

libxime
2.12.5

Создано системой Doxygen 1.8.2

Пт 23 Окт 2020 14:09:11

Оглавление

1	Библиотека libximc	1
1.1	Что делает контроллер 8SMC4-USB и 8SMC5-USB.	1
1.2	Что умеет библиотека libximc.	1
1.3	Содействие.	2
2	Введение	3
2.1	О библиотеке	3
2.2	Требования к установленному программному обеспечению	3
2.2.1	Для сборки библиотеки	3
2.2.2	Для использования библиотеки	4
3	Как пересобрать библиотеку	5
3.1	Сборка для UNIX	5
3.2	Сборка для Linux на основе Debian	5
3.3	Сборка для Linux на основе RedHat	5
3.4	Сборка для Mac OS X	6
3.5	Сборка в ОС Windows	6
3.6	Доступ к исходным кодам	6
4	Как использовать с...	7
4.1	Использование на C	7
4.1.1	Visual C++	7
4.1.2	CodeBlocks	7
4.1.3	MinGW	8
4.1.4	C++ Builder	8
4.1.5	XCode	8
4.1.6	GCC	8
4.2	.NET	9
4.3	Delphi	9
4.4	Java	9
4.5	Python	10
4.6	MATLAB	10

4.7	Логирование в файл	11
4.8	Требуемые права доступа	11
4.9	Си-профили	11
5	Работа с пользовательскими единицами	12
5.1	Структура пересчета единиц <code>calibration_t</code>	12
5.2	Функции дублиеры для работы с пользовательскими единицами и структуры данных для них	12
5.3	Таблица коррекции координат для более точного позиционирования	13
6	Структуры данных	14
6.1	Структура <code>accessories_settings_t</code>	14
6.1.1	Подробное описание	14
6.1.2	Поля	15
6.1.2.1	<code>LimitSwitchesSettings</code>	15
6.1.2.2	<code>MagneticBrakeInfo</code>	15
6.1.2.3	<code>MBRatedCurrent</code>	15
6.1.2.4	<code>MBRatedVoltage</code>	15
6.1.2.5	<code>MBSettings</code>	15
6.1.2.6	<code>MBTorque</code>	15
6.1.2.7	<code>TemperatureSensorInfo</code>	15
6.1.2.8	<code>TSGrad</code>	15
6.1.2.9	<code>TSMax</code>	15
6.1.2.10	<code>TSTMin</code>	15
6.1.2.11	<code>TSSettings</code>	15
6.2	Структура <code>analog_data_t</code>	16
6.2.1	Подробное описание	17
6.2.2	Поля	17
6.2.2.1	<code>A1Voltage</code>	17
6.2.2.2	<code>A1Voltage_ADC</code>	17
6.2.2.3	<code>A2Voltage</code>	17
6.2.2.4	<code>A2Voltage_ADC</code>	17
6.2.2.5	<code>ACurrent</code>	17
6.2.2.6	<code>ACurrent_ADC</code>	17
6.2.2.7	<code>B1Voltage</code>	18
6.2.2.8	<code>B1Voltage_ADC</code>	18
6.2.2.9	<code>B2Voltage</code>	18
6.2.2.10	<code>B2Voltage_ADC</code>	18
6.2.2.11	<code>BCurrent</code>	18
6.2.2.12	<code>BCurrent_ADC</code>	18
6.2.2.13	<code>FullCurrent</code>	18

6.2.2.14	FullCurrent_ADC	18
6.2.2.15	H5	18
6.2.2.16	Joy	18
6.2.2.17	Joy_ADC	18
6.2.2.18	L5	18
6.2.2.19	L5_ADC	19
6.2.2.20	Pot	19
6.2.2.21	SupVoltage	19
6.2.2.22	SupVoltage_ADC	19
6.2.2.23	Temp	19
6.2.2.24	Temp_ADC	19
6.3	Структура brake_settings_t	19
6.3.1	Подробное описание	19
6.3.2	Поля	20
6.3.2.1	BrakeFlags	20
6.3.2.2	t1	20
6.3.2.3	t2	20
6.3.2.4	t3	20
6.3.2.5	t4	20
6.4	Структура calibration_settings_t	20
6.4.1	Подробное описание	21
6.4.2	Поля	21
6.4.2.1	CSS1_A	21
6.4.2.2	CSS1_B	21
6.4.2.3	CSS2_A	21
6.4.2.4	CSS2_B	21
6.4.2.5	FullCurrent_A	21
6.4.2.6	FullCurrent_B	21
6.5	Структура calibration_t	21
6.5.1	Подробное описание	21
6.6	Структура chart_data_t	22
6.6.1	Подробное описание	22
6.6.2	Поля	22
6.6.2.1	DutyCycle	22
6.6.2.2	Joy	22
6.6.2.3	Pot	23
6.6.2.4	WindingCurrentA	23
6.6.2.5	WindingCurrentB	23
6.6.2.6	WindingCurrentC	23
6.6.2.7	WindingVoltageA	23

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	24
6.7.2	Поля	24
6.7.2.1	Flags	24
6.7.2.2	MaxClickTime	24
6.7.2.3	MaxSpeed	24
6.7.2.4	Timeout	24
6.8	Структура control_settings_t	24
6.8.1	Подробное описание	25
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	26
6.9	Структура controller_name_t	26
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 ctp_settings_t	26
6.10.1	Подробное описание	27
6.10.2	Поля	27
6.10.2.1	CTPFlags	27
6.10.2.2	CTPMinError	27
6.11	Структура debug_read_t	27
6.11.1	Подробное описание	27
6.11.2	Поля	28
6.11.2.1	DebugData	28
6.12	Структура debug_write_t	28
6.12.1	Подробное описание	28
6.12.2	Поля	28
6.12.2.1	DebugData	28
6.13	Структура device_information_t	28
6.13.1	Подробное описание	29
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>	30
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 <code>edges_settings_t</code>	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 <code>emf_settings_t</code>	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	32
6.18	Структура <code>encoder_information_t</code>	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 <code>encoder_settings_t</code>	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 <code>engine_advansed_setup_t</code>	34
6.20.1 Подробное описание	35
6.20.2 Поля	35
6.20.2.1 <code>stepcloseloop_Kp_high</code>	35
6.20.2.2 <code>stepcloseloop_Kp_low</code>	35
6.20.2.3 <code>stepcloseloop_Kw</code>	35
6.21 Структура <code>engine_settings_calb_t</code>	35
6.21.1 Подробное описание	36
6.21.2 Поля	36
6.21.2.1 <code>Antiplay</code>	36
6.21.2.2 <code>EngineFlags</code>	36
6.21.2.3 <code>MicrostepMode</code>	36
6.21.2.4 <code>NomCurrent</code>	36
6.21.2.5 <code>NomSpeed</code>	36
6.21.2.6 <code>NomVoltage</code>	36
6.21.2.7 <code>StepsPerRev</code>	36
6.22 Структура <code>engine_settings_t</code>	37
6.22.1 Подробное описание	37
6.22.2 Поля	37
6.22.2.1 <code>Antiplay</code>	37
6.22.2.2 <code>EngineFlags</code>	37
6.22.2.3 <code>MicrostepMode</code>	38
6.22.2.4 <code>NomCurrent</code>	38
6.22.2.5 <code>NomSpeed</code>	38
6.22.2.6 <code>NomVoltage</code>	38
6.22.2.7 <code>StepsPerRev</code>	38
6.22.2.8 <code>uNomSpeed</code>	38
6.23 Структура <code>entype_settings_t</code>	38
6.23.1 Подробное описание	38
6.23.2 Поля	39
6.23.2.1 <code>DriverType</code>	39
6.23.2.2 <code>EngineType</code>	39
6.24 Структура <code>extended_settings_t</code>	39
6.24.1 Подробное описание	39
6.25 Структура <code>extio_settings_t</code>	39
6.25.1 Подробное описание	40
6.25.2 Поля	40
6.25.2.1 <code>EXTIOModeFlags</code>	40
6.25.2.2 <code>EXTIOSetupFlags</code>	40
6.26 Структура <code>feedback_settings_t</code>	40

6.26.1	Подробное описание	40
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	41
6.27.2	Поля	41
6.27.2.1	Manufacturer	41
6.27.2.2	PartNumber	42
6.28	Структура gear_settings_t	42
6.28.1	Подробное описание	42
6.28.2	Поля	42
6.28.2.1	Efficiency	42
6.28.2.2	InputInertia	42
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	43
6.29.1	Подробное описание	43
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	44
6.30.2	Поля	44
6.30.2.1	EncPosition	44
6.30.2.2	uPosition	44
6.31	Структура globally_unique_idenfier_t	44
6.31.1	Подробное описание	45
6.31.2	Поля	45
6.31.2.1	UniqueID0	45
6.31.2.2	UniqueID1	45
6.31.2.3	UniqueID2	45
6.31.2.4	UniqueID3	45
6.32	Структура hallsensor_information_t	45
6.32.1	Подробное описание	46

6.32.2 Поля	46
6.32.2.1 Manufacturer	46
6.32.2.2 PartNumber	46
6.33 Структура hallsensor_settings_t	46
6.33.1 Подробное описание	46
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	47
6.34 Структура home_settings_calb_t	47
6.34.1 Подробное описание	47
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	48
6.35.1 Подробное описание	48
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	49
6.35.2.5 uFastHome	49
6.35.2.6 uHomeDelta	49
6.35.2.7 uSlowHome	49
6.36 Структура init_random_t	49
6.36.1 Подробное описание	50
6.36.2 Поля	50
6.36.2.1 key	50
6.37 Структура joystick_settings_t	50
6.37.1 Подробное описание	50
6.37.2 Поля	51
6.37.2.1 DeadZone	51
6.37.2.2 ExpFactor	51
6.37.2.3 JoyCenter	51
6.37.2.4 JoyFlags	51
6.37.2.5 JoyHighEnd	51
6.37.2.6 JoyLowEnd	51

6.38 Структура <code>measurements_t</code>	51
6.38.1 Подробное описание	52
6.38.2 Поля	52
6.38.2.1 <code>Error</code>	52
6.38.2.2 <code>Length</code>	52
6.38.2.3 <code>Speed</code>	52
6.39 Структура <code>motor_information_t</code>	52
6.39.1 Подробное описание	52
6.39.2 Поля	52
6.39.2.1 <code>Manufacturer</code>	52
6.39.2.2 <code>PartNumber</code>	53
6.40 Структура <code>motor_settings_t</code>	53
6.40.1 Подробное описание	54
6.40.2 Поля	54
6.40.2.1 <code>DetentTorque</code>	54
6.40.2.2 <code>MaxCurrent</code>	54
6.40.2.3 <code>MaxCurrentTime</code>	54
6.40.2.4 <code>MaxSpeed</code>	54
6.40.2.5 <code>MechanicalTimeConstant</code>	54
6.40.2.6 <code>MotorType</code>	55
6.40.2.7 <code>NoLoadCurrent</code>	55
6.40.2.8 <code>NoLoadSpeed</code>	55
6.40.2.9 <code>NominalCurrent</code>	55
6.40.2.10 <code>NominalPower</code>	55
6.40.2.11 <code>NominalSpeed</code>	55
6.40.2.12 <code>NominalTorque</code>	55
6.40.2.13 <code>NominalVoltage</code>	55
6.40.2.14 <code>Phases</code>	55
6.40.2.15 <code>Poles</code>	55
6.40.2.16 <code>RotorInertia</code>	56
6.40.2.17 <code>SpeedConstant</code>	56
6.40.2.18 <code>SpeedTorqueGradient</code>	56
6.40.2.19 <code>StallTorque</code>	56
6.40.2.20 <code>TorqueConstant</code>	56
6.40.2.21 <code>WindingInductance</code>	56
6.40.2.22 <code>WindingResistance</code>	56
6.41 Структура <code>move_settings_calb_t</code>	56
6.41.1 Подробное описание	57
6.41.2 Поля	57
6.41.2.1 <code>Accel</code>	57

6.41.2.2	AntiplaySpeed	57
6.41.2.3	Decel	57
6.41.2.4	MoveFlags	57
6.41.2.5	Speed	57
6.42	Структура move_settings_t	57
6.42.1	Подробное описание	58
6.42.2	Поля	58
6.42.2.1	Accel	58
6.42.2.2	AntiplaySpeed	58
6.42.2.3	Decel	58
6.42.2.4	MoveFlags	58
6.42.2.5	Speed	58
6.42.2.6	uAntiplaySpeed	58
6.42.2.7	uSpeed	59
6.43	Структура nonvolatile_memory_t	59
6.43.1	Подробное описание	59
6.43.2	Поля	59
6.43.2.1	UserData	59
6.44	Структура pid_settings_t	59
6.44.1	Подробное описание	60
6.45	Структура power_settings_t	60
6.45.1	Подробное описание	60
6.45.2	Поля	60
6.45.2.1	CurrentSetTime	60
6.45.2.2	CurrReductDelay	60
6.45.2.3	HoldCurrent	61
6.45.2.4	PowerFlags	61
6.45.2.5	PowerOffDelay	61
6.46	Структура secure_settings_t	61
6.46.1	Подробное описание	61
6.46.2	Поля	62
6.46.2.1	CriticalIpwr	62
6.46.2.2	CriticalIusb	62
6.46.2.3	CriticalUpwr	62
6.46.2.4	CriticalUusb	62
6.46.2.5	Flags	62
6.46.2.6	LowUpwrOff	62
6.46.2.7	MinimumUusb	62
6.47	Структура serial_number_t	62
6.47.1	Подробное описание	63

6.47.2 Поля	63
6.47.2.1 Key	63
6.47.2.2 Major	63
6.47.2.3 Minor	63
6.47.2.4 Release	63
6.47.2.5 SN	63
6.48 Структура set_position_calb_t	63
6.48.1 Подробное описание	63
6.48.2 Поля	64
6.48.2.1 EncPosition	64
6.48.2.2 PosFlags	64
6.48.2.3 Position	64
6.49 Структура set_position_t	64
6.49.1 Подробное описание	64
6.49.2 Поля	64
6.49.2.1 EncPosition	64
6.49.2.2 PosFlags	64
6.49.2.3 uPosition	65
6.50 Структура stage_information_t	65
6.50.1 Подробное описание	65
6.50.2 Поля	65
6.50.2.1 Manufacturer	65
6.50.2.2 PartNumber	65
6.51 Структура stage_name_t	65
6.51.1 Подробное описание	66
6.51.2 Поля	66
6.51.2.1 PositionerName	66
6.52 Структура stage_settings_t	66
6.52.1 Подробное описание	66
6.52.2 Поля	67
6.52.2.1 HorizontalLoadCapacity	67
6.52.2.2 LeadScrewPitch	67
6.52.2.3 MaxCurrentConsumption	67
6.52.2.4 MaxSpeed	67
6.52.2.5 SupplyVoltageMax	67
6.52.2.6 SupplyVoltageMin	67
6.52.2.7 TravelRange	67
6.52.2.8 Units	67
6.52.2.9 VerticalLoadCapacity	67
6.53 Структура status_calb_t	68

6.53.1	Подробное описание	68
6.53.2	Поля	69
6.53.2.1	CmdBufFreeSpace	69
6.53.2.2	CurPosition	69
6.53.2.3	CurSpeed	69
6.53.2.4	CurT	69
6.53.2.5	EncPosition	69
6.53.2.6	EncSts	69
6.53.2.7	Flags	69
6.53.2.8	GPIOFlags	69
6.53.2.9	Ipwr	69
6.53.2.10	Iusb	69
6.53.2.11	MoveSts	69
6.53.2.12	MvCmdSts	70
6.53.2.13	PWRSts	70
6.53.2.14	Upwr	70
6.53.2.15	Uusb	70
6.53.2.16	WindSts	70
6.54	Структура status_t	70
6.54.1	Подробное описание	71
6.54.2	Поля	71
6.54.2.1	CmdBufFreeSpace	71
6.54.2.2	CurPosition	71
6.54.2.3	CurSpeed	71
6.54.2.4	CurT	71
6.54.2.5	EncPosition	71
6.54.2.6	EncSts	72
6.54.2.7	Flags	72
6.54.2.8	GPIOFlags	72
6.54.2.9	Ipwr	72
6.54.2.10	Iusb	72
6.54.2.11	MoveSts	72
6.54.2.12	MvCmdSts	72
6.54.2.13	PWRSts	72
6.54.2.14	uCurPosition	72
6.54.2.15	uCurSpeed	72
6.54.2.16	Upwr	72
6.54.2.17	Uusb	73
6.54.2.18	WindSts	73
6.55	Структура sync_in_settings_calb_t	73

6.55.1	Подробное описание	73
6.55.2	Поля	73
6.55.2.1	ClutterTime	73
6.55.2.2	Position	73
6.55.2.3	Speed	73
6.55.2.4	SyncInFlags	73
6.56	Структура sync_in_settings_t	74
6.56.1	Подробное описание	74
6.56.2	Поля	74
6.56.2.1	ClutterTime	74
6.56.2.2	Speed	74
6.56.2.3	SyncInFlags	74
6.56.2.4	uPosition	74
6.56.2.5	uSpeed	75
6.57	Структура sync_out_settings_calb_t	75
6.57.1	Подробное описание	75
6.57.2	Поля	75
6.57.2.1	Accuracy	75
6.57.2.2	SyncOutFlags	75
6.57.2.3	SyncOutPeriod	75
6.57.2.4	SyncOutPulseSteps	76
6.58	Структура sync_out_settings_t	76
6.58.1	Подробное описание	76
6.58.2	Поля	76
6.58.2.1	Accuracy	76
6.58.2.2	SyncOutFlags	76
6.58.2.3	SyncOutPeriod	76
6.58.2.4	SyncOutPulseSteps	77
6.58.2.5	uAccuracy	77
6.59	Структура uart_settings_t	77
6.59.1	Подробное описание	77
6.59.2	Поля	77
6.59.2.1	UARTSetupFlags	77
7	Файлы	78
7.1	Файл ximc.h	78
7.1.1	Подробное описание	102
7.1.2	Макросы	102
7.1.2.1	ALARM_ON_DRIVER_OVERHEATING	102
7.1.2.2	BACK_EMF_INDUCTANCE_AUTO	102

7.1.2.3	BACK_EMF_KM_AUTO	102
7.1.2.4	BACK_EMF_RESISTANCE_AUTO	102
7.1.2.5	BORDER_IS_ENCODER	102
7.1.2.6	BORDER_STOP_LEFT	102
7.1.2.7	BORDER_STOP_RIGHT	103
7.1.2.8	BORDERS_SWAP_MISSET_DETECTION	103
7.1.2.9	BRAKE_ENABLED	103
7.1.2.10	BRAKE_ENG_PWROFF	103
7.1.2.11	CONTROL_BTN_LEFT_PUSHED_OPEN	103
7.1.2.12	CONTROL_BTN_RIGHT_PUSHED_OPEN	103
7.1.2.13	CONTROL_MODE_BITS	103
7.1.2.14	CONTROL_MODE_JOY	103
7.1.2.15	CONTROL_MODE_LR	103
7.1.2.16	CONTROL_MODE_OFF	103
7.1.2.17	CTP_ALARM_ON_ERROR	103
7.1.2.18	CTP_BASE	103
7.1.2.19	CTP_ENABLED	104
7.1.2.20	CTP_ERROR_CORRECTION	104
7.1.2.21	DRIVER_TYPE_DISCRETE_FET	104
7.1.2.22	DRIVER_TYPE_EXTERNAL	104
7.1.2.23	DRIVER_TYPE_INTEGRATE	104
7.1.2.24	EEPROM_PRECEDENCE	104
7.1.2.25	ENC_STATE_ABSENT	104
7.1.2.26	ENC_STATE_MALFUNC	104
7.1.2.27	ENC_STATE_OK	104
7.1.2.28	ENC_STATE_REVERS	104
7.1.2.29	ENC_STATE_UNKNOWN	104
7.1.2.30	ENDER_SW1_ACTIVE_LOW	104
7.1.2.31	ENDER_SW2_ACTIVE_LOW	105
7.1.2.32	ENDER_SWAP	105
7.1.2.33	ENGINE_ACCEL_ON	105
7.1.2.34	ENGINE_ANTIPLAY	105
7.1.2.35	ENGINE_CURRENT_AS_RMS	105
7.1.2.36	ENGINE_LIMIT_CURR	105
7.1.2.37	ENGINE_LIMIT_RPM	105
7.1.2.38	ENGINE_LIMIT_VOLT	105
7.1.2.39	ENGINE_MAX_SPEED	105
7.1.2.40	ENGINE_REVERSE	106
7.1.2.41	ENGINE_TYPE_2DC	106
7.1.2.42	ENGINE_TYPE_BRUSHLESS	106

7.1.2.43	ENGINE_TYPE_DC	106
7.1.2.44	ENGINE_TYPE_NONE	106
7.1.2.45	ENGINE_TYPE_STEP	106
7.1.2.46	ENGINE_TYPE_TEST	106
7.1.2.47	ENUMERATE_PROBE	106
7.1.2.48	EXTIO_SETUP_INVERT	106
7.1.2.49	EXTIO_SETUP_MODE_IN_ALARM	106
7.1.2.50	EXTIO_SETUP_MODE_IN_BITS	106
7.1.2.51	EXTIO_SETUP_MODE_IN_HOME	107
7.1.2.52	EXTIO_SETUP_MODE_IN_MOVR	107
7.1.2.53	EXTIO_SETUP_MODE_IN_NOP	107
7.1.2.54	EXTIO_SETUP_MODE_IN_PWOF	107
7.1.2.55	EXTIO_SETUP_MODE_IN_STOP	107
7.1.2.56	EXTIO_SETUP_MODE_OUT_ALARM	107
7.1.2.57	EXTIO_SETUP_MODE_OUT_BITS	107
7.1.2.58	EXTIO_SETUP_MODE_OUT_MOTOR_ON	107
7.1.2.59	EXTIO_SETUP_MODE_OUT_MOVING	107
7.1.2.60	EXTIO_SETUP_MODE_OUT_OFF	107
7.1.2.61	EXTIO_SETUP_MODE_OUT_ON	107
7.1.2.62	EXTIO_SETUP_OUTPUT	107
7.1.2.63	FEEDBACK_EMF	107
7.1.2.64	FEEDBACK_ENC_REVERSE	108
7.1.2.65	FEEDBACK_ENC_TYPE_AUTO	108
7.1.2.66	FEEDBACK_ENC_TYPE_BITS	108
7.1.2.67	FEEDBACK_ENC_TYPE_DIFFERENTIAL	108
7.1.2.68	FEEDBACK_ENC_TYPE_SINGLE_ENDED	108
7.1.2.69	FEEDBACK_ENCODER	108
7.1.2.70	FEEDBACK_ENCODER_MEDIATED	108
7.1.2.71	FEEDBACK_NONE	108
7.1.2.72	HOME_DIR_FIRST	108
7.1.2.73	HOME_DIR_SECOND	108
7.1.2.74	HOME_HALF_MV	108
7.1.2.75	HOME_MV_SEC_EN	109
7.1.2.76	HOME_STOP_FIRST_BITS	109
7.1.2.77	HOME_STOP_FIRST_LIM	109
7.1.2.78	HOME_STOP_FIRST_REV	109
7.1.2.79	HOME_STOP_FIRST_SYN	109
7.1.2.80	HOME_STOP_SECOND_BITS	109
7.1.2.81	HOME_STOP_SECOND_LIM	109
7.1.2.82	HOME_STOP_SECOND_REV	109

7.1.2.83 HOME_STOP_SECOND_SYN	109
7.1.2.84 HOME_USE_FAST	109
7.1.2.85 JOY_REVERSE	109
7.1.2.86 LOW_UPWR_PROTECTION	109
7.1.2.87 LS_SHORTED	110
7.1.2.88 MICROSTEP_MODE_FRAC_128	110
7.1.2.89 MICROSTEP_MODE_FRAC_16	110
7.1.2.90 MICROSTEP_MODE_FRAC_2	110
7.1.2.91 MICROSTEP_MODE_FRAC_256	110
7.1.2.92 MICROSTEP_MODE_FRAC_32	110
7.1.2.93 MICROSTEP_MODE_FRAC_4	110
7.1.2.94 MICROSTEP_MODE_FRAC_64	110
7.1.2.95 MICROSTEP_MODE_FRAC_8	110
7.1.2.96 MICROSTEP_MODE_FULL	110
7.1.2.97 MOVE_STATE_ANTIPLAY	110
7.1.2.98 MOVE_STATE_MOVING	110
7.1.2.99 MOVE_STATE_TARGET_SPEED	111
7.1.2.100 MVCMD_ERROR	111
7.1.2.101 MVCMD_HOME	111
7.1.2.102 MVCMD_LEFT	111
7.1.2.103 MVCMD_LOFT	111
7.1.2.104 MVCMD_MOVE	111
7.1.2.105 MVCMD_MOVR	111
7.1.2.106 MVCMD_NAME_BITS	111
7.1.2.107 MVCMD_RIGHT	111
7.1.2.108 MVCMD_RUNNING	111
7.1.2.109 MVCMD_SSTP	111
7.1.2.110 MVCMD_STOP	111
7.1.2.111 MVCMD_UKNWN	112
7.1.2.112 POWER_OFF_ENABLED	112
7.1.2.113 POWER_REDUCT_ENABLED	112
7.1.2.114 POWER_SMOOTH_CURRENT	112
7.1.2.115 PWR_STATE_MAX	112
7.1.2.116 PWR_STATE_NORM	112
7.1.2.117 PWR_STATE_OFF	112
7.1.2.118 PWR_STATE_REDUCT	112
7.1.2.119 PWR_STATE_UNKNOWN	112
7.1.2.120 REV_SENS_INV	112
7.1.2.121 RPM_DIV_1000	113
7.1.2.122 SETPOS_IGNORE_ENCODER	113

7.1.2.123 SETPOS_IGNORE_POSITION	113
7.1.2.124 STATE_ALARM	113
7.1.2.125 STATE_BORDERS_SWAP_MISSET	113
7.1.2.126 STATE_BRAKE	113
7.1.2.127 STATE_BUTTON_LEFT	113
7.1.2.128 STATE_BUTTON_RIGHT	113
7.1.2.129 STATE_CONTR	113
7.1.2.130 STATE_CONTROLLER_OVERHEAT	113
7.1.2.131 STATE_CTP_ERROR	113
7.1.2.132 STATE_DIG_SIGNAL	114
7.1.2.133 STATE_EEPROM_CONNECTED	114
7.1.2.134 STATE_ENC_A	114
7.1.2.135 STATE_ENC_B	114
7.1.2.136 STATE_ENGINE_RESPONSE_ERROR	114
7.1.2.137 STATE_ERRC	114
7.1.2.138 STATE_ERRD	114
7.1.2.139 STATE_ERRV	114
7.1.2.140 STATE_EXTIO_ALARM	114
7.1.2.141 STATE_GPIO_LEVEL	114
7.1.2.142 STATE_GPIO_PINOUT	114
7.1.2.143 STATE_LEFT_EDGE	114
7.1.2.144 STATE_LOW_USB_VOLTAGE	115
7.1.2.145 STATE_OVERLOAD_POWER_CURRENT	115
7.1.2.146 STATE_OVERLOAD_POWER_VOLTAGE	115
7.1.2.147 STATE_OVERLOAD_USB_CURRENT	115
7.1.2.148 STATE_OVERLOAD_USB_VOLTAGE	115
7.1.2.149 STATE_POWER_OVERHEAT	115
7.1.2.150 STATE_REV_SENSOR	115
7.1.2.151 STATE_RIGHT_EDGE	115
7.1.2.152 STATE_SECUR	115
7.1.2.153 STATE_SYNC_INPUT	115
7.1.2.154 STATE_SYNC_OUTPUT	115
7.1.2.155 SYNCIN_ENABLED	115
7.1.2.156 SYNCIN_INVERT	116
7.1.2.157 SYNCOUT_ENABLED	116
7.1.2.158 SYNCOUT_IN_STEPS	116
7.1.2.159 SYNCOUT_INVERT	116
7.1.2.160 SYNCOUT_ONPERIOD	116
7.1.2.161 SYNCOUT_ONSTART	116
7.1.2.162 SYNCOUT_ONSTOP	116

7.1.2.163	SYNCOUT_STATE	116
7.1.2.164	TS_TYPE_BITS	116
7.1.2.165	UART_PARITY_BITS	116
7.1.2.166	WIND_A_STATE_ABSENT	116
7.1.2.167	WIND_A_STATE_MALFUNC	117
7.1.2.168	WIND_A_STATE_OK	117
7.1.2.169	WIND_A_STATE_UNKNOWN	117
7.1.2.170	WIND_B_STATE_ABSENT	117
7.1.2.171	WIND_B_STATE_MALFUNC	117
7.1.2.172	WIND_B_STATE_OK	117
7.1.2.173	WIND_B_STATE_UNKNOWN	117
7.1.2.174	XIMC_API	117
7.1.3	Типы	117
7.1.3.1	logging_callback_t	117
7.1.4	Функции	117
7.1.4.1	close_device	117
7.1.4.2	command_clear_fram	118
7.1.4.3	command_eeread_settings	118
7.1.4.4	command_eesave_settings	118
7.1.4.5	command_home	118
7.1.4.6	command_homezero	119
7.1.4.7	command_left	119
7.1.4.8	command_loft	119
7.1.4.9	command_move	119
7.1.4.10	command_move_calb	120
7.1.4.11	command_movr	120
7.1.4.12	command_movr_calb	120
7.1.4.13	command_power_off	121
7.1.4.14	command_read_robust_settings	121
7.1.4.15	command_read_settings	121
7.1.4.16	command_reset	121
7.1.4.17	command_right	122
7.1.4.18	command_save_robust_settings	122
7.1.4.19	command_save_settings	122
7.1.4.20	command_sstp	122
7.1.4.21	command_start_measurements	122
7.1.4.22	command_stop	122
7.1.4.23	command_update_firmware	123
7.1.4.24	command_wait_for_stop	123
7.1.4.25	command_zero	123

7.1.4.26	<code>enumerate_devices</code>	123
7.1.4.27	<code>free_enumerate_devices</code>	124
7.1.4.28	<code>get_accessories_settings</code>	124
7.1.4.29	<code>get_analog_data</code>	124
7.1.4.30	<code>get_bootloader_version</code>	125
7.1.4.31	<code>get_brake_settings</code>	125
7.1.4.32	<code>get_calibration_settings</code>	125
7.1.4.33	<code>get_chart_data</code>	125
7.1.4.34	<code>get_control_settings</code>	126
7.1.4.35	<code>get_control_settings_calb</code>	126
7.1.4.36	<code>get_controller_name</code>	126
7.1.4.37	<code>get_ctp_settings</code>	126
7.1.4.38	<code>get_debug_read</code>	127
7.1.4.39	<code>get_device_count</code>	127
7.1.4.40	<code>get_device_information</code>	127
7.1.4.41	<code>get_device_name</code>	127
7.1.4.42	<code>get_edges_settings</code>	128
7.1.4.43	<code>get_edges_settings_calb</code>	128
7.1.4.44	<code>get_emf_settings</code>	128
7.1.4.45	<code>get_encoder_information</code>	129
7.1.4.46	<code>get_encoder_settings</code>	129
7.1.4.47	<code>get_engine_advanced_setup</code>	129
7.1.4.48	<code>get_engine_settings</code>	129
7.1.4.49	<code>get_engine_settings_calb</code>	130
7.1.4.50	<code>get_entype_settings</code>	130
7.1.4.51	<code>get_enumerate_device_controller_name</code>	130
7.1.4.52	<code>get_enumerate_device_information</code>	131
7.1.4.53	<code>get_enumerate_device_network_information</code>	131
7.1.4.54	<code>get_enumerate_device_serial</code>	131
7.1.4.55	<code>get_enumerate_device_stage_name</code>	131
7.1.4.56	<code>get_extended_settings</code>	132
7.1.4.57	<code>get_extio_settings</code>	132
7.1.4.58	<code>get_feedback_settings</code>	132
7.1.4.59	<code>get_firmware_version</code>	133
7.1.4.60	<code>get_gear_information</code>	133
7.1.4.61	<code>get_gear_settings</code>	133
7.1.4.62	<code>get_globally_unique_identifier</code>	133
7.1.4.63	<code>get_hallsensor_information</code>	133
7.1.4.64	<code>get_hallsensor_settings</code>	134
7.1.4.65	<code>get_home_settings</code>	134

7.1.4.66	<code>get_home_settings_calb</code>	134
7.1.4.67	<code>get_init_random</code>	135
7.1.4.68	<code>get_joystick_settings</code>	135
7.1.4.69	<code>get_measurements</code>	135
7.1.4.70	<code>get_motor_information</code>	135
7.1.4.71	<code>get_motor_settings</code>	136
7.1.4.72	<code>get_move_settings</code>	136
7.1.4.73	<code>get_move_settings_calb</code>	136
7.1.4.74	<code>get_nonvolatile_memory</code>	136
7.1.4.75	<code>get_pid_settings</code>	137
7.1.4.76	<code>get_position</code>	137
7.1.4.77	<code>get_position_calb</code>	137
7.1.4.78	<code>get_power_settings</code>	137
7.1.4.79	<code>get_secure_settings</code>	138
7.1.4.80	<code>get_serial_number</code>	138
7.1.4.81	<code>get_stage_information</code>	138
7.1.4.82	<code>get_stage_name</code>	138
7.1.4.83	<code>get_stage_settings</code>	138
7.1.4.84	<code>get_status</code>	139
7.1.4.85	<code>get_status_calb</code>	139
7.1.4.86	<code>get_sync_in_settings</code>	139
7.1.4.87	<code>get_sync_in_settings_calb</code>	140
7.1.4.88	<code>get_sync_out_settings</code>	140
7.1.4.89	<code>get_sync_out_settings_calb</code>	140
7.1.4.90	<code>get_uart_settings</code>	140
7.1.4.91	<code>goto_firmware</code>	141
7.1.4.92	<code>has_firmware</code>	141
7.1.4.93	<code>load_correction_table</code>	141
7.1.4.94	<code>logging_callback_stderr_narrow</code>	142
7.1.4.95	<code>logging_callback_stderr_wide</code>	142
7.1.4.96	<code>msec_sleep</code>	142
7.1.4.97	<code>open_device</code>	142
7.1.4.98	<code>probe_device</code>	143
7.1.4.99	<code>service_command_updf</code>	143
7.1.4.100	<code>set_accessories_settings</code>	143
7.1.4.101	<code>set_bindy_key</code>	143
7.1.4.102	<code>set_brake_settings</code>	143
7.1.4.103	<code>set_calibration_settings</code>	143
7.1.4.104	<code>set_control_settings</code>	144
7.1.4.105	<code>set_control_settings_calb</code>	144

7.1.4.106 set_controller_name	144
7.1.4.107 set_ctp_settings	145
7.1.4.108 set_debug_write	145
7.1.4.109 set_edges_settings	145
7.1.4.110 set_edges_settings_calb	145
7.1.4.111 set_emf_settings	146
7.1.4.112 set_encoder_information	146
7.1.4.113 set_encoder_settings	146
7.1.4.114 set_engine_advansed_setup	147
7.1.4.115 set_engine_settings	147
7.1.4.116 set_engine_settings_calb	147
7.1.4.117 set_entype_settings	148
7.1.4.118 set_extended_settings	148
7.1.4.119 set_extio_settings	148
7.1.4.120 set_feedback_settings	149
7.1.4.121 set_gear_information	149
7.1.4.122 set_gear_settings	149
7.1.4.123 set_hallsensor_information	149
7.1.4.124 set_hallsensor_settings	150
7.1.4.125 set_home_settings	150
7.1.4.126 set_home_settings_calb	150
7.1.4.127 set_joystick_settings	150
7.1.4.128 set_logging_callback	151
7.1.4.129 set_motor_information	151
7.1.4.130 set_motor_settings	151
7.1.4.131 set_move_settings	152
7.1.4.132 set_move_settings_calb	152
7.1.4.133 set_nonvolatile_memory	152
7.1.4.134 set_pid_settings	152
7.1.4.135 set_position	153
7.1.4.136 set_position_calb	153
7.1.4.137 set_power_settings	153
7.1.4.138 set_secure_settings	153
7.1.4.139 set_serial_number	154
7.1.4.140 set_stage_information	154
7.1.4.141 set_stage_name	154
7.1.4.142 set_stage_settings	154
7.1.4.143 set_sync_in_settings	154
7.1.4.144 set_sync_in_settings_calb	155
7.1.4.145 set_sync_out_settings	155

7.1.4.146 <code>set_sync_out_settings_calb</code>	155
7.1.4.147 <code>set_uart_settings</code>	156
7.1.4.148 <code>write_key</code>	156
7.1.4.149 <code>ximc_fix_usbser_sys</code>	156
7.1.4.150 <code>ximc_version</code>	156
 Алфавитный указатель	 157

Глава 1

Библиотека libximc

Документация для библиотеки libximc.

Libximc - кроссплатформенная 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](#).

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 оборудованием с помощью готового ПО или с помощью библиотек для языков программирования: C/C++, C#, JAVA , Visual Basic, Python 2/3, .NET, Delphi, интеграция со средами программирования MS Visual Studio, gcc, Xcode.
- Работает с научными средами разработки путем интеграции LabVIEW и MATLAB;

1.2 Что умеет библиотека libximc.

- Libximc управляет оборудованием 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.

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 предложения, ошибки и идеи. Мы ценим ваши предложения и стараемся сделать наш продукт лучше. Пожалуйста, оставляйте свои вопросы [сюда](#). Идеи и комментарии отправляйте нам на почту 8smc4@standa.lt

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, в том числе Windows 7, Windows Vista, Windows XP, Windows Server 2003, Windows 2000, Linux, Mac OS X. 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.

2.2 Требования к установленному программному обеспечению

2.2.1 Для сборки библиотеки

Для Windows:

- Windows 2000 или старше, 64-битная система (если планируется собирать обе архитектуры) или 32-битная система
- Microsoft Visual C++ 2013 или старше
- cygwin с tar, bison, flex, curl
- 7z

Для Linux:

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

- mercurial (для сборки версии для разработки из hg)

Для Mac OS X:

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

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

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

2.2.2 Для использования библиотеки

Поддерживаемые операционные системы (32 и 64 бита) и требования к окружению:

- Mac OS X 10.6
- Windows 2000 или старше
- Autotools-совместимый unix. Библиотека устанавливается из бинарного вида.
- Linux на основе debian 32 и 64 бита. 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
- Java 7 64 бит или 32 бит
- .NET 2.0 (только 32 бит)
- Delphi (только 32 бит)

Требования сборки:

- Windows: Microsoft Visual C++ 2013 или mingw (в данный момент не поддерживается)
- UNIX: gcc 4, gmake
- Mac OS X: XCode 4
- JDK 7

Глава 3

Как пересобрать библиотеку

3.1 Сборка для UNIX

Обобщенная версия собирается обычными autotools.

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

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

3.2 Сборка для Linux на основе Debian

Требования: 64-битная или 32-битная система на основе debian, ubuntu Примерный набор пакетов: gcc, autotools, autoconf, libtool, dpkg-dev, flex, bison, doxygen, texlive, mercurial Полный набор пакетов: apt-get install ruby1.9.1 debhelper vim sudo g++ mercurial git curl make cmake autotools-dev automake autoconf libtool default-jre-headless default-jdk openjdk-6-jdk dpkg-dev lintian texlive texlive-latex-extra texlive-lang-cyrillic dh-autoreconf hardening-wrapper bison flex doxygen lsb-release pkg-config check Для кросс-компиляции ARM установите gcc-arm-linux-gnueabi из вашего инстументария ARM.

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

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

Примерный набор пакетов: gcc, autotools, autoconf, libtool, flex, bison, doxygen, texlive, mercurial Полный набор пакетов: autoconf automake bison doxygen flex gcc gcc-32bit gcc-c++ gcc-c++-32bit java-1_7_0-openjdk java-1_7_0-openjdk-devel libtool lsb-release make mercurial rpm-build rpm-devel rpmlint texlive texlive-fonts-extra texlive-latex

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`.

3.4 Сборка для Mac OS X

Для сборки библиотеки и пакета запустите скрипт:

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

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

3.5 Сборка в ОС Windows

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

Запустите скрипт:

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

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

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

3.6 Доступ к исходным кодам

Исходные коды XIMC могут быть выданы по отдельному запросу.

Глава 4

Как использовать с...

Для приобретения первых навыков использования библиотеки создано простое тестовое приложение `testapp`. Языки, отличные от C-подобных, поддерживаются с помощью вызовов с преобразованием аргументов типа `stdcall`. Простое тестовое приложение на языке C расположено в директории `'examples/testapp'`, проект на C# - в `'examples/testcs'`, на VB.NET - в `'examples/testvbnet'`, для delphi 6 - в `'example/testdelphi'`, для matlab - `'examples/testmatlab'`, для Java - `'examples/testjava'`, для Python - `'examples/testpython'`. Библиотеки, заголовочные файлы и другие необходимые файлы расположены в директориях `'win32'`, `'win64'`, `'macosx'` и подобных. В комплект разработчика также входят уже скомпилированные примеры: `testapp` и `testappeasy` в варианте 32 и 64 бита под windows и только 64 бита под osx, `testcs`, `testvbnet`, `testdelphi` - только 32 бита, `testjava` - кроссплатформенный, `testmatlab` и `testpython` не требуют компиляции.

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

ЗАМЕЧАНИЕ: Для работы на Linux требуется установить оба пакета `libxime7_x.x.x` и `libxime7-dev_x.x.x`. Для установки пакетов можно воспользоваться `.deb` командой: `dpkg -i имя_пакета.deb`, где `имя_пакета.deb` — это имя файла пакета (пакеты в Debian имеют расширение `.deb`). Запускать `dpkg` необходимо с правами суперпользователя (`root`).

4.1 Использование на C

4.1.1 Visual C++

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

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

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

4.1.2 CodeBlocks

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

Откройте проект `examples/testcodeblocks/testcodeblocks.cbp`, выполните сборку и запустите приложение из среды разработки.

4.1.3 MinGW

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

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

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

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

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

4.1.4 C++ Builder

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

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

Затем скомпилируйте тестовое приложение:

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

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

4.1.5 XCode

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

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

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

4.1.6 GCC

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

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

Скопируйте файл `/usr/share/libximc/keyfile.sqlite` в директорию с проектом командой

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

testapp может быть собран следующим образом с установленной библиотекой:

```
$ make
```

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

Затем запустите приложение с помощью:

```
$ make run
```

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

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

4.2 .NET

Для использования в .NET предлагается обертка `wrappers/csharp/ximcnet.dll`. Она распространяется в двух различных архитектурах. Поддерживает платформу .NET от 2.0. до 4.0.

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

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

4.3 Delphi

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

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

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

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле `testdelphi.dpr` перед сборкой нужно прописать IP адрес Ethernet-адаптера (переменная `enum_hints`).

4.4 Java

Как запустить пример на Linux. Перейдите в `ximc-2.x.x/examples/testjava/compiled/` и выполните

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

Как запустить пример на Windows или Mac. Перейдите в `ximc-2.x.x/examples/testjava/compiled/`. Скопируйте содержимое `ximc-2.x.x/ximc/win64/` или `ximc-2.x.x/ximc/macosx/` соответственно в текущую директорию. Затем запустите:

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

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

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

Затем пересоберите исходные тексты:


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

или для Windows или Mac:

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

Затем соберите jar:

```
$ jar cmf MANIFEST.MF testjava.jar ru
```

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

4.5 Python

Измените текущую директорию на examples/testpython. Для корректного использования библиотеки libximc, в примере используется файл обертка, crossplatform\wrappers\python\pyximc.py с описанием структур библиотеки.

Перед запуском:

На OS X: скопируйте библиотеку ximc/macosx/libximc.framework в текущую директорию.

На Linux: может понадобиться установить LD_LIBRARY_PATH, чтобы Python мог найти библиотеки с RPATH. Например, запустите:

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

На Windows: перед запуском ничего делать не нужно. Все необходимые связи и зависимости прописаны в коде примера. Используются библиотеки: bindy.dll, libximc.dll, xiwrapper.dll. Расположенные в папке для соответствующих версий Windows.

Запустите Python 2 или Python 3:

```
python testpython.py
```

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

4.6 MATLAB

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

Перед запуском:

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

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

Для проверки совместимых XCode и gcc проверьте документы <https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/support/sysreq/files/SystemRequirements-Release2014a-SupportedCompilers.pdf> или похожие.

На Windows: перед запуском ничего делать не нужно

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

testximc

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

4.7 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.

4.8 Требуемые права доступа

Библиотеке не требуются особые права для выполнения, а нужен только доступ 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.

4.9 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 "testcprofile".

Глава 5

Работа с пользовательскими единицами

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

- Структура пересчета единиц `calibration_t`
- Функции дублиеры 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.

$$\text{user_value} = A * (\text{step} + \text{mstep} / \text{pow}(2, \text{MicrostepMode} - 1))$$

- Пересчет из пользовательских единиц.

$$\begin{aligned} \text{step} &= (\text{int})(\text{user_value} / A) \\ \text{mstep} &= (\text{user_value} / A - \text{step}) * \text{pow}(2, \text{MicrostepMode} - 1) \end{aligned}$$

5.2 Функции дублиеры 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. Формат данных `_calb` структур описан подробно. 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.

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 А" необработанные данные с АЦП.
- unsigned int [A2Voltage_ADC](#)
"Выходное напряжение на 2 выводе обмотки А" необработанные данные с АЦП.
- unsigned int [B1Voltage_ADC](#)
"Выходное напряжение на 1 выводе обмотки В" необработанные данные с АЦП.
- unsigned int [B2Voltage_ADC](#)
"Выходное напряжение на 2 выводе обмотки В" необработанные данные с АЦП.
- unsigned int [SupVoltage_ADC](#)
"Напряжение питания ключей H-моста" необработанные данные с АЦП.
- unsigned int [ACurrent_ADC](#)
"Ток через обмотку А" необработанные данные с АЦП.
- unsigned int [BCurrent_ADC](#)
"Ток через обмотку В" 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 [L5_ADC](#)
Напряжение питания USB после current sense резистора, необработанные данные с АЦП.
- unsigned int [H5_ADC](#)
Напряжение питания USB, необработанные данные с АЦП
- int [A1Voltage](#)
"Выходное напряжение на 1 выводе обмотки А" откалиброванные данные (в десятках мВ).
- int [A2Voltage](#)
"Выходное напряжение на 2 выводе обмотки А" откалиброванные данные (в десятках мВ).
- int [B1Voltage](#)
"Выходное напряжение на 1 выводе обмотки В" откалиброванные данные (в десятках мВ).
- int [B2Voltage](#)
"Выходное напряжение на 2 выводе обмотки В" откалиброванные данные (в десятках мВ).
- int [SupVoltage](#)
"Напряжение питания ключей H-моста" откалиброванные данные (в десятках мВ).
- int [ACurrent](#)
"Ток через обмотку А" откалиброванные данные (в мА).
- int [BCurrent](#)
"Ток через обмотку В" 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 [L5](#)
Напряжение питания USB после current sense резистора (в десятках мВ).
- int [H5](#)
Напряжение питания USB (в десятках мВ).
- unsigned int deprecated
- 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 В" откалиброванные данные (в десятках мВ).

6.2.2.8 unsigned int B1Voltage_ADC

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

6.2.2.9 int B2Voltage

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

6.2.2.10 unsigned int B2Voltage_ADC

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

6.2.2.11 int BCurrent

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

6.2.2.12 unsigned int BCurrent_ADC

"Ток через обмотку В" необработанные данные с АЦП.

6.2.2.13 int FullCurrent

"Полный ток" откалиброванные данные (в мА).

6.2.2.14 unsigned int FullCurrent_ADC

"Полный ток" необработанные данные с АЦП.

6.2.2.15 int H5

Напряжение питания USB (в десятках мВ).

6.2.2.16 int Joy

Джойстик во внутренних единицах.

Диапазон: 0..10000

6.2.2.17 unsigned int Joy_ADC

Джойстик, необработанные данные с АЦП.

6.2.2.18 int L5

Напряжение питания USB после current sense резистора (в десятках мВ).

6.2.2.19 `unsigned int L5_ADC`

Напряжение питания USB после current sense резистора, необработанные данные с АЦП.

6.2.2.20 `int Pot`

Аналоговый вход во внутренних единицах.

Диапазон: 0..10000

6.2.2.21 `int SupVoltage`

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

6.2.2.22 `unsigned int SupVoltage_ADC`

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

6.2.2.23 `int Temp`

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

6.2.2.24 `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 А.
- float [CSS1_B](#)
Коэффициент сдвига для аналоговых измерений тока в обмотке А.
- float [CSS2_A](#)
Коэффициент масштабирования для аналоговых измерений тока в обмотке В.
- float [CSS2_B](#)
Коэффициент сдвига для аналоговых измерений тока в обмотке В.
- 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.

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](#)
Multiplier.
- unsigned int [MicrostepMode](#)
Microstep mode.

6.5.1 Подробное описание

Структура калибровок

6.6 Структура `chart_data_t`

Дополнительное состояние устройства.

Поля данных

- `int` [WindingVoltageA](#)
В случае ШД, напряжение на обмотке А (в десятках мВ); в случае бесщеточного, напряжение на первой обмотке; в случае DC на единственной.
- `int` [WindingVoltageB](#)
В случае ШД, напряжение на обмотке В (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 А (в мА); в случае бесщеточного, ток в первой обмотке; в случае DC в единственной.
- `int` [WindingCurrentB](#)
В случае ШД, ток в обмотке В (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 десяти тысячных долях.
- `int` [DutyCycle](#)
Коэффициент заполнения ШИМ.

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` DutyCycle

Коэффициент заполнения ШИМ.

6.6.2.2 `unsigned int` Joy

Положение джойстика в десяти тысячных 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

Команда чтения 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

Команда чтения информации о контроллере.

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

См. также

```
get_device_information
get_device_information_impl
```

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 address, passed in network byte order (big-endian byte order)
- `char nodename [16]`
Name of the Bindy node which hosts the device.
- `uint32_t axis_state`
Flags representing device state.
- `char locker_username [16]`
Name of the user who locked the device (if any)
- `char locker_nodename [16]`
Bindy node name, which was used to lock the device (if any)
- `time_t locked_time`
Time the lock was acquired at (UTC, microseconds since the epoch)

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](#)
Электрохимический коэффициент двигателя.
- unsigned int [BackEMFFlags](#)
Флаги автоопределения характеристик обмоток двигателя.

6.17.1 Подробное описание

Настройки EMF.

Эта структура содержит данные электрохимических характеристик(EMF) двигателя. Они определяют индуктивность, сопротивление и электрохимический коэффициент двигателя. Эти данные хранятся во flash памяти 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

Электрохимический 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 Kp, Ki, Kd и L, R, Km. Эти данные хранятся во flash памяти 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 mA).

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

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 грм для DC и шагового двигателя в режиме ведущего энкодера).

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

id	идентификатор устройства
EngineType	тип мотора
DriverType	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 EAS.

Поля данных

- unsigned int Param1

6.24.1 Подробное описание

Настройки EAS.

Эта структура EST. Эти данные хранятся во flash памяти контроллера.

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

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 см2).

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.

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.

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

См. также

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

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

Буфер вмещает не более 25и точек.

Поля данных

- int [Speed](#) [25]

Текущая скорость в микрошагах в секунду (целые шаги пересчитываются с учетом текущего режима деления шага) или отсчетах энкодера в секунду.

- int [Error](#) [25]

Текущая скорость в микрошагах в секунду (целые шаги пересчитываются с учетом текущего режима деления шага) или отсчетах энкодера в секунду.

- unsigned int [Length](#)

Длина фактических данных в буфере.

6.38.1 Подробное описание

Буфер вмещает не более 25и точек.

Точная длина полученного буфера отражена в поле `Length`.

См. также

`measurements`
[get_measurements](#)

6.38.2 Поля

6.38.2.1 `int Error[25]`

Текущая скорость в микрошагах в секунду (целые шаги пересчитываются с учетом текущего режима деления шага) или отсчетах энкодера в секунду.

6.38.2.2 `unsigned int Length`

Длина фактических данных в буфере.

6.38.2.3 `int Speed[25]`

Текущая скорость в микрошагах в секунду (целые шаги пересчитываются с учетом текущего режима деления шага) или отсчетах энкодера в секунду.

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

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

6.40.1 Подробное описание

Физический 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

Флаг 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 см²).

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`

Сопротивление 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](#)
Заданная скорость.
- float [Accel](#)

- Ускорение, заданное в шагах в секунду²(ШД) или в оборотах в минуту за секунду(DC).
- float [Decel](#)
Торможение, заданное в шагах в секунду²(ШД) или в оборотах в минуту за секунду(DC).
- 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, заданное в шагах в секунду²(ШД) или в оборотах в минуту за секунду(DC).

6.41.2.2 float AntiplaySpeed

Скорость в режиме антилюфта.

6.41.2.3 float Decel

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

6.41.2.4 unsigned int MoveFlags

[Флаги параметров движения.](#)

6.41.2.5 float Speed

Заданная скорость.

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 секунду²(ШД) или в оборотах в минуту за секунду(DC).
- unsigned int [Decel](#)
Торможение, заданное в шагах в секунду²(ШД) или в оборотах в минуту за секунду(DC).
- unsigned int [AntipplaySpeed](#)
Скорость в режиме антилюфта, заданная в целых шагах/с(ШД) или в оборотах/с(DC).
- unsigned int [uAntipplaySpeed](#)
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 секунду²(ШД) или в оборотах в минуту за секунду(DC).

Диапазон: 1..65535.

6.42.2.2 unsigned int AntipplaySpeed

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 секунду²(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 uAntipplaySpeed

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 `nonvolatile_memory_t`

Пользовательские данные для сохранения во FRAM.

Поля данных

- `unsigned int UserData [7]`
Пользовательские данные.

6.43.1 Подробное описание

Пользовательские данные для сохранения во FRAM.

См. также

[`get_nonvolatile_memory`](#), [`set_nonvolatile_memory`](#)

6.43.2 Поля

6.43.2.1 `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.

6.44 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.

6.44.1 Подробное описание

Настройки ПИД.

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

См. также

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

6.45 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.](#)

6.45.1 Подробное описание

Настройки питания шагового мотора.

См. также

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

6.45.2 Поля

6.45.2.1 unsigned int CurrentSetTime

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

6.45.2.2 unsigned int CurrReductDelay

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

6.45.2.3 unsigned int `HoldCurrent`

Ток мотора в режиме удержания, в процентах от номинального.

Диапазон: 0..100.

6.45.2.4 unsigned int `PowerFlags`

Флаги параметров питания шагового мотора.

6.45.2.5 unsigned int `PowerOffDelay`

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

6.46 Структура `secure_settings_t`

Эта структура содержит необработанные данные с АЦП и нормированные значения.

Поля данных

- unsigned int `LowUpwrOff`
Нижний порог напряжения на силовой части для выключения, 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`
Максимальный ток USB, вызывающий состояние ALARM, в мА.
- unsigned int `CriticalUusb`
Максимальное напряжение на USB, вызывающее состояние ALARM, десятки мВ.
- unsigned int `MinimumUusb`
Минимальное напряжение на USB, вызывающее состояние ALARM, десятки мВ.
- unsigned int `Flags`
Флаги критических параметров.

6.46.1 Подробное описание

Эта структура содержит необработанные данные с АЦП и нормированные значения.

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

См. также

`get_secure_settings`
`set_secure_settings`
`get_secure_settings, set_secure_settings`

6.46.2 Поля

6.46.2.1 `unsigned int CriticalIpwr`

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

6.46.2.2 `unsigned int CriticalIusb`

Максимальный ток USB, вызывающий состояние ALARM, в мА.

6.46.2.3 `unsigned int CriticalUpwr`

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

6.46.2.4 `unsigned int CriticalUusb`

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

6.46.2.5 `unsigned int Flags`

[Флаги критических параметров.](#)

6.46.2.6 `unsigned int LowUpwrOff`

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

6.46.2.7 `unsigned int MinimumUusb`

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

6.47 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.

6.47.1 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](#)

6.47.2 Поля

6.47.2.1 `uint8_t Key[32]`

Ключ защиты для установки серийного номера (256 бит).

6.47.2.2 `unsigned int Major`

Основной номер версии железа.

6.47.2.3 `unsigned int Minor`

Второстепенный номер версии железа.

6.47.2.4 `unsigned int Release`

Номер правок этой версии железа.

6.47.2.5 `unsigned int SN`

Новый серийный номер платы.

6.48 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.

6.48.1 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](#)

6.48.2 Поля

6.48.2.1 `long_t` `EncPosition`

Позиция энкодера.

6.48.2.2 `unsigned int` `PosFlags`

[Флаги установки положения.](#)

6.48.2.3 `float` `Position`

Позиция двигателя.

6.49 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.](#)

6.49.1 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](#)

6.49.2 Поля

6.49.2.1 `long_t` `EncPosition`

Позиция энкодера.

6.49.2.2 `unsigned int` `PosFlags`

[Флаги установки положения.](#)

6.49.2.3 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).

6.50 Структура stage_information_t

Информация о позиционере.

Поля данных

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

6.50.1 Подробное описание

Информация о позиционере.

См. также

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

6.50.2 Поля

6.50.2.1 char Manufacturer[17]

Производитель.

Максимальная длина строки: 16 символов.

6.50.2.2 char PartNumber[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

6.51 Структура stage_name_t

Пользовательское имя подвижки.

Поля данных

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

6.51.1 Подробное описание

Пользовательское имя подвижки.

См. также

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

6.51.2 Поля

6.51.2.1 char PositionerName[17]

Пользовательское имя подвижки.

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

6.52 Структура stage_settings_t

Настройки позиционера.

Поля данных

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

6.52.1 Подробное описание

Настройки позиционера.

См. также

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

6.52.2 Поля

6.52.2.1 float HorizontalLoadCapacity

Горизонтальная грузоподъемность (кг).

Тип данных: float.

6.52.2.2 float LeadScrewPitch

Шаг ходового винта в мм.

Тип данных: float.

6.52.2.3 float MaxCurrentConsumption

Максимальный ток потребления (А).

Тип данных: float.

6.52.2.4 float MaxSpeed

Максимальная скорость (Units/c).

Тип данных: float.

6.52.2.5 float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

6.52.2.6 float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

6.52.2.7 float TravelRange

Диапазон перемещения (Units).

Тип данных: float.

6.52.2.8 char Units[9]

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

6.52.2.9 float VerticalLoadCapacity

Вертикальная грузоподъемность (кг).

Тип данных: float.

6.53 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 `Ipwr`
Ток потребления силовой части, мА.
- int `Upwr`
Напряжение на силовой части, десятки мВ.
- int `Iusb`
Ток потребления по USB, мА.
- int `Uusb`
Напряжение на USB, десятки мВ.
- int `CurT`
Температура процессора в десятых долях градусов цельсия.
- unsigned int `Flags`
Флаги состояния.
- unsigned int `GPIOFlags`
Флаги состояния GPIO входов.
- unsigned int `CmdBufFreeSpace`
Данное поле служебное.

6.53.1 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`

6.53.2 Поля

6.53.2.1 unsigned int CmdBufFreeSpace

Данное поле служебное.

Оно показывает количество свободных ячеек буфера цепочки синхронизации.

6.53.2.2 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.

6.53.2.3 float CurSpeed

Текущая скорость.

6.53.2.4 int CurT

Температура процессора в десятых долях градусов цельсия.

6.53.2.5 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.

6.53.2.6 unsigned int EncSts

Состояние энкодера.

6.53.2.7 unsigned int Flags

Флаги состояния.

6.53.2.8 unsigned int GPIOFlags

Флаги состояния GPIO входов.

6.53.2.9 int Ipwr

Ток потребления силовой части, мА.

6.53.2.10 int Iusb

Ток потребления по USB, мА.

6.53.2.11 unsigned int MoveSts

Флаги состояния движения.

6.53.2.12 unsigned int MvCmdSts

Состояние команды движения.

6.53.2.13 unsigned int PWRSts

Флаги состояния питания шагового мотора.

6.53.2.14 int Upwr

Напряжение на силовой части, десятки мВ.

6.53.2.15 int Uusb

Напряжение на USB, десятки мВ.

6.53.2.16 unsigned int WindSts

Состояние обмоток.

6.54 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 Ipwr
Ток потребления силовой части, мА.

- int Upwr
Напряжение на силовой части, десятки мВ.
- int Iusb
Ток потребления по USB, мА.
- int Uusb
Напряжение на USB, десятки мВ.
- int CurT
Температура процессора в десятых долях градусов цельсия.
- unsigned int Flags
Флаги состояния.
- unsigned int GPIOFlags
Флаги состояния GPIO входов.
- unsigned int CmdBufFreeSpace
Данное поле служебное.

6.54.1 Подробное описание

Состояние устройства.

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

См. также

get_status_impl

6.54.2 Поля

6.54.2.1 unsigned int CmdBufFreeSpace

Данное поле служебное.

Оно показывает количество свободных ячеек буфера цепочки синхронизации.

6.54.2.2 int CurPosition

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

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

6.54.2.3 int CurSpeed

Текущая скорость.

6.54.2.4 int CurT

Температура процессора в десятых долях градусов цельсия.

6.54.2.5 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.

6.54.2.6 unsigned int EncSts

Состояние энкодера.

6.54.2.7 unsigned int Flags

Флаги состояния.

6.54.2.8 unsigned int GPIOFlags

Флаги состояния GPIO входов.

6.54.2.9 int Ipwr

Ток потребления силовой части, мА.

6.54.2.10 int Iusb

Ток потребления по USB, мА.

6.54.2.11 unsigned int MoveSts

Флаги состояния движения.

6.54.2.12 unsigned int MvCmdSts

Состояние команды движения.

6.54.2.13 unsigned int PWRSts

Флаги состояния питания шагового мотора.

6.54.2.14 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.

6.54.2.15 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.

6.54.2.16 int Upwr

Напряжение на силовой части, десятки мВ.

6.54.2.17 `int Uusb`

Напряжение на USB, десятки мВ.

6.54.2.18 `unsigned int WindSts`

[Состояние обмоток.](#)

6.55 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.

6.55.1 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](#)

6.55.2 Поля

6.55.2.1 `unsigned int ClutterTime`

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

6.55.2.2 `float Position`

Желаемая позиция или смещение.

6.55.2.3 `float Speed`

Заданная скорость.

6.55.2.4 `unsigned int SyncInFlags`

[Флаги настроек синхронизации входа.](#)

6.56 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.

6.56.1 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
```

6.56.2 Поля

6.56.2.1 unsigned int `ClutterTime`

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

6.56.2.2 unsigned int `Speed`

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

Диапазон: 0..100000.

6.56.2.3 unsigned int `SyncInFlags`

Флаги настроек синхронизации входа.

6.56.2.4 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`).

6.56.2.5 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.

6.57 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.

6.57.1 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
```

6.57.2 Поля

6.57.2.1 float `Accuracy`

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

6.57.2.2 unsigned int `SyncOutFlags`

Флаги настроек синхронизации выхода.

6.57.2.3 unsigned int `SyncOutPeriod`

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

6.57.2.4 `unsigned int SyncOutPulseSteps`

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

6.58 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).

6.58.1 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
```

6.58.2 Поля

6.58.2.1 `unsigned int Accuracy`

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

6.58.2.2 `unsigned int SyncOutFlags`

Флаги настроек синхронизации выхода.

6.58.2.3 `unsigned int SyncOutPeriod`

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

6.58.2.4 unsigned int SyncOutPulseSteps

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

6.58.2.5 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`).

6.59 Структура `uart_settings_t`

Настройки UART.

Поля данных

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

6.59.1 Подробное описание

Настройки UART.

Эта структура содержит настройки UART.

См. также

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

6.59.2 Поля

6.59.2.1 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](#)
Эта структура содержит необработанные данные с АЦП и нормированные значения.
- struct [edges_settings_t](#)
Настройки границ.
- struct [edges_settings_calb_t](#)
Настройки границ с использованием пользовательских единиц.
- struct [pid_settings_t](#)

- struct [device_information_t](#)
Команда чтения 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](#)
Library import macro Macros allows to automatically import function from shared library.
- `#define` [XIMC_CALLCONV](#)
Library calling convention macros.
- `#define` [XIMC_RETTYPE](#) void*
- `#define` [device_undefined](#) -1
Макрос, означающий неопределенное устройство

Результаты выполнения команд

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

Уровень логирования

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

Флаги поиска устройств

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

Флаги состояния движения

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

См. также

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

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

Флаги состояния питания шагового мотора

Возвращаются командой `get_status`.

См. также

`get_status`
`status_t::PWRSts, get_status_impl`

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

Флаги состояния

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

См. также

`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`
 Нарушение целостности данных.
- `#define STATE_ERRV 0x00000004`
 Недопустимое значение данных.
- `#define STATE_EEPROM_CONNECTED 0x00000010`
 Подключена память EEPROM с настройками.
- `#define STATE_IS_HOMED 0x00000020`
 Калибровка выполнена
- `#define STATE_SECUR 0x1B3FFC0`
 Флаги опасности.
- `#define STATE_ALARM 0x00000040`
 Контроллер находится в состоянии ALARM, показывая, что случилась какая-то опасная ситуация.
- `#define STATE_CTP_ERROR 0x00000080`
 Контроль позиции нарушен(используется только с шаговым двигателем).
- `#define STATE_POWER_OVERHEAT 0x00000100`
 Перегрелась силовая часть платы.
- `#define STATE_CONTROLLER_OVERHEAT 0x00000200`
 Перегрелась микросхема контроллера.
- `#define STATE_OVERLOAD_POWER_VOLTAGE 0x00000400`
 Превышено напряжение на силовой части.
- `#define STATE_OVERLOAD_POWER_CURRENT 0x00000800`
 Превышен максимальный ток потребления силовой части.
- `#define STATE_OVERLOAD_USB_VOLTAGE 0x0001000`
 Превышено напряжение на USB.
- `#define STATE_LOW_USB_VOLTAGE 0x0002000`
 Слишком низкое напряжение на USB.
- `#define STATE_OVERLOAD_USB_CURRENT 0x0004000`
 Превышен максимальный ток потребления USB.
- `#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 отличаются друг от друга слишком сильно
- `#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 входным сигналом.

Флаги состояния GPIO входов

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 А энкодера(флаг "1", если энкодер активен).
- `#define STATE_ENC_B 0x4000`
Состояние ножки В энкодера(флаг "1", если энкодер активен).

Состояние энкодера

Состояние энкодера, подключенного к контроллеру.

См. также

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

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

Состояние обмоток

Состояние обмоток двигателя, подключенного к контроллеру.

См. также

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

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

Состояние команды движения

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

См. также

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

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

- Команда left.
#define [MVCMD_RIGHT](#) 0x04
- Команда rigt.
#define [MVCMD_STOP](#) 0x05
- Команда stop.
#define [MVCMD_HOME](#) 0x06
- Команда home.
#define [MVCMD_LOFT](#) 0x07
- Команда loft.
#define [MVCMD_SSTP](#) 0x08
- Команда плавной остановки(SSTP).
#define [MVCMD_ERROR](#) 0x40
- Состояние завершения движения (1 - команда движения выполнена с ошибкой, 0 - команда движения выполнена корректно).
#define [MVCMD_RUNNING](#) 0x80
- Состояние команды движения (0 - команда движения выполнена, 1 - команда движения сейчас выполняется).

Флаги параметров движения

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 грм.

Флаги параметров мотора

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

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

См. также

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

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

Флаги, определяющие тип мотора

Определяют тип мотора. Возвращаются командой `get_entype_settings`.

См. также

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

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

Флаги, определяющие тип силового драйвера

Определяют тип силового драйвера. Возвращаются командой `get_entype_settings`.

См. также

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

- `#define DRIVER_TYPE_DISCRETE_FET 0x01`

- Силовой драйвер на дискретных мосфет-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

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.

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 концевика.
- #define `ALARM_FLAGS_STICKING` 0x10
Если флаг установлен, то только по команде STOP возможен сброс всех флагов ALARM.
- #define `USB_BREAK_RECONNECT` 0x20
Если флаг установлен, то будет включен блок перезагрузки USB при поломке связи.
- #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

Возвращаются командой `get_position`.

См. также

[get_position](#)
[set_position](#)
[set_position_t::PosFlags](#), [set_position](#)

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

Тип обратной связи.

См. также

[set_feedback_settings](#)
[get_feedback_settings](#)
[feedback_settings_t::FeedbackType](#), [get_feedback_settings](#), [set_feedback_settings](#)

- `#define FEEDBACK_ENCODER 0x01`
 Обратная связь с помощью энкодера.
- `#define FEEDBACK_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.

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_TYPE_BITS 0xC0`
 Биты, отвечающие за тип энкодера.
- `#define FEEDBACK_ENC_TYPE_AUTO 0x00`
 Определять тип энкодера автоматически.
- `#define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40`
 Недифференциальный энкодер.
- `#define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80`
 Дифференциальный энкодер.

Флаги настроек синхронизации входа

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

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`
Выдать импульс синхронизации после прохождения SyncOutPeriod отсчётов.

Флаги настройки работы внешнего ввода/вывода

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

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

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

См. также

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

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

Флаги концевых выключателей

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

См. также

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

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

Флаги настроек тормоза

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

См. также

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

- `#define BRAKE_ENABLED 0x01`
Управление тормозом включено, если флаг установлен.

- `#define BRAKE_ENG_PWROFF 0x02`
Тормоз отключает питание шагового мотора, если флаг установлен.

Флаги управления

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

См. также

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

- `#define CONTROL_MODE_BITS 0x03`
Биты управления мотором с помощью джойстика или кнопок влево/вправо.
- `#define CONTROL_MODE_OFF 0x00`
Управление отключено.
- `#define CONTROL_MODE_JOY 0x01`
Управление с помощью джойстика.
- `#define CONTROL_MODE_LR 0x02`
Управление с помощью кнопок left/right.
- `#define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04`
Левая кнопка нормально разомкнутая, если флаг установлен.
- `#define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08`
Правая кнопка нормально разомкнутая, если флаг установлен.

Флаги джойстика

Управляют состояниями джойстика.

См. также

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

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

Флаги контроля позиции

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

См. также

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

- `#define CTP_ENABLED 0x01`
Контроль позиции включен, если флаг установлен.
- `#define CTP_BASE 0x02`
Опорой является датчик оборотов, если флаг установлен; иначе - энкодер.
- `#define CTP_ALARM_ON_ERROR 0x04`
Войти в состояние ALARM при расхождении позиции, если флаг установлен.
- `#define REV_SENS_INV 0x08`
Сенсор считается активным, когда на нём 0, а инвертирование делает активным уровнем 1.
- `#define CTP_ERROR_CORRECTION 0x10`
Корректировать ошибки, возникающие при проскальзывании, если флаг установлен.

Флаги настроек команды home

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

См. также

[get_home_settings](#)
[set_home_settings](#)
[command_home](#)
[home_settings_t::HomeFlags](#), [get_home_settings](#), [set_home_settings](#)

- `#define HOME_DIR_FIRST 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 концевика.
- `#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 концевика.
- `#define HOME_USE_FAST 0x100`
 Если флаг установлен, используется быстрый поиск домашней позиции; иначе - традиционный.

Флаги настроек четности команды uart

См. также

[uart_settings_t::UARTSetupFlags](#), [get_uart_settings](#), [set_uart_settings](#)

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

Флаг типа двигателя

См. также

[motor_settings_t::MotorType](#), [get_motor_settings](#), [set_motor_settings](#)

- `#define MOTOR_TYPE_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

См. также

[accessories_settings_t::MBSettings](#), [get_accessories_settings](#), [set_accessories_settings](#)

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

Флаги настроек температурного датчика

См. также

[accessories_settings_t::LimitSwitchesSettings](#), [get_accessories_settings](#), [set_accessories_settings](#)

- `#define TS_TYPE_BITS 0x07`
Биты, отвечающие за тип температурного датчика.
- `#define TS_TYPE_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 концевик, подключенный к ножке SW1, доступен
- `#define LS_ON_SW2_AVAILABLE 0x02`
Если флаг установлен, то концевик, подключенный к ножке SW2, доступен

- `#define LS_SW1_ACTIVE_LOW 0x04`
Если флаг установлен, то концевик, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте
- `#define LS_SW2_ACTIVE_LOW 0x08`
Если флаг установлен, то концевик, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте
- `#define LS_SHORTED 0x10`
Если флаг установлен, то концевики закорочены.

Флаги автоопределения характеристик обмоток двигателя.

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 конечных 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_calibration_settings (device_t id, const calibration_settings_t *calibration_settings)`
Команда записи калибровочных коэффициентов.
- `result_t XIMC_API get_calibration_settings (device_t id, calibration_settings_t *calibration_settings)`
Команда чтения калибровочных коэффициентов.
- `result_t XIMC_API set_controller_name (device_t id, const controller_name_t *controller_name)`
Запись пользовательского имени контроллера и настроек в FRAM.
- `result_t XIMC_API get_controller_name (device_t id, controller_name_t *controller_name)`
Чтение пользовательского имени контроллера и настроек из FRAM.
- `result_t XIMC_API set_nonvolatile_memory (device_t id, const nonvolatile_memory_t *nonvolatile_memory)`
Запись пользовательских данных во FRAM.
- `result_t XIMC_API get_nonvolatile_memory (device_t id, nonvolatile_memory_t *nonvolatile_memory)`
Чтение пользовательских данных из FRAM.
- `result_t XIMC_API set_emf_settings (device_t id, const emf_settings_t *emf_settings)`
Запись электромеханических настроек шагового двигателя.
- `result_t XIMC_API get_emf_settings (device_t id, emf_settings_t *emf_settings)`
Чтение электромеханических настроек шагового двигателя.
- `result_t XIMC_API set_engine_advanced_setup (device_t id, const engine_advanced_setup_t *engine_advanced_setup)`
Запись расширенных настроек.
- `result_t XIMC_API get_engine_advanced_setup (device_t id, engine_advanced_setup_t *engine_advanced_setup)`
Чтение расширенных настроек.
- `result_t XIMC_API set_extended_settings (device_t id, const extended_settings_t *extended_settings)`
Запись расширенных настроек.
- `result_t XIMC_API get_extended_settings (device_t id, extended_settings_t *extended_settings)`
Чтение расширенных настроек.

Группа команд управления движением

- `result_t XIMC_API command_stop (device_t id)`
Немедленная остановка двигателя, переход в состояние STOP, ключи в режиме BREAK (обмотки накоротко замкнуты), режим "удержания" деактивируется для ДС 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)`
Поля скоростей знаковые.
- `result_t XIMC_API command_left (device_t id)`
При получении команды "left" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), влево.
- `result_t XIMC_API command_right (device_t id)`
При получении команды "right" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), вправо.
- `result_t XIMC_API command_loft (device_t id)`
При получении команды "loft" двигатель смещается из текущей точки на расстояние GENG:::-Antiplay, затем двигается в ту же точку.
- `result_t XIMC_API command_sstp (device_t id)`
Плавная остановка.
- `result_t XIMC_API get_position (device_t id, get_position_t *the_get_position)`
Считывает значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.
- `result_t XIMC_API get_position_calb (device_t id, get_position_calb_t *the_get_position_calb, const calibration_t *calibration)`
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 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 и позицию в которую осуществляется движение по командам move и movr равными нулю для всех случаев, кроме движения к позиции назначения.

Группа команд сохранения и загрузки настроек

- `result_t XIMC_API command_save_settings (device_t id)`
При получении команды контроллер выполняет операцию сохранения текущих настроек во встроенную энергонезависимую память контроллера.
- `result_t XIMC_API command_read_settings (device_t id)`
Чтение всех настроек контроллера из flash памяти в оперативную, заменяя текущие настройки.
- `result_t XIMC_API command_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, которые непосредственно связаны с позиционером и не меняются без его механической переделки.
- `result_t XIMC_API command_eeread_settings (device_t id)`
Чтение настроек контроллера из EEPROM памяти позиционера, которые непосредственно связаны с позиционером и не меняются без его механической переделки.
- `result_t XIMC_API 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 прошивки 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.
- `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)`
Устанавливает ключ шифрования сетевой подсистемы (bindy).
- `device_enumeration_t XIMC_API enumerate_devices (int enumerate_flags, const char *hints)`
Перечисляет все XIMC-совместимые устройства.
- `result_t XIMC_API free_enumerate_devices (device_enumeration_t device_enumeration)`
Освобождает память, выделенную enumerate_devices.
- `int XIMC_API get_device_count (device_enumeration_t device_enumeration)`
Возвращает количество подключенных устройств.
- `pchar XIMC_API get_device_name (device_enumeration_t device_enumeration, int device_index)`
Возвращает имя подключенного устройства из перечисления устройств.
- `result_t XIMC_API get_enumerate_device_serial (device_enumeration_t device_enumeration, int device_index, uint32_t *serial)`
Возвращает серийный номер подключенного устройства из перечисления устройств.
- `result_t XIMC_API get_enumerate_device_information (device_enumeration_t device_enumeration, int device_index, device_information_t *device_information)`
Возвращает 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 ()`
Снимает блокировку библиотеки в экстренном случае.
- `result_t XIMC_API ximc_fix_usbser_sys (const char *device_uri)`
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 при достижении обеих границ.

Нужен для предотвращения поломки двигателя при неправильных настройках концевиков

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 CONTROL_BTN_LEFT_PUSHED_OPEN 0x04`

Левая кнопка нормально разомкнутая, если флаг установлен.

7.1.2.12 `#define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08`

Правая кнопка нормально разомкнутая, если флаг установлен.

7.1.2.13 `#define CONTROL_MODE_BITS 0x03`

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

7.1.2.14 `#define CONTROL_MODE_JOY 0x01`

Управление с помощью джойстика.

7.1.2.15 `#define CONTROL_MODE_LR 0x02`

Управление с помощью кнопок left/right.

7.1.2.16 `#define CONTROL_MODE_OFF 0x00`

Управление отключено.

7.1.2.17 `#define CTP_ALARM_ON_ERROR 0x04`

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

7.1.2.18 `#define CTP_BASE 0x02`

Опорой является датчик оборотов, если флаг установлен; иначе - энкодер.

7.1.2.19 `#define CTP_ENABLED 0x01`

Контроль позиции включен, если флаг установлен.

7.1.2.20 `#define CTP_ERROR_CORRECTION 0x10`

Корректировать ошибки, возникающие при проскальзывании, если флаг установлен.

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

7.1.2.21 `#define DRIVER_TYPE_DISCRETE_FET 0x01`

Силовой драйвер на дискретных мосфет-ключках.

Используется по умолчанию.

7.1.2.22 `#define DRIVER_TYPE_EXTERNAL 0x03`

Внешний силовой драйвер.

7.1.2.23 `#define DRIVER_TYPE_INTEGRATE 0x02`

Силовой драйвер с использованием ключей, интегрированных в микросхему.

7.1.2.24 `#define EEPROM_PRECEDENCE 0x01`

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

7.1.2.25 `#define ENC_STATE_ABSENT 0x00`

Энкодер не подключен.

7.1.2.26 `#define ENC_STATE_MALFUNC 0x02`

Энкодер подключен и неисправен.

7.1.2.27 `#define ENC_STATE_OK 0x04`

Энкодер подключен и работает адекватно.

7.1.2.28 `#define ENC_STATE_REVERS 0x03`

Энкодер подключен и исправен, но считает в другую сторону.

7.1.2.29 `#define ENC_STATE_UNKNOWN 0x01`

Состояние энкодера неизвестно.

7.1.2.30 `#define ENDER_SW1_ACTIVE_LOW 0x02`

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

7.1.2.31 `#define ENDER_SW2_ACTIVE_LOW 0x04`

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

7.1.2.32 `#define ENDER_SWAP 0x01`

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

7.1.2.33 `#define ENGINE_ACCEL_ON 0x10`

Ускорение.

Если флаг установлен, движение происходит с ускорением.

7.1.2.34 `#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.

7.1.2.35 `#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 (blde).

7.1.2.36 `#define ENGINE_LIMIT_CURR 0x40`

Номинальный ток мотора.

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

7.1.2.37 `#define ENGINE_LIMIT_RPM 0x80`

Номинальная частота вращения мотора.

Если флаг установлен, частота вращения ограничивается заданным номинальным значением.

7.1.2.38 `#define ENGINE_LIMIT_VOLT 0x20`

Номинальное напряжение мотора.

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

7.1.2.39 `#define ENGINE_MAX_SPEED 0x04`

Флаг максимальной скорости.

Если флаг установлен, движение происходит на максимальной скорости.

7.1.2.40 `#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.

7.1.2.41 `#define ENGINE_TYPE_2DC 0x02`

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

7.1.2.42 `#define ENGINE_TYPE_BRUSHLESS 0x05`

Безщеточный мотор.

7.1.2.43 `#define ENGINE_TYPE_DC 0x01`

Мотор постоянного тока.

7.1.2.44 `#define ENGINE_TYPE_NONE 0x00`

Это значение не нужно использовать.

7.1.2.45 `#define ENGINE_TYPE_STEP 0x03`

Шаговый мотор.

7.1.2.46 `#define ENGINE_TYPE_TEST 0x04`

Скважность в обмотках фиксирована.

Используется только производителем.

7.1.2.47 `#define ENUMERATE_PROBE 0x01`

Проверять, является ли устройство XIMC-совместимым.

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

7.1.2.48 `#define EXTIO_SETUP_INVERT 0x02`

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

7.1.2.49 `#define EXTIO_SETUP_MODE_IN_ALARM 0x05`

Войти в состояние ALARM при переходе сигнала в активное состояние.

7.1.2.50 `#define EXTIO_SETUP_MODE_IN_BITS 0x0F`

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

7.1.2.51 `#define EXTIO_SETUP_MODE_IN_HOME 0x04`

Выполняется команда HOME.

7.1.2.52 `#define EXTIO_SETUP_MODE_IN_MOVR 0x03`

Выполняется команда MOVR с последними настройками.

7.1.2.53 `#define EXTIO_SETUP_MODE_IN_NOP 0x00`

Ничего не делать.

7.1.2.54 `#define EXTIO_SETUP_MODE_IN_PWOF 0x02`

Выполняет команду PWOF, обесточивая обмотки двигателя.

7.1.2.55 `#define EXTIO_SETUP_MODE_IN_STOP 0x01`

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

7.1.2.56 `#define EXTIO_SETUP_MODE_OUT_ALARM 0x30`

Ножка находится в активном состоянии при нахождении в состоянии ALARM.

7.1.2.57 `#define EXTIO_SETUP_MODE_OUT_BITS 0xF0`

Биты выбора поведения на выходе.

7.1.2.58 `#define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40`

Ножка находится в активном состоянии при подаче питания на обмотки.

7.1.2.59 `#define EXTIO_SETUP_MODE_OUT_MOVING 0x20`

Ножка находится в активном состоянии при движении.

7.1.2.60 `#define EXTIO_SETUP_MODE_OUT_OFF 0x00`

Ножка всегда в неактивном состоянии.

7.1.2.61 `#define EXTIO_SETUP_MODE_OUT_ON 0x10`

Ножка всегда в активном состоянии.

7.1.2.62 `#define EXTIO_SETUP_OUTPUT 0x01`

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

7.1.2.63 `#define FEEDBACK_EMF 0x04`

Обратная связь по ЭДС.

7.1.2.64 `#define FEEDBACK_ENC_REVERSE 0x01`

Обратный счет у энкодера.

7.1.2.65 `#define FEEDBACK_ENC_TYPE_AUTO 0x00`

Определять тип энкодера автоматически.

7.1.2.66 `#define FEEDBACK_ENC_TYPE_BITS 0xC0`

Биты, отвечающие за тип энкодера.

7.1.2.67 `#define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80`

Дифференциальный энкодер.

7.1.2.68 `#define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40`

Недифференциальный энкодер.

7.1.2.69 `#define FEEDBACK_ENCODER 0x01`

Обратная связь с помощью энкодера.

7.1.2.70 `#define FEEDBACK_ENCODER_MEDIATED 0x06`

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

7.1.2.71 `#define FEEDBACK_NONE 0x05`

Обратная связь отсутствует.

7.1.2.72 `#define HOME_DIR_FIRST 0x001`

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

7.1.2.73 `#define HOME_DIR_SECOND 0x002`

Определяет направление второго движения мотора.

Если флаг установлен - вправо; иначе - влево.

7.1.2.74 `#define HOME_HALF_MV 0x008`

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

7.1.2.75 `#define HOME_MV_SEC_EN 0x004`

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

7.1.2.76 `#define HOME_STOP_FIRST_BITS 0x030`

Биты, отвечающие за выбор сигнала завершения первого движения.

7.1.2.77 `#define HOME_STOP_FIRST_LIM 0x030`

Первое движение завершается по сигналу с концевика.

7.1.2.78 `#define HOME_STOP_FIRST_REV 0x010`

Первое движение завершается по сигналу с Revolution sensor.

7.1.2.79 `#define HOME_STOP_FIRST_SYN 0x020`

Первое движение завершается по сигналу со входа синхронизации.

7.1.2.80 `#define HOME_STOP_SECOND_BITS 0x0C0`

Биты, отвечающие за выбор сигнала завершения второго движения.

7.1.2.81 `#define HOME_STOP_SECOND_LIM 0x0C0`

Второе движение завершается по сигналу с концевика.

7.1.2.82 `#define HOME_STOP_SECOND_REV 0x040`

Второе движение завершается по сигналу с Revolution sensor.

7.1.2.83 `#define HOME_STOP_SECOND_SYN 0x080`

Второе движение завершается по сигналу со входа синхронизации.

7.1.2.84 `#define HOME_USE_FAST 0x100`

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

7.1.2.85 `#define JOY_REVERSE 0x01`

Реверс воздействия джойстика.

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

7.1.2.86 `#define LOW_UPWR_PROTECTION 0x02`

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

7.1.2.87 `#define LS_SHORTED 0x10`

Если флаг установлен, то концевики закорочены.

7.1.2.88 `#define MICROSTEP_MODE_FRAC_128 0x08`

Деление шага $1/128$.

7.1.2.89 `#define MICROSTEP_MODE_FRAC_16 0x05`

Деление шага $1/16$.

7.1.2.90 `#define MICROSTEP_MODE_FRAC_2 0x02`

Деление шага $1/2$.

7.1.2.91 `#define MICROSTEP_MODE_FRAC_256 0x09`

Деление шага $1/256$.

7.1.2.92 `#define MICROSTEP_MODE_FRAC_32 0x06`

Деление шага $1/32$.

7.1.2.93 `#define MICROSTEP_MODE_FRAC_4 0x03`

Деление шага $1/4$.

7.1.2.94 `#define MICROSTEP_MODE_FRAC_64 0x07`

Деление шага $1/64$.

7.1.2.95 `#define MICROSTEP_MODE_FRAC_8 0x04`

Деление шага $1/8$.

7.1.2.96 `#define MICROSTEP_MODE_FULL 0x01`

Полношаговый режим.

7.1.2.97 `#define MOVE_STATE_ANTIPLAY 0x04`

Выполняется компенсация люфта, если флаг установлен.

7.1.2.98 `#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`.

7.1.2.99 `#define MOVE_STATE_TARGET_SPEED 0x02`

Флаг устанавливается при достижении заданной скорости.

7.1.2.100 `#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.

7.1.2.101 `#define MVCMD_HOME 0x06`

Команда home.

7.1.2.102 `#define MVCMD_LEFT 0x03`

Команда left.

7.1.2.103 `#define MVCMD_LOFT 0x07`

Команда loft.

7.1.2.104 `#define MVCMD_MOVE 0x01`

Команда move.

7.1.2.105 `#define MVCMD_MOVR 0x02`

Команда movr.

7.1.2.106 `#define MVCMD_NAME_BITS 0x3F`

Битовая маска активной команды.

7.1.2.107 `#define MVCMD_RIGHT 0x04`

Команда rigt.

7.1.2.108 `#define MVCMD_RUNNING 0x80`

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

7.1.2.109 `#define MVCMD_SSTP 0x08`

Команда плавной остановки(SSTP).

7.1.2.110 `#define MVCMD_STOP 0x05`

Команда stop.

7.1.2.111 `#define MVCMD_UNKNWN 0x00`

Неизвестная команда.

7.1.2.112 `#define POWER_OFF_ENABLED 0x02`

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

Иначе - не снимать.

7.1.2.113 `#define POWER_REDUCT_ENABLED 0x01`

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

Иначе - не уменьшать.

7.1.2.114 `#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.

7.1.2.115 `#define PWR_STATE_MAX 0x05`

Обмотки запитаны максимально доступным током, который может выдать схема при данном напряжении питания.

7.1.2.116 `#define PWR_STATE_NORM 0x03`

Обмотки запитаны номинальным током.

7.1.2.117 `#define PWR_STATE_OFF 0x01`

Обмотки мотора разомкнуты и не управляются драйвером.

7.1.2.118 `#define PWR_STATE_REDUCT 0x04`

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

7.1.2.119 `#define PWR_STATE_UNKNOWN 0x00`

Неизвестное состояние, которое не должно никогда реализовываться.

7.1.2.120 `#define REV_SENS_INV 0x08`

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

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

7.1.2.121 `#define RPM_DIV_1000 0x01`

Флаг указывает на то что рабочая скорость указанная в команде задана в милли грм.
Применим только для режима обратной связи ENCODER и только для BLDC моторов.

7.1.2.122 `#define SETPOS_IGNORE_ENCODER 0x02`

Если установлен, то счётчик энкодера не обновляется.

7.1.2.123 `#define SETPOS_IGNORE_POSITION 0x01`

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

7.1.2.124 `#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.

7.1.2.125 `#define STATE_BORDERS_SWAP_MISSET 0x0008000`

Достижение неверной границы.

7.1.2.126 `#define STATE_BRAKE 0x0200`

Состояние вывода управления тормозом.

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

7.1.2.127 `#define STATE_BUTTON_LEFT 0x0008`

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

7.1.2.128 `#define STATE_BUTTON_RIGHT 0x0004`

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

7.1.2.129 `#define STATE_CONTR 0x000003F`

Флаги состояния контроллера.

7.1.2.130 `#define STATE_CONTROLLER_OVERHEAT 0x0000200`

Перегрелась микросхема контроллера.

7.1.2.131 `#define STATE_CTP_ERROR 0x0000080`

Контроль позиции нарушен(используется только с шаговым двигателем).

7.1.2.132 `#define STATE_DIG_SIGNAL 0xFFFF`

Флаги цифровых сигналов.

7.1.2.133 `#define STATE_EEPROM_CONNECTED 0x0000010`

Подключена память EEPROM с настройками.

7.1.2.134 `#define STATE_ENC_A 0x2000`

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

7.1.2.135 `#define STATE_ENC_B 0x4000`

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

7.1.2.136 `#define STATE_ENGINE_RESPONSE_ERROR 0x0800000`

Ошибка реакции двигателя на управляющее воздействие.

7.1.2.137 `#define STATE_ERRC 0x0000001`

Недопустимая команда.

7.1.2.138 `#define STATE_ERRD 0x0000002`

Нарушение целостности данных.

7.1.2.139 `#define STATE_ERRV 0x0000004`

Недопустимое значение данных.

7.1.2.140 `#define STATE_EXTIO_ALARM 0x1000000`

Ошибка вызвана входным сигналом.

7.1.2.141 `#define STATE_GPIO_LEVEL 0x0020`

Состояние ввода/вывода общего назначения.

7.1.2.142 `#define STATE_GPIO_PINOUT 0x0010`

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

7.1.2.143 `#define STATE_LEFT_EDGE 0x0002`

Достижение левой границы.

7.1.2.144 `#define STATE_LOW_USB_VOLTAGE 0x0002000`

Слишком низкое напряжение на USB.

7.1.2.145 `#define STATE_OVERLOAD_POWER_CURRENT 0x0000800`

Превышен максимальный ток потребления силовой части.

7.1.2.146 `#define STATE_OVERLOAD_POWER_VOLTAGE 0x0000400`

Превышено напряжение на силовой части.

7.1.2.147 `#define STATE_OVERLOAD_USB_CURRENT 0x0004000`

Превышен максимальный ток потребления USB.

7.1.2.148 `#define STATE_OVERLOAD_USB_VOLTAGE 0x0001000`

Превышено напряжение на USB.

7.1.2.149 `#define STATE_POWER_OVERHEAT 0x0000100`

Перегрелась силовая часть платы.

7.1.2.150 `#define STATE_REV_SENSOR 0x0400`

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

7.1.2.151 `#define STATE_RIGHT_EDGE 0x0001`

Достижение правой границы.

7.1.2.152 `#define STATE_SECUR 0x1B3FFC0`

Флаги опасности.

7.1.2.153 `#define STATE_SYNC_INPUT 0x0800`

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

7.1.2.154 `#define STATE_SYNC_OUTPUT 0x1000`

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

7.1.2.155 `#define SYNCIN_ENABLED 0x01`

Включение необходимости импульса синхронизации для начала движения.

7.1.2.156 `#define SYNCIN_INVERT 0x02`

Если установлен - срабатывает по переходу из 1 в 0.

Иначе - из 0 в 1.

7.1.2.157 `#define SYNCOUT_ENABLED 0x01`

Синхронизация выхода работает согласно настройкам, если флаг установлен.

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

7.1.2.158 `#define SYNCOUT_IN_STEPS 0x08`

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

7.1.2.159 `#define SYNCOUT_INVERT 0x04`

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

7.1.2.160 `#define SYNCOUT_ONPERIOD 0x40`

Выдать импульс синхронизации после прохождения `SyncOutPeriod` отсчётов.

7.1.2.161 `#define SYNCOUT_ONSTART 0x10`

Генерация синхронизирующего импульса при начале движения.

7.1.2.162 `#define SYNCOUT_ONSTOP 0x20`

Генерация синхронизирующего импульса при остановке.

7.1.2.163 `#define SYNCOUT_STATE 0x02`

Когда значение выхода управляется напрямую (см.

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

7.1.2.164 `#define TS_TYPE_BITS 0x07`

Биты, отвечающие за тип температурного датчика.

7.1.2.165 `#define UART_PARITY_BITS 0x03`

Биты, отвечающие за выбор четности.

7.1.2.166 `#define WIND_A_STATE_ABSENT 0x00`

Обмотка А не подключена.

7.1.2.167 `#define WIND_A_STATE_MALFUNC 0x02`

Короткое замыкание на обмотке A.

7.1.2.168 `#define WIND_A_STATE_OK 0x03`

Обмотка A работает адекватно.

7.1.2.169 `#define WIND_A_STATE_UNKNOWN 0x01`

Состояние обмотки A неизвестно.

7.1.2.170 `#define WIND_B_STATE_ABSENT 0x00`

Обмотка B не подключена.

7.1.2.171 `#define WIND_B_STATE_MALFUNC 0x20`

Короткое замыкание на обмотке B.

7.1.2.172 `#define WIND_B_STATE_OK 0x30`

Обмотка B работает адекватно.

7.1.2.173 `#define WIND_B_STATE_UNKNOWN 0x10`

Состояние обмотки B неизвестно.

7.1.2.174 `#define XIMC_API`

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

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

7.1.3 Типы

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

Прототип функции обратного вызова для логирования

Аргументы

loglevel	уровень логирования
message	сообщение

7.1.4 Функции

7.1.4.1 `result_t XIMC_API close_device (device_t * id)`

Закрывает устройство

Аргументы

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

7.1.4.2 result_t XIMC_API command_clear_fram (device_t id)

Очистка FRAM памяти контроллера.

Функция используется только производителем.

Аргументы

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

7.1.4.3 result_t XIMC_API command_eeread_settings (device_t id)

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

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

Аргументы

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

7.1.4.4 result_t XIMC_API command_eesave_settings (device_t id)

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

Функция должна использоваться только производителем.

Аргументы

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

7.1.4.5 result_t XIMC_API command_home (device_t id)

Поля скоростей знаковые.

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

Аргументы

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

См. также

[home_settings_t](#)
[get_home_settings](#)
[set_home_settings](#)

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

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

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

id	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 GENG:::-Antiplay, затем двигается в ту же точку.

Аргументы

id	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

id	идентификатор устройства
Position	заданная позиция.

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

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

id	идентификатор устройства
Position	позиция для перемещения
calibration	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 DC мотора это поле не используется.

Аргументы

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

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 "ТТЛСинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), влево или вправо (зависит от знака DeltaPosition) на расстояние указанное в поле DeltaPosition.

Аргументы

DeltaPosition	смещение.
id	идентификатор устройства
calibration	настройки пользовательских единиц

Заметки

Конечная координата вычисляемая с помощью `DeltaPosition`, корректируется таблицей коррекции. Для корректного расчета координат, при использовании корректирующей таблицы, не нужно выполнять команды `movt` пакетами.

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 отключении после остановки следует использовать систему управления электропитанием.

Аргументы

id	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.

Аргументы

id	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

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

7.1.4.16 `result_t XIMC_API command_reset (device_t id)`

Перезагрузка контроллера.

Функция используется только производителем.

Аргументы

id	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

id	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.

Аргументы

id	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

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

7.1.4.20 result_t XIMC_API command_sstp (device_t id)

Плавная остановка.

Двигатель останавливается с ускорением замедления.

Аргументы

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

7.1.4.21 result_t XIMC_API command_start_measurements (device_t id)

Начать измерения и буферизацию скорости, ошибки следования.

Аргументы

id	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).

Аргументы

id	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

Аргументы

uri	идентификатор устройства
data	указатель на массив байтов прошивки
data_size	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

	id	идентификатор устройства
	refresh_interval_ms	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	ret	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 и позицию в которую осуществляется движение по командам `move` и `movt` равными нулю для всех случаев, кроме движения к позиции назначения.

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

Аргументы

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

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

Перечисляет все XIMC-совместимые устройства.

Аргументы

in	enumerate_ - flags	флаги поиска устройств
in	hints	дополнительная информация для поиска hints это строка вида "ключ1=значение1 \n ключ2=значение2". Неизвестные пары ключ-значение игнорируются. Список ключей: addr - используется вместе с флагом ENUMERATE_NETWORK. Ненулевое значение это адрес или список адресов с перечислением через запятую удаленных хостов на которых происходит поиск устройств, отсутствующее значение это подключение посредством широковещательного запроса. adapter_addr - используется вместе с флагом ENUMERATE_NETWORK. Ненулевое значение это IP адрес сетевого адаптера. Сетевое устройство ximc должно быть в локальной сети, к которой подключён этот адаптер. При использовании ключа adapter_addr обязательно установить ключ addr. Пример: "addr= \n adapter_addr=192.168.0.100". Для перечисления сетевых устройств обязательно нужно сначала вызвать функцию установки сетевого ключа set_bindy_key .

7.1.4.27 result_t XIMC_API free_enumerate_devices (device_enumeration_t device_enumeration)

Освобождает память, выделенную enumerate_devices.

Аргументы

in	device_ - enumeration	закрытый указатель на данные о перечисленных устройствах
----	-----------------------	--

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

	id	идентификатор устройства
out	accessories_ - settings	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

	id	идентификатор устройства
out	analog_data	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 прошивки контроллера.

Аргументы

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

	id	идентификатор устройства
out	brake_settings	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.

Эта функция заполняет структуру калибровочных коэффициентов.

См. также

[calibration_settings_t](#)

Аргументы

	id	идентификатор устройства
out	calibration_settings	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

	id	идентификатор устройства
out	chart_data	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

	id	идентификатор устройства
out	control_settings	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

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

	id	идентификатор устройства
out	controller_name	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 оборот (GENG::StepsPerRev) и разрешение энкодера

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

Аргументы

	id	идентификатор устройства
out	ctp_settings	структура, содержащая настройки контроля позиции

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.

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

Аргументы

	id	идентификатор устройства
out	debug_read	Данные для отладки.

7.1.4.39 int XIMC_API get_device_count (device_enumeration_t device_enumeration)

Возвращает количество подключенных устройств.

Аргументы

in	device_enumeration	закрытый указатель на данные о перечисленных 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

	id	идентификатор устройства.
out	device_information	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	device_ - enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	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

	id	идентификатор устройства
out	edges_settings	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

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

Заметки

Внимание! Некоторые параметры структуры edges_settings_calb корректируются таблицей коррекции координат.

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

	id	идентификатор устройства
out	emf_settings	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

	id	идентификатор устройства
out	encoder_information	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

	id	идентификатор устройства
out	encoder_settings	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

	id	идентификатор устройства
out	engine_advanced_setup	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

	id	идентификатор устройства
out	engine_settings	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

	id	идентификатор устройства
out	engine_settings_calb	структура с настройками мотора
	calibration	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

	id	идентификатор устройства
out	entype_settings	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	device_enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	номер устройства
out	controller	name 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	device_ - enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	номер устройства
out	device_ - information	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	device_ - enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	номер устройства
out	device_ - network_ - information	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	device_ - enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	номер устройства
in	serial	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 подвигки для подключенного устройства из перечисления устройств.

Возвращает имя подвигки устройства с номером device_index.

Аргументы

in	device_ - enumeration	закрытый указатель на данные о перечисленных устройствах
----	--------------------------	--

in	device_index	номер устройства
out	stage	name имя подвижки

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

Чтение расширенных настроек.

См. также

[set_extended_settings](#)

Аргументы

	id	идентификатор устройства
out	extended_settings	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

	id	идентификатор устройства
out	extio_settings	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

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

```
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

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

	id	идентификатор устройства
out	gear_ - information	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

	id	идентификатор устройства
out	gear_settings	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.

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

Аргументы

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

```
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

	id	идентификатор устройства
out	hallsensor_ information	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

	id	идентификатор устройства
out	hallsensor_ settings	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

	id	идентификатор устройства
out	home_settings	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, используемых для калибровки позиции, в память контроллера.

См. также

[home_settings_calb_t](#)

Аргументы

	id	идентификатор устройства
out	home_settings_ calb	настройки калибровки позиции
	calibration	настройки пользовательских единиц

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

Чтение случайного числа из контроллера.

Аргументы

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

Аргументы

	id	идентификатор устройства
out	joystick_settings	структура, содержащая настройки джойстика

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". Буффер вмещает 25 точек, точки снимаются с периодом 1 мс. Для создания устойчивой системы следует считывать данные каждые 20 мс, если буффер полностью заполнен, то рекомендуется повторять считывания каждые 5 мс до момента пока буффер вновь не станет заполнен 20-ю точками.

См. также

[measurements_t](#)

Аргументы

	id	идентификатор устройства
out	measurements	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

	id	идентификатор устройства
out	motor_ information	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

	id	идентификатор устройства
out	motor_settings	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 антилюфта).

Аргументы

	id	идентификатор устройства
out	move_settings	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

	id	идентификатор устройства
out	move_settings- _calb	структура, содержащая настройки движения: скорость, ускорение, и т.д.
	calibration	настройки пользовательских единиц

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

Чтение пользовательских данных из FRAM.

Аргументы

	id	идентификатор устройства
out	nonvolatile_ memory	структура, содержащая установленные пользовательские данные

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

Чтение ПИД коэффициентов.

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

См. также

[set_pid_settings](#)

Аргументы

	id	идентификатор устройства
out	pid_settings	настройки ПИД

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

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

Аргументы

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

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

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

Аргументы

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

Заметки

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

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

Команда чтения параметров питания мотора.

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

Аргументы

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


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

Команда записи установок защит.

Аргументы

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

См. также

status_t::flags

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

Чтение серийного номера контроллера.

Аргументы

	id	идентификатор устройства
out	SerialNumber	серийный номер контроллера

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

Чтение информации о позиционере из EEPROM.

Аргументы

	id	идентификатор устройства
out	stage_information	структура, содержащая информацию о позиционере

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

Чтение пользовательского имени подвижки из EEPROM.

Аргументы

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

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

Чтение настроек позиционера из EEPROM.

Аргументы

	id	идентификатор устройства
out	stage_settings	структура, содержащая настройки позиционера

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

Возвращает информацию о текущем состоянии устройства.

Аргументы

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

См. также

[get_status](#)

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

Возвращает информацию о текущем состоянии устройства.

Аргументы

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

См. также

[get_status](#)

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

Чтение настроек для входного импульса синхронизации.

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

См. также

[set_sync_in_settings](#)

Аргументы

	id	идентификатор устройства
out	sync_in_settings	настройки синхронизации

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

Чтение настроек для входного импульса синхронизации с использованием пользовательских единиц. Эта функция считывает структуру с настройками синхронизации, определяющими поведение входа синхронизации, в память контроллера.

См. также

[set_sync_in_settings_calb](#)

Аргументы

	id	идентификатор устройства
out	sync_in_settings_calb	настройки синхронизации
	calibration	настройки пользовательских единиц

```
7.1.4.88 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.

```
7.1.4.89 result_t XIMC_API get_sync_out_settings_calb ( device_t id, sync_out_settings_calb_t
* sync_out_settings_calb, const calibration_t * calibration )
```

Чтение настроек для выходного импульса синхронизации с использованием пользовательских единиц.

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

См. также

[set_sync_in_settings_calb](#)

Аргументы

	id	идентификатор устройства
in	sync_out_settings_calb	настройки синхронизации
	calibration	настройки пользовательских единиц

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

Команда чтения настроек UART.

Эта функция заполняет структуру настроек UART.

См. также

[uart_settings_t](#)

Аргументы

	Speed	Скорость UART
out	uart_settings	настройки UART

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

Перезагрузка в прошивку в контроллере

Аргументы

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

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

Проверка наличия прошивки в контроллере

Аргументы

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

7.1.4.93 `result_t XIMC_API load_correction_table (device_t * id, const char * namefile)`

Команда загрузки корректирующей таблицы из текстового файла.

Таблица используется для коррекции положения в случае механических неточностей. Работает для некоторых параметров в `_calb` командах.

Аргументы

	id	- идентификатор устройства
in	namefile	- 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.

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](#)

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

Простая функция логирования на stderr в узких (однобайтных) символах

Аргументы

loglevel	уровень логирования
message	сообщение

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

Простая функция логирования на stderr в широких символах

Аргументы

loglevel	уровень логирования
message	сообщение

7.1.4.96 void XIMC_API msec_sleep (unsigned int msec)

Приостанавливает работу на указанное время

Аргументы

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

7.1.4.97 device_t XIMC_API open_device (const char * uri)

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

Аргументы

in	uri	- уникальный идентификатор устройства. Uri устройства имеет вид "xi-com:port" или "xi-net://host/serial" или "xi-emu:///file". Для USB-COM устройства "port" это uri устройства в ОС. Например "xi-com:\\.\\COM3" в Windows или "xi-com:/dev/tty.s123" 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". Замечание: для открытия сетевого устройства обязательно нужно сначала вызвать функцию установки сетевого ключа set_bindy_key . Для виртуального устройства "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.
----	-----	--

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

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

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

Аргументы

in	uri	- уникальный идентификатор устройства
----	-----	---------------------------------------

7.1.4.99 `result_t XIMC_API service_command_updf (device_t id)`

Команда переводит контроллер в режим обновления прошивки.

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

7.1.4.100 `result_t XIMC_API set_accessories_settings (device_t id, const accessories_settings_t * accessories_settings)`

Запись информации о дополнительных аксессуарах в EEPROM.

Функция должна использоваться только производителем.

Аргументы

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

7.1.4.101 `result_t XIMC_API set_bindy_key (const char * keyfilepath)`

Устанавливает ключ шифрования сетевой подсистемы (bindy).

Аргументы

in	keyfilepath	полный путь к файлу ключа В случае использования сетевых устройств эта функция должна быть вызвана до функций enumerate_devices и open_device .
----	-------------	---

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

Запись настроек управления тормозом.

Аргументы

	id	идентификатор устройства
in	brake_settings	структура, содержащая настройки управления тормозом

7.1.4.103 `result_t XIMC_API set_calibration_settings (device_t id, const calibration_settings_t * calibration_settings)`

Команда записи калибровочных коэффициентов.

Эта функция записывает структуру калибровочных коэффициентов в память контроллера.

См. также

[calibration_settings_t](#)

Аргументы

	id	идентификатор устройства
in	calibration_ settings	калибровочные коэффициенты

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

Запись настроек управления мотором.

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

Аргументы

	id	идентификатор устройства
in	control_ settings	структура, содержащая настройки управления мотором с помощью джойстика или кнопок влево/вправо.

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

Запись настроек управления мотором с использованием пользовательских единиц.

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

Аргументы

	id	идентификатор устройства
in	control_ settings_calb	структура, содержащая настройки управления мотором с помощью джойстика или кнопок влево/вправо.
	calibration	настройки пользовательских единиц

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

Запись пользовательского имени контроллера и настроек в FRAM.

Аргументы

	id	идентификатор устройства
in	controller_information	структура, содержащая информацию о контроллере

7.1.4.107 `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

	id	идентификатор устройства
in	ctp_settings	структура, содержащая настройки контроля позиции

7.1.4.108 `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.

Аргументы

	id	идентификатор устройства
in	debug_write	Данные для отладки.

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

Запись настроек границ и концевых выключателей.

См. также

[get_edges_settings](#)

Аргументы

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

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

Запись настроек границ и концевых выключателей с использованием пользовательских единиц.

См. также

[get_edges_settings_calb](#)

Аргументы

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

Заметки

Внимание! Некоторые параметры структуры edges_settings_calb корректируются таблицей коррекции координат.

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

Запись электромеханических настроек шагового двигателя.

Настройки различны для разных двигателей. Пожалуйста, загружайте новые настройки, когда вы меняете мотор.

См. также

[get_emf_settings](#)

Аргументы

	id	идентификатор устройства
in	emf_settings	настройки EMF

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

Запись информации об энкодере в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	encoder_information	структура, содержащая информацию об энкодере

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

Запись настроек энкодера в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	encoder _ - settings	структура, содержащая настройки энкодера

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

Запись расширенных настроек.

См. также

[get_engine_advansed_setup](#)

Аргументы

	id	идентификатор устройства
in	engine _ - advansed _ - setup	настройки EAS

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

Запись настроек мотора.

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

См. также

[get_engine_settings](#)

Аргументы

	id	идентификатор устройства
in	engine _ - settings	структура с настройками мотора

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

Запись настроек мотора с использованием пользовательских единиц.

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

См. также

[get_engine_settings](#)

Аргументы

	id	идентификатор устройства
in	engine_ settings_calb	структура с настройками мотора
	calibration	настройки пользовательских единиц

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

Запись информации о типе мотора и типе силового драйвера.

Аргументы

	id	идентификатор устройства
in	entype_ settings	структура, содержащая настройки типа мотора и типа силового драйвера

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

Запись расширенных настроек.

См. также

[get_extended_settings](#)

Аргументы

	id	идентификатор устройства
in	extended_ settings	настройки EST

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

Команда записи параметров настройки режимов внешнего ввода/вывода.

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

См. также

[get_extio_settings](#)

Аргументы

	id	идентификатор устройства
in	extio_settings	настройки EXTIO

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

Запись настроек обратной связи.

Аргументы

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

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

Запись информации о редукторе в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	gear_information	структура, содержащая информацию о редукторе

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

Запись настроек редуктора в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	gear_settings	структура, содержащая настройки редуктора

```
7.1.4.123 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

	id	идентификатор устройства
in	hallsensor_information	структура, содержащая информацию о датчиках Холла

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

Запись настроек датчиков Холла в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	hallsensor_settings	структура, содержащая настройки датчиков Холла

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

Команда записи настроек для подхода в home position.

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

См. также

[home_settings_t](#)

Аргументы

	id	идентификатор устройства
out	home_settings	настройки калибровки позиции

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

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

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

См. также

[home_settings_calb_t](#)

Аргументы

	id	идентификатор устройства
in	home_settings_calb	настройки калибровки позиции
	calibration	настройки пользовательских единиц

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

Запись настроек джойстика.

При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения,

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

Аргументы

	id	идентификатор устройства
in	joystick_ settings	структура, содержащая настройки джойстика

7.1.4.128 void XIMC_API set_logging_callback (logging_callback_t logging_callback, void * user_data)

Устанавливает функцию обратного вызова для логирования.

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

Аргументы

logging_ callback	указатель на функцию обратного вызова
----------------------	---------------------------------------

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

Запись информации о двигателе в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	motor_ information	структура, содержащая информацию о двигателе

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

Запись настроек двигателя в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	motor_settings	структура, содержащая настройки двигателя

```
7.1.4.131 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

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

```
7.1.4.132 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

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

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

Запись пользовательских данных во FRAM.

Аргументы

	id	идентификатор устройства
in	nonvolatile_memory	структура, содержащая установленные пользовательские данные

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

Запись ПИД коэффициентов.

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

См. также

[get_pid_settings](#)

Аргументы

	id	идентификатор устройства
in	pid_settings	настройки ПИД

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

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

То есть меняется основной показатель положения.

Аргументы

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

7.1.4.136 `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.

То есть меняется основной показатель положения.

Аргументы

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

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

Команда записи параметров питания мотора.

Используется только с шаговым двигателем.

Аргументы

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

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

Команда записи установок защит.

Аргументы

	id	идентификатор устройства
secure_settings		структура с настройками критических значений

См. также

`status_t::flags`


```
7.1.4.139 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

	id	идентификатор устройства
in	serial_number	структура, содержащая серийный номер, версию железа и ключ.

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

Запись информации о позиционере в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	stage_information	структура, содержащая информацию о позиционере

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

Запись пользовательского имени подвижки в EEPROM.

Аргументы

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

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

Запись настроек позиционера в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	stage_settings	структура, содержащая настройки позиционера

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

Запись настроек для входного импульса синхронизации.

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

См. также

[get_sync_in_settings](#)

Аргументы

	id	идентификатор устройства
in	sync_in_settings	настройки синхронизации

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

Запись настроек для входного импульса синхронизации с использованием пользовательских единиц. Эта функция записывает структуру с настройками входного импульса синхронизации, определяющими поведение входа синхронизации, в память контроллера.

См. также

[get_sync_in_settings_calb](#)

Аргументы

	id	идентификатор устройства
in	sync_in_settings_calb	настройки синхронизации
	calibration	настройки пользовательских единиц

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

Запись настроек для выходного импульса синхронизации.

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

См. также

[get_sync_in_settings](#)

Аргументы

	id	идентификатор устройства
in	sync_out_settings	настройки синхронизации

```
7.1.4.146 result_t XIMC_API set_sync_out_settings_calb ( device_t id, const
sync_out_settings_calb_t * sync_out_settings_calb, const calibration_t * calibration )
```

Запись настроек для выходного импульса синхронизации с использованием пользовательских единиц.

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

См. также

[get_sync_in_settings_calb](#)

Аргументы

	id	идентификатор устройства
in	sync_out_settings_calb	настройки синхронизации
	calibration	настройки пользовательских единиц

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

Команда записи настроек UART.

Эта функция записывает структуру настроек UART в память контроллера.

См. также

[uart_settings_t](#)

Аргументы

	Speed	Скорость UART
in	uart_settings	настройки UART

```
7.1.4.148 result_t XIMC_API write_key ( const char * uri, uint8_t * key )
```

Запись ключа защиты. Функция используется только производителем.

Аргументы

	uri	идентификатор устройства
in	key	ключ защиты. Диапазон: 0..4294967295

```
7.1.4.149 result_t XIMC_API ximc_fix_usbser_sys ( const char * device_uri )
```

Исправление ошибки драйвера USB в Windows.

Подсистема USB-COM на Windows не всегда работает корректно. При работе возможны следующие неисправности: все попытки открыть устройство заканчиваются неудачно, или устройство можно открыть и писать в него данные, но в ответ данные не приходят. Эти проблемы лечатся переподключением устройства или удалением и повторным поиском устройства в диспетчере устройств. Функция [ximc_fix_usbser_sys\(\)](#) автоматизирует процесс удаления-обнаружения. Имеет смысл вызывать эту функцию, если библиотека не может открыть устройство, при том что оно физически не было удалено из системы, или если устройство не отвечает.

```
7.1.4.150 void XIMC_API ximc_version ( char * version )
```

Возвращает версию библиотеки

Аргументы

<code>version</code>	буфер для строки с версией, 32 байт достаточно
----------------------	--

Предметный указатель

- A1Voltage
 - analog_data_t, [17](#)
- A1Voltage_ADC
 - analog_data_t, [17](#)
- A2Voltage
 - analog_data_t, [17](#)
- A2Voltage_ADC
 - analog_data_t, [17](#)
- ACurrent
 - analog_data_t, [17](#)
- ACurrent_ADC
 - analog_data_t, [17](#)
- Accel
 - move_settings_calb_t, [57](#)
 - move_settings_t, [58](#)
- accessories_settings_t, [14](#)
 - LimitSwitchesSettings, [15](#)
 - MBRatedCurrent, [15](#)
 - MBRatedVoltage, [15](#)
 - MBSettings, [15](#)
 - MBTorque, [15](#)
 - MagneticBrakeInfo, [15](#)
 - TSGrad, [15](#)
 - TSMMax, [15](#)
 - TSMIn, [15](#)
 - TSSettings, [15](#)
 - TemperatureSensorInfo, [15](#)
- Accuracy
 - sync_out_settings_calb_t, [75](#)
 - sync_out_settings_t, [76](#)
- analog_data_t, [16](#)
 - A1Voltage, [17](#)
 - A1Voltage_ADC, [17](#)
 - A2Voltage, [17](#)
 - A2Voltage_ADC, [17](#)
 - ACurrent, [17](#)
 - ACurrent_ADC, [17](#)
 - B1Voltage, [17](#)
 - B1Voltage_ADC, [18](#)
 - B2Voltage, [18](#)
 - B2Voltage_ADC, [18](#)
 - BCurrent, [18](#)
 - BCurrent_ADC, [18](#)
 - FullCurrent, [18](#)
 - FullCurrent_ADC, [18](#)
 - H5, [18](#)
 - Joy, [18](#)
 - Joy_ADC, [18](#)
 - L5, [18](#)
 - L5_ADC, [18](#)
 - Pot, [19](#)
 - SupVoltage, [19](#)
 - SupVoltage_ADC, [19](#)
 - Temp, [19](#)
 - Temp_ADC, [19](#)
- Antiplay
 - engine_settings_calb_t, [36](#)
 - engine_settings_t, [37](#)
- AntiplaySpeed
 - move_settings_calb_t, [57](#)
 - move_settings_t, [58](#)
- B1Voltage
 - analog_data_t, [17](#)
- B1Voltage_ADC
 - analog_data_t, [18](#)
- B2Voltage
 - analog_data_t, [18](#)
- B2Voltage_ADC
 - analog_data_t, [18](#)
- BACK_EMF_KM_AUTO
 - ximc.h, [102](#)
- BCurrent
 - analog_data_t, [18](#)
- BCurrent_ADC
 - analog_data_t, [18](#)
- BORDER_IS_ENCODER
 - ximc.h, [102](#)
- BORDER_STOP_LEFT
 - ximc.h, [102](#)
- BORDER_STOP_RIGHT
 - ximc.h, [102](#)
- BRAKE_ENABLED
 - ximc.h, [103](#)
- BRAKE_ENG_PWROFF
 - ximc.h, [103](#)
- BackEMFFlags
 - emf_settings_t, [32](#)
- BorderFlags
 - edges_settings_calb_t, [30](#)
 - edges_settings_t, [31](#)
- brake_settings_t, [19](#)
 - BrakeFlags, [20](#)
 - t1, [20](#)
 - t2, [20](#)
 - t3, [20](#)
 - t4, [20](#)
- BrakeFlags

- brake_settings_t, 20
- CONTROL_MODE_BITS
 - ximc.h, 103
- CONTROL_MODE_JOY
 - ximc.h, 103
- CONTROL_MODE_LR
 - ximc.h, 103
- CONTROL_MODE_OFF
 - ximc.h, 103
- CSS1_A
 - calibration_settings_t, 21
- CSS1_B
 - calibration_settings_t, 21
- CSS2_A
 - calibration_settings_t, 21
- CSS2_B
 - calibration_settings_t, 21
- CTP_ALARM_ON_ERROR
 - ximc.h, 103
- CTP_BASE
 - ximc.h, 103
- CTP_ENABLED
 - ximc.h, 103
- CTP_ERROR_CORRECTION
 - ximc.h, 104
- CTPFlags
 - ctp_settings_t, 27
- CTPMinError
 - ctp_settings_t, 27
- calibration_settings_t, 20
 - CSS1_A, 21
 - CSS1_B, 21
 - CSS2_A, 21
 - CSS2_B, 21
 - FullCurrent_A, 21
 - FullCurrent_B, 21
- calibration_t, 21
- chart_data_t, 22
 - DutyCycle, 22
 - Joy, 22
 - Pot, 22
 - WindingCurrentA, 23
 - WindingCurrentB, 23
 - WindingCurrentC, 23
 - WindingVoltageA, 23
 - WindingVoltageB, 23
 - WindingVoltageC, 23
- close_device
 - ximc.h, 117
- ClutterTime
 - sync_in_settings_calb_t, 73
 - sync_in_settings_t, 74
- CmdBufFreeSpace
 - status_calb_t, 69
 - status_t, 71
- command_clear_fram
 - ximc.h, 118
- command_eeread_settings
 - ximc.h, 118
- command_eesave_settings
 - ximc.h, 118
- command_home
 - ximc.h, 118
- command_homezero
 - ximc.h, 119
- command_left
 - ximc.h, 119
- command_loft
 - ximc.h, 119
- command_move
 - ximc.h, 119
- command_move_calb
 - ximc.h, 120
- command_movr
 - ximc.h, 120
- command_movr_calb
 - ximc.h, 120
- command_power_off
 - ximc.h, 121
- command_read_robust_settings
 - ximc.h, 121
- command_read_settings
 - ximc.h, 121
- command_reset
 - ximc.h, 121
- command_right
 - ximc.h, 121
- command_save_robust_settings
 - ximc.h, 122
- command_save_settings
 - ximc.h, 122
- command_sstp
 - ximc.h, 122
- command_start_measurements
 - ximc.h, 122
- command_stop
 - ximc.h, 122
- command_update_firmware
 - ximc.h, 123
- command_wait_for_stop
 - ximc.h, 123
- command_zero
 - ximc.h, 123
- control_settings_calb_t, 23
 - Flags, 24
 - MaxClickTime, 24
 - MaxSpeed, 24
 - Timeout, 24
- control_settings_t, 24
 - Flags, 25
 - MaxClickTime, 25
 - MaxSpeed, 25
 - Timeout, 25
 - uDeltaPosition, 25
 - uMaxSpeed, 26
- controller_name_t, 26

- ControllerName, [26](#)
- CtrlFlags, [26](#)
- ControllerName
 - controller_name_t, [26](#)
- CountsPerTurn
 - feedback_settings_t, [41](#)
- CriticalIpwr
 - secure_settings_t, [62](#)
- CriticalIusb
 - secure_settings_t, [62](#)
- CriticalUpwr
 - secure_settings_t, [62](#)
- CriticalUusb
 - secure_settings_t, [62](#)
- ctp_settings_t, [26](#)
- CTPFlags, [27](#)
- CTPMinError, [27](#)
- CtrlFlags
 - controller_name_t, [26](#)
- CurPosition
 - status_calb_t, [69](#)
 - status_t, [71](#)
- CurSpeed
 - status_calb_t, [69](#)
 - status_t, [71](#)
- CurT
 - status_calb_t, [69](#)
 - status_t, [71](#)
- CurrReductDelay
 - power_settings_t, [60](#)
- CurrentSetTime
 - power_settings_t, [60](#)
- DRIVER_TYPE_EXTERNAL
 - ximc.h, [104](#)
- DeadZone
 - joystick_settings_t, [51](#)
- debug_read_t, [27](#)
- DebugData, [28](#)
- debug_write_t, [28](#)
- DebugData, [28](#)
- DebugData
 - debug_read_t, [28](#)
 - debug_write_t, [28](#)
- Decel
 - move_settings_calb_t, [57](#)
 - move_settings_t, [58](#)
- DetentTorque
 - motor_settings_t, [54](#)
- device_information_t, [28](#)
- Major, [29](#)
- Minor, [29](#)
- Release, [29](#)
- device_network_information_t, [29](#)
- DriverType
 - entype_settings_t, [39](#)
- DutyCycle
 - chart_data_t, [22](#)
- EEPROM_PRECEDENCE
 - ximc.h, [104](#)
- ENC_STATE_ABSENT
 - ximc.h, [104](#)
- ENC_STATE_MALFUNC
 - ximc.h, [104](#)
- ENC_STATE_OK
 - ximc.h, [104](#)
- ENC_STATE_REVERS
 - ximc.h, [104](#)
- ENC_STATE_UNKNOWN
 - ximc.h, [104](#)
- ENDER_SW1_ACTIVE_LOW
 - ximc.h, [104](#)
- ENDER_SW2_ACTIVE_LOW
 - ximc.h, [104](#)
- ENDER_SWAP
 - ximc.h, [105](#)
- ENGINE_ACCEL_ON
 - ximc.h, [105](#)
- ENGINE_ANTIPLAY
 - ximc.h, [105](#)
- ENGINE_LIMIT_CURR
 - ximc.h, [105](#)
- ENGINE_LIMIT_RPM
 - ximc.h, [105](#)
- ENGINE_LIMIT_VOLT
 - ximc.h, [105](#)
- ENGINE_MAX_SPEED
 - ximc.h, [105](#)
- ENGINE_REVERSE
 - ximc.h, [105](#)
- ENGINE_TYPE_2DC
 - ximc.h, [106](#)
- ENGINE_TYPE_DC
 - ximc.h, [106](#)
- ENGINE_TYPE_NONE
 - ximc.h, [106](#)
- ENGINE_TYPE_STEP
 - ximc.h, [106](#)
- ENGINE_TYPE_TEST
 - ximc.h, [106](#)
- ENUMERATE_PROBE
 - ximc.h, [106](#)
- EXTIO_SETUP_INVERT
 - ximc.h, [106](#)
- EXTIO_SETUP_OUTPUT
 - ximc.h, [107](#)
- EXTIOModeFlags
 - extio_settings_t, [40](#)
- EXTIOSetupFlags
 - extio_settings_t, [40](#)
- edges_settings_calb_t, [30](#)
- 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, 42
- emf_settings_t, 32
 - BackEMFFlags, 32
 - Km, 32
 - L, 32
 - R, 32
- EncPosition
 - get_position_calb_t, 44
 - get_position_t, 44
 - set_position_calb_t, 64
 - set_position_t, 64
 - status_calb_t, 69
 - status_t, 71
- EncSts
 - status_calb_t, 69
 - status_t, 71
- 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, 36
 - StepsPerRev, 36
- engine_settings_t, 37
 - Antiplay, 37
 - EngineFlags, 37
 - MicrostepMode, 37
 - NomCurrent, 38
 - NomSpeed, 38
 - NomVoltage, 38
 - StepsPerRev, 38
 - uNomSpeed, 38
- EngineFlags
 - engine_settings_calb_t, 36
 - engine_settings_t, 37
- EngineType
 - entype_settings_t, 39
- entype_settings_t, 38
 - DriverType, 39
 - EngineType, 39
- enumerate_devices
 - ximc.h, 123
- Error
 - measurements_t, 52
- ExpFactor
 - joystick_settings_t, 51
- extended_settings_t, 39
- extio_settings_t, 39
 - EXTIOModeFlags, 40
 - EXTIOSetupFlags, 40
- FEEDBACK_EMF
 - ximc.h, 107
- FEEDBACK_ENC_REVERSE
 - ximc.h, 107
- FEEDBACK_ENCODER
 - ximc.h, 108
- FEEDBACK_NONE
 - ximc.h, 108
- 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, 24
 - control_settings_t, 25
 - secure_settings_t, 62
 - status_calb_t, 69
 - status_t, 72
- free_enumerate_devices
 - ximc.h, 124
- FullCurrent
 - analog_data_t, 18
- FullCurrent_A
 - calibration_settings_t, 21
- FullCurrent_ADC
 - analog_data_t, 18
- FullCurrent_B
 - calibration_settings_t, 21
- GPIOFlags
 - status_calb_t, 69
 - status_t, 72

- gear_information_t, [41](#)
 - Manufacturer, [41](#)
 - PartNumber, [41](#)
- gear_settings_t, [42](#)
 - Efficiency, [42](#)
 - InputInertia, [42](#)
 - MaxOutputBacklash, [42](#)
 - RatedInputSpeed, [43](#)
 - RatedInputTorque, [43](#)
 - ReductionIn, [43](#)
 - ReductionOut, [43](#)
- get_accessories_settings
 - ximc.h, [124](#)
- get_analog_data
 - ximc.h, [124](#)
- get_bootloader_version
 - ximc.h, [124](#)
- get_brake_settings
 - ximc.h, [125](#)
- get_calibration_settings
 - ximc.h, [125](#)
- get_chart_data
 - ximc.h, [125](#)
- get_control_settings
 - ximc.h, [125](#)
- get_control_settings_calb
 - ximc.h, [126](#)
- get_controller_name
 - ximc.h, [126](#)
- get_ctp_settings
 - ximc.h, [126](#)
- get_debug_read
 - ximc.h, [127](#)
- get_device_count
 - ximc.h, [127](#)
- get_device_information
 - ximc.h, [127](#)
- get_device_name
 - ximc.h, [127](#)
- get_edges_settings
 - ximc.h, [128](#)
- get_edges_settings_calb
 - ximc.h, [128](#)
- get_emf_settings
 - ximc.h, [128](#)
- get_encoder_information
 - ximc.h, [129](#)
- get_encoder_settings
 - ximc.h, [129](#)
- get_engine_advansed_setup
 - ximc.h, [129](#)
- get_engine_settings
 - ximc.h, [129](#)
- get_engine_settings_calb
 - ximc.h, [130](#)
- get_entype_settings
 - ximc.h, [130](#)
- get_enumerate_device_controller_name
 - ximc.h, [130](#)
- get_enumerate_device_information
 - ximc.h, [130](#)
- get_enumerate_device_network_information
 - ximc.h, [131](#)
- get_enumerate_device_serial
 - ximc.h, [131](#)
- get_enumerate_device_stage_name
 - ximc.h, [131](#)
- get_extended_settings
 - ximc.h, [132](#)
- get_extio_settings
 - ximc.h, [132](#)
- get_feedback_settings
 - ximc.h, [132](#)
- get_firmware_version
 - ximc.h, [132](#)
- get_gear_information
 - ximc.h, [133](#)
- get_gear_settings
 - ximc.h, [133](#)
- get_globally_unique_identifier
 - ximc.h, [133](#)
- get_hallsensor_information
 - ximc.h, [133](#)
- get_hallsensor_settings
 - ximc.h, [134](#)
- get_home_settings
 - ximc.h, [134](#)
- get_home_settings_calb
 - ximc.h, [134](#)
- get_init_random
 - ximc.h, [134](#)
- get_joystick_settings
 - ximc.h, [135](#)
- get_measurements
 - ximc.h, [135](#)
- get_motor_information
 - ximc.h, [135](#)
- get_motor_settings
 - ximc.h, [136](#)
- get_move_settings
 - ximc.h, [136](#)
- get_move_settings_calb
 - ximc.h, [136](#)
- get_nonvolatile_memory
 - ximc.h, [136](#)
- get_pid_settings
 - ximc.h, [136](#)
- get_position
 - ximc.h, [137](#)
- get_position_calb
 - ximc.h, [137](#)
- get_position_calb_t, [43](#)
 - EncPosition, [44](#)
 - Position, [44](#)
- get_position_t, [44](#)
 - EncPosition, [44](#)

- uPosition, 44
- get_power_settings
 - ximc.h, 137
- get_secure_settings
 - ximc.h, 138
- get_serial_number
 - ximc.h, 138
- get_stage_information
 - ximc.h, 138
- get_stage_name
 - ximc.h, 138
- get_stage_settings
 - ximc.h, 138
- get_status
 - ximc.h, 138
- get_status_calb
 - ximc.h, 139
- get_sync_in_settings
 - ximc.h, 139
- get_sync_in_settings_calb
 - ximc.h, 139
- get_sync_out_settings
 - ximc.h, 140
- get_sync_out_settings_calb
 - ximc.h, 140
- get_uart_settings
 - ximc.h, 140
- globally_unique_identifier_t, 44
 - UniqueID0, 45
 - UniqueID1, 45
 - UniqueID2, 45
 - UniqueID3, 45
- goto_firmware
 - ximc.h, 141
- H5
 - analog_data_t, 18
- HOME_DIR_FIRST
 - ximc.h, 108
- HOME_DIR_SECOND
 - ximc.h, 108
- HOME_HALF_MV
 - ximc.h, 108
- HOME_MV_SEC_EN
 - ximc.h, 108
- HOME_STOP_FIRST_LIM
 - ximc.h, 109
- HOME_STOP_FIRST_REV
 - ximc.h, 109
- HOME_STOP_FIRST_SYN
 - ximc.h, 109
- HOME_USE_FAST
 - ximc.h, 109
- hallsensor_information_t, 45
 - Manufacturer, 46
 - PartNumber, 46
- hallsensor_settings_t, 46
 - MaxCurrentConsumption, 47
 - MaxOperatingFrequency, 47
 - SupplyVoltageMax, 47
 - SupplyVoltageMin, 47
- has_firmware
 - ximc.h, 141
- HoldCurrent
 - power_settings_t, 60
- home_settings_calb_t, 47
 - FastHome, 48
 - HomeDelta, 48
 - HomeFlags, 48
 - SlowHome, 48
- home_settings_t, 48
 - FastHome, 49
 - HomeDelta, 49
 - HomeFlags, 49
 - SlowHome, 49
 - uFastHome, 49
 - uHomeDelta, 49
 - uSlowHome, 49
- HomeDelta
 - home_settings_calb_t, 48
 - home_settings_t, 49
- HomeFlags
 - home_settings_calb_t, 48
 - home_settings_t, 49
- HorizontalLoadCapacity
 - stage_settings_t, 67
- IPS
 - feedback_settings_t, 41
- init_random_t, 49
 - key, 50
- InputInertia
 - gear_settings_t, 42
- Ipwr
 - status_calb_t, 69
 - status_t, 72
- Iusb
 - status_calb_t, 69
 - status_t, 72
- JOY_REVERSE
 - ximc.h, 109
- Joy
 - analog_data_t, 18
 - chart_data_t, 22
- Joy_ADC
 - analog_data_t, 18
- JoyCenter
 - joystick_settings_t, 51
- JoyFlags
 - joystick_settings_t, 51
- JoyHighEnd
 - joystick_settings_t, 51
- JoyLowEnd
 - joystick_settings_t, 51
- joystick_settings_t, 50
 - DeadZone, 51
 - ExpFactor, 51

- JoyCenter, [51](#)
- JoyFlags, [51](#)
- JoyHighEnd, [51](#)
- JoyLowEnd, [51](#)
- Key
 - serial_number_t, [63](#)
- key
 - init_random_t, [50](#)
- Km
 - emf_settings_t, [32](#)
- L
 - emf_settings_t, [32](#)
- L5
 - analog_data_t, [18](#)
- L5_ADC
 - analog_data_t, [18](#)
- LOW_UPWR_PROTECTION
 - ximc.h, [109](#)
- LS_SHORTED
 - ximc.h, [109](#)
- LeadScrewPitch
 - stage_settings_t, [67](#)
- LeftBorder
 - edges_settings_calb_t, [30](#)
 - edges_settings_t, [31](#)
- Length
 - measurements_t, [52](#)
- LimitSwitchesSettings
 - accessories_settings_t, [15](#)
- load_correction_table
 - ximc.h, [141](#)
- logging_callback_stderr_narrow
 - ximc.h, [142](#)
- logging_callback_stderr_wide
 - ximc.h, [142](#)
- logging_callback_t
 - ximc.h, [117](#)
- LowUpwrOff
 - secure_settings_t, [62](#)
- MBRatedCurrent
 - accessories_settings_t, [15](#)
- MBRatedVoltage
 - accessories_settings_t, [15](#)
- MBSettings
 - accessories_settings_t, [15](#)
- MBTorque
 - accessories_settings_t, [15](#)
- MICROSTEP_MODE_FULL
 - ximc.h, [110](#)
- MOVE_STATE_ANTIPLAY
 - ximc.h, [110](#)
- MOVE_STATE_MOVING
 - ximc.h, [110](#)
- MVCMD_ERROR
 - ximc.h, [111](#)
- MVCMD_HOME
 - ximc.h, [111](#)
- MVCMD_LEFT
 - ximc.h, [111](#)
- MVCMD_LOFT
 - ximc.h, [111](#)
- MVCMD_MOVE
 - ximc.h, [111](#)
- MVCMD_MOVR
 - ximc.h, [111](#)
- MVCMD_NAME_BITS
 - ximc.h, [111](#)
- MVCMD_RIGHT
 - ximc.h, [111](#)
- MVCMD_RUNNING
 - ximc.h, [111](#)
- MVCMD_SSTP
 - ximc.h, [111](#)
- MVCMD_STOP
 - ximc.h, [111](#)
- MVCMD_UKNWN
 - ximc.h, [111](#)
- MagneticBrakeInfo
 - accessories_settings_t, [15](#)
- Major
 - device_information_t, [29](#)
 - serial_number_t, [63](#)
- Manufacturer
 - encoder_information_t, [33](#)
 - gear_information_t, [41](#)
 - hallsensor_information_t, [46](#)
 - motor_information_t, [52](#)
 - stage_information_t, [65](#)
- MaxClickTime
 - control_settings_calb_t, [24](#)
 - control_settings_t, [25](#)
- MaxCurrent
 - motor_settings_t, [54](#)
- MaxCurrentConsumption
 - encoder_settings_t, [34](#)
 - hallsensor_settings_t, [47](#)
 - stage_settings_t, [67](#)
- MaxCurrentTime
 - motor_settings_t, [54](#)
- MaxOperatingFrequency
 - encoder_settings_t, [34](#)
 - hallsensor_settings_t, [47](#)
- MaxOutputBacklash
 - gear_settings_t, [42](#)
- MaxSpeed
 - control_settings_calb_t, [24](#)
 - control_settings_t, [25](#)
 - motor_settings_t, [54](#)
 - stage_settings_t, [67](#)
- measurements_t, [51](#)
 - Error, [52](#)
 - Length, [52](#)
 - Speed, [52](#)
- MechanicalTimeConstant

- motor_settings_t, 54
- MicrostepMode
 - engine_settings_calb_t, 36
 - engine_settings_t, 37
- MinimumUusb
 - secure_settings_t, 62
- Minor
 - device_information_t, 29
 - serial_number_t, 63
- motor_information_t, 52
 - Manufacturer, 52
 - PartNumber, 52
- motor_settings_t, 53
 - DetentTorque, 54
 - MaxCurrent, 54
 - MaxCurrentTime, 54
 - MaxSpeed, 54
 - MechanicalTimeConstant, 54
 - MotorType, 54
 - NoLoadCurrent, 55
 - NoLoadSpeed, 55
 - NominalCurrent, 55
 - NominalPower, 55
 - NominalSpeed, 55
 - NominalTorque, 55
 - NominalVoltage, 55
 - Phases, 55
 - Poles, 55
 - RotorInertia, 55
 - SpeedConstant, 56
 - SpeedTorqueGradient, 56
 - StallTorque, 56
 - TorqueConstant, 56
 - WindingInductance, 56
 - WindingResistance, 56
- MotorType
 - motor_settings_t, 54
- move_settings_calb_t, 56
 - Accel, 57
 - AntiplaySpeed, 57
 - Decel, 57
 - MoveFlags, 57
 - Speed, 57
- move_settings_t, 57
 - Accel, 58
 - AntiplaySpeed, 58
 - Decel, 58
 - MoveFlags, 58
 - Speed, 58
 - uAntiplaySpeed, 58
 - uSpeed, 58
- MoveFlags
 - move_settings_calb_t, 57
 - move_settings_t, 58
- MoveSts
 - status_calb_t, 69
 - status_t, 72
- msec_sleep
 - ximc.h, 142
- MvCmdSts
 - status_calb_t, 69
 - status_t, 72
- NoLoadCurrent
 - motor_settings_t, 55
- NoLoadSpeed
 - motor_settings_t, 55
- 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, 36
 - engine_settings_t, 38
- NominalCurrent
 - motor_settings_t, 55
- NominalPower
 - motor_settings_t, 55
- NominalSpeed
 - motor_settings_t, 55
- NominalTorque
 - motor_settings_t, 55
- NominalVoltage
 - motor_settings_t, 55
- nonvolatile_memory_t, 59
 - UserData, 59
- open_device
 - ximc.h, 142
- POWER_OFF_ENABLED
 - ximc.h, 112
- POWER_REDUCT_ENABLED
 - ximc.h, 112
- POWER_SMOOTH_CURRENT
 - ximc.h, 112
- PWR_STATE_MAX
 - ximc.h, 112
- PWR_STATE_NORM
 - ximc.h, 112
- PWR_STATE_OFF
 - ximc.h, 112
- PWR_STATE_REDUCT
 - ximc.h, 112
- PWR_STATE_UNKNOWN
 - ximc.h, 112
- PWRSts
 - status_calb_t, 70
 - status_t, 72
- PartNumber
 - encoder_information_t, 33
 - gear_information_t, 41
 - hallsensor_information_t, 46
 - motor_information_t, 52
 - stage_information_t, 65

- Phases
 - motor_settings_t, 55
- pid_settings_t, 59
- Poles
 - motor_settings_t, 55
- PosFlags
 - set_position_calb_t, 64
 - set_position_t, 64
- Position
 - get_position_calb_t, 44
 - set_position_calb_t, 64
 - sync_in_settings_calb_t, 73
- PositionerName
 - stage_name_t, 66
- Pot
 - analog_data_t, 19
 - chart_data_t, 22
- power_settings_t, 60
 - CurrReductDelay, 60
 - CurrentSetTime, 60
 - HoldCurrent, 60
 - PowerFlags, 61
 - PowerOffDelay, 61
- PowerFlags
 - power_settings_t, 61
- PowerOffDelay
 - power_settings_t, 61
- probe_device
 - ximc.h, 142
- R
 - emf_settings_t, 32
- REV_SENS_INV
 - ximc.h, 112
- RPM_DIV_1000
 - ximc.h, 112
- 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, 63
- RightBorder
 - edges_settings_calb_t, 30
 - edges_settings_t, 31
- RotorInertia
 - motor_settings_t, 55
- SN
 - serial_number_t, 63
- STATE_ALARM
 - ximc.h, 113
- STATE_BRAKE
 - ximc.h, 113
- STATE_BUTTON_LEFT
 - ximc.h, 113
- STATE_BUTTON_RIGHT
 - ximc.h, 113
- STATE_CONTR
 - ximc.h, 113
- STATE_CTP_ERROR
 - ximc.h, 113
- STATE_DIG_SIGNAL
 - ximc.h, 113
- STATE_ENC_A
 - ximc.h, 114
- STATE_ENC_B
 - ximc.h, 114
- STATE_ERRC
 - ximc.h, 114
- STATE_ERRD
 - ximc.h, 114
- STATE_ERRV
 - ximc.h, 114
- STATE_EXTIO_ALARM
 - ximc.h, 114
- STATE_GPIO_LEVEL
 - ximc.h, 114
- STATE_GPIO_PINOUT
 - ximc.h, 114
- STATE_LEFT_EDGE
 - ximc.h, 114
- STATE_POWER_OVERHEAT
 - ximc.h, 115
- STATE_REV_SENSOR
 - ximc.h, 115
- STATE_RIGHT_EDGE
 - ximc.h, 115
- STATE_SECUR
 - ximc.h, 115
- STATE_SYNC_INPUT
 - ximc.h, 115
- STATE_SYNC_OUTPUT
 - ximc.h, 115
- SYNCIN_ENABLED
 - ximc.h, 115
- SYNCIN_INVERT
 - ximc.h, 115
- SYNCOUT_ENABLED
 - ximc.h, 116
- SYNCOUT_IN_STEPS
 - ximc.h, 116
- SYNCOUT_INVERT
 - ximc.h, 116
- SYNCOUT_ONPERIOD
 - ximc.h, 116
- SYNCOUT_ONSTART
 - ximc.h, 116
- SYNCOUT_ONSTOP
 - ximc.h, 116
- SYNCOUT_STATE
 - ximc.h, 116

- secure_settings_t, 61
 - CriticalIpwr, 62
 - CriticalIusb, 62
 - CriticalUpwr, 62
 - CriticalUusb, 62
 - Flags, 62
 - LowUpwrOff, 62
 - MinimumUusb, 62
- serial_number_t, 62
 - Key, 63
 - Major, 63
 - Minor, 63
 - Release, 63
 - SN, 63
- service_command_updf
 - ximc.h, 143
- set_accessories_settings
 - ximc.h, 143
- set_bindy_key
 - ximc.h, 143
- set_brake_settings
 - ximc.h, 143
- set_calibration_settings
 - ximc.h, 143
- set_control_settings
 - ximc.h, 144
- set_control_settings_calb
 - ximc.h, 144
- set_controller_name
 - ximc.h, 144
- set_ctp_settings
 - ximc.h, 145
- set_debug_write
 - ximc.h, 145
- set_edges_settings
 - ximc.h, 145
- set_edges_settings_calb
 - ximc.h, 145
- set_emf_settings
 - ximc.h, 146
- set_encoder_information
 - ximc.h, 146
- set_encoder_settings
 - ximc.h, 146
- set_engine_advansed_setup
 - ximc.h, 147
- set_engine_settings
 - ximc.h, 147
- set_engine_settings_calb
 - ximc.h, 147
- set_entype_settings
 - ximc.h, 148
- set_extended_settings
 - ximc.h, 148
- set_extio_settings
 - ximc.h, 148
- set_feedback_settings
 - ximc.h, 148
- set_gear_information
 - ximc.h, 149
- set_gear_settings
 - ximc.h, 149
- set_hallsensor_information
 - ximc.h, 149
- set_hallsensor_settings
 - ximc.h, 149
- set_home_settings
 - ximc.h, 150
- set_home_settings_calb
 - ximc.h, 150
- set_joystick_settings
 - ximc.h, 150
- set_logging_callback
 - ximc.h, 151
- set_motor_information
 - ximc.h, 151
- set_motor_settings
 - ximc.h, 151
- set_move_settings
 - ximc.h, 151
- set_move_settings_calb
 - ximc.h, 152
- set_nonvolatile_memory
 - ximc.h, 152
- set_pid_settings
 - ximc.h, 152
- set_position
 - ximc.h, 152
- set_position_calb
 - ximc.h, 153
- set_position_calb_t, 63
 - EncPosition, 64
 - PosFlags, 64
 - Position, 64
- set_position_t, 64
 - EncPosition, 64
 - PosFlags, 64
 - uPosition, 64
- set_power_settings
 - ximc.h, 153
- set_secure_settings
 - ximc.h, 153
- set_serial_number
 - ximc.h, 153
- set_stage_information
 - ximc.h, 154
- set_stage_name
 - ximc.h, 154
- set_stage_settings
 - ximc.h, 154
- set_sync_in_settings
 - ximc.h, 154
- set_sync_in_settings_calb
 - ximc.h, 155
- set_sync_out_settings
 - ximc.h, 155

- set_sync_out_settings_calb
 - ximc.h, [155](#)
- set_uart_settings
 - ximc.h, [156](#)
- SlowHome
 - home_settings_calb_t, [48](#)
 - home_settings_t, [49](#)
- Speed
 - measurements_t, [52](#)
 - move_settings_calb_t, [57](#)
 - move_settings_t, [58](#)
 - sync_in_settings_calb_t, [73](#)
 - sync_in_settings_t, [74](#)
- SpeedConstant
 - motor_settings_t, [56](#)
- SpeedTorqueGradient
 - motor_settings_t, [56](#)
- stage_information_t, [65](#)
 - Manufacturer, [65](#)
 - PartNumber, [65](#)
- stage_name_t, [65](#)
 - PositionerName, [66](#)
- stage_settings_t, [66](#)
 - HorizontalLoadCapacity, [67](#)
 - LeadScrewPitch, [67](#)
 - MaxCurrentConsumption, [67](#)
 - MaxSpeed, [67](#)
 - SupplyVoltageMax, [67](#)
 - SupplyVoltageMin, [67](#)
 - TravelRange, [67](#)
 - Units, [67](#)
 - VerticalLoadCapacity, [67](#)
- StallTorque
 - motor_settings_t, [56](#)
- status_calb_t, [68](#)
 - CmdBufFreeSpace, [69](#)
 - CurPosition, [69](#)
 - CurSpeed, [69](#)
 - CurT, [69](#)
 - EncPosition, [69](#)
 - EncSts, [69](#)
 - Flags, [69](#)
 - GPIOFlags, [69](#)
 - Ipwr, [69](#)
 - Iusb, [69](#)
 - MoveSts, [69](#)
 - MvCmdSts, [69](#)
 - PWRSts, [70](#)
 - Upwr, [70](#)
 - Uusb, [70](#)
 - WindSts, [70](#)
- status_t, [70](#)
 - CmdBufFreeSpace, [71](#)
 - CurPosition, [71](#)
 - CurSpeed, [71](#)
 - CurT, [71](#)
 - EncPosition, [71](#)
 - EncSts, [71](#)
 - Flags, [72](#)
 - GPIOFlags, [72](#)
 - Ipwr, [72](#)
 - Iusb, [72](#)
 - MoveSts, [72](#)
 - MvCmdSts, [72](#)
 - PWRSts, [72](#)
 - uCurPosition, [72](#)
 - uCurSpeed, [72](#)
 - Upwr, [72](#)
 - Uusb, [72](#)
 - WindSts, [73](#)
- 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, [36](#)
 - engine_settings_t, [38](#)
- SupVoltage
 - analog_data_t, [19](#)
- SupVoltage_ADC
 - analog_data_t, [19](#)
- SupplyVoltageMax
 - encoder_settings_t, [34](#)
 - hallsensor_settings_t, [47](#)
 - stage_settings_t, [67](#)
- SupplyVoltageMin
 - encoder_settings_t, [34](#)
 - hallsensor_settings_t, [47](#)
 - stage_settings_t, [67](#)
- sync_in_settings_calb_t, [73](#)
 - ClutterTime, [73](#)
 - Position, [73](#)
 - Speed, [73](#)
 - SyncInFlags, [73](#)
- sync_in_settings_t, [74](#)
 - ClutterTime, [74](#)
 - Speed, [74](#)
 - SyncInFlags, [74](#)
 - uPosition, [74](#)
 - uSpeed, [74](#)
- sync_out_settings_calb_t, [75](#)
 - Accuracy, [75](#)
 - SyncOutFlags, [75](#)
 - SyncOutPeriod, [75](#)
 - SyncOutPulseSteps, [75](#)
- sync_out_settings_t, [76](#)
 - Accuracy, [76](#)
 - SyncOutFlags, [76](#)
 - SyncOutPeriod, [76](#)
 - SyncOutPulseSteps, [76](#)
 - uAccuracy, [77](#)
- SyncInFlags
 - sync_in_settings_calb_t, [73](#)
 - sync_in_settings_t, [74](#)

- SyncOutFlags
 - sync_out_settings_calb_t, 75
 - sync_out_settings_t, 76
- SyncOutPeriod
 - sync_out_settings_calb_t, 75
 - sync_out_settings_t, 76
- SyncOutPulseSteps
 - sync_out_settings_calb_t, 75
 - sync_out_settings_t, 76
- t1
 - brake_settings_t, 20
- t2
 - brake_settings_t, 20
- t3
 - brake_settings_t, 20
- t4
 - brake_settings_t, 20
- TS_TYPE_BITS
 - ximc.h, 116
- TSGrad
 - accessories_settings_t, 15
- TSMMax
 - accessories_settings_t, 15
- TSMIn
 - accessories_settings_t, 15
- TSSettings
 - accessories_settings_t, 15
- Temp
 - analog_data_t, 19
- Temp_ADC
 - analog_data_t, 19
- TemperatureSensorInfo
 - accessories_settings_t, 15
- Timeout
 - control_settings_calb_t, 24
 - control_settings_t, 25
- TorqueConstant
 - motor_settings_t, 56
- TravelRange
 - stage_settings_t, 67
- UART_PARITY_BITS
 - ximc.h, 116
- UARTSetupFlags
 - uart_settings_t, 77
- uAccuracy
 - sync_out_settings_t, 77
- uAntiplaySpeed
 - move_settings_t, 58
- uCurPosition
 - status_t, 72
- uCurSpeed
 - status_t, 72
- uDeltaPosition
 - control_settings_t, 25
- uFastHome
 - home_settings_t, 49
- uHomeDelta
 - home_settings_t, 49
- uLeftBorder
 - edges_settings_t, 31
- uMaxSpeed
 - control_settings_t, 26
- uNomSpeed
 - engine_settings_t, 38
- uPosition
 - get_position_t, 44
 - set_position_t, 64
 - sync_in_settings_t, 74
- uRightBorder
 - edges_settings_t, 31
- uSlowHome
 - home_settings_t, 49
- uSpeed
 - move_settings_t, 58
 - sync_in_settings_t, 74
- uart_settings_t, 77
 - UARTSetupFlags, 77
- UniqueID0
 - globally_unique_identifier_t, 45
- UniqueID1
 - globally_unique_identifier_t, 45
- UniqueID2
 - globally_unique_identifier_t, 45
- UniqueID3
 - globally_unique_identifier_t, 45
- Units
 - stage_settings_t, 67
- Upwr
 - status_calb_t, 70
 - status_t, 72
- UserData
 - nonvolatile_memory_t, 59
- Uusb
 - status_calb_t, 70
 - status_t, 72
- VerticalLoadCapacity
 - stage_settings_t, 67
- WIND_A_STATE_ABSENT
 - ximc.h, 116
- WIND_A_STATE_OK
 - ximc.h, 117
- WIND_B_STATE_ABSENT
 - ximc.h, 117
- WIND_B_STATE_OK
 - ximc.h, 117
- WindSts
 - status_calb_t, 70
 - status_t, 73
- WindingCurrentA
 - chart_data_t, 23
- WindingCurrentB
 - chart_data_t, 23
- WindingCurrentC
 - chart_data_t, 23

- WindingInductance
 - motor_settings_t, 56
- WindingResistance
 - motor_settings_t, 56
- WindingVoltageA
 - chart_data_t, 23
- WindingVoltageB
 - chart_data_t, 23
- WindingVoltageC
 - chart_data_t, 23
- write_key
 - ximc.h, 156
- XIMC_API
 - ximc.h, 117
- ximc.h, 78
 - BACK_EMF_KM_AUTO, 102
 - BORDER_IS_ENCODER, 102
 - BORDER_STOP_LEFT, 102
 - BORDER_STOP_RIGHT, 102
 - BRAKE_ENABLED, 103
 - BRAKE_ENG_PWROFF, 103
 - CONTROL_MODE_BITS, 103
 - CONTROL_MODE_JOY, 103
 - CONTROL_MODE_LR, 103
 - CONTROL_MODE_OFF, 103
 - CTP_ALARM_ON_ERROR, 103
 - CTP_BASE, 103
 - CTP_ENABLED, 103
 - close_device, 117
 - command_clear_fram, 118
 - command_eeread_settings, 118
 - command_eesave_settings, 118
 - command_home, 118
 - command_homezero, 119
 - command_left, 119
 - command_loft, 119
 - command_move, 119
 - command_move_calb, 120
 - command_movr, 120
 - command_movr_calb, 120
 - command_power_off, 121
 - command_read_robust_settings, 121
 - command_read_settings, 121
 - command_reset, 121
 - command_right, 121
 - command_save_robust_settings, 122
 - command_save_settings, 122
 - command_sstp, 122
 - command_start_measurements, 122
 - command_stop, 122
 - command_update_firmware, 123
 - command_wait_for_stop, 123
 - command_zero, 123
 - EEPROM_PRECEDENCE, 104
 - ENC_STATE_ABSENT, 104
 - ENC_STATE_MALFUNC, 104
 - ENC_STATE_OK, 104
 - ENC_STATE_REVERS, 104
 - ENC_STATE_UNKNOWN, 104
 - ENDER_SWAP, 105
 - ENGINE_ACCEL_ON, 105
 - ENGINE_ANTIPLAY, 105
 - ENGINE_LIMIT_CURR, 105
 - ENGINE_LIMIT_RPM, 105
 - ENGINE_LIMIT_VOLT, 105
 - ENGINE_MAX_SPEED, 105
 - ENGINE_REVERSE, 105
 - ENGINE_TYPE_2DC, 106
 - ENGINE_TYPE_DC, 106
 - ENGINE_TYPE_NONE, 106
 - ENGINE_TYPE_STEP, 106
 - ENGINE_TYPE_TEST, 106
 - ENUMERATE_PROBE, 106
 - EXTIO_SETUP_INVERT, 106
 - EXTIO_SETUP_OUTPUT, 107
 - enumerate_devices, 123
 - FEEDBACK_EMF, 107
 - FEEDBACK_ENCODER, 108
 - FEEDBACK_NONE, 108
 - free_enumerate_devices, 124
 - get_accessories_settings, 124
 - get_analog_data, 124
 - get_bootloader_version, 124
 - get_brake_settings, 125
 - get_calibration_settings, 125
 - get_chart_data, 125
 - get_control_settings, 125
 - get_control_settings_calb, 126
 - get_controller_name, 126
 - get_ctp_settings, 126
 - get_debug_read, 127
 - get_device_count, 127
 - get_device_information, 127
 - get_device_name, 127
 - get_edges_settings, 128
 - get_edges_settings_calb, 128
 - get_emf_settings, 128
 - get_encoder_information, 129
 - get_encoder_settings, 129
 - get_engine_advansed_setup, 129
 - get_engine_settings, 129
 - get_engine_settings_calb, 130
 - get_entype_settings, 130
 - get_enumerate_device_controller_name, 130
 - get_enumerate_device_information, 130
 - get_enumerate_device_network_information, 131
 - get_enumerate_device_serial, 131
 - get_enumerate_device_stage_name, 131
 - get_extended_settings, 132
 - get_extio_settings, 132
 - get_feedback_settings, 132
 - get_firmware_version, 132
 - get_gear_information, 133
 - get_gear_settings, 133
 - get_globally_unique_identifier, 133

get_hallsensor_information, 133
get_hallsensor_settings, 134
get_home_settings, 134
get_home_settings_calb, 134
get_init_random, 134
get_joystick_settings, 135
get_measurements, 135
get_motor_information, 135
get_motor_settings, 136
get_move_settings, 136
get_move_settings_calb, 136
get_nonvolatile_memory, 136
get_pid_settings, 136
get_position, 137
get_position_calb, 137
get_power_settings, 137
get_secure_settings, 138
get_serial_number, 138
get_stage_information, 138
get_stage_name, 138
get_stage_settings, 138
get_status, 138
get_status_calb, 139
get_sync_in_settings, 139
get_sync_in_settings_calb, 139
get_sync_out_settings, 140
get_sync_out_settings_calb, 140
get_uart_settings, 140
goto_firmware, 141
HOME_DIR_FIRST, 108
HOME_DIR_SECOND, 108
HOME_HALF_MV, 108
HOME_MV_SEC_EN, 108
HOME_USE_FAST, 109
has_firmware, 141
JOY_REVERSE, 109
LOW_UPWR_PROTECTION, 109
LS_SHORTED, 109
load_correction_table, 141
logging_callback_stderr_narrow, 142
logging_callback_stderr_wide, 142
logging_callback_t, 117
MICROSTEP_MODE_FULL, 110
MOVE_STATE_ANTIPLAY, 110
MOVE_STATE_MOVING, 110
MVCMD_ERROR, 111
MVCMD_HOME, 111
MVCMD_LEFT, 111
MVCMD_LOFT, 111
MVCMD_MOVE, 111
MVCMD_MOVR, 111
MVCMD_NAME_BITS, 111
MVCMD_RIGHT, 111
MVCMD_RUNNING, 111
MVCMD_SSTP, 111
MVCMD_STOP, 111
MVCMD_UKNWN, 111
msec_sleep, 142
open_device, 142
POWER_OFF_ENABLED, 112
PWR_STATE_MAX, 112
PWR_STATE_NORM, 112
PWR_STATE_OFF, 112
PWR_STATE_REDUCT, 112
PWR_STATE_UNKNOWN, 112
probe_device, 142
REV_SENS_INV, 112
RPM_DIV_1000, 112
STATE_ALARM, 113
STATE_BRAKE, 113
STATE_BUTTON_LEFT, 113
STATE_BUTTON_RIGHT, 113
STATE_CONTR, 113
STATE_CTP_ERROR, 113
STATE_DIG_SIGNAL, 113
STATE_ENC_A, 114
STATE_ENC_B, 114
STATE_ERRC, 114
STATE_ERRD, 114
STATE_ERRV, 114
STATE_EXTIO_ALARM, 114
STATE_GPIO_LEVEL, 114
STATE_GPIO_PINOUT, 114
STATE_LEFT_EDGE, 114
STATE_REV_SENSOR, 115
STATE_RIGHT_EDGE, 115
STATE_SECUR, 115
STATE_SYNC_INPUT, 115
STATE_SYNC_OUTPUT, 115
SYNCIN_ENABLED, 115
SYNCIN_INVERT, 115
SYNCOUT_ENABLED, 116
SYNCOUT_IN_STEPS, 116
SYNCOUT_INVERT, 116
SYNCOUT_ONPERIOD, 116
SYNCOUT_ONSTART, 116
SYNCOUT_ONSTOP, 116
SYNCOUT_STATE, 116
service_command_updf, 143
set_accessories_settings, 143
set_bindy_key, 143
set_brake_settings, 143
set_calibration_settings, 143
set_control_settings, 144
set_control_settings_calb, 144
set_controller_name, 144
set_ctp_settings, 145
set_debug_write, 145
set_edges_settings, 145
set_edges_settings_calb, 145
set_emf_settings, 146
set_encoder_information, 146
set_encoder_settings, 146
set_engine_advansed_setup, 147
set_engine_settings, 147
set_engine_settings_calb, 147

- set_entye_settings, [148](#)
- set_extended_settings, [148](#)
- set_extio_settings, [148](#)
- set_feedback_settings, [148](#)
- set_gear_information, [149](#)
- set_gear_settings, [149](#)
- set_hallsensor_information, [149](#)
- set_hallsensor_settings, [149](#)
- set_home_settings, [150](#)
- set_home_settings_calb, [150](#)
- set_joystick_settings, [150](#)
- set_logging_callback, [151](#)
- set_motor_information, [151](#)
- set_motor_settings, [151](#)
- set_move_settings, [151](#)
- set_move_settings_calb, [152](#)
- set_nonvolatile_memory, [152](#)
- set_pid_settings, [152](#)
- set_position, [152](#)
- set_position_calb, [153](#)
- set_power_settings, [153](#)
- set_secure_settings, [153](#)
- set_serial_number, [153](#)
- set_stage_information, [154](#)
- set_stage_name, [154](#)
- set_stage_settings, [154](#)
- set_sync_in_settings, [154](#)
- set_sync_in_settings_calb, [155](#)
- set_sync_out_settings, [155](#)
- set_sync_out_settings_calb, [155](#)
- set_uart_settings, [156](#)
- TS_TYPE_BITS, [116](#)
- UART_PARITY_BITS, [116](#)
- WIND_A_STATE_OK, [117](#)
- WIND_B_STATE_OK, [117](#)
- write_key, [156](#)
- XIMC_API, [117](#)
- ximc_fix_usbser_sys, [156](#)
- ximc_version, [156](#)
- ximc_fix_usbser_sys
 - ximc.h, [156](#)
- ximc_version
 - ximc.h, [156](#)