 121
- STATE\_ERRC, 121
- STATE\_ERRD, 121
- STATE\_ERRV, 121
- STATE\_EXTIO\_ALARM, 121
- STATE\_GPIO\_LEVEL, 121
- STATE\_GPIO\_PINOUT, 121
- STATE\_IS\_HOMED, 121
- STATE\_LEFT\_EDGE, 122
- STATE\_REV\_SENSOR, 122
- STATE\_RIGHT\_EDGE, 122
- STATE\_SECUR, 123
- STATE\_SYNC\_INPUT, 123
- STATE\_SYNC\_OUTPUT, 123
- SYNCIN\_ENABLED, 123
- SYNCIN\_INVERT, 123
- SYNCOUT\_ENABLED, 123
- SYNCOUT\_IN\_STEPS, 123
- SYNCOUT\_INVERT, 123
- SYNCOUT\_ONPERIOD, 123
- SYNCOUT\_ONSTART, 124
- SYNCOUT\_ONSTOP, 124
- SYNCOUT\_STATE, 124
- service\_command\_updf, 154
- set\_accessories\_settings, 154
- set\_bindy\_key, 155
- set\_brake\_settings, 155
- set\_calibration\_settings, 155
- set\_control\_settings, 155
- set\_control\_settings\_calb, 156
- set\_controller\_name, 156
- set\_correction\_table, 156
- set\_ctp\_settings, 157
- set\_debug\_write, 157
- set\_edges\_settings, 158
- set\_edges\_settings\_calb, 158
- set\_emf\_settings, 158
- set\_encoder\_information, 159
- set\_encoder\_settings, 159
- set\_engine\_advanced\_setup, 159
- set\_engine\_settings, 159
- set\_engine\_settings\_calb, 160
- set\_entype\_settings, 160
- set\_extended\_settings, 160
- set\_extio\_settings, 161
- set\_feedback\_settings, 161
- set\_gear\_information, 161
- set\_gear\_settings, 162
- set\_hallsensor\_information, 162
- set\_hallsensor\_settings, 162
- set\_home\_settings, 162
- set\_home\_settings\_calb, 163
- set\_joystick\_settings, 163
- set\_logging\_callback, 163
- set\_motor\_information, 164
- set\_motor\_settings, 164
- set\_move\_settings, 164
- set\_move\_settings\_calb, 164
- set\_network\_settings, 165



- set\_nonvolatile\_memory, [165](#)
- set\_password\_settings, [165](#)
- set\_pid\_settings, [165](#)
- set\_position, [166](#)
- set\_position\_calb, [166](#)
- set\_power\_settings, [166](#)
- set\_secure\_settings, [166](#)
- set\_serial\_number, [167](#)
- set\_stage\_information, [167](#)
- set\_stage\_name, [167](#)
- set\_stage\_settings, [167](#)
- set\_sync\_in\_settings, [167](#)
- set\_sync\_in\_settings\_calb, [168](#)
- set\_sync\_out\_settings, [168](#)
- set\_sync\_out\_settings\_calb, [168](#)
- set\_uart\_settings, [169](#)
- TS\_TYPE\_BITS, [124](#)
- UART\_PARITY\_BITS, [124](#)
- WIND\_A\_STATE\_OK, [124](#)
- WIND\_B\_STATE\_OK, [124](#)
- write\_key, [169](#)
- XIMC\_API, [125](#)
- ximc\_fix\_usbser\_sys, [169](#)
- ximc\_version, [169](#)
- ximc\_fix\_usbser\_sys
  - ximc.h, [169](#)
- ximc\_version
  - ximc.h, [169](#)