

libximc

2.13.3

Создано системой Doxygen 1.8.1.2

Пн 29 Ноя 2021 21:26:05

# Оглавление

<b>1 Библиотека libximc</b>	<b>1</b>
1.1 Что делает контроллер 8SMC4-USB и 8SMC5-USB.	1
1.2 Что умеет библиотека libximc.	1
1.3 Содействие.	2
<b>2 Введение</b>	<b>3</b>
2.1 О библиотеке	3
2.2 Требования к установленному программному обеспечению	3
2.2.1 Для сборки библиотеки	3
2.2.2 Для использования библиотеки	4
<b>3 Как пересобрать библиотеку</b>	<b>5</b>
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
<b>4 Как использовать с...</b>	<b>7</b>
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
<b>5</b>	<b>Работа с пользовательскими единицами</b>	<b>12</b>
5.1	Структура пересчета единиц calibration_t	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
<b>6</b>	<b>Структуры данных</b>	<b>14</b>
6.1	Структура accessories_settings_t	14
6.1.1	Подробное описание	14
6.1.2	Поля	15
6.1.2.1	LimitSwitchesSettings	15
6.1.2.2	MagneticBrakeInfo	15
6.1.2.3	MBRatedCurrent	15
6.1.2.4	MBRatedVoltage	15
6.1.2.5	MBSettings	15
6.1.2.6	MBTorque	15
6.1.2.7	TemperatureSensorInfo	15
6.1.2.8	TSGrad	15
6.1.2.9	TSMax	15
6.1.2.10	TSTMin	16
6.1.2.11	TSSettings	16
6.2	Структура analog_data_t	16
6.2.1	Подробное описание	17
6.2.2	Поля	17
6.2.2.1	A1Voltage	17
6.2.2.2	A1Voltage_ADC	17
6.2.2.3	A2Voltage	17
6.2.2.4	A2Voltage_ADC	18
6.2.2.5	ACurrent	18
6.2.2.6	ACurrent_ADC	18
6.2.2.7	B1Voltage	18
6.2.2.8	B1Voltage_ADC	18
6.2.2.9	B2Voltage	18
6.2.2.10	B2Voltage_ADC	18
6.2.2.11	BCurrent	18

6.2.2.12	BCurrent_ADC	18
6.2.2.13	FullCurrent	18
6.2.2.14	FullCurrent_ADC	18
6.2.2.15	H5	18
6.2.2.16	Joy	19
6.2.2.17	Joy_ADC	19
6.2.2.18	L5	19
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	20
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	22
6.5	Структура calibration_t	22
6.5.1	Подробное описание	22
6.6	Структура chart_data_t	22
6.6.1	Подробное описание	23
6.6.2	Поля	23
6.6.2.1	DutyCycle	23
6.6.2.2	Joy	23
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	24
6.7	Структура control_settings_calb_t	24
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	25
6.7.2.4	Timeout	25
6.8	Структура control_settings_t	25
6.8.1	Подробное описание	25
6.8.2	Поля	26
6.8.2.1	Flags	26
6.8.2.2	MaxClickTime	26
6.8.2.3	MaxSpeed	26
6.8.2.4	Timeout	26
6.8.2.5	uDeltaPosition	26
6.8.2.6	uMaxSpeed	26
6.9	Структура controller_name_t	26
6.9.1	Подробное описание	27
6.9.2	Поля	27
6.9.2.1	ControllerName	27
6.9.2.2	CtrlFlags	27
6.10	Структура ctr_settings_t	27
6.10.1	Подробное описание	27
6.10.2	Поля	28
6.10.2.1	CTPFlags	28
6.10.2.2	CTPMinError	28
6.11	Структура debug_read_t	28
6.11.1	Подробное описание	28
6.11.2	Поля	28
6.11.2.1	DebugData	28
6.12	Структура debug_write_t	28
6.12.1	Подробное описание	29
6.12.2	Поля	29
6.12.2.1	DebugData	29

6.13 Структура <code>device_information_t</code> . . . . .	29
6.13.1 Подробное описание . . . . .	29
6.13.2 Поля . . . . .	29
6.13.2.1 Major . . . . .	29
6.13.2.2 Minor . . . . .	30
6.13.2.3 Release . . . . .	30
6.14 Структура <code>device_network_information_t</code> . . . . .	30
6.14.1 Подробное описание . . . . .	30
6.15 Структура <code>edges_settings_calb_t</code> . . . . .	30
6.15.1 Подробное описание . . . . .	31
6.15.2 Поля . . . . .	31
6.15.2.1 BorderFlags . . . . .	31
6.15.2.2 EnderFlags . . . . .	31
6.15.2.3 LeftBorder . . . . .	31
6.15.2.4 RightBorder . . . . .	31
6.16 Структура <code>edges_settings_t</code> . . . . .	31
6.16.1 Подробное описание . . . . .	32
6.16.2 Поля . . . . .	32
6.16.2.1 BorderFlags . . . . .	32
6.16.2.2 EnderFlags . . . . .	32
6.16.2.3 LeftBorder . . . . .	32
6.16.2.4 RightBorder . . . . .	32
6.16.2.5 uLeftBorder . . . . .	32
6.16.2.6 uRightBorder . . . . .	32
6.17 Структура <code>emf_settings_t</code> . . . . .	32
6.17.1 Подробное описание . . . . .	33
6.17.2 Поля . . . . .	33
6.17.2.1 BackEMFFlags . . . . .	33
6.17.2.2 Km . . . . .	33
6.17.2.3 L . . . . .	33
6.17.2.4 R . . . . .	33
6.18 Структура <code>encoder_information_t</code> . . . . .	33
6.18.1 Подробное описание . . . . .	34
6.18.2 Поля . . . . .	34
6.18.2.1 Manufacturer . . . . .	34
6.18.2.2 PartNumber . . . . .	34
6.19 Структура <code>encoder_settings_t</code> . . . . .	34
6.19.1 Подробное описание . . . . .	35
6.19.2 Поля . . . . .	35

6.19.2.1	EncoderSettings	35
6.19.2.2	MaxCurrentConsumption	35
6.19.2.3	MaxOperatingFrequency	35
6.19.2.4	SupplyVoltageMax	35
6.19.2.5	SupplyVoltageMin	35
6.20	Структура engine_advanced_setup_t	35
6.20.1	Подробное описание	36
6.20.2	Поля	36
6.20.2.1	stepcloseloop_Kp_high	36
6.20.2.2	stepcloseloop_Kp_low	36
6.20.2.3	stepcloseloop_Kw	36
6.21	Структура engine_settings_calb_t	36
6.21.1	Подробное описание	37
6.21.2	Поля	37
6.21.2.1	Antiplay	37
6.21.2.2	EngineFlags	37
6.21.2.3	MicrostepMode	37
6.21.2.4	NomCurrent	37
6.21.2.5	NomSpeed	37
6.21.2.6	NomVoltage	38
6.21.2.7	StepsPerRev	38
6.22	Структура engine_settings_t	38
6.22.1	Подробное описание	38
6.22.2	Поля	39
6.22.2.1	Antiplay	39
6.22.2.2	EngineFlags	39
6.22.2.3	MicrostepMode	39
6.22.2.4	NomCurrent	39
6.22.2.5	NomSpeed	39
6.22.2.6	NomVoltage	39
6.22.2.7	StepsPerRev	39
6.22.2.8	uNomSpeed	39
6.23	Структура entype_settings_t	40
6.23.1	Подробное описание	40
6.23.2	Поля	40
6.23.2.1	DriverType	40
6.23.2.2	EngineType	40
6.24	Структура extended_settings_t	40
6.24.1	Подробное описание	40

6.25 Структура <code>extio_settings_t</code> . . . . .	41
6.25.1 Подробное описание . . . . .	41
6.25.2 Поля . . . . .	41
6.25.2.1 <code>EXTIOModeFlags</code> . . . . .	41
6.25.2.2 <code>EXTIOSetupFlags</code> . . . . .	41
6.26 Структура <code>feedback_settings_t</code> . . . . .	41
6.26.1 Подробное описание . . . . .	42
6.26.2 Поля . . . . .	42
6.26.2.1 <code>CountsPerTurn</code> . . . . .	42
6.26.2.2 <code>FeedbackFlags</code> . . . . .	42
6.26.2.3 <code>FeedbackType</code> . . . . .	42
6.26.2.4 <code>IPS</code> . . . . .	42
6.27 Структура <code>gear_information_t</code> . . . . .	42
6.27.1 Подробное описание . . . . .	43
6.27.2 Поля . . . . .	43
6.27.2.1 <code>Manufacturer</code> . . . . .	43
6.27.2.2 <code>PartNumber</code> . . . . .	43
6.28 Структура <code>gear_settings_t</code> . . . . .	43
6.28.1 Подробное описание . . . . .	43
6.28.2 Поля . . . . .	44
6.28.2.1 <code>Efficiency</code> . . . . .	44
6.28.2.2 <code>InputInertia</code> . . . . .	44
6.28.2.3 <code>MaxOutputBacklash</code> . . . . .	44
6.28.2.4 <code>RatedInputSpeed</code> . . . . .	44
6.28.2.5 <code>RatedInputTorque</code> . . . . .	44
6.28.2.6 <code>ReductionIn</code> . . . . .	44
6.28.2.7 <code>ReductionOut</code> . . . . .	44
6.29 Структура <code>get_position_calb_t</code> . . . . .	44
6.29.1 Подробное описание . . . . .	45
6.29.2 Поля . . . . .	45
6.29.2.1 <code>EncPosition</code> . . . . .	45
6.29.2.2 <code>Position</code> . . . . .	45
6.30 Структура <code>get_position_t</code> . . . . .	45
6.30.1 Подробное описание . . . . .	45
6.30.2 Поля . . . . .	46
6.30.2.1 <code>EncPosition</code> . . . . .	46
6.30.2.2 <code>uPosition</code> . . . . .	46
6.31 Структура <code>globally_unique_identifier_t</code> . . . . .	46
6.31.1 Подробное описание . . . . .	46



6.31.2 Поля	46
6.31.2.1 UniqueID0	46
6.31.2.2 UniqueID1	46
6.31.2.3 UniqueID2	46
6.31.2.4 UniqueID3	47
6.32 Структура hallsensor_information_t	47
6.32.1 Подробное описание	47
6.32.2 Поля	47
6.32.2.1 Manufacturer	47
6.32.2.2 PartNumber	47
6.33 Структура hallsensor_settings_t	47
6.33.1 Подробное описание	48
6.33.2 Поля	48
6.33.2.1 MaxCurrentConsumption	48
6.33.2.2 MaxOperatingFrequency	48
6.33.2.3 SupplyVoltageMax	48
6.33.2.4 SupplyVoltageMin	48
6.34 Структура home_settings_calb_t	48
6.34.1 Подробное описание	49
6.34.2 Поля	49
6.34.2.1 FastHome	49
6.34.2.2 HomeDelta	49
6.34.2.3 HomeFlags	49
6.34.2.4 SlowHome	49
6.35 Структура home_settings_t	49
6.35.1 Подробное описание	50
6.35.2 Поля	50
6.35.2.1 FastHome	50
6.35.2.2 HomeDelta	50
6.35.2.3 HomeFlags	50
6.35.2.4 SlowHome	50
6.35.2.5 uFastHome	50
6.35.2.6 uHomeDelta	51
6.35.2.7 uSlowHome	51
6.36 Структура init_random_t	51
6.36.1 Подробное описание	51
6.36.2 Поля	51
6.36.2.1 key	51
6.37 Структура joystick_settings_t	51

6.37.1	Подробное описание	52
6.37.2	Поля	52
6.37.2.1	DeadZone	52
6.37.2.2	ExpFactor	52
6.37.2.3	JoyCenter	52
6.37.2.4	JoyFlags	52
6.37.2.5	JoyHighEnd	53
6.37.2.6	JoyLowEnd	53
6.38	Структура measurements_t	53
6.38.1	Подробное описание	53
6.38.2	Поля	53
6.38.2.1	Error	53
6.38.2.2	Length	53
6.38.2.3	Speed	53
6.39	Структура motor_information_t	54
6.39.1	Подробное описание	54
6.39.2	Поля	54
6.39.2.1	Manufacturer	54
6.39.2.2	PartNumber	54
6.40	Структура motor_settings_t	54
6.40.1	Подробное описание	55
6.40.2	Поля	56
6.40.2.1	DetentTorque	56
6.40.2.2	MaxCurrent	56
6.40.2.3	MaxCurrentTime	56
6.40.2.4	MaxSpeed	56
6.40.2.5	MechanicalTimeConstant	56
6.40.2.6	MotorType	56
6.40.2.7	NoLoadCurrent	56
6.40.2.8	NoLoadSpeed	56
6.40.2.9	NominalCurrent	56
6.40.2.10	NominalPower	57
6.40.2.11	NominalSpeed	57
6.40.2.12	NominalTorque	57
6.40.2.13	NominalVoltage	57
6.40.2.14	Phases	57
6.40.2.15	Poles	57
6.40.2.16	RotorInertia	57
6.40.2.17	SpeedConstant	57

6.40.2.18SpeedTorqueGradient	57
6.40.2.19StallTorque	58
6.40.2.20TorqueConstant	58
6.40.2.21WindingInductance	58
6.40.2.22WindingResistance	58
6.41 Структура move_settings_calb_t	58
6.41.1 Подробное описание	58
6.41.2 Поля	59
6.41.2.1 Accel	59
6.41.2.2 AntiplaySpeed	59
6.41.2.3 Decel	59
6.41.2.4 MoveFlags	59
6.41.2.5 Speed	59
6.42 Структура move_settings_t	59
6.42.1 Подробное описание	60
6.42.2 Поля	60
6.42.2.1 Accel	60
6.42.2.2 AntiplaySpeed	60
6.42.2.3 Decel	60
6.42.2.4 MoveFlags	60
6.42.2.5 Speed	60
6.42.2.6 uAntiplaySpeed	60
6.42.2.7 uSpeed	60
6.43 Структура nonvolatile_memory_t	61
6.43.1 Подробное описание	61
6.43.2 Поля	61
6.43.2.1 UserData	61
6.44 Структура pid_settings_t	61
6.44.1 Подробное описание	62
6.45 Структура power_settings_t	62
6.45.1 Подробное описание	62
6.45.2 Поля	62
6.45.2.1 CurrentSetTime	62
6.45.2.2 CurrReductDelay	62
6.45.2.3 HoldCurrent	63
6.45.2.4 PowerFlags	63
6.45.2.5 PowerOffDelay	63
6.46 Структура secure_settings_t	63
6.46.1 Подробное описание	63

6.46.2 Поля	64
6.46.2.1 Criticalpwr	64
6.46.2.2 Criticalusb	64
6.46.2.3 CriticalUpwr	64
6.46.2.4 CriticalUusb	64
6.46.2.5 Flags	64
6.46.2.6 LowUpwrOff	64
6.46.2.7 MinimumUusb	64
6.47 Структура serial_number_t	64
6.47.1 Подробное описание	65
6.47.2 Поля	65
6.47.2.1 Key	65
6.47.2.2 Major	65
6.47.2.3 Minor	65
6.47.2.4 Release	65
6.47.2.5 SN	65
6.48 Структура set_position_calb_t	65
6.48.1 Подробное описание	66
6.48.2 Поля	66
6.48.2.1 EncPosition	66
6.48.2.2 PosFlags	66
6.48.2.3 Position	66
6.49 Структура set_position_t	66
6.49.1 Подробное описание	66
6.49.2 Поля	67
6.49.2.1 EncPosition	67
6.49.2.2 PosFlags	67
6.49.2.3 uPosition	67
6.50 Структура stage_information_t	67
6.50.1 Подробное описание	67
6.50.2 Поля	67
6.50.2.1 Manufacturer	67
6.50.2.2 PartNumber	67
6.51 Структура stage_name_t	68
6.51.1 Подробное описание	68
6.51.2 Поля	68
6.51.2.1 PositionerName	68
6.52 Структура stage_settings_t	68
6.52.1 Подробное описание	69

6.52.2 Поля	69
6.52.2.1 HorizontalLoadCapacity	69
6.52.2.2 LeadScrewPitch	69
6.52.2.3 MaxCurrentConsumption	69
6.52.2.4 MaxSpeed	69
6.52.2.5 SupplyVoltageMax	69
6.52.2.6 SupplyVoltageMin	69
6.52.2.7 TravelRange	69
6.52.2.8 Units	70
6.52.2.9 VerticalLoadCapacity	70
6.53 Структура status_calb_t	70
6.53.1 Подробное описание	71
6.53.2 Поля	71
6.53.2.1 CmdBufFreeSpace	71
6.53.2.2 CurPosition	71
6.53.2.3 CurSpeed	71
6.53.2.4 CurT	71
6.53.2.5 EncPosition	71
6.53.2.6 EncSts	71
6.53.2.7 Flags	71
6.53.2.8 GPIOFlags	71
6.53.2.9 lpwr	72
6.53.2.10 lusb	72
6.53.2.11 MoveSts	72
6.53.2.12 MvCmdSts	72
6.53.2.13 PWRSts	72
6.53.2.14 Upwr	72
6.53.2.15 Uusb	72
6.53.2.16 WindSts	72
6.54 Структура status_t	72
6.54.1 Подробное описание	73
6.54.2 Поля	73
6.54.2.1 CmdBufFreeSpace	73
6.54.2.2 CurPosition	74
6.54.2.3 CurSpeed	74
6.54.2.4 CurT	74
6.54.2.5 EncPosition	74
6.54.2.6 EncSts	74
6.54.2.7 Flags	74

6.54.2.8 GPIOFlags	74
6.54.2.9 lpwr	74
6.54.2.10 lusb	74
6.54.2.11 MoveSts	74
6.54.2.12 MvCmdSts	74
6.54.2.13 PWRSts	75
6.54.2.14 uCurPosition	75
6.54.2.15 uCurSpeed	75
6.54.2.16 Upwr	75
6.54.2.17 Uusb	75
6.54.2.18 WindSts	75
6.55 Структура sync_in_settings_calb_t	75
6.55.1 Подробное описание	76
6.55.2 Поля	76
6.55.2.1 ClutterTime	76
6.55.2.2 Position	76
6.55.2.3 Speed	76
6.55.2.4 SyncInFlags	76
6.56 Структура sync_in_settings_t	76
6.56.1 Подробное описание	77
6.56.2 Поля	77
6.56.2.1 ClutterTime	77
6.56.2.2 Speed	77
6.56.2.3 SyncInFlags	77
6.56.2.4 uPosition	77
6.56.2.5 uSpeed	77
6.57 Структура sync_out_settings_calb_t	77
6.57.1 Подробное описание	78
6.57.2 Поля	78
6.57.2.1 Accuracy	78
6.57.2.2 SyncOutFlags	78
6.57.2.3 SyncOutPeriod	78
6.57.2.4 SyncOutPulseSteps	78
6.58 Структура sync_out_settings_t	78
6.58.1 Подробное описание	79
6.58.2 Поля	79
6.58.2.1 Accuracy	79
6.58.2.2 SyncOutFlags	79
6.58.2.3 SyncOutPeriod	79

6.58.2.4 SyncOutPulseSteps . . . . .	79
6.58.2.5 uAccuracy . . . . .	79
6.59 Структура uart_settings_t . . . . .	80
6.59.1 Подробное описание . . . . .	80
6.59.2 Поля . . . . .	80
6.59.2.1 UARTSetupFlags . . . . .	80
<b>7 Файлы . . . . .</b>	<b>81</b>
7.1 Файл ximc.h . . . . .	81
7.1.1 Подробное описание . . . . .	106
7.1.2 Макросы . . . . .	106
7.1.2.1 ALARM_ON_DRIVER_OVERHEATING . . . . .	106
7.1.2.2 BACK_EMF_INDUCTANCE_AUTO . . . . .	106
7.1.2.3 BACK_EMF_KM_AUTO . . . . .	106
7.1.2.4 BACK_EMF_RESISTANCE_AUTO . . . . .	106
7.1.2.5 BORDER_IS_ENCODER . . . . .	107
7.1.2.6 BORDER_STOP_LEFT . . . . .	107
7.1.2.7 BORDER_STOP_RIGHT . . . . .	107
7.1.2.8 BORDERS_SWAP_MISSET_DETECTION . . . . .	107
7.1.2.9 BRAKE_ENABLED . . . . .	107
7.1.2.10 BRAKE_ENG_PWROFF . . . . .	107
7.1.2.11 CONTROL_BTN_LEFT_PUSHED_OPEN . . . . .	107
7.1.2.12 CONTROL_BTN_RIGHT_PUSHED_OPEN . . . . .	107
7.1.2.13 CONTROL_MODE_BITS . . . . .	107
7.1.2.14 CONTROL_MODE_JOY . . . . .	107
7.1.2.15 CONTROL_MODE_LR . . . . .	107
7.1.2.16 CONTROL_MODE_OFF . . . . .	108
7.1.2.17 CTP_ALARM_ON_ERROR . . . . .	108
7.1.2.18 CTP_BASE . . . . .	108
7.1.2.19 CTP_ENABLED . . . . .	108
7.1.2.20 CTP_ERROR_CORRECTION . . . . .	108
7.1.2.21 DRIVER_TYPE_DISCRETE_FET . . . . .	108
7.1.2.22 DRIVER_TYPE_EXTERNAL . . . . .	108
7.1.2.23 DRIVER_TYPE_INTEGRATE . . . . .	108
7.1.2.24 EEPROM_PRECEDENCE . . . . .	108
7.1.2.25 ENC_STATE_ABSENT . . . . .	108
7.1.2.26 ENC_STATE_MALFUNC . . . . .	108
7.1.2.27 ENC_STATE_OK . . . . .	109
7.1.2.28 ENC_STATE_REVERS . . . . .	109

7.1.2.29	ENC_STATE_UNKNOWN	109
7.1.2.30	ENDER_SW1_ACTIVE_LOW	109
7.1.2.31	ENDER_SW2_ACTIVE_LOW	109
7.1.2.32	ENDER_SWAP	109
7.1.2.33	ENGINE_ACCEL_ON	109
7.1.2.34	ENGINE_ANTIPLAY	109
7.1.2.35	ENGINE_CURRENT_AS_RMS	109
7.1.2.36	ENGINE_LIMIT_CURR	109
7.1.2.37	ENGINE_LIMIT_RPM	110
7.1.2.38	ENGINE_LIMIT_VOLT	110
7.1.2.39	ENGINE_MAX_SPEED	110
7.1.2.40	ENGINE_REVERSE	110
7.1.2.41	ENGINE_TYPE_2DC	110
7.1.2.42	ENGINE_TYPE_BRUSHLESS	110
7.1.2.43	ENGINE_TYPE_DC	110
7.1.2.44	ENGINE_TYPE_NONE	110
7.1.2.45	ENGINE_TYPE_STEP	110
7.1.2.46	ENGINE_TYPE_TEST	111
7.1.2.47	ENUMERATE_PROBE	111
7.1.2.48	EXTIO_SETUP_INVERT	111
7.1.2.49	EXTIO_SETUP_MODE_IN_ALARM	111
7.1.2.50	EXTIO_SETUP_MODE_IN_BITS	111
7.1.2.51	EXTIO_SETUP_MODE_IN_HOME	111
7.1.2.52	EXTIO_SETUP_MODE_IN_MOVR	111
7.1.2.53	EXTIO_SETUP_MODE_IN_NOP	111
7.1.2.54	EXTIO_SETUP_MODE_IN_PWOF	111
7.1.2.55	EXTIO_SETUP_MODE_IN_STOP	111
7.1.2.56	EXTIO_SETUP_MODE_OUT_ALARM	111
7.1.2.57	EXTIO_SETUP_MODE_OUT_BITS	112
7.1.2.58	EXTIO_SETUP_MODE_OUT_MOTOR_ON	112
7.1.2.59	EXTIO_SETUP_MODE_OUT_MOVING	112
7.1.2.60	EXTIO_SETUP_MODE_OUT_OFF	112
7.1.2.61	EXTIO_SETUP_MODE_OUT_ON	112
7.1.2.62	EXTIO_SETUP_OUTPUT	112
7.1.2.63	FEEDBACK_EMF	112
7.1.2.64	FEEDBACK_ENC_REVERSE	112
7.1.2.65	FEEDBACK_ENC_TYPE_AUTO	112
7.1.2.66	FEEDBACK_ENC_TYPE_BITS	112
7.1.2.67	FEEDBACK_ENC_TYPE_DIFFERENTIAL	112



7.1.2.68	FEEDBACK_ENC_TYPE_SINGLE_ENDED	112
7.1.2.69	FEEDBACK_ENCODER	113
7.1.2.70	FEEDBACK_ENCODER_MEDIATED	113
7.1.2.71	FEEDBACK_NONE	113
7.1.2.72	HOME_DIR_FIRST	113
7.1.2.73	HOME_DIR_SECOND	113
7.1.2.74	HOME_HALF_MV	113
7.1.2.75	HOME_MV_SEC_EN	113
7.1.2.76	HOME_STOP_FIRST_BITS	113
7.1.2.77	HOME_STOP_FIRST_LIM	113
7.1.2.78	HOME_STOP_FIRST_REV	113
7.1.2.79	HOME_STOP_FIRST_SYN	113
7.1.2.80	HOME_STOP_SECOND_BITS	114
7.1.2.81	HOME_STOP_SECOND_LIM	114
7.1.2.82	HOME_STOP_SECOND_REV	114
7.1.2.83	HOME_STOP_SECOND_SYN	114
7.1.2.84	HOME_USE_FAST	114
7.1.2.85	JOY_REVERSE	114
7.1.2.86	LOW_UPWR_PROTECTION	114
7.1.2.87	LS_SHORTED	114
7.1.2.88	MICROSTEP_MODE_FRAC_128	114
7.1.2.89	MICROSTEP_MODE_FRAC_16	114
7.1.2.90	MICROSTEP_MODE_FRAC_2	114
7.1.2.91	MICROSTEP_MODE_FRAC_256	114
7.1.2.92	MICROSTEP_MODE_FRAC_32	115
7.1.2.93	MICROSTEP_MODE_FRAC_4	115
7.1.2.94	MICROSTEP_MODE_FRAC_64	115
7.1.2.95	MICROSTEP_MODE_FRAC_8	115
7.1.2.96	MICROSTEP_MODE_FULL	115
7.1.2.97	MOVE_STATE_ANTIPLAY	115
7.1.2.98	MOVE_STATE_MOVING	115
7.1.2.99	MOVE_STATE_TARGET_SPEED	115
7.1.2.100	MVCMD_ERROR	115
7.1.2.101	MVCMD_HOME	115
7.1.2.102	MVCMD_LEFT	115
7.1.2.103	MVCMD_LOFT	116
7.1.2.104	MVCMD_MOVE	116
7.1.2.105	MVCMD_MOVR	116
7.1.2.106	MVCMD_NAME_BITS	116

7.1.2.107MVCMD_RIGHT	116
7.1.2.108MVCMD_RUNNING	116
7.1.2.109MVCMD_SSTP	116
7.1.2.110MVCMD_STOP	116
7.1.2.111MVCMD_UKNWN	116
7.1.2.112POWER_OFF_ENABLED	116
7.1.2.113POWER_REDUCT_ENABLED	116
7.1.2.114POWER_SMOOTH_CURRENT	117
7.1.2.115PWR_STATE_MAX	117
7.1.2.116PWR_STATE_NORM	117
7.1.2.117PWR_STATE_OFF	117
7.1.2.118PWR_STATE_REDUCT	117
7.1.2.119PWR_STATE_UNKNOWN	117
7.1.2.120REV_SENS_INV	117
7.1.2.121RPM_DIV_1000	117
7.1.2.122SETPOS_IGNORE_ENCODER	117
7.1.2.123SETPOS_IGNORE_POSITION	117
7.1.2.124STATE_ALARM	118
7.1.2.125STATE_BORDERS_SWAP_MISSET	118
7.1.2.126STATE_BRAKE	118
7.1.2.127STATE_BUTTON_LEFT	118
7.1.2.128STATE_BUTTON_RIGHT	118
7.1.2.129STATE_CONTR	118
7.1.2.130STATE_CONTROLLER_OVERHEAT	118
7.1.2.131STATE_CTP_ERROR	118
7.1.2.132STATE_DIG_SIGNAL	118
7.1.2.133STATE_EEPROM_CONNECTED	118
7.1.2.134STATE_ENC_A	118
7.1.2.135STATE_ENC_B	119
7.1.2.136STATE_ENGINE_RESPONSE_ERROR	119
7.1.2.137STATE_ERRC	119
7.1.2.138STATE_ERRD	119
7.1.2.139STATE_ERRV	119
7.1.2.140STATE_EXTIO_ALARM	119
7.1.2.141STATE_GPIO_LEVEL	119
7.1.2.142STATE_GPIO_PINOUT	119
7.1.2.143STATE_LEFT_EDGE	119
7.1.2.144STATE_LOW_USB_VOLTAGE	119
7.1.2.145STATE_OVERLOAD_POWER_CURRENT	119

7.1.2.146	STATE_OVERLOAD_POWER_VOLTAGE	119
7.1.2.147	STATE_OVERLOAD_USB_CURRENT	120
7.1.2.148	STATE_OVERLOAD_USB_VOLTAGE	120
7.1.2.149	STATE_POWER_OVERHEAT	120
7.1.2.150	STATE_REV_SENSOR	120
7.1.2.151	STATE_RIGHT_EDGE	120
7.1.2.152	STATE_SECUR	120
7.1.2.153	STATE_SYNC_INPUT	120
7.1.2.154	STATE_SYNC_OUTPUT	120
7.1.2.155	SYNCIN_ENABLED	120
7.1.2.156	SYNCIN_INVERT	120
7.1.2.157	SYNCOUT_ENABLED	120
7.1.2.158	SYNCOUT_IN_STEPS	121
7.1.2.159	SYNCOUT_INVERT	121
7.1.2.160	SYNCOUT_ONPERIOD	121
7.1.2.161	SYNCOUT_ONSTART	121
7.1.2.162	SYNCOUT_ONSTOP	121
7.1.2.163	SYNCOUT_STATE	121
7.1.2.164	TS_TYPE_BITS	121
7.1.2.165	UART_PARITY_BITS	121
7.1.2.166	WIND_A_STATE_ABSENT	121
7.1.2.167	WIND_A_STATE_MALFUNC	121
7.1.2.168	WIND_A_STATE_OK	121
7.1.2.169	WIND_A_STATE_UNKNOWN	122
7.1.2.170	WIND_B_STATE_ABSENT	122
7.1.2.171	WIND_B_STATE_MALFUNC	122
7.1.2.172	WIND_B_STATE_OK	122
7.1.2.173	WIND_B_STATE_UNKNOWN	122
7.1.2.174	XIMC_API	122
7.1.3	Типы	122
7.1.3.1	logging_callback_t	122
7.1.4	Функции	122
7.1.4.1	close_device	122
7.1.4.2	command_clear_fram	123
7.1.4.3	command_eeread_settings	123
7.1.4.4	command_eesave_settings	123
7.1.4.5	command_home	123
7.1.4.6	command_homezero	124
7.1.4.7	command_left	124

7.1.4.8	<code>command_loft</code>	124
7.1.4.9	<code>command_move</code>	124
7.1.4.10	<code>command_move_calb</code>	125
7.1.4.11	<code>command_movr</code>	125
7.1.4.12	<code>command_movr_calb</code>	125
7.1.4.13	<code>command_power_off</code>	126
7.1.4.14	<code>command_read_robust_settings</code>	126
7.1.4.15	<code>command_read_settings</code>	126
7.1.4.16	<code>command_reset</code>	126
7.1.4.17	<code>command_right</code>	127
7.1.4.18	<code>command_save_robust_settings</code>	127
7.1.4.19	<code>command_save_settings</code>	127
7.1.4.20	<code>command_sstp</code>	127
7.1.4.21	<code>command_start_measurements</code>	127
7.1.4.22	<code>command_stop</code>	127
7.1.4.23	<code>command_update_firmware</code>	128
7.1.4.24	<code>command_wait_for_stop</code>	128
7.1.4.25	<code>command_zero</code>	128
7.1.4.26	<code>enumerate_devices</code>	129
7.1.4.27	<code>free_enumerate_devices</code>	129
7.1.4.28	<code>get_accessories_settings</code>	129
7.1.4.29	<code>get_analog_data</code>	129
7.1.4.30	<code>get_bootloader_version</code>	130
7.1.4.31	<code>get_brake_settings</code>	130
7.1.4.32	<code>get_calibration_settings</code>	130
7.1.4.33	<code>get_chart_data</code>	130
7.1.4.34	<code>get_control_settings</code>	131
7.1.4.35	<code>get_control_settings_calb</code>	131
7.1.4.36	<code>get_controller_name</code>	131
7.1.4.37	<code>get_ctp_settings</code>	132
7.1.4.38	<code>get_debug_read</code>	132
7.1.4.39	<code>get_device_count</code>	132
7.1.4.40	<code>get_device_information</code>	132
7.1.4.41	<code>get_device_name</code>	133
7.1.4.42	<code>get_edges_settings</code>	133
7.1.4.43	<code>get_edges_settings_calb</code>	133
7.1.4.44	<code>get_emf_settings</code>	134
7.1.4.45	<code>get_encoder_information</code>	134
7.1.4.46	<code>get_encoder_settings</code>	134

7.1.4.47	<a href="#">get_engine_advanced_setup</a>	134
7.1.4.48	<a href="#">get_engine_settings</a>	135
7.1.4.49	<a href="#">get_engine_settings_calb</a>	135
7.1.4.50	<a href="#">get_entype_settings</a>	135
7.1.4.51	<a href="#">get_enumerate_device_controller_name</a>	136
7.1.4.52	<a href="#">get_enumerate_device_information</a>	136
7.1.4.53	<a href="#">get_enumerate_device_network_information</a>	136
7.1.4.54	<a href="#">get_enumerate_device_serial</a>	137
7.1.4.55	<a href="#">get_enumerate_device_stage_name</a>	137
7.1.4.56	<a href="#">get_extended_settings</a>	137
7.1.4.57	<a href="#">get_extio_settings</a>	137
7.1.4.58	<a href="#">get_feedback_settings</a>	138
7.1.4.59	<a href="#">get_firmware_version</a>	138
7.1.4.60	<a href="#">get_gear_information</a>	138
7.1.4.61	<a href="#">get_gear_settings</a>	138
7.1.4.62	<a href="#">get_globally_unique_identifier</a>	139
7.1.4.63	<a href="#">get_hallsensor_information</a>	139
7.1.4.64	<a href="#">get_hallsensor_settings</a>	139
7.1.4.65	<a href="#">get_home_settings</a>	139
7.1.4.66	<a href="#">get_home_settings_calb</a>	140
7.1.4.67	<a href="#">get_init_random</a>	140
7.1.4.68	<a href="#">get_joystick_settings</a>	140
7.1.4.69	<a href="#">get_measurements</a>	141
7.1.4.70	<a href="#">get_motor_information</a>	141
7.1.4.71	<a href="#">get_motor_settings</a>	141
7.1.4.72	<a href="#">get_move_settings</a>	142
7.1.4.73	<a href="#">get_move_settings_calb</a>	142
7.1.4.74	<a href="#">get_nonvolatile_memory</a>	142
7.1.4.75	<a href="#">get_pid_settings</a>	142
7.1.4.76	<a href="#">get_position</a>	143
7.1.4.77	<a href="#">get_position_calb</a>	143
7.1.4.78	<a href="#">get_power_settings</a>	143
7.1.4.79	<a href="#">get_secure_settings</a>	143
7.1.4.80	<a href="#">get_serial_number</a>	144
7.1.4.81	<a href="#">get_stage_information</a>	144
7.1.4.82	<a href="#">get_stage_name</a>	144
7.1.4.83	<a href="#">get_stage_settings</a>	144
7.1.4.84	<a href="#">get_status</a>	144
7.1.4.85	<a href="#">get_status_calb</a>	145

7.1.4.86	get_sync_in_settings	145
7.1.4.87	get_sync_in_settings_calb	146
7.1.4.88	get_sync_out_settings	146
7.1.4.89	get_sync_out_settings_calb	146
7.1.4.90	get_uart_settings	146
7.1.4.91	goto_firmware	147
7.1.4.92	has_firmware	147
7.1.4.93	load_correction_table	147
7.1.4.94	logging_callback_stderr_narrow	148
7.1.4.95	logging_callback_stderr_wide	148
7.1.4.96	msec_sleep	148
7.1.4.97	open_device	148
7.1.4.98	probe_device	149
7.1.4.99	reset_locks	149
7.1.4.100	service_command_updf	149
7.1.4.101	set_accessories_settings	149
7.1.4.102	set_bindy_key	150
7.1.4.103	set_brake_settings	150
7.1.4.104	set_calibration_settings	150
7.1.4.105	set_control_settings	150
7.1.4.106	set_control_settings_calb	151
7.1.4.107	set_controller_name	151
7.1.4.108	set_correction_table	151
7.1.4.109	set_ctp_settings	152
7.1.4.110	set_debug_write	152
7.1.4.111	set_edges_settings	152
7.1.4.112	set_edges_settings_calb	153
7.1.4.113	set_emf_settings	153
7.1.4.114	set_encoder_information	153
7.1.4.115	set_encoder_settings	154
7.1.4.116	set_engine_advansed_setup	154
7.1.4.117	set_engine_settings	154
7.1.4.118	set_engine_settings_calb	155
7.1.4.119	set_entype_settings	155
7.1.4.120	set_extended_settings	155
7.1.4.121	set_extio_settings	155
7.1.4.122	set_feedback_settings	156
7.1.4.123	set_gear_information	156
7.1.4.124	set_gear_settings	156

7.1.4.125set_hallsensor_information . . . . .	157
7.1.4.126set_hallsensor_settings . . . . .	157
7.1.4.127set_home_settings . . . . .	157
7.1.4.128set_home_settings_calb . . . . .	157
7.1.4.129set_joystick_settings . . . . .	158
7.1.4.130set_logging_callback . . . . .	158
7.1.4.131set_motor_information . . . . .	158
7.1.4.132set_motor_settings . . . . .	159
7.1.4.133set_move_settings . . . . .	159
7.1.4.134set_move_settings_calb . . . . .	159
7.1.4.135set_nonvolatile_memory . . . . .	159
7.1.4.136set_pid_settings . . . . .	159
7.1.4.137set_position . . . . .	160
7.1.4.138set_position_calb . . . . .	160
7.1.4.139set_power_settings . . . . .	160
7.1.4.140set_secure_settings . . . . .	161
7.1.4.141set_serial_number . . . . .	161
7.1.4.142set_stage_information . . . . .	161
7.1.4.143set_stage_name . . . . .	161
7.1.4.144set_stage_settings . . . . .	162
7.1.4.145set_sync_in_settings . . . . .	162
7.1.4.146set_sync_in_settings_calb . . . . .	162
7.1.4.147set_sync_out_settings . . . . .	162
7.1.4.148set_sync_out_settings_calb . . . . .	163
7.1.4.149set_uart_settings . . . . .	163
7.1.4.150write_key . . . . .	163
7.1.4.151ximc_fix_usbser_sys . . . . .	164
7.1.4.152ximc_version . . . . .	164

# Глава 1

## Библиотека libximc

Документация для библиотеки libximc.

Libximc - **потокобезопасная**, 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.



### Предупреждения

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

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

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

## 1.3 Содействие.

Большое спасибо всем, кто отправляет 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](mailto: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 инструментария 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/test_CSharp'`, на VB.NET - в `'examples/test_VBNET'`, для delphi 6 - в `'example/test_Delphi'`, для matlab - `'examples/test_MATLAB'`, для Java - `'examples/test_Java'`, для Python - `'examples/test_Python'`. 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, `test_CSharp`, `test_VBNET`, `test_Delphi` - только 32 бита, `test_Java` - кроссплатформенный, `test_MATLAB` и `test_Python` 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 `libximc7_x.x.x` и `libximc7-dev_x.x.x` целевой архитектуры в указанном порядке. 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 `test_CodeBlocks.cbp`. Для компиляции необходимо использовать также MS Visual C++, `mingw-library` не поддерживается. Убедитесь, что Microsoft Visual C++ Redistributable Package установлен.

Откройте проект `examples/test_CodeBlocks/test_CodeBlocks.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 tarballs) 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 `test_CSharp` (для C#) и `test_VBNET` (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 `'test_Delphi'`. 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 `test_Delphi.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/test_Java/compiled/` и выполните

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

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

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

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

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



Затем пересоберите исходные тексты:

```
$ javac -classpath /usr/share/java/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 test_Java.jar ru
```

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

## 4.5 Python

Измените текущую директорию на examples/test\_Python. 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 test_Python.py
```

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле test\_Python.py перед запуском нужно прописать IP адрес Ethernet-адаптера (переменная enum\_hints).

## 4.6 MATLAB

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

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](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 дублиры для работы с пользовательскими единицами и структуры данных для них
- Таблица коррекции координат для более точного позиционирования

### 5.1 Структура пересчета единиц `calibration_t`

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

Формулы пересчета:

- Пересчет в пользовательские единицы.

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

- Пересчет из пользовательских единиц.

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

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

Структуры и функции для работы с пользовательскими единицами имеют постфикс `_calb`. Пользователь используя данные функции может выполнять все действия в собственных единицах не беспокоясь о том, что и как считает контроллер. 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 A" необработанные данные с АЦП.*
- `unsigned int` [A2Voltage\\_ADC](#)  
*"Выходное напряжение на 2 выводе обмотки A" необработанные данные с АЦП.*
- `unsigned int` [B1Voltage\\_ADC](#)  
*"Выходное напряжение на 1 выводе обмотки B" необработанные данные с АЦП.*
- `unsigned int` [B2Voltage\\_ADC](#)  
*"Выходное напряжение на 2 выводе обмотки B" необработанные данные с АЦП.*
- `unsigned int` [SupVoltage\\_ADC](#)  
*"Напряжение питания ключей H-моста" необработанные данные с АЦП.*
- `unsigned int` [ACurrent\\_ADC](#)  
*"Ток через обмотку A" необработанные данные с АЦП.*
- `unsigned int` [BCurrent\\_ADC](#)  
*"Ток через обмотку B" необработанные данные с АЦП.*
- `unsigned int` [FullCurrent\\_ADC](#)  
*"Полный ток" необработанные данные с АЦП.*
- `unsigned int` [Temp\\_ADC](#)  
*Напряжение с датчика температуры, необработанные данные с АЦП.*
- `unsigned int` [Joy\\_ADC](#)  
*Джойстик, необработанные данные с АЦП.*
- `unsigned int` [Pot\\_ADC](#)  
*Напряжение на аналоговом входе, необработанные данные с АЦП.*
- `unsigned int` [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 A" откалиброванные данные (в десятках мВ).*
- `int` [A2Voltage](#)  
*"Выходное напряжение на 2 выводе обмотки A" откалиброванные данные (в десятках мВ).*
- `int` [B1Voltage](#)  
*"Выходное напряжение на 1 выводе обмотки B" откалиброванные данные (в десятках мВ).*
- `int` [B2Voltage](#)

- `int SupVoltage`  
"Выходное напряжение на 2 выводе обмотки B" откалиброванные данные (в десятках мВ).
- `int ACurrent`  
"Напряжение питания ключей H-моста" откалиброванные данные (в десятках мВ).
- `int BCurrent`  
"Ток через обмотку A" 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 B" откалиброванные данные (в десятках мВ).

6.2.2.8 `unsigned int B1Voltage_ADC`

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

6.2.2.9 `int B2Voltage`

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

6.2.2.10 `unsigned int B2Voltage_ADC`

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

6.2.2.11 `int BCurrent`

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

6.2.2.12 `unsigned int BCurrent_ADC`

"Ток через обмотку B" необработанные данные с АЦП.

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

#### 6.4.1 Подробное описание

Калибровочные коэффициенты.

Эта структура содержит калибровочные коэффициенты.

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](#)  
*Multplier.*
- 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 XIMC 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](#)  
*Электромеханический коэффициент двигателя.*
- unsigned int [BackEMFFlags](#)  
*Флаги автоопределения характеристик обмоток двигателя.*

### 6.17.1 Подробное описание

Настройки EMF.

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

См. также

```
set_emf_settings  
get_emf_settings  
get_emf_settings, set_emf_settings
```

### 6.17.2 Поля

#### 6.17.2.1 unsigned int BackEMFFlags

*Флаги автоопределения характеристик обмоток двигателя.*

#### 6.17.2.2 float Km

*Электромеханический коэффициент двигателя.*

#### 6.17.2.3 float L

*Индуктивность обмоток двигателя.*

#### 6.17.2.4 float R

*Сопротивление обмоток двигателя.*

## 6.18 Структура `encoder_information_t`

Информация об энкодере.



Поля данных

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

#### 6.18.1 Подробное описание

Информация об энкодере.

См. также

`set_encoder_information`  
`get_encoder_information`  
`get_encoder_information, set_encoder_information`

#### 6.18.2 Поля

##### 6.18.2.1 `char Manufacturer`[17]

Производитель.

Максимальная длина строки: 16 символов.

##### 6.18.2.2 `char PartNumber`[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

### 6.19 Структура `encoder_settings_t`

Настройки энкодера.

Поля данных

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

## 6.19.1 Подробное описание

Настройки энкодера.

См. также

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

## 6.19.2 Поля

## 6.19.2.1 unsigned int EncoderSettings

[Флаги настроек энкодера.](#)

## 6.19.2.2 float MaxCurrentConsumption

Максимальное потребление тока (мА).

Тип данных: float.

## 6.19.2.3 float MaxOperatingFrequency

Максимальная частота (кГц).

Тип данных: float.

## 6.19.2.4 float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

## 6.19.2.5 float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

## 6.20 Структура engine\_advanced\_setup\_t

Настройки EAS.

Поля данных

- unsigned int [stepcloseloop\\_Kw](#)  
*Коэффициент смещения реальной и заданной скорости, диапазон [0, 100], значение по умолчанию 50.*
- unsigned int [stepcloseloop\\_Kp\\_low](#)  
*Обратная связь по позиции в зоне малых скоростей, диапазон [0, 65535], значение по умолчанию 1000.*

- unsigned int [stepcloseloop\\_Kp\\_high](#)

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

### 6.20.1 Подробное описание

Настройки EAS.

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

## 6.22.1 Подробное описание

Ограничения и настройки движения, связанные с двигателем.

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

См. также

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

## 6.22.2 Поля

## 6.22.2.1 int Antiplay

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

Используется, если установлен флаг ENGINE\_ANTIPLAY.

## 6.22.2.2 unsigned int EngineFlags

Флаги параметров мотора.

## 6.22.2.3 unsigned int MicrostepMode

Флаги параметров микрошагового режима.

## 6.22.2.4 unsigned int NomCurrent

Номинальный ток через мотор (в мА).

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

## 6.22.2.5 unsigned int NomSpeed

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

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

## 6.22.2.6 unsigned int NomVoltage

Номинальное напряжение мотора в десятках мВ.

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

## 6.22.2.7 unsigned int StepsPerRev

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

Диапазон: 1..65535.

## 6.22.2.8 unsigned int uNomSpeed

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

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

## 6.23 Структура `entype_settings_t`

Настройки типа мотора и типа силового драйвера.

Поля данных

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

### 6.23.1 Подробное описание

Настройки типа мотора и типа силового драйвера.

Эта структура содержит настройки типа мотора и типа силового драйвера.

Аргументы

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

См. также

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

### 6.23.2 Поля

#### 6.23.2.1 unsigned int `DriverType`

*Флаги, определяющие тип силового драйвера.*

#### 6.23.2.2 unsigned int `EngineType`

*Флаги, определяющие тип мотора.*

## 6.24 Структура `extended_settings_t`

Настройки 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 см<sup>2</sup>).*
- float [Efficiency](#)  
*КПД редуктора (%).*

### 6.28.1 Подробное описание

Настройки редуктора.

См. также

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

## 6.28.2 Поля

### 6.28.2.1 float Efficiency

КПД редуктора (%).

Тип данных: float.

### 6.28.2.2 float InputInertia

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

Тип данных: float.

### 6.28.2.3 float MaxOutputBacklash

Выходной люфт редуктора (градус).

Тип данных: float.

### 6.28.2.4 float RatedInputSpeed

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

Тип данных: float.

### 6.28.2.5 float RatedInputTorque

Максимальный крутящий момент (Н м).

Тип данных: float.

### 6.28.2.6 float ReductionIn

Входной коэффициент редуктора.

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

### 6.28.2.7 float ReductionOut

Выходной коэффициент редуктора.

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

## 6.29 Структура `get_position_calb_t`

Данные о позиции.

Поля данных

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

#### 6.29.1 Подробное описание

Данные о позиции.

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

См. также

[get\\_position](#)

#### 6.29.2 Поля

##### 6.29.2.1 long\_t EncPosition

Позиция энкодера.

##### 6.29.2.2 float Position

Позиция двигателя.

Корректируется таблицей.

### 6.30 Структура `get_position_t`

Данные о позиции.

Поля данных

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

#### 6.30.1 Подробное описание

Данные о позиции.

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

См. также

[get\\_position](#)

### 6.30.2 Поля

#### 6.30.2.1 `long_t` `EncPosition`

Позиция энкодера.

#### 6.30.2.2 `int` `uPosition`

Позиция в микрошагах (используется только с шаговыми двигателями).

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

## 6.31 Структура `globally_unique_identifier_t`

Глобальный уникальный идентификатор.

Поля данных

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

### 6.31.1 Подробное описание

Глобальный уникальный идентификатор.

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](#)  
*Флаги типа двигателя.*
- unsigned int [ReservedField](#)  
*Зарезервировано*
- unsigned int [Poles](#)  
*Кол-во пар полюсов у DC или BLDC двигателя или кол-во шагов на оборот для шагового двигателя.*
- unsigned int [Phases](#)  
*Кол-во фаз у BLDC двигателя.*
- float [NominalVoltage](#)

- Номинальное напряжение на обмотке (В).
- float `NominalCurrent`  
Максимальный постоянный ток в обмотке для DC и BLDC двигателей, номинальный ток в обмотке для шаговых двигателей (А).
- float `NominalSpeed`  
Не используется.
- float `NominalTorque`  
Номинальный крутящий момент (мН м).
- float `NominalPower`  
Номинальная мощность (Вт).
- float `WindingResistance`  
Сопротивление обмотки DC двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC двигателя (Ом).
- float `WindingInductance`  
Индуктивность обмотки DC двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC двигателя (мГн).
- float `RotorInertia`  
Инерция ротора (г см<sup>2</sup>).
- float `StallTorque`  
Крутящий момент удержания позиции для шагового двигателя или крутящий момент при неподвижном роторе для других типов двигателей (мН м).
- float `DetentTorque`  
Момент удержания позиции с незапитанными обмотками (мН м).
- float `TorqueConstant`  
Константа крутящего момента, определяющая коэффициент пропорциональности максимального момента силы ротора от протекающего в обмотке тока (мН м/А).
- float `SpeedConstant`  
Константа скорости, определяющая значение или амплитуду напряжения наведённой индукции при вращении ротора DC или BLDC двигателя (об/мин / В) или шагового двигателя (шаг/с / В).
- float `SpeedTorqueGradient`  
Градиент крутящего момента (об/мин / мН м).
- float `MechanicalTimeConstant`  
Механическая постоянная времени (мс).
- float `MaxSpeed`  
Максимальная разрешённая скорость для шаговых двигателей (шаг/с) или для DC и BLDC двигателей (об/мин).
- float `MaxCurrent`  
Максимальный ток в обмотке (А).
- float `MaxCurrentTime`  
Безопасная длительность максимального тока в обмотке (мс).
- float `NoLoadCurrent`  
Ток потребления в холостом режиме (А).
- float `NoLoadSpeed`  
Скорость в холостом режиме (об/мин).

#### 6.40.1 Подробное описание

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`

Флаги типа двигателя.

6.40.2.7 `float NoLoadCurrent`

Ток потребления в холостом режиме (А).

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

6.40.2.8 `float NoLoadSpeed`

Скорость в холостом режиме (об/мин).

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

6.40.2.9 `float NominalCurrent`

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

Тип данных: `float`.

## 6.40.2.10 float NominalPower

Номинальная мощность(Вт).

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

## 6.40.2.11 float NominalSpeed

Не используется.

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

## 6.40.2.12 float NominalTorque

Номинальный крутящий момент (мН м).

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

## 6.40.2.13 float NominalVoltage

Номинальное напряжение на обмотке (В).

Тип данных: float.

## 6.40.2.14 unsigned int Phases

Кол-во фаз у BLDC двигателя.

## 6.40.2.15 unsigned int Poles

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

## 6.40.2.16 float RotorInertia

Инерция ротора (г см<sup>2</sup>).

Тип данных: float.

## 6.40.2.17 float SpeedConstant

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

Тип данных: float.

## 6.40.2.18 float SpeedTorqueGradient

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

Тип данных: float.



## 6.40.2.19 float StallTorque

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

Тип данных: float.

## 6.40.2.20 float TorqueConstant

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

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

## 6.40.2.21 float WindingInductance

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

Тип данных: float.

## 6.40.2.22 float WindingResistance

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 секунду<sup>2</sup>(ШД) или в оборотах в минуту за секунду(DC).*
- float [Decel](#)  
*Торможение, заданное в шагах в секунду<sup>2</sup>(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 секунду<sup>2</sup>(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 секунду<sup>2</sup>(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 секунду<sup>2</sup>(ШД) или в оборотах в минуту за секунду(DC).*
- unsigned int [Decel](#)  
*Торможение, заданное в шагах в секунду<sup>2</sup>(ШД) или в оборотах в минуту за секунду(DC).*
- unsigned int [AntiplaySpeed](#)  
*Скорость в режиме антилюфта, заданная в целых шагах/с(ШД) или в оборотах/с(DC).*
- unsigned int [uAntiplaySpeed](#)  
*Скорость в режиме антилюфта, выраженная в микрошагах в секунду.*
- unsigned int [MoveFlags](#)  
*[Флаги параметров движения](#).*

## 6.42.1 Подробное описание

Настройки движения.

См. также

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

## 6.42.2 Поля

6.42.2.1 `unsigned int Accel`

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

Диапазон: 1..65535.

6.42.2.2 `unsigned int AntiplaySpeed`

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

Диапазон: 0..100000.

6.42.2.3 `unsigned int Decel`

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

Диапазон: 1..65535.

6.42.2.4 `unsigned int MoveFlags`

[Флаги параметров движения.](#)

6.42.2.5 `unsigned int Speed`

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

Диапазон: 0..100000.

6.42.2.6 `unsigned int uAntiplaySpeed`

Скорость в режиме антилюфта, выраженная в микрошагах в секунду.

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

6.42.2.7 `unsigned int uSpeed`

Заданная скорость в единицах деления микрошага в секунду.

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

## 6.43 Структура `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 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 `Criticalpwr`  
*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 `Criticalusb`  
*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 CriticalPwr`

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

6.46.2.2 `unsigned int CriticalUsb`

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 MinimumUsb`

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

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

#### 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 lpwr

Ток потребления силовой части, мА.

6.53.2.10 int lusb

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

#### 6.54.1 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`  
*Настройки ПИД.*
- struct `sync_in_settings_t`  
*Настройки входной синхронизации.*
- struct `sync_in_settings_calb_t`  
*Настройки входной синхронизации с использованием пользовательских единиц.*
- struct `sync_out_settings_t`  
*Настройки выходной синхронизации.*
- struct `sync_out_settings_calb_t`  
*Настройки выходной синхронизации с использованием пользовательских единиц.*
- struct `extio_settings_t`  
*Настройки EXTIO.*
- struct `brake_settings_t`  
*Настройки тормоза.*
- struct `control_settings_t`  
*Настройки управления.*
- struct `control_settings_calb_t`  
*Настройки управления с использованием пользовательских единиц.*
- struct `joystick_settings_t`  
*Настройки джойстика.*
- struct `ctp_settings_t`  
*Настройки контроля позиции(для шагового двигателя).*
- struct `uart_settings_t`  
*Настройки UART.*
- struct `calibration_settings_t`  
*Калибровочные коэффициенты.*
- struct `controller_name_t`  
*Пользовательское имя контроллера и флаги настройки.*
- struct `nonvolatile_memory_t`  
*Пользовательские данные для сохранения во FRAM.*
- struct `emf_settings_t`  
*Настройки EMF.*
- struct `engine_advansed_setup_t`  
*Настройки EAS.*
- struct `extended_settings_t`  
*Настройки EAS.*
- struct `get_position_t`  
*Данные о позиции.*
- struct `get_position_calb_t`  
*Данные о позиции.*
- struct `set_position_t`  
*Данные о позиции.*
- struct `set_position_calb_t`  
*Данные о позиции с использованием пользовательских единиц.*
- struct `status_t`  
*Состояние устройства.*
- struct `status_calb_t`  
*Состояние устройства с использованием пользовательских единиц.*
- struct `measurements_t`

- `struct chart_data_t`  
*Буфер вмещает не более 25и точек.*
- `struct device_information_t`  
*Дополнительное состояние устройства.*
- `struct serial_number_t`  
*Команда чтения информации о контроллере.*
- `struct analog_data_t`  
*Структура с серийным номером и версией железа.*
- `struct debug_read_t`  
*Аналоговые данные.*
- `struct debug_write_t`  
*Отладочные данные.*
- `struct stage_name_t`  
*Отладочные данные.*
- `struct stage_information_t`  
*Пользовательское имя подвижки.*
- `struct stage_settings_t`  
*Информация о позиционере.*
- `struct motor_information_t`  
*Настройки позиционера.*
- `struct motor_settings_t`  
*Информация о двигателе.*
- `struct encoder_information_t`  
*Физический характеристики и ограничения мотора.*
- `struct encoder_settings_t`  
*Информация об энкодере.*
- `struct hallsensor_information_t`  
*Настройки энкодера.*
- `struct hallsensor_settings_t`  
*Информация о датчиках Холла.*
- `struct gear_information_t`  
*Настройки датчиков Холла.*
- `struct gear_settings_t`  
*Информация о редукторе.*
- `struct accessories_settings_t`  
*Настройки редуктора.*
- `struct init_random_t`  
*Информация о дополнительных аксессуарах.*
- `struct globally_unique_identifier_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*`

*Thread return type.*

- #define `device_undefined` -1  
*Макрос, означающий неопределенное устройство*

### Результаты выполнения команд

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

### Уровень логирования

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

### Флаги поиска устройств

*Это битовая маска для побитовых операций.*

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

### Флаги состояния движения

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

См. также

`get_status`  
`status_t::MoveSts, get_status_impl`

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

### Флаги настроек контроллера

*Это битовая маска для побитовых операций.*

См. также

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

- `#define EEPROM_PRECEDENCE 0x01`

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

### Флаги состояния питания шагового мотора

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

См. также

`get_status`  
`status_t::PWRSts, get_status_impl`

- `#define PWR_STATE_UNKNOWN 0x00`

*Неизвестное состояние, которое не должно никогда реализовываться.*

- `#define PWR_STATE_OFF 0x01`

*Обмотки мотора разомкнуты и не управляются драйвером.*

- `#define PWR_STATE_NORM 0x03`

*Обмотки запитаны номинальным током.*

- `#define PWR_STATE_REDUCT 0x04`

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

- `#define PWR_STATE_MAX 0x05`

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

### Флаги состояния

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

См. также

`get_status`  
`status_t::Flags, get_status_impl`

- `#define STATE_CONTR 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 0x0000200
- *Перегрелась микросхема контроллера.*
- #define STATE\_OVERLOAD\_POWER\_VOLTAGE 0x0000400
- *Превышено напряжение на силовой части.*
- #define STATE\_OVERLOAD\_POWER\_CURRENT 0x0000800
- *Превышен максимальный ток потребления силовой части.*
- #define STATE\_OVERLOAD\_USB\_VOLTAGE 0x0001000
- *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

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

См. также

[get\\_status](#)

[status\\_t::GPIOFlags](#), [get\\_status\\_impl](#)

- #define STATE\_DIG\_SIGNAL 0xFFFF
- *Флаги цифровых сигналов.*
- #define STATE\_RIGHT\_EDGE 0x0001
- *Достижение правой границы.*
- #define STATE\_LEFT\_EDGE 0x0002
- *Достижение левой границы.*
- #define STATE\_BUTTON\_RIGHT 0x0004
- *Состояние кнопки "вправо" (1, если нажата).*
- #define STATE\_BUTTON\_LEFT 0x0008
- *Состояние кнопки "влево" (1, если нажата).*
- #define STATE\_GPIO\_PINOUT 0x0010
- *Если флаг установлен, ввод/вывод общего назначения работает как выход; если флаг сброшен, ввод/вывод работает как вход.*
- #define STATE\_GPIO\_LEVEL 0x0020
- *Состояние ввода/вывода общего назначения.*
- #define STATE\_BRAKE 0x0200
- *Состояние вывода управления тормозом.*
- #define STATE\_REV\_SENSOR 0x0400
- *Состояние вывода датчика оборотов(флаг "1", если датчик активен).*
- #define STATE\_SYNC\_INPUT 0x0800

- `#define STATE_SYNC_OUTPUT 0x1000`  
Состояние входа синхронизации(1, если вход синхронизации активен).
- `#define STATE_ENC_A 0x2000`  
Состояние выхода синхронизации(1, если выход синхронизации активен).
- `#define STATE_ENC_B 0x4000`  
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

Это битовая маска для побитовых операций. 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 должным образом.

### Состояние обмоток

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

См. также

`get_status`  
`status_t::WindSts, get_status_impl`

- `#define WIND_A_STATE_ABSENT 0x00`  
Обмотка 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

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

См. также

[get\\_status](#)  
[status\\_t::MvCmdSts](#), [get\\_status\\_impl](#)

- #define [MVCMD\\_NAME\\_BITS](#) 0x3F  
*Битовая маска активной команды.*
- #define [MVCMD\\_UKNWN](#) 0x00  
*Неизвестная команда.*
- #define [MVCMD\\_MOVE](#) 0x01  
*Команда move.*
- #define [MVCMD\\_MOVR](#) 0x02  
*Команда movr.*
- #define [MVCMD\\_LEFT](#) 0x03  
*Команда left.*
- #define [MVCMD\\_RIGHT](#) 0x04  
*Команда rigt.*
- #define [MVCMD\\_STOP](#) 0x05  
*Команда stop.*
- #define [MVCMD\\_HOME](#) 0x06  
*Команда home.*
- #define [MVCMD\\_LOFT](#) 0x07  
*Команда loft.*
- #define [MVCMD\\_SSTP](#) 0x08  
*Команда плавной остановки(SSTP).*
- #define [MVCMD\\_ERROR](#) 0x40  
*Состояние завершения движения (1 - команда движения выполнена с ошибкой, 0 - команда движения выполнена корректно).*
- #define [MVCMD\\_RUNNING](#) 0x80  
*Состояние команды движения (0 - команда движения выполнена, 1 - команда движения сейчас выполняется).*

### Флаги параметров движения

Это битовая маска для побитовых операций. Определяют настройки параметров движения. Возвращаются командой [get\\_move\\_settings](#).

См. также

[set\\_move\\_settings](#)  
[get\\_move\\_settings](#)  
[move\\_settings\\_t::MoveFlags](#), [get\\_move\\_settings](#), [set\\_move\\_settings](#)

- #define [RPM\\_DIV\\_1000](#) 0x01  
*Флаг указывает на то что рабочая скорость указанная в команде задана в милли rpm.*

### Флаги параметров мотора

Это битовая маска для побитовых операций. Определяют настройки движения и работу ограничителей. Возвращаются командой [get\\_engine\\_settings](#). Могут быть объединены с помощью логического ИЛИ.

См. также

[set\\_engine\\_settings](#)  
[get\\_engine\\_settings](#)  
[engine\\_settings\\_t::EngineFlags](#), [get\\_engine\\_settings](#), [set\\_engine\\_settings](#)

- #define [ENGINE\\_REVERSE](#) 0x01  
*Флаг реверса.*
- #define [ENGINE\\_CURRENT\\_AS\\_RMS](#) 0x02

- Флаг интерпретации значения тока.  
• `#define ENGINE_MAX_SPEED 0x04`
- Флаг максимальной скорости.  
• `#define ENGINE_ANTIPLAY 0x08`
- Компенсация люфта.  
• `#define ENGINE_ACCEL_ON 0x10`
- Ускорение.  
• `#define ENGINE_LIMIT_VOLT 0x20`
- Номинальное напряжение мотора.  
• `#define ENGINE_LIMIT_CURR 0x40`
- Номинальный ток мотора.  
• `#define ENGINE_LIMIT_RPM 0x80`
- Номинальная частота вращения мотора.

### Флаги параметров микрошагового режима

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

См. также

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

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

### Флаги, определяющие тип мотора

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

См. также

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

- `#define ENGINE_TYPE_NONE 0x00`  
Это значение не нужно использовать.



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

### Флаги, определяющие тип силового драйвера

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

См. также

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

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

### Флаги параметров питания шагового мотора

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

См. также

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

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

### Флаги критических параметров.

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

См. также

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

- `#define ALARM_ON_DRIVER_OVERHEATING 0x01`  
*Если флаг установлен, то войти в состояние `Alarm` при получении сигнала подступающего перегрева с драйвера.*

- `#define LOW_UPWR_PROTECTION 0x02`  
*Если установлен, то выключать силовую часть при напряжении меньшем LowUpwrOff.*
- `#define H_BRIDGE_ALERT 0x04`  
*Если установлен, то выключать силовую часть при сигнале неполадки в одном из транзисторных мостов.*
- `#define ALARM_ON_BORDERS_SWAP_MISSET 0x08`  
*Если флаг установлен, то войти в состояние Alarm при получении сигнала с противоположного 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

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

См. также

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

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

### Тип обратной связи.

Это битовая маска для побитовых операций.

См. также

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

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

### Флаги обратной связи.

Это битовая маска для побитовых операций.

См. также

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

- #define [FEEDBACK\\_ENC\\_REVERSE](#) 0x01  
Обратный счет у энкодера.
- #define [FEEDBACK\\_ENC\\_TYPE\\_BITS](#) 0xC0  
Биты, отвечающие за тип энкодера.
- #define [FEEDBACK\\_ENC\\_TYPE\\_AUTO](#) 0x00  
Определяет тип энкодера автоматически.
- #define [FEEDBACK\\_ENC\\_TYPE\\_SINGLE\\_ENDED](#) 0x40  
Недифференциальный энкодер.
- #define [FEEDBACK\\_ENC\\_TYPE\\_DIFFERENTIAL](#) 0x80  
Дифференциальный энкодер.

### Флаги настроек синхронизации входа

Это битовая маска для побитовых операций.

См. также

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

- #define [SYNCIN\\_ENABLED](#) 0x01  
Включение необходимости импульса синхронизации для начала движения.
- #define [SYNCIN\\_INVERT](#) 0x02  
Если установлен - срабатывает по переходу из 1 в 0.
- #define [SYNCIN\\_GOTOPOSITION](#) 0x04  
Если флаг установлен, то двигатель смещается к позиции, установленной в *Position* и *uPosition*, иначе двигатель смещается на *Position* и *uPosition*.

### Флаги настроек синхронизации выхода

Это битовая маска для побитовых операций.

См. также

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

- #define [SYNCOUT\\_ENABLED](#) 0x01  
Синхронизация выхода работает согласно настройкам, если флаг установлен.
- #define [SYNCOUT\\_STATE](#) 0x02  
Когда значение выхода управляется напрямую (см.
- #define [SYNCOUT\\_INVERT](#) 0x04  
Нулевой логический уровень является активным, если флаг установлен, а единичный - если флаг сброшен.
- #define [SYNCOUT\\_IN\\_STEPS](#) 0x08  
Если флаг установлен использовать шаги/импульсы энкодера для выходных импульсов синхронизации вместо миллисекунд.
- #define [SYNCOUT\\_ONSTART](#) 0x10  
Генерация синхронизирующего импульса при начале движения.
- #define [SYNCOUT\\_ONSTOP](#) 0x20  
Генерация синхронизирующего импульса при остановке.
- #define [SYNCOUT\\_ONPERIOD](#) 0x40  
Выдает импульс синхронизации после прохождения *SyncOutPeriod* отсчётов.

### Флаги настройки работы внешнего ввода/вывода

Это битовая маска для побитовых операций.

См. также

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

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

### Флаги настройки режимов внешнего ввода/вывода

Это битовая маска для побитовых операций.

См. также

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

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

### Флаги границ

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

См. также

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

- #define **BORDER\_IS\_ENCODER** 0x01  
*Если флаг установлен, границы определяются предустановленными точками на шкале позиции.*
- #define **BORDER\_STOP\_LEFT** 0x02  
*Если флаг установлен, мотор останавливается при достижении левой границы.*
- #define **BORDER\_STOP\_RIGHT** 0x04  
*Если флаг установлен, мотор останавливается при достижении правой границы.*
- #define **BORDERS\_SWAP\_MISSET\_DETECTION** 0x08  
*Если флаг установлен, мотор останавливается при достижении обеих границ.*

#### Флаги концевых выключателей

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

См. также

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

- #define **ENDER\_SWAP** 0x01  
*Если флаг установлен, первый концевой выключатель находится справа; иначе - слева.*
- #define **ENDER\_SW1\_ACTIVE\_LOW** 0x02  
*1 - 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

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

См. также

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

- #define **BRAKE\_ENABLED** 0x01  
*Управление тормозом включено, если флаг установлен.*
- #define **BRAKE\_ENG\_PWROFF** 0x02  
*Тормоз отключает питание шагового мотора, если флаг установлен.*

#### Флаги управления

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

См. также

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

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

### Флаги джойстика

Это битовая маска для побитовых операций. Управляют состояниями джойстика.

См. также

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

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

### Флаги контроля позиции

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

См. также

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

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

### Флаги настроек команды home

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

См. также

[`get\_home\_settings`](#)  
[`set\_home\_settings`](#)  
[`command\_home`](#)  
[`home\_settings\_t::HomeFlags, get\_home\_settings, set\_home\_settings`](#)

- `#define HOME_DIR_FIRST 0x001`  
*Определяет направление первоначального движения мотора после поступления команды HOME.*
- `#define HOME_DIR_SECOND 0x002`  
*Определяет направление второго движения мотора.*
- `#define HOME_MV_SEC_EN 0x004`  
*Если флаг установлен, реализуется второй этап доводки в домашнюю позицию; иначе - этап пропускается.*
- `#define HOME_HALF_MV 0x008`  
*Если флаг установлен, в начале второго движения первые пол оборота сигналы завершения движения игнорируются.*
- `#define HOME_STOP_FIRST_BITS 0x030`  
*Биты, отвечающие за выбор сигнала завершения первого движения.*
- `#define HOME_STOP_FIRST_REV 0x010`  
*Первое движение завершается по сигналу с Revolution sensor.*
- `#define HOME_STOP_FIRST_SYN 0x020`  
*Первое движение завершается по сигналу со входа синхронизации.*
- `#define HOME_STOP_FIRST_LIM 0x030`  
*Первое движение завершается по сигналу с 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`

Это битовая маска для побитовых операций.

См. также

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

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

### Флаги типа двигателя

Это битовая маска для побитовых операций.

См. также

`motor_settings_t::MotorType, get_motor_settings, set_motor_settings`

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

### Флаги настроек энкодера

Это битовая маска для побитовых операций.

См. также

`accessories_settings_t::MBSettings, get_accessories_settings, set_accessories_settings`

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

### Флаги настроек температурного датчика

Это битовая маска для побитовых операций.

См. также

`accessories_settings_t::LimitSwitchesSettings, get_accessories_settings, set_accessories_settings`

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



- Если флаг установлен, то 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 замкнуты.

### Флаги автоопределения характеристик обмоток двигателя.

Это битовая маска для побитовых операций.

См. также

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

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

### Определения типов

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

### Функции

#### Группа команд настройки контроллера

Функции для чтения/записи большинства настроек контроллера.

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

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

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

- Запись настроек джойстика.*

  - `result_t XIMC_API get_joystick_settings (device_t id, joystick_settings_t *joystick_settings)`
- Чтение настроек джойстика.*

  - `result_t XIMC_API set_ctp_settings (device_t id, const ctp_settings_t *ctp_settings)`
- Запись настроек контроля позиции(для шагового двигателя).*

  - `result_t XIMC_API get_ctp_settings (device_t id, ctp_settings_t *ctp_settings)`
- Чтение настроек контроля позиции(для шагового двигателя).*

  - `result_t XIMC_API set_uart_settings (device_t id, const uart_settings_t *uart_settings)`
- Команда записи настроек UART.*

  - `result_t XIMC_API get_uart_settings (device_t id, uart_settings_t *uart_settings)`
- Команда чтения настроек UART.*

  - `result_t XIMC_API set_calibration_settings (device_t id, const calibration_settings_t *calibration_settings)`
- Команда записи калибровочных коэффициентов.*

  - `result_t XIMC_API get_calibration_settings (device_t id, calibration_settings_t *calibration_settings)`
- Команда чтения калибровочных коэффициентов.*

  - `result_t XIMC_API set_controller_name (device_t id, const controller_name_t *controller_name)`
- Запись пользовательского имени контроллера и настроек в FRAM.*

  - `result_t XIMC_API get_controller_name (device_t id, controller_name_t *controller_name)`
- Чтение пользовательского имени контроллера и настроек из FRAM.*

  - `result_t XIMC_API set_nonvolatile_memory (device_t id, const nonvolatile_memory_t *nonvolatile_memory)`
- Запись пользовательских данных во FRAM.*

  - `result_t XIMC_API get_nonvolatile_memory (device_t id, nonvolatile_memory_t *nonvolatile_memory)`
- Чтение пользовательских данных из FRAM.*

  - `result_t XIMC_API set_emf_settings (device_t id, const emf_settings_t *emf_settings)`
- Запись электромеханических настроек шагового двигателя.*

  - `result_t XIMC_API get_emf_settings (device_t id, emf_settings_t *emf_settings)`
- Чтение электромеханических настроек шагового двигателя.*

  - `result_t XIMC_API set_engine_advansed_setup (device_t id, const engine_advansed_setup_t *engine_advansed_setup)`
- Запись расширенных настроек.*

  - `result_t XIMC_API get_engine_advansed_setup (device_t id, engine_advansed_setup_t *engine_advansed_setup)`
- Чтение расширенных настроек.*

  - `result_t XIMC_API set_extended_settings (device_t id, const extended_settings_t *extended_settings)`
- Запись расширенных настроек.*

  - `result_t XIMC_API get_extended_settings (device_t id, extended_settings_t *extended_settings)`
- Чтение расширенных настроек.*

### Группа команд управления движением

- `result_t XIMC_API command_stop (device_t id)`

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

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

*При получении команды "move" двигатель начинает перемещаться (если не используется режим "ТТЛ СинхроВхода"), 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 set\_correction\_table (device\_t id, const char \*namefile)  
*Команда загрузки корректирующей таблицы из текстового файла.*
- result\_t XIMC\_API probe\_device (const char \*uri)  
*Проверяет, является ли устройство с уникальным идентификатором uri XIMC-совместимым.*
- result\_t XIMC\_API set\_bindy\_key (const char \*keyfilepath)  
*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`

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

7.1.2.12 `#define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08`

Нажатая правая кнопка соответствует открытому контакту, если этот 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 влево/вправо.

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`

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

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-ключах.  
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 должным образом.

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 (bldc).

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`

Бесщеточный 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`

Продолжительность включения 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`

Определяет 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 замкнуты.

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 `right`.

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

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, инвертирование делает активным уровень 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 rpm.

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`

Выдает 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 A 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

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

### 7.1.4 Функции

7.1.4.1 `result_t XIMC_API close_device ( device_t * id )`

Закрывает устройство

Аргументы

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

Заметки

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

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

Очистка FRAM памяти контроллера.

Функция используется только производителем.

Аргументы

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

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

Чтение настроек контроллера из EEPROM памяти позиционера, 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

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

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

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

Функция должна использоваться только производителем.

Аргументы

<i>id</i>	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

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

См. также

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

#### 7.1.4.6 **result\_t XIMC\_API** command\_homezero ( **device\_t** id )

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

Это удобный путь для калибровки нулевой позиции.

Аргументы

	<i>id</i>	идентификатор устройства
<i>out</i>	<i>ret</i>	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

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

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

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

Аргументы

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

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

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

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

Аргументы

<i>id</i>	идентификатор устройства
<i>Position</i>	заданная позиция.
<i>uPosition</i>	часть позиции в микрошагах. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле 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 "ТТЛ-СинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), к точке указанной в поле Position.

Аргументы

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

Заметки

Параметр Position корректируется таблицей коррекции.

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

Перемещение на заданное смещение.

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

Аргументы

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

7.1.4.12 **result\_t** XIMC\_API command\_movr\_calb ( **device\_t** id, float DeltaPosition, const **calibration\_t** \* calibration )

Перемещение на заданное смещение с использованием пользовательских единиц.

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

Аргументы

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

## Заметки

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

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

См. также

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

7.1.4.14 `result_t XIMC_API command_read_robust_settings ( device_t id )`

Чтение важных настроек (калибровочные коэффициенты и т.

п.) контроллера из flash памяти в оперативную, заменяя текущие настройки.

## Аргументы

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

7.1.4.15 `result_t XIMC_API command_read_settings ( device_t id )`

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

## Аргументы

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

7.1.4.16 `result_t XIMC_API command_reset ( device_t id )`

Перезагрузка контроллера.

Функция используется только производителем.

## Аргументы

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

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

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

Аргументы

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

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

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

п.) во встроенную энергонезависимую память контроллера.

Аргументы

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

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

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

Аргументы

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

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

Плавная остановка.

Двигатель останавливается с ускорением замедления.

Аргументы

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

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

Начать измерения и буферизацию скорости, ошибки следования.

Аргументы

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

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

Немедленная остановка двигателя, переход в состояние STOP, ключи в режиме BREAK (обмотки накоротко замкнуты), режим "удержания" деактивируется для DC двигателей, удержание тока в

обмотках для шаговых двигателей (с учётом Power management настроек).

Аргументы

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

7.1.4.23 **result\_t XIMC\_API** `command_update_firmware ( const char * uri, const uint8_t * data, uint32_t data_size )`

Обновление прошивки

Аргументы

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

7.1.4.24 **result\_t XIMC\_API** `command_wait_for_stop ( device_t id, uint32_t refresh_interval_ms )`

Ожидание остановки контроллера

Аргументы

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

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 `movg` 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

<i>id</i>	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	<i>enumerate_flags</i>	флаги поиска устройств
in	<i>hints</i>	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 <a href="#">set_bindy_key</a> .

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

Освобождает память, выделенную *enumerate\_devices*.

Аргументы

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

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

Чтение информации о дополнительных аксессуарах из EEPROM.

Аргументы

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

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

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

Эта функция используется для тестирования и калибровки устройства.



Аргументы

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

7.1.4.30 **result\_t XIMC\_API** `get_bootloader_version ( device_t id, unsigned int * Major, unsigned int * Minor, unsigned int * Release )`

Чтение номера версии прошивки контроллера.

Аргументы

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

7.1.4.31 **result\_t XIMC\_API** `get_brake_settings ( device_t id, brake_settings_t * brake_settings )`

Чтение настроек управления тормозом.

Аргументы

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

7.1.4.32 **result\_t XIMC\_API** `get_calibration_settings ( device_t id, calibration_settings_t * calibration_settings )`

Команда чтения калибровочных коэффициентов.

Эта функция заполняет структуру калибровочных коэффициентов.

См. также

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

Аргументы

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

7.1.4.33 **result\_t XIMC\_API** `get_chart_data ( device_t id, chart_data_t * chart_data )`

Команда чтения состояния обмоток и других не часто используемых данных.

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

См. также

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

Аргументы

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

7.1.4.34 **result\_t XIMC\_API** `get_control_settings ( device_t id, control_settings_t * control_settings )`

Чтение настроек управления мотором.

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

Аргументы

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

7.1.4.35 **result\_t XIMC\_API** `get_control_settings_calb ( device_t id, control_settings_calb_t * control_settings_calb, const calibration_t * calibration )`

Чтение настроек управления мотором с использованием пользовательских единиц.

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

Аргументы

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

7.1.4.36 **result\_t XIMC\_API** `get_controller_name ( device_t id, controller_name_t * controller_name )`

Чтение пользовательского имени контроллера и настроек из FRAM.

## Аргументы

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

7.1.4.37 **result\_t** **XIMC\_API** `get_ctp_settings ( device_t id, ctp_settings_t * ctp_settings )`

Чтение настроек контроля позиции(для шагового двигателя).

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

## Аргументы

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

7.1.4.38 **result\_t** **XIMC\_API** `get_debug_read ( device_t id, debug_read_t * debug_read )`

Чтение данных из прошивки для отладки и поиска неисправностей.

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

## Аргументы

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

7.1.4.39 **int** **XIMC\_API** `get_device_count ( device_enumeration_t device_enumeration )`

Возвращает количество подключенных устройств.

## Аргументы

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

7.1.4.40 **result\_t** **XIMC\_API** `get_device_information ( device_t id, device_information_t *  
device_information )`

Возвращает информацию об устройстве.

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

Аргументы

	<i>id</i>	идентификатор устройства.
out	<i>device_ - information</i>	информация об устройстве Информация об устройстве.

См. также

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

7.1.4.41 **pchar XIMC\_API** `get_device_name ( device_enumeration_t device_enumeration, int device_index )`

Возвращает имя подключенного устройства из перечисления устройств.

Возвращает имя устройства с номером *device\_index*.

Аргументы

in	<i>device_ - enumeration</i>	закрытый указатель на данные о перечисленных устройствах
in	<i>device_index</i>	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

	<i>id</i>	идентификатор устройства
out	<i>edges_settings</i>	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

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

## Заметки

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

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

Чтение электромеханических настроек шагового двигателя.

Настройки различны для разных двигателей.

См. также

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

## Аргументы

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

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

Чтение информации об энкодере из EEPROM.

## Аргументы

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

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

Чтение настроек энкодера из EEPROM.

## Аргументы

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

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

Чтение расширенных настроек.

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>engine_ - advanced_ - setup</i>	настройки EAS

7.1.4.48 **result\_t XIMC\_API** `get_engine_settings ( device_t id, engine_settings_t * engine_settings )`

Чтение настроек мотора.

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

См. также

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

Аргументы

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

7.1.4.49 **result\_t XIMC\_API** `get_engine_settings_calb ( device_t id, engine_settings_calb_t * engine_settings_calb, const calibration_t * calibration )`

Чтение настроек мотора с использованием пользовательских единиц.

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

См. также

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

Аргументы

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

7.1.4.50 **result\_t XIMC\_API** `get_entype_settings ( device_t id, entype_settings_t * entype_settings )`

Возвращает информацию о типе мотора и силового драйвера.

Аргументы

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

7.1.4.51 **result\_t XIMC\_API** get\_enumerate\_device\_controller\_name ( **device\_enumeration\_t** device\_enumeration, int device\_index, **controller\_name\_t** \* controller\_name )

Возвращает имя подключенного устройства из перечисления устройств.

Возвращает имя устройства с номером *device\_index*.

Аргументы

in	<i>device_ - enumeration</i>	закрытый указатель на данные о перечисленных устройствах
in	<i>device_index</i>	номер устройства
out	<i>controller</i>	наименование имени 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	<i>device_ - enumeration</i>	закрытый указатель на данные о перечисленных устройствах
in	<i>device_index</i>	номер устройства
out	<i>device_ - information</i>	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	<i>device_ - enumeration</i>	закрытый указатель на данные о перечисленных устройствах
in	<i>device_index</i>	номер устройства
out	<i>device_ - network_ - information</i>	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	<i>device_enumeration</i>	закрытый указатель на данные о перечисленных устройствах
in	<i>device_index</i>	номер устройства
in	<i>serial</i>	серийный номер устройства

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

Возвращает имя подвижки для подключенного устройства из перечисления устройств.

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

Аргументы

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

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

Чтение расширенных настроек.

См. также

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

Аргументы

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

7.1.4.57 **result\_t XIMC\_API** get\_extio\_settings ( **device\_t** id, **extio\_settings\_t** \* extio\_settings )

Команда чтения параметров настройки режимов внешнего ввода/вывода.

См. также

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



Аргументы

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

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

Чтение настроек обратной связи

Аргументы

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

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

Чтение номера версии прошивки контроллера.

Аргументы

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

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

Чтение информации о редукторе из EEPROM.

Аргументы

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

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

Чтение настроек редуктора из EEPROM.

## Аргументы

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

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

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

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

## Аргументы

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

7.1.4.63 **result\_t XIMC\_API** get\_hallsensor\_information ( **device\_t** id, **hallsensor\_information\_t** \* hallsensor\_information )

Чтение информации о датчиках Холла из EEPROM.

## Аргументы

	<i>id</i>	идентификатор устройства
out	<i>hallsensor_information</i>	структура, содержащая информацию о датчиках Холла

7.1.4.64 **result\_t XIMC\_API** get\_hallsensor\_settings ( **device\_t** id, **hallsensor\_settings\_t** \* hallsensor\_settings )

Чтение настроек датчиков Холла из EEPROM.

## Аргументы

	<i>id</i>	идентификатор устройства
out	<i>hallsensor_settings</i>	структура, содержащая настройки датчиков Холла

7.1.4.65 **result\_t XIMC\_API** get\_home\_settings ( **device\_t** id, **home\_settings\_t** \* home\_settings )

Команда чтения настроек для подхода в home position.

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

См. также

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

Аргументы

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

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

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

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

См. также

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

Аргументы

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

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

Чтение случайного числа из контроллера.

Аргументы

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

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

Чтение настроек джойстика.

При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость, а 100% отклонения соответствует Max-Speed *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. Нуле-

вой параметр нелинейности соответствует линейной зависимости.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>joystick_ - settings</i>	структура, содержащая настройки джойстика

7.1.4.69 **result\_t XIMC\_API** get\_measurements ( **device\_t** id, **measurements\_t** \* measurements )

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

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

См. также

[measurements\\_t](#)

Аргументы

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

7.1.4.70 **result\_t XIMC\_API** get\_motor\_information ( **device\_t** id, **motor\_information\_t** \* motor\_information )

Чтение информации о двигателе из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>motor_ - information</i>	структура, содержащая информацию о двигателе

7.1.4.71 **result\_t XIMC\_API** get\_motor\_settings ( **device\_t** id, **motor\_settings\_t** \* motor\_settings )

Чтение настроек двигателя из EEPROM.

Аргументы

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

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

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

Аргументы

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

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

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

Аргументы

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

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

Чтение пользовательских данных из FRAM.

Аргументы

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

7.1.4.75 `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

	<i>id</i>	идентификатор устройства
out	<i>pid_settings</i>	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

	<i>id</i>	идентификатор устройства
out	<i>position</i>	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

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

	<i>id</i>	идентификатор устройства
out	<i>power_settings</i>	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

	<i>id</i>	идентификатор устройства
out	<i>secure_settings</i>	настройки, определяющие максимально допустимые параметры, для защиты оборудования

См. также

status\_t::flags

7.1.4.80 **result\_t XIMC\_API** get\_serial\_number ( **device\_t** id, unsigned int \* SerialNumber )

Чтение серийного номера контроллера.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>SerialNumber</i>	серийный номер контроллера

7.1.4.81 **result\_t XIMC\_API** get\_stage\_information ( **device\_t** id, **stage\_information\_t** \* stage\_information )

Чтение информации о позиционере из EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>stage_information</i>	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

	<i>id</i>	идентификатор устройства
out	<i>stage_name</i>	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

	<i>id</i>	идентификатор устройства
out	<i>stage_settings</i>	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

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

См. также

[get\\_status](#)

7.1.4.85 **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

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

См. также

[get\\_status](#)

7.1.4.86 **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

	<i>id</i>	идентификатор устройства
<i>out</i>	<i>sync_in_settings</i>	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

	<i>id</i>	идентификатор устройства
out	<i>sync_in_settings_calb</i>	настройки синхронизации
	<i>calibration</i>	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

	<i>id</i>	идентификатор устройства
in	<i>sync_out_settings_calb</i>	настройки синхронизации
	<i>calibration</i>	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

	<i>Speed</i>	Скорость UART
out	<i>uart_settings</i>	настройки UART

7.1.4.91 **result\_t XIMC\_API** goto\_firmware ( **device\_t** id, uint8\_t \* ret )

Перезагрузка в прошивку в контроллере

Аргументы

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

7.1.4.92 **result\_t XIMC\_API** has\_firmware ( const char \* uri, uint8\_t \* ret )

Проверка наличия прошивки в контроллере

Аргументы

	<i>uri</i>	уникальный идентификатор ресурса устройства
out	<i>ret</i>	ноль, если прошивка присутствует

7.1.4.93 **result\_t XIMC\_API** load\_correction\_table ( **device\_t** \* id, const char \* namefile )

Команда загрузки корректирующей таблицы из текстового файла.

Данная функция устарела. Данная функция устарела. Используйте функцию [set\\_correction\\_table\(device\\_t id, const char\\* namefile\)](#). Таблица используется для коррекции положения в случае механических неточностей. Работает для некоторых параметров в \_calb командах.

Аргументы

	<i>id</i>	- идентификатор устройства
in	<i>namefile</i>	- имя файла должно быть полным. Если используется короткое имя, файл должен находиться в директории приложения. Если имя файла равно NULL таблица коррекции будет очищена. Формат файла: два столбца разделенных табуляцией. Заголовки столбцов строковые. Данные действительные разделитель точка. Первый столбец координата. Второй - отклонение вызванное ошибкой механики. Между координатами отклонение рассчитывается линейно. За диапазоном константа равная отклонению на границе. Максимальная длина таблицы 100 строк.

## Заметки

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

См. также

```
command_move
command_movr
get_position_calb
get_position_calb_t
get_status_calb
status_calb_t
get_edges_settings_calb
set_edges_settings_calb
edges_settings_calb_t
```

7.1.4.94 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

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

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

7.1.4.96 void **XIMC\_API** msec\_sleep ( unsigned int msec )

Приостанавливает работу на указанное время

Аргументы

<i>msec</i>	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

<code>in</code>	<code>uri</code>	- 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 <code>set_bindy_key</code> . 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

<code>in</code>	<code>uri</code>	- уникальный идентификатор устройства
-----------------	------------------	---------------------------------------

7.1.4.99 `result_t XIMC_API reset_locks ( )`

Снимает блокировку библиотеки в экстренном случае.

Эта функция возвращает только 0 (ОК). Если отправка некоторой команды `libximc` заканчивается неправильной передачей данных (ошибкой), любая последующая команда всегда возвращает -1 (типично для Windows). Вызов этой команды, сбрасывает ошибку.

7.1.4.100 `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.

7.1.4.101 `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

	<code>id</code>	идентификатор устройства
<code>in</code>	<code>accessories_settings</code>	структура, содержащая информацию о дополнительных аксессуарах

7.1.4.102 **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 <a href="#">enumerate_devices</a> и <a href="#">open_device</a> .
----	-------------	--

7.1.4.103 **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

7.1.4.104 **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

7.1.4.105 **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
--	----	--------------------------

<code>in</code>	<code>control_ - settings</code>	структура, содержащая настройки управления мотором с помощью джойстика или кнопок влево/вправо.
-----------------	----------------------------------	---

7.1.4.106 **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

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

7.1.4.107 **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

	<code>id</code>	идентификатор устройства
<code>in</code>	<code>controller_ - information</code>	структура, содержащая информацию о контроллере

7.1.4.108 **result\_t** **XIMC\_API** set\_correction\_table ( **device\_t** id, const char \* namefile )

Команда загрузки корректирующей таблицы из текстового файла.

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

Аргументы

	<code>id</code>	- идентификатор устройства
<code>in</code>	<code>namefile</code>	- 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

```

7.1.4.109 **result\_t** **XIMC\_API** `set_ctp_settings` ( **device\_t** `id`, const **ctp\_settings\_t** \* `ctp_settings` )

Запись настроек контроля позиции(для шагового двигателя).

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

Аргументы

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

7.1.4.110 **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

	<i>id</i>	идентификатор устройства
<code>in</code>	<i>debug_write</i>	Данные для отладки.

7.1.4.111 **result\_t** **XIMC\_API** `set_edges_settings` ( **device\_t** `id`, const **edges\_settings\_t** \* `edges_settings` )

Запись настроек границ и концевых выключателей.

См. также

```

get_edges_settings

```

Аргументы

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

7.1.4.112 **result\_t** **XIMC\_API** set\_edges\_settings\_calb ( **device\_t** id, const **edges\_settings\_calb\_t** \* edges\_settings\_calb, const **calibration\_t** \* calibration )

Запись настроек границ и концевых выключателей с использованием пользовательских единиц.

См. также

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

Аргументы

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

Заметки

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

7.1.4.113 **result\_t** **XIMC\_API** set\_emf\_settings ( **device\_t** id, const **emf\_settings\_t** \* emf\_settings )

Запись электромеханических настроек шагового двигателя.

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

См. также

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

Аргументы

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

7.1.4.114 **result\_t** **XIMC\_API** set\_encoder\_information ( **device\_t** id, const **encoder\_information\_t** \* encoder\_information )

Запись информации об энкодере в EEPROM.

Функция должна использоваться только производителем.



## Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>encoder_ - information</i>	структура, содержащая информацию об энкодере

7.1.4.115 **result\_t XIMC\_API** `set_encoder_settings ( device_t id, const encoder_settings_t * encoder_settings )`

Запись настроек энкодера в EEPROM.

Функция должна использоваться только производителем.

## Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>encoder_ - settings</i>	структура, содержащая настройки энкодера

7.1.4.116 **result\_t XIMC\_API** `set_engine_advanced_setup ( device_t id, const engine_advanced_setup_t * engine_advanced_setup )`

Запись расширенных настроек.

См. также

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

## Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>engine_ - advanced_ - setup</i>	настройки EAS

7.1.4.117 **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

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

7.1.4.118 **result\_t XIMC\_API** `set_engine_settings_calb ( device_t id, const engine_settings_calb_t * engine_settings_calb, const calibration_t * calibration )`

Запись настроек мотора с использованием пользовательских единиц.

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

См. также

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

Аргументы

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

7.1.4.119 **result\_t XIMC\_API** `set_entype_settings ( device_t id, const entype_settings_t * entype_settings )`

Запись информации о типе мотора и типе силового драйвера.

Аргументы

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

7.1.4.120 **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

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

7.1.4.121 **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 еди-

ничное состояние считается активным выходом.

См. также

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

Аргументы

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

7.1.4.122 **result\_t** XIMC\_API set\_feedback\_settings ( **device\_t** id, const **feedback\_settings\_t** \* feedback\_settings )

Запись настроек обратной связи.

Аргументы

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

7.1.4.123 **result\_t** XIMC\_API set\_gear\_information ( **device\_t** id, const **gear\_information\_t** \* gear\_information )

Запись информации о редукторе в EEPROM.

Функция должна использоваться только производителем.

Аргументы

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

7.1.4.124 **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

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

7.1.4.125 **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

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

7.1.4.126 **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

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

7.1.4.127 **result\_t XIMC\_API** set\_home\_settings ( **device\_t** id, const **home\_settings\_t** \* home\_settings )

Команда записи настроек для подхода в home position.

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

См. также

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

Аргументы

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

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

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

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

См. также

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

## Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>home_settings-_calb</i>	настройки калибровки позиции
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.129 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>joystick_-_settings</i>	структура, содержащая настройки джойстика

7.1.4.130 **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

<i>logging_-_callback</i>	указатель на функцию обратного вызова
---------------------------	---------------------------------------

7.1.4.131 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>motor_-_information</i>	структура, содержащая информацию о двигателе

7.1.4.132 **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

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

7.1.4.133 **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

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

7.1.4.134 **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

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

7.1.4.135 **result\_t** **XIMC\_API** set\_nonvolatile\_memory ( **device\_t** id, const **nonvolatile\_memory\_t** \* nonvolatile\_memory )

Запись пользовательских данных во FRAM.

Аргументы

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

7.1.4.136 **result\_t** **XIMC\_API** set\_pid\_settings ( **device\_t** id, const **pid\_settings\_t** \* pid\_settings )

Запись ПИД коэффициентов.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>pid_settings</i>	настройки ПИД

7.1.4.137 **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

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

7.1.4.138 **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

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

7.1.4.139 **result\_t** XIMC\_API set\_power\_settings ( **device\_t** id, const **power\_settings\_t** \* power\_settings )

Команда записи параметров питания мотора.

Используется только с шаговым двигателем.

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>power_settings</i>	структура, содержащая настройки питания шагового мотора

7.1.4.140 **result\_t XIMC\_API** set\_secure\_settings ( **device\_t** id, const **secure\_settings\_t** \* secure\_settings )

Команда записи установок защит.

Аргументы

<i>id</i>	идентификатор устройства
<i>secure_settings</i>	структура с настройками критических значений

См. также

status\_t::flags

7.1.4.141 **result\_t XIMC\_API** set\_serial\_number ( **device\_t** id, const **serial\_number\_t** \* serial\_number )

Запись серийного номера и версии железа во flash память контроллера.

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>serial_number</i>	структура, содержащая серийный номер, версию железа и ключ.

7.1.4.142 **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

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>stage_information</i>	структура, содержащая информацию о позиционере

7.1.4.143 **result\_t XIMC\_API** set\_stage\_name ( **device\_t** id, const **stage\_name\_t** \* stage\_name )

Запись пользовательского имени подвижки в EEPROM.

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>stage_name</i>	структура, содержащая установленное пользовательское имя позиционера



7.1.4.144 **result\_t XIMC\_API** `set_stage_settings ( device_t id, const stage_settings_t * stage_settings )`

Запись настроек позиционера в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>stage_settings</i>	структура, содержащая настройки позиционера

7.1.4.145 **result\_t XIMC\_API** `set_sync_in_settings ( device_t id, const sync_in_settings_t * sync_in_settings )`

Запись настроек для входного импульса синхронизации.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>sync_in_settings</i>	настройки синхронизации

7.1.4.146 **result\_t XIMC\_API** `set_sync_in_settings_calb ( device_t id, const sync_in_settings_calb_t * sync_in_settings_calb, const calibration_t * calibration )`

Запись настроек для входного импульса синхронизации с использованием пользовательских единиц.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>sync_in_settings_calb</i>	настройки синхронизации
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.147 **result\_t XIMC\_API** `set_sync_out_settings ( device_t id, const sync_out_settings_t * sync_out_settings )`

Запись настроек для выходного импульса синхронизации.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>sync_out_settings</i>	настройки синхронизации

7.1.4.148 **result\_t** XIMC\_API set\_sync\_out\_settings\_calb ( **device\_t** id, const **sync\_out\_settings\_calb\_t** \* sync\_out\_settings\_calb, const **calibration\_t** \* calibration )

Запись настроек для выходного импульса синхронизации с использованием пользовательских единиц.

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

См. также

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

Аргументы

	<i>id</i>	идентификатор устройства
<i>in</i>	<i>sync_out_settings_calb</i>	настройки синхронизации
	<i>calibration</i>	настройки пользовательских единиц

7.1.4.149 **result\_t** XIMC\_API set\_uart\_settings ( **device\_t** id, const **uart\_settings\_t** \* uart\_settings )

Команда записи настроек UART.

Эта функция записывает структуру настроек UART в память контроллера.

См. также

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

Аргументы

	<i>Speed</i>	Скорость UART
<i>in</i>	<i>uart_settings</i>	настройки UART

7.1.4.150 **result\_t** XIMC\_API write\_key ( const char \* uri, uint8\_t \* key )

Запись ключа защиты. Функция используется только производителем.

## Аргументы

	<i>uri</i>	идентификатор устройства
<i>in</i>	<i>key</i>	ключ защиты. Диапазон: 0..4294967295

7.1.4.151 **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.

7.1.4.152 **void XIMC\_API** `ximc_version ( char * version )`

Возвращает версию библиотеки

## Аргументы

<i>version</i>	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, 18
- ACurrent\_ADC
  - analog\_data\_t, 18
- Accel
  - move\_settings\_calb\_t, 59
  - move\_settings\_t, 60
- accessories\_settings\_t, 14
  - LimitSwitchesSettings, 15
  - MBRatedCurrent, 15
  - MBRatedVoltage, 15
  - MBSettings, 15
  - MBTorque, 15
  - MagneticBrakeInfo, 15
  - TSGrad, 15
  - TSMaх, 15
  - TSMin, 15
  - TSSettings, 16
  - TemperatureSensorInfo, 15
- Accuracy
  - sync\_out\_settings\_calb\_t, 78
  - sync\_out\_settings\_t, 79
- analog\_data\_t, 16
  - A1Voltage, 17
  - A1Voltage\_ADC, 17
  - A2Voltage, 17
  - A2Voltage\_ADC, 17
  - ACurrent, 18
  - ACurrent\_ADC, 18
  - B1Voltage, 18
  - B1Voltage\_ADC, 18
  - B2Voltage, 18
  - B2Voltage\_ADC, 18
  - BCurrent, 18
  - BCurrent\_ADC, 18
  - FullCurrent, 18
  - FullCurrent\_ADC, 18
  - H5, 18
  - Joy, 18
  - Joy\_ADC, 19
  - L5, 19
  - L5\_ADC, 19
  - Pot, 19
  - SupVoltage, 19
  - SupVoltage\_ADC, 19
  - Temp, 19
  - Temp\_ADC, 19
- Antiplay
  - engine\_settings\_calb\_t, 37
  - engine\_settings\_t, 39
- AntiplaySpeed
  - move\_settings\_calb\_t, 59
  - move\_settings\_t, 60
- B1Voltage
  - analog\_data\_t, 18
- B1Voltage\_ADC
  - analog\_data\_t, 18
- B2Voltage
  - analog\_data\_t, 18
- B2Voltage\_ADC
  - analog\_data\_t, 18
- BACK\_EMF\_KM\_AUTO
  - ximc.h, 106
- BCurrent
  - analog\_data\_t, 18
- BCurrent\_ADC
  - analog\_data\_t, 18
- BORDER\_IS\_ENCODER
  - ximc.h, 106
- BORDER\_STOP\_LEFT
  - ximc.h, 107
- BORDER\_STOP\_RIGHT
  - ximc.h, 107
- BRAKE\_ENABLED
  - ximc.h, 107
- BRAKE\_ENG\_PWROFF
  - ximc.h, 107
- BackEMFFlags
  - emf\_settings\_t, 33
- BorderFlags
  - edges\_settings\_calb\_t, 31
  - edges\_settings\_t, 32
- brake\_settings\_t, 19
  - BrakeFlags, 20
  - t1, 20

- t2, [20](#)
- t3, [20](#)
- t4, [20](#)
- BrakeFlags
  - brake\_settings\_t, [20](#)
- CONTROL\_MODE\_BITS
  - ximc.h, [107](#)
- CONTROL\_MODE\_JOY
  - ximc.h, [107](#)
- CONTROL\_MODE\_LR
  - ximc.h, [107](#)
- CONTROL\_MODE\_OFF
  - ximc.h, [107](#)
- 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, [108](#)
- CTP\_BASE
  - ximc.h, [108](#)
- CTP\_ENABLED
  - ximc.h, [108](#)
- CTP\_ERROR\_CORRECTION
  - ximc.h, [108](#)
- CTPFlags
  - ctp\_settings\_t, [28](#)
- CTPMinError
  - ctp\_settings\_t, [28](#)
- calibration\_settings\_t, [20](#)
  - CSS1\_A, [21](#)
  - CSS1\_B, [21](#)
  - CSS2\_A, [21](#)
  - CSS2\_B, [21](#)
  - FullCurrent\_A, [21](#)
  - FullCurrent\_B, [21](#)
- calibration\_t, [22](#)
- chart\_data\_t, [22](#)
  - DutyCycle, [23](#)
  - Joy, [23](#)
  - Pot, [23](#)
  - WindingCurrentA, [23](#)
  - WindingCurrentB, [23](#)
  - WindingCurrentC, [23](#)
  - WindingVoltageA, [23](#)
  - WindingVoltageB, [23](#)
  - WindingVoltageC, [23](#)
- close\_device
  - ximc.h, [122](#)
- ClutterTime
  - sync\_in\_settings\_calb\_t, [76](#)
- sync\_in\_settings\_t, [77](#)
- CmdBufFreeSpace
  - status\_calb\_t, [71](#)
  - status\_t, [73](#)
- command\_clear\_fram
  - ximc.h, [122](#)
- command\_eeread\_settings
  - ximc.h, [123](#)
- command\_eesave\_settings
  - ximc.h, [123](#)
- command\_home
  - ximc.h, [123](#)
- command\_homezero
  - ximc.h, [124](#)
- command\_left
  - ximc.h, [124](#)
- command\_loft
  - ximc.h, [124](#)
- command\_move
  - ximc.h, [124](#)
- command\_move\_calb
  - ximc.h, [125](#)
- command\_movr
  - ximc.h, [125](#)
- command\_movr\_calb
  - ximc.h, [125](#)
- command\_power\_off
  - ximc.h, [126](#)
- command\_read\_robust\_settings
  - ximc.h, [126](#)
- command\_read\_settings
  - ximc.h, [126](#)
- command\_reset
  - ximc.h, [126](#)
- command\_right
  - ximc.h, [126](#)
- command\_save\_robust\_settings
  - ximc.h, [127](#)
- command\_save\_settings
  - ximc.h, [127](#)
- command\_sstp
  - ximc.h, [127](#)
- command\_start\_measurements
  - ximc.h, [127](#)
- command\_stop
  - ximc.h, [127](#)
- command\_update\_firmware
  - ximc.h, [128](#)
- command\_wait\_for\_stop
  - ximc.h, [128](#)
- command\_zero
  - ximc.h, [128](#)
- control\_settings\_calb\_t, [24](#)
  - Flags, [24](#)
  - MaxClickTime, [24](#)
  - MaxSpeed, [24](#)

- Timeout, [25](#)
- control\_settings\_t, [25](#)
  - Flags, [26](#)
  - MaxClickTime, [26](#)
  - MaxSpeed, [26](#)
  - Timeout, [26](#)
  - uDeltaPosition, [26](#)
  - uMaxSpeed, [26](#)
- controller\_name\_t, [26](#)
  - ControllerName, [27](#)
  - CtrlFlags, [27](#)
- ControllerName
  - controller\_name\_t, [27](#)
- CountsPerTurn
  - feedback\_settings\_t, [42](#)
- CriticalIpwr
  - secure\_settings\_t, [64](#)
- CriticalIusb
  - secure\_settings\_t, [64](#)
- CriticalUpwr
  - secure\_settings\_t, [64](#)
- CriticalUusb
  - secure\_settings\_t, [64](#)
- ctp\_settings\_t, [27](#)
  - CTPFlags, [28](#)
  - CTPMinError, [28](#)
- CtrlFlags
  - controller\_name\_t, [27](#)
- CurPosition
  - status\_calb\_t, [71](#)
  - status\_t, [73](#)
- CurSpeed
  - status\_calb\_t, [71](#)
  - status\_t, [74](#)
- CurT
  - status\_calb\_t, [71](#)
  - status\_t, [74](#)
- CurrReductDelay
  - power\_settings\_t, [62](#)
- CurrentSetTime
  - power\_settings\_t, [62](#)
- DRIVER\_TYPE\_EXTERNAL
  - ximc.h, [108](#)
- DeadZone
  - joystick\_settings\_t, [52](#)
- debug\_read\_t, [28](#)
  - DebugData, [28](#)
- debug\_write\_t, [28](#)
  - DebugData, [29](#)
- DebugData
  - debug\_read\_t, [28](#)
  - debug\_write\_t, [29](#)
- Decel
  - move\_settings\_calb\_t, [59](#)
  - move\_settings\_t, [60](#)
- DetentTorque
  - motor\_settings\_t, [56](#)
- device\_information\_t, [29](#)
  - Major, [29](#)
  - Minor, [29](#)
  - Release, [30](#)
- device\_network\_information\_t, [30](#)
- DriverType
  - entype\_settings\_t, [40](#)
- DutyCycle
  - chart\_data\_t, [23](#)
- EEPROM\_PRECEDENCE
  - ximc.h, [108](#)
- ENC\_STATE\_ABSENT
  - ximc.h, [108](#)
- ENC\_STATE\_MALFUNC
  - ximc.h, [108](#)
- ENC\_STATE\_OK
  - ximc.h, [108](#)
- ENC\_STATE\_REVERS
  - ximc.h, [109](#)
- ENC\_STATE\_UNKNOWN
  - ximc.h, [109](#)
- ENDER\_SW1\_ACTIVE\_LOW
  - ximc.h, [109](#)
- ENDER\_SW2\_ACTIVE\_LOW
  - ximc.h, [109](#)
- ENDER\_SWAP
  - ximc.h, [109](#)
- ENGINE\_ACCEL\_ON
  - ximc.h, [109](#)
- ENGINE\_ANTIPLAY
  - ximc.h, [109](#)
- ENGINE\_LIMIT\_CURR
  - ximc.h, [109](#)
- ENGINE\_LIMIT\_RPM
  - ximc.h, [110](#)
- ENGINE\_LIMIT\_VOLT
  - ximc.h, [110](#)
- ENGINE\_MAX\_SPEED
  - ximc.h, [110](#)
- ENGINE\_REVERSE
  - ximc.h, [110](#)
- ENGINE\_TYPE\_2DC
  - ximc.h, [110](#)
- ENGINE\_TYPE\_DC
  - ximc.h, [110](#)
- ENGINE\_TYPE\_NONE
  - ximc.h, [110](#)
- ENGINE\_TYPE\_STEP
  - ximc.h, [110](#)
- ENGINE\_TYPE\_TEST
  - ximc.h, [110](#)
- ENUMERATE\_PROBE
  - ximc.h, [111](#)

- EXTIO\_SETUP\_INVERT
  - ximc.h, [111](#)
- EXTIO\_SETUP\_OUTPUT
  - ximc.h, [112](#)
- EXTIOModeFlags
  - extio\_settings\_t, [41](#)
- EXTIOSetupFlags
  - extio\_settings\_t, [41](#)
- edges\_settings\_calb\_t, [30](#)
  - BorderFlags, [31](#)
  - EnderFlags, [31](#)
  - LeftBorder, [31](#)
  - RightBorder, [31](#)
- edges\_settings\_t, [31](#)
  - BorderFlags, [32](#)
  - EnderFlags, [32](#)
  - LeftBorder, [32](#)
  - RightBorder, [32](#)
  - uLeftBorder, [32](#)
  - uRightBorder, [32](#)
- Efficiency
  - gear\_settings\_t, [44](#)
- emf\_settings\_t, [32](#)
  - BackEMFFlags, [33](#)
  - Km, [33](#)
  - L, [33](#)
  - R, [33](#)
- EncPosition
  - get\_position\_calb\_t, [45](#)
  - get\_position\_t, [46](#)
  - set\_position\_calb\_t, [66](#)
  - set\_position\_t, [67](#)
  - status\_calb\_t, [71](#)
  - status\_t, [74](#)
- EncSts
  - status\_calb\_t, [71](#)
  - status\_t, [74](#)
- encoder\_information\_t, [33](#)
  - Manufacturer, [34](#)
  - PartNumber, [34](#)
- encoder\_settings\_t, [34](#)
  - EncoderSettings, [35](#)
  - MaxCurrentConsumption, [35](#)
  - MaxOperatingFrequency, [35](#)
  - SupplyVoltageMax, [35](#)
  - SupplyVoltageMin, [35](#)
- EncoderSettings
  - encoder\_settings\_t, [35](#)
- EnderFlags
  - edges\_settings\_calb\_t, [31](#)
  - edges\_settings\_t, [32](#)
- engine\_advanced\_setup\_t, [35](#)
  - stepcloseloop\_Kp\_high, [36](#)
  - stepcloseloop\_Kp\_low, [36](#)
  - stepcloseloop\_Kw, [36](#)
- engine\_settings\_calb\_t, [36](#)
  - Antiplay, [37](#)
  - EngineFlags, [37](#)
  - MicrostepMode, [37](#)
  - NomCurrent, [37](#)
  - NomSpeed, [37](#)
  - NomVoltage, [37](#)
  - StepsPerRev, [38](#)
- engine\_settings\_t, [38](#)
  - Antiplay, [39](#)
  - EngineFlags, [39](#)
  - MicrostepMode, [39](#)
  - NomCurrent, [39](#)
  - NomSpeed, [39](#)
  - NomVoltage, [39](#)
  - StepsPerRev, [39](#)
  - uNomSpeed, [39](#)
- EngineFlags
  - engine\_settings\_calb\_t, [37](#)
  - engine\_settings\_t, [39](#)
- EngineType
  - entype\_settings\_t, [40](#)
- entype\_settings\_t, [40](#)
  - DriverType, [40](#)
  - EngineType, [40](#)
- enumerate\_devices
  - ximc.h, [128](#)
- Error
  - measurements\_t, [53](#)
- ExpFactor
  - joystick\_settings\_t, [52](#)
- extended\_settings\_t, [40](#)
- extio\_settings\_t, [41](#)
  - EXTIOModeFlags, [41](#)
  - EXTIOSetupFlags, [41](#)
- FEEDBACK\_EMF
  - ximc.h, [112](#)
- FEEDBACK\_ENC\_REVERSE
  - ximc.h, [112](#)
- FEEDBACK\_ENCODER
  - ximc.h, [112](#)
- FEEDBACK\_NONE
  - ximc.h, [113](#)
- FastHome
  - home\_settings\_calb\_t, [49](#)
  - home\_settings\_t, [50](#)
- feedback\_settings\_t, [41](#)
  - CountsPerTurn, [42](#)
  - FeedbackFlags, [42](#)
  - FeedbackType, [42](#)
  - IPS, [42](#)
- FeedbackFlags
  - feedback\_settings\_t, [42](#)
- FeedbackType
  - feedback\_settings\_t, [42](#)
- Flags

- control\_settings\_calb\_t, 24
- control\_settings\_t, 26
- secure\_settings\_t, 64
- status\_calb\_t, 71
- status\_t, 74
- free\_enumerate\_devices
  - ximc.h, 129
- 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, 71
  - status\_t, 74
- gear\_information\_t, 42
  - Manufacturer, 43
  - PartNumber, 43
- gear\_settings\_t, 43
  - Efficiency, 44
  - InputInertia, 44
  - MaxOutputBacklash, 44
  - RatedInputSpeed, 44
  - RatedInputTorque, 44
  - ReductionIn, 44
  - ReductionOut, 44
- get\_accessories\_settings
  - ximc.h, 129
- get\_analog\_data
  - ximc.h, 129
- get\_bootloader\_version
  - ximc.h, 130
- get\_brake\_settings
  - ximc.h, 130
- get\_calibration\_settings
  - ximc.h, 130
- get\_chart\_data
  - ximc.h, 130
- get\_control\_settings
  - ximc.h, 131
- get\_control\_settings\_calb
  - ximc.h, 131
- get\_controller\_name
  - ximc.h, 131
- get\_ctp\_settings
  - ximc.h, 132
- get\_debug\_read
  - ximc.h, 132
- get\_device\_count
  - ximc.h, 132
- get\_device\_information
  - ximc.h, 132
- get\_device\_name
  - ximc.h, 133
- get\_edges\_settings
  - ximc.h, 133
- get\_edges\_settings\_calb
  - ximc.h, 133
- get\_emf\_settings
  - ximc.h, 134
- get\_encoder\_information
  - ximc.h, 134
- get\_encoder\_settings
  - ximc.h, 134
- get\_engine\_advansed\_setup
  - ximc.h, 134
- get\_engine\_settings
  - ximc.h, 135
- get\_engine\_settings\_calb
  - ximc.h, 135
- get\_entype\_settings
  - ximc.h, 135
- get\_enumerate\_device\_controller\_name
  - ximc.h, 136
- get\_enumerate\_device\_information
  - ximc.h, 136
- get\_enumerate\_device\_network\_information
  - ximc.h, 136
- get\_enumerate\_device\_serial
  - ximc.h, 136
- get\_enumerate\_device\_stage\_name
  - ximc.h, 137
- get\_extended\_settings
  - ximc.h, 137
- get\_extio\_settings
  - ximc.h, 137
- get\_feedback\_settings
  - ximc.h, 138
- get\_firmware\_version
  - ximc.h, 138
- get\_gear\_information
  - ximc.h, 138
- get\_gear\_settings
  - ximc.h, 138
- get\_globally\_unique\_identifier
  - ximc.h, 139
- get\_hallsensor\_information
  - ximc.h, 139
- get\_hallsensor\_settings
  - ximc.h, 139
- get\_home\_settings
  - ximc.h, 139
- get\_home\_settings\_calb
  - ximc.h, 140
- get\_init\_random
  - ximc.h, 140
- get\_joystick\_settings
  - ximc.h, 140



- get\_measurements
  - ximc.h, [141](#)
- get\_motor\_information
  - ximc.h, [141](#)
- get\_motor\_settings
  - ximc.h, [141](#)
- get\_move\_settings
  - ximc.h, [141](#)
- get\_move\_settings\_calb
  - ximc.h, [142](#)
- get\_nonvolatile\_memory
  - ximc.h, [142](#)
- get\_pid\_settings
  - ximc.h, [142](#)
- get\_position
  - ximc.h, [142](#)
- get\_position\_calb
  - ximc.h, [143](#)
- get\_position\_calb\_t, [44](#)
  - EncPosition, [45](#)
  - Position, [45](#)
- get\_position\_t, [45](#)
  - EncPosition, [46](#)
  - uPosition, [46](#)
- get\_power\_settings
  - ximc.h, [143](#)
- get\_secure\_settings
  - ximc.h, [143](#)
- get\_serial\_number
  - ximc.h, [144](#)
- get\_stage\_information
  - ximc.h, [144](#)
- get\_stage\_name
  - ximc.h, [144](#)
- get\_stage\_settings
  - ximc.h, [144](#)
- get\_status
  - ximc.h, [144](#)
- get\_status\_calb
  - ximc.h, [145](#)
- get\_sync\_in\_settings
  - ximc.h, [145](#)
- get\_sync\_in\_settings\_calb
  - ximc.h, [145](#)
- get\_sync\_out\_settings
  - ximc.h, [146](#)
- get\_sync\_out\_settings\_calb
  - ximc.h, [146](#)
- get\_uart\_settings
  - ximc.h, [146](#)
- globally\_unique\_identifier\_t, [46](#)
  - UniqueID0, [46](#)
  - UniqueID1, [46](#)
  - UniqueID2, [46](#)
  - UniqueID3, [46](#)
- goto\_firmware
  - ximc.h, [147](#)
- H5
  - analog\_data\_t, [18](#)
- HOME\_DIR\_FIRST
  - ximc.h, [113](#)
- HOME\_DIR\_SECOND
  - ximc.h, [113](#)
- HOME\_HALF\_MV
  - ximc.h, [113](#)
- HOME\_MV\_SEC\_EN
  - ximc.h, [113](#)
- HOME\_STOP\_FIRST\_LIM
  - ximc.h, [113](#)
- HOME\_STOP\_FIRST\_REV
  - ximc.h, [113](#)
- HOME\_STOP\_FIRST\_SYN
  - ximc.h, [113](#)
- HOME\_USE\_FAST
  - ximc.h, [114](#)
- hallsensor\_information\_t, [47](#)
  - Manufacturer, [47](#)
  - PartNumber, [47](#)
- hallsensor\_settings\_t, [47](#)
  - MaxCurrentConsumption, [48](#)
  - MaxOperatingFrequency, [48](#)
  - SupplyVoltageMax, [48](#)
  - SupplyVoltageMin, [48](#)
- has\_firmware
  - ximc.h, [147](#)
- HoldCurrent
  - power\_settings\_t, [62](#)
- home\_settings\_calb\_t, [48](#)
  - FastHome, [49](#)
  - HomeDelta, [49](#)
  - HomeFlags, [49](#)
  - SlowHome, [49](#)
- home\_settings\_t, [49](#)
  - FastHome, [50](#)
  - HomeDelta, [50](#)
  - HomeFlags, [50](#)
  - SlowHome, [50](#)
  - uFastHome, [50](#)
  - uHomeDelta, [50](#)
  - uSlowHome, [51](#)
- HomeDelta
  - home\_settings\_calb\_t, [49](#)
  - home\_settings\_t, [50](#)
- HomeFlags
  - home\_settings\_calb\_t, [49](#)
  - home\_settings\_t, [50](#)
- HorizontalLoadCapacity
  - stage\_settings\_t, [69](#)
- IPS
  - feedback\_settings\_t, [42](#)
- init\_random\_t, [51](#)

- key, [51](#)
- InputInertia
  - gear\_settings\_t, [44](#)
- lpwr
  - status\_calb\_t, [71](#)
  - status\_t, [74](#)
- lusb
  - status\_calb\_t, [72](#)
  - status\_t, [74](#)
- JOY\_REVERSE
  - ximc.h, [114](#)
- Joy
  - analog\_data\_t, [18](#)
  - chart\_data\_t, [23](#)
- Joy\_ADC
  - analog\_data\_t, [19](#)
- JoyCenter
  - joystick\_settings\_t, [52](#)
- JoyFlags
  - joystick\_settings\_t, [52](#)
- JoyHighEnd
  - joystick\_settings\_t, [52](#)
- JoyLowEnd
  - joystick\_settings\_t, [53](#)
- joystick\_settings\_t, [51](#)
  - DeadZone, [52](#)
  - ExpFactor, [52](#)
  - JoyCenter, [52](#)
  - JoyFlags, [52](#)
  - JoyHighEnd, [52](#)
  - JoyLowEnd, [53](#)
- Key
  - serial\_number\_t, [65](#)
- key
  - init\_random\_t, [51](#)
- Km
  - emf\_settings\_t, [33](#)
- L
  - emf\_settings\_t, [33](#)
- L5
  - analog\_data\_t, [19](#)
- L5\_ADC
  - analog\_data\_t, [19](#)
- LOW\_UPWR\_PROTECTION
  - ximc.h, [114](#)
- LS\_SHORTED
  - ximc.h, [114](#)
- LeadScrewPitch
  - stage\_settings\_t, [69](#)
- LeftBorder
  - edges\_settings\_calb\_t, [31](#)
  - edges\_settings\_t, [32](#)
- Length
  - measurements\_t, [53](#)
- LimitSwitchesSettings
  - accessories\_settings\_t, [15](#)
- load\_correction\_table
  - ximc.h, [147](#)
- logging\_callback\_stderr\_narrow
  - ximc.h, [148](#)
- logging\_callback\_stderr\_wide
  - ximc.h, [148](#)
- logging\_callback\_t
  - ximc.h, [122](#)
- LowUpwrOff
  - secure\_settings\_t, [64](#)
- 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, [115](#)
- MOVE\_STATE\_ANTIPLAY
  - ximc.h, [115](#)
- MOVE\_STATE\_MOVING
  - ximc.h, [115](#)
- MVCMD\_ERROR
  - ximc.h, [115](#)
- MVCMD\_HOME
  - ximc.h, [115](#)
- MVCMD\_LEFT
  - ximc.h, [115](#)
- MVCMD\_LOFT
  - ximc.h, [115](#)
- MVCMD\_MOVE
  - ximc.h, [116](#)
- MVCMD\_MOVR
  - ximc.h, [116](#)
- MVCMD\_NAME\_BITS
  - ximc.h, [116](#)
- MVCMD\_RIGHT
  - ximc.h, [116](#)
- MVCMD\_RUNNING
  - ximc.h, [116](#)
- MVCMD\_SSTP
  - ximc.h, [116](#)
- MVCMD\_STOP
  - ximc.h, [116](#)
- MVCMD\_UKNWN
  - ximc.h, [116](#)
- MagneticBrakeInfo
  - accessories\_settings\_t, [15](#)
- Major
  - device\_information\_t, [29](#)
  - serial\_number\_t, [65](#)

- Manufacturer
  - encoder\_information\_t, 34
  - gear\_information\_t, 43
  - hallsensor\_information\_t, 47
  - motor\_information\_t, 54
  - stage\_information\_t, 67
- MaxClickTime
  - control\_settings\_calb\_t, 24
  - control\_settings\_t, 26
- MaxCurrent
  - motor\_settings\_t, 56
- MaxCurrentConsumption
  - encoder\_settings\_t, 35
  - hallsensor\_settings\_t, 48
  - stage\_settings\_t, 69
- MaxCurrentTime
  - motor\_settings\_t, 56
- MaxOperatingFrequency
  - encoder\_settings\_t, 35
  - hallsensor\_settings\_t, 48
- MaxOutputBacklash
  - gear\_settings\_t, 44
- MaxSpeed
  - control\_settings\_calb\_t, 24
  - control\_settings\_t, 26
  - motor\_settings\_t, 56
  - stage\_settings\_t, 69
- measurements\_t, 53
  - Error, 53
  - Length, 53
  - Speed, 53
- MechanicalTimeConstant
  - motor\_settings\_t, 56
- MicrostepMode
  - engine\_settings\_calb\_t, 37
  - engine\_settings\_t, 39
- MinimumUusb
  - secure\_settings\_t, 64
- Minor
  - device\_information\_t, 29
  - serial\_number\_t, 65
- motor\_information\_t, 54
  - Manufacturer, 54
  - PartNumber, 54
- motor\_settings\_t, 54
  - DetentTorque, 56
  - MaxCurrent, 56
  - MaxCurrentTime, 56
  - MaxSpeed, 56
  - MechanicalTimeConstant, 56
  - MotorType, 56
  - NoLoadCurrent, 56
  - NoLoadSpeed, 56
  - NominalCurrent, 56
  - NominalPower, 56
  - NominalSpeed, 57
  - NominalTorque, 57
  - NominalVoltage, 57
  - Phases, 57
  - Poles, 57
  - RotorInertia, 57
  - SpeedConstant, 57
  - SpeedTorqueGradient, 57
  - StallTorque, 57
  - TorqueConstant, 58
  - WindingInductance, 58
  - WindingResistance, 58
- MotorType
  - motor\_settings\_t, 56
- move\_settings\_calb\_t, 58
  - Accel, 59
  - AntiplaySpeed, 59
  - Decel, 59
  - MoveFlags, 59
  - Speed, 59
- move\_settings\_t, 59
  - Accel, 60
  - AntiplaySpeed, 60
  - Decel, 60
  - MoveFlags, 60
  - Speed, 60
  - uAntiplaySpeed, 60
  - uSpeed, 60
- MoveFlags
  - move\_settings\_calb\_t, 59
  - move\_settings\_t, 60
- MoveSts
  - status\_calb\_t, 72
  - status\_t, 74
- msec\_sleep
  - ximc.h, 148
- MvCmdSts
  - status\_calb\_t, 72
  - status\_t, 74
- NoLoadCurrent
  - motor\_settings\_t, 56
- NoLoadSpeed
  - motor\_settings\_t, 56
- NomCurrent
  - engine\_settings\_calb\_t, 37
  - engine\_settings\_t, 39
- NomSpeed
  - engine\_settings\_calb\_t, 37
  - engine\_settings\_t, 39
- NomVoltage
  - engine\_settings\_calb\_t, 37
  - engine\_settings\_t, 39
- NominalCurrent
  - motor\_settings\_t, 56
- NominalPower
  - motor\_settings\_t, 56

- NominalSpeed
  - motor\_settings\_t, 57
- NominalTorque
  - motor\_settings\_t, 57
- NominalVoltage
  - motor\_settings\_t, 57
- nonvolatile\_memory\_t, 61
  - UserData, 61
- open\_device
  - ximc.h, 148
- POWER\_OFF\_ENABLED
  - ximc.h, 116
- POWER\_REDUCE\_ENABLED
  - ximc.h, 116
- POWER\_SMOOTH\_CURRENT
  - ximc.h, 116
- PWR\_STATE\_MAX
  - ximc.h, 117
- PWR\_STATE\_NORM
  - ximc.h, 117
- PWR\_STATE\_OFF
  - ximc.h, 117
- PWR\_STATE\_REDUCE
  - ximc.h, 117
- PWR\_STATE\_UNKNOWN
  - ximc.h, 117
- PWRSts
  - status\_calb\_t, 72
  - status\_t, 74
- PartNumber
  - encoder\_information\_t, 34
  - gear\_information\_t, 43
  - hallsensor\_information\_t, 47
  - motor\_information\_t, 54
  - stage\_information\_t, 67
- Phases
  - motor\_settings\_t, 57
- pid\_settings\_t, 61
- Poles
  - motor\_settings\_t, 57
- PosFlags
  - set\_position\_calb\_t, 66
  - set\_position\_t, 67
- Position
  - get\_position\_calb\_t, 45
  - set\_position\_calb\_t, 66
  - sync\_in\_settings\_calb\_t, 76
- PositionerName
  - stage\_name\_t, 68
- Pot
  - analog\_data\_t, 19
  - chart\_data\_t, 23
- power\_settings\_t, 62
  - CurrReductDelay, 62
  - CurrentSetTime, 62
  - HoldCurrent, 62
  - PowerFlags, 63
  - PowerOffDelay, 63
- PowerFlags
  - power\_settings\_t, 63
- PowerOffDelay
  - power\_settings\_t, 63
- probe\_device
  - ximc.h, 149
- R
  - emf\_settings\_t, 33
- REV\_SENS\_INV
  - ximc.h, 117
- RPM\_DIV\_1000
  - ximc.h, 117
- RatedInputSpeed
  - gear\_settings\_t, 44
- RatedInputTorque
  - gear\_settings\_t, 44
- ReductionIn
  - gear\_settings\_t, 44
- ReductionOut
  - gear\_settings\_t, 44
- Release
  - device\_information\_t, 30
  - serial\_number\_t, 65
- reset\_locks
  - ximc.h, 149
- RightBorder
  - edges\_settings\_calb\_t, 31
  - edges\_settings\_t, 32
- RotorInertia
  - motor\_settings\_t, 57
- SN
  - serial\_number\_t, 65
- STATE\_ALARM
  - ximc.h, 117
- STATE\_BRAKE
  - ximc.h, 118
- STATE\_BUTTON\_LEFT
  - ximc.h, 118
- STATE\_BUTTON\_RIGHT
  - ximc.h, 118
- STATE\_CONTR
  - ximc.h, 118
- STATE\_CTP\_ERROR
  - ximc.h, 118
- STATE\_DIG\_SIGNAL
  - ximc.h, 118
- STATE\_ENC\_A
  - ximc.h, 118
- STATE\_ENC\_B
  - ximc.h, 118
- STATE\_ERRC
  - ximc.h, 119

STATE\_ERRD  
ximc.h, [119](#)

STATE\_ERRV  
ximc.h, [119](#)

STATE\_EXTIO\_ALARM  
ximc.h, [119](#)

STATE\_GPIO\_LEVEL  
ximc.h, [119](#)

STATE\_GPIO\_PINOUT  
ximc.h, [119](#)

STATE\_LEFT\_EDGE  
ximc.h, [119](#)

STATE\_POWER\_OVERHEAT  
ximc.h, [120](#)

STATE\_REV\_SENSOR  
ximc.h, [120](#)

STATE\_RIGHT\_EDGE  
ximc.h, [120](#)

STATE\_SECUR  
ximc.h, [120](#)

STATE\_SYNC\_INPUT  
ximc.h, [120](#)

STATE\_SYNC\_OUTPUT  
ximc.h, [120](#)

SYNCIN\_ENABLED  
ximc.h, [120](#)

SYNCIN\_INVERT  
ximc.h, [120](#)

SYNCOUT\_ENABLED  
ximc.h, [120](#)

SYNCOUT\_IN\_STEPS  
ximc.h, [120](#)

SYNCOUT\_INVERT  
ximc.h, [121](#)

SYNCOUT\_ONPERIOD  
ximc.h, [121](#)

SYNCOUT\_ONSTART  
ximc.h, [121](#)

SYNCOUT\_ONSTOP  
ximc.h, [121](#)

SYNCOUT\_STATE  
ximc.h, [121](#)

secure\_settings\_t, [63](#)  
CriticalIpwr, [64](#)  
CriticalIusb, [64](#)  
CriticalUpwr, [64](#)  
CriticalUusb, [64](#)  
Flags, [64](#)  
LowUpwrOff, [64](#)  
MinimumUusb, [64](#)

serial\_number\_t, [64](#)  
Key, [65](#)  
Major, [65](#)  
Minor, [65](#)  
Release, [65](#)  
SN, [65](#)

service\_command\_updf  
ximc.h, [149](#)

set\_accessories\_settings  
ximc.h, [149](#)

set\_bindy\_key  
ximc.h, [149](#)

set\_brake\_settings  
ximc.h, [150](#)

set\_calibration\_settings  
ximc.h, [150](#)

set\_control\_settings  
ximc.h, [150](#)

set\_control\_settings\_calb  
ximc.h, [151](#)

set\_controller\_name  
ximc.h, [151](#)

set\_correction\_table  
ximc.h, [151](#)

set\_ctp\_settings  
ximc.h, [152](#)

set\_debug\_write  
ximc.h, [152](#)

set\_edges\_settings  
ximc.h, [152](#)

set\_edges\_settings\_calb  
ximc.h, [153](#)

set\_emf\_settings  
ximc.h, [153](#)

set\_encoder\_information  
ximc.h, [153](#)

set\_encoder\_settings  
ximc.h, [154](#)

set\_engine\_advansed\_setup  
ximc.h, [154](#)

set\_engine\_settings  
ximc.h, [154](#)

set\_engine\_settings\_calb  
ximc.h, [154](#)

set\_entype\_settings  
ximc.h, [155](#)

set\_extended\_settings  
ximc.h, [155](#)

set\_extio\_settings  
ximc.h, [155](#)

set\_feedback\_settings  
ximc.h, [156](#)

set\_gear\_information  
ximc.h, [156](#)

set\_gear\_settings  
ximc.h, [156](#)

set\_hallsensor\_information  
ximc.h, [157](#)

set\_hallsensor\_settings  
ximc.h, [157](#)

set\_home\_settings  
ximc.h, [157](#)

- set\_home\_settings\_calb
  - ximc.h, [157](#)
- set\_joystick\_settings
  - ximc.h, [158](#)
- set\_logging\_callback
  - ximc.h, [158](#)
- set\_motor\_information
  - ximc.h, [158](#)
- set\_motor\_settings
  - ximc.h, [159](#)
- set\_move\_settings
  - ximc.h, [159](#)
- set\_move\_settings\_calb
  - ximc.h, [159](#)
- set\_nonvolatile\_memory
  - ximc.h, [159](#)
- set\_pid\_settings
  - ximc.h, [159](#)
- set\_position
  - ximc.h, [160](#)
- set\_position\_calb
  - ximc.h, [160](#)
- set\_position\_calb\_t, [65](#)
  - EncPosition, [66](#)
  - PosFlags, [66](#)
  - Position, [66](#)
- set\_position\_t, [66](#)
  - EncPosition, [67](#)
  - PosFlags, [67](#)
  - uPosition, [67](#)
- set\_power\_settings
  - ximc.h, [160](#)
- set\_secure\_settings
  - ximc.h, [160](#)
- set\_serial\_number
  - ximc.h, [161](#)
- set\_stage\_information
  - ximc.h, [161](#)
- set\_stage\_name
  - ximc.h, [161](#)
- set\_stage\_settings
  - ximc.h, [161](#)
- set\_sync\_in\_settings
  - ximc.h, [162](#)
- set\_sync\_in\_settings\_calb
  - ximc.h, [162](#)
- set\_sync\_out\_settings
  - ximc.h, [162](#)
- set\_sync\_out\_settings\_calb
  - ximc.h, [163](#)
- set\_uart\_settings
  - ximc.h, [163](#)
- SlowHome
  - home\_settings\_calb\_t, [49](#)
  - home\_settings\_t, [50](#)
- Speed
  - measurements\_t, [53](#)
  - move\_settings\_calb\_t, [59](#)
  - move\_settings\_t, [60](#)
  - sync\_in\_settings\_calb\_t, [76](#)
  - sync\_in\_settings\_t, [77](#)
- SpeedConstant
  - motor\_settings\_t, [57](#)
- SpeedTorqueGradient
  - motor\_settings\_t, [57](#)
- stage\_information\_t, [67](#)
  - Manufacturer, [67](#)
  - PartNumber, [67](#)
- stage\_name\_t, [68](#)
  - PositionerName, [68](#)
- stage\_settings\_t, [68](#)
  - HorizontalLoadCapacity, [69](#)
  - LeadScrewPitch, [69](#)
  - MaxCurrentConsumption, [69](#)
  - MaxSpeed, [69](#)
  - SupplyVoltageMax, [69](#)
  - SupplyVoltageMin, [69](#)
  - TravelRange, [69](#)
  - Units, [69](#)
  - VerticalLoadCapacity, [70](#)
- StallTorque
  - motor\_settings\_t, [57](#)
- status\_calb\_t, [70](#)
  - CmdBufFreeSpace, [71](#)
  - CurPosition, [71](#)
  - CurSpeed, [71](#)
  - CurT, [71](#)
  - EncPosition, [71](#)
  - EncSts, [71](#)
  - Flags, [71](#)
  - GPIOFlags, [71](#)
  - lpwr, [71](#)
  - lusb, [72](#)
  - MoveSts, [72](#)
  - MvCmdSts, [72](#)
  - PWRSts, [72](#)
  - Upwr, [72](#)
  - Uusb, [72](#)
  - WindSts, [72](#)
- status\_t, [72](#)
  - CmdBufFreeSpace, [73](#)
  - CurPosition, [73](#)
  - CurSpeed, [74](#)
  - CurT, [74](#)
  - EncPosition, [74](#)
  - EncSts, [74](#)
  - Flags, [74](#)
  - GPIOFlags, [74](#)
  - lpwr, [74](#)
  - lusb, [74](#)
  - MoveSts, [74](#)
  - MvCmdSts, [74](#)

- PWRSts, [74](#)
- uCurPosition, [75](#)
- uCurSpeed, [75](#)
- Upwr, [75](#)
- Uusb, [75](#)
- WindSts, [75](#)
- stepcloseloop\_Kp\_high
  - engine\_advanced\_setup\_t, [36](#)
- stepcloseloop\_Kp\_low
  - engine\_advanced\_setup\_t, [36](#)
- stepcloseloop\_Kw
  - engine\_advanced\_setup\_t, [36](#)
- StepsPerRev
  - engine\_settings\_calb\_t, [38](#)
  - engine\_settings\_t, [39](#)
- SupVoltage
  - analog\_data\_t, [19](#)
- SupVoltage\_ADC
  - analog\_data\_t, [19](#)
- SupplyVoltageMax
  - encoder\_settings\_t, [35](#)
  - hallsensor\_settings\_t, [48](#)
  - stage\_settings\_t, [69](#)
- SupplyVoltageMin
  - encoder\_settings\_t, [35](#)
  - hallsensor\_settings\_t, [48](#)
  - stage\_settings\_t, [69](#)
- sync\_in\_settings\_calb\_t, [75](#)
  - ClutterTime, [76](#)
  - Position, [76](#)
  - Speed, [76](#)
  - SyncInFlags, [76](#)
- sync\_in\_settings\_t, [76](#)
  - ClutterTime, [77](#)
  - Speed, [77](#)
  - SyncInFlags, [77](#)
  - uPosition, [77](#)
  - uSpeed, [77](#)
- sync\_out\_settings\_calb\_t, [77](#)
  - Accuracy, [78](#)
  - SyncOutFlags, [78](#)
  - SyncOutPeriod, [78](#)
  - SyncOutPulseSteps, [78](#)
- sync\_out\_settings\_t, [78](#)
  - Accuracy, [79](#)
  - SyncOutFlags, [79](#)
  - SyncOutPeriod, [79](#)
  - SyncOutPulseSteps, [79](#)
  - uAccuracy, [79](#)
- SyncInFlags
  - sync\_in\_settings\_calb\_t, [76](#)
  - sync\_in\_settings\_t, [77](#)
- SyncOutFlags
  - sync\_out\_settings\_calb\_t, [78](#)
  - sync\_out\_settings\_t, [79](#)
- SyncOutPeriod
  - sync\_out\_settings\_calb\_t, [78](#)
  - sync\_out\_settings\_t, [79](#)
- sync\_out\_settings\_calb\_t, [78](#)
- sync\_out\_settings\_t, [79](#)
- SyncOutPulseSteps
  - sync\_out\_settings\_calb\_t, [78](#)
  - sync\_out\_settings\_t, [79](#)
- 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, [121](#)
- TSGrad
  - accessories\_settings\_t, [15](#)
- TSMMax
  - accessories\_settings\_t, [15](#)
- TSMIn
  - accessories\_settings\_t, [15](#)
- TSSettings
  - accessories\_settings\_t, [16](#)
- Temp
  - analog\_data\_t, [19](#)
- Temp\_ADC
  - analog\_data\_t, [19](#)
- TemperatureSensorInfo
  - accessories\_settings\_t, [15](#)
- Timeout
  - control\_settings\_calb\_t, [25](#)
  - control\_settings\_t, [26](#)
- TorqueConstant
  - motor\_settings\_t, [58](#)
- TravelRange
  - stage\_settings\_t, [69](#)
- UART\_PARITY\_BITS
  - ximc.h, [121](#)
- UARTSetupFlags
  - uart\_settings\_t, [80](#)
- uAccuracy
  - sync\_out\_settings\_t, [79](#)
- uAntiplaySpeed
  - move\_settings\_t, [60](#)
- uCurPosition
  - status\_t, [75](#)
- uCurSpeed
  - status\_t, [75](#)
- uDeltaPosition
  - control\_settings\_t, [26](#)
- uFastHome
  - home\_settings\_t, [50](#)
- uHomeDelta
  - home\_settings\_t, [50](#)
- uLeftBorder

- edges\_settings\_t, 32
- uMaxSpeed
  - control\_settings\_t, 26
- uNomSpeed
  - engine\_settings\_t, 39
- uPosition
  - get\_position\_t, 46
  - set\_position\_t, 67
  - sync\_in\_settings\_t, 77
- uRightBorder
  - edges\_settings\_t, 32
- uSlowHome
  - home\_settings\_t, 51
- uSpeed
  - move\_settings\_t, 60
  - sync\_in\_settings\_t, 77
- uart\_settings\_t, 80
  - UARTSetupFlags, 80
- UniquelD0
  - globally\_unique\_identifier\_t, 46
- UniquelD1
  - globally\_unique\_identifier\_t, 46
- UniquelD2
  - globally\_unique\_identifier\_t, 46
- UniquelD3
  - globally\_unique\_identifier\_t, 46
- Units
  - stage\_settings\_t, 69
- Upwr
  - status\_calb\_t, 72
  - status\_t, 75
- UserData
  - nonvolatile\_memory\_t, 61
- Uusb
  - status\_calb\_t, 72
  - status\_t, 75
- VerticalLoadCapacity
  - stage\_settings\_t, 70
- WIND\_A\_STATE\_ABSENT
  - ximc.h, 121
- WIND\_A\_STATE\_OK
  - ximc.h, 121
- WIND\_B\_STATE\_ABSENT
  - ximc.h, 122
- WIND\_B\_STATE\_OK
  - ximc.h, 122
- WindSts
  - status\_calb\_t, 72
  - status\_t, 75
- WindingCurrentA
  - chart\_data\_t, 23
- WindingCurrentB
  - chart\_data\_t, 23
- WindingCurrentC
  - chart\_data\_t, 23
- WindingInductance
  - motor\_settings\_t, 58
- WindingResistance
  - motor\_settings\_t, 58
- WindingVoltageA
  - chart\_data\_t, 23
- WindingVoltageB
  - chart\_data\_t, 23
- WindingVoltageC
  - chart\_data\_t, 23
- write\_key
  - ximc.h, 163
- XIMC\_API
  - ximc.h, 122
- ximc.h, 81
  - BACK\_EMF\_KM\_AUTO, 106
  - BORDER\_IS\_ENCODER, 106
  - BORDER\_STOP\_LEFT, 107
  - BORDER\_STOP\_RIGHT, 107
  - BRAKE\_ENABLED, 107
  - BRAKE\_ENG\_PWROFF, 107
  - CONTROL\_MODE\_BITS, 107
  - CONTROL\_MODE\_JOY, 107
  - CONTROL\_MODE\_LR, 107
  - CONTROL\_MODE\_OFF, 107
  - CTP\_ALARM\_ON\_ERROR, 108
  - CTP\_BASE, 108
  - CTP\_ENABLED, 108
  - close\_device, 122
  - command\_clear\_fram, 122
  - command\_eeread\_settings, 123
  - command\_eesave\_settings, 123
  - command\_home, 123
  - command\_homezero, 124
  - command\_left, 124
  - command\_loft, 124
  - command\_move, 124
  - command\_move\_calb, 125
  - command\_movr, 125
  - command\_movr\_calb, 125
  - command\_power\_off, 126
  - command\_read\_robust\_settings, 126
  - command\_read\_settings, 126
  - command\_reset, 126
  - command\_right, 126
  - command\_save\_robust\_settings, 127
  - command\_save\_settings, 127
  - command\_sstp, 127
  - command\_start\_measurements, 127
  - command\_stop, 127
  - command\_update\_firmware, 128
  - command\_wait\_for\_stop, 128
  - command\_zero, 128
  - EEPROM\_PRECEDENCE, 108
  - ENC\_STATE\_ABSENT, 108



- ENC\_STATE\_MALFUNC, 108
- ENC\_STATE\_OK, 108
- ENC\_STATE\_REVERS, 109
- ENC\_STATE\_UNKNOWN, 109
- ENDER\_SWAP, 109
- ENGINE\_ACCEL\_ON, 109
- ENGINE\_ANTIPLAY, 109
- ENGINE\_LIMIT\_CURR, 109
- ENGINE\_LIMIT\_RPM, 110
- ENGINE\_LIMIT\_VOLT, 110
- ENGINE\_MAX\_SPEED, 110
- ENGINE\_REVERSE, 110
- ENGINE\_TYPE\_2DC, 110
- ENGINE\_TYPE\_DC, 110
- ENGINE\_TYPE\_NONE, 110
- ENGINE\_TYPE\_STEP, 110
- ENGINE\_TYPE\_TEST, 110
- ENUMERATE\_PROBE, 111
- EXTIO\_SETUP\_INVERT, 111
- EXTIO\_SETUP\_OUTPUT, 112
- enumerate\_devices, 128
- FEEDBACK\_EMF, 112
- FEEDBACK\_ENCODER, 112
- FEEDBACK\_NONE, 113
- free\_enumerate\_devices, 129
- get\_accessories\_settings, 129
- get\_analog\_data, 129
- get\_bootloader\_version, 130
- get\_brake\_settings, 130
- get\_calibration\_settings, 130
- get\_chart\_data, 130
- get\_control\_settings, 131
- get\_control\_settings\_calb, 131
- get\_controller\_name, 131
- get\_ctp\_settings, 132
- get\_debug\_read, 132
- get\_device\_count, 132
- get\_device\_information, 132
- get\_device\_name, 133
- get\_edges\_settings, 133
- get\_edges\_settings\_calb, 133
- get\_emf\_settings, 134
- get\_encoder\_information, 134
- get\_encoder\_settings, 134
- get\_engine\_advanced\_setup, 134
- get\_engine\_settings, 135
- get\_engine\_settings\_calb, 135
- get\_entype\_settings, 135
- get\_enumerate\_device\_controller\_name, 136
- get\_enumerate\_device\_information, 136
- get\_enumerate\_device\_network\_information, 136
- get\_enumerate\_device\_serial, 136
- get\_enumerate\_device\_stage\_name, 137
- get\_extended\_settings, 137
- get\_extio\_settings, 137
- get\_feedback\_settings, 138
- get\_firmware\_version, 138
- get\_gear\_information, 138
- get\_gear\_settings, 138
- get\_globally\_unique\_identifier, 139
- get\_hallsensor\_information, 139
- get\_hallsensor\_settings, 139
- get\_home\_settings, 139
- get\_home\_settings\_calb, 140
- get\_init\_random, 140
- get\_joystick\_settings, 140
- get\_measurements, 141
- get\_motor\_information, 141
- get\_motor\_settings, 141
- get\_move\_settings, 141
- get\_move\_settings\_calb, 142
- get\_nonvolatile\_memory, 142
- get\_pid\_settings, 142
- get\_position, 142
- get\_position\_calb, 143
- get\_power\_settings, 143
- get\_secure\_settings, 143
- get\_serial\_number, 144
- get\_stage\_information, 144
- get\_stage\_name, 144
- get\_stage\_settings, 144
- get\_status, 144
- get\_status\_calb, 145
- get\_sync\_in\_settings, 145
- get\_sync\_in\_settings\_calb, 145
- get\_sync\_out\_settings, 146
- get\_sync\_out\_settings\_calb, 146
- get\_uart\_settings, 146
- goto\_firmware, 147
- HOME\_DIR\_FIRST, 113
- HOME\_DIR\_SECOND, 113
- HOME\_HALF\_MV, 113
- HOME\_MV\_SEC\_EN, 113
- HOME\_USE\_FAST, 114
- has\_firmware, 147
- JOY\_REVERSE, 114
- LOW\_UPWR\_PROTECTION, 114
- LS\_SHORTED, 114
- load\_correction\_table, 147
- logging\_callback\_stderr\_narrow, 148
- logging\_callback\_stderr\_wide, 148
- logging\_callback\_t, 122
- MICROSTEP\_MODE\_FULL, 115
- MOVE\_STATE\_ANTIPLAY, 115
- MOVE\_STATE\_MOVING, 115
- MVCMD\_ERROR, 115
- MVCMD\_HOME, 115
- MVCMD\_LEFT, 115
- MVCMD\_LOFT, 115
- MVCMD\_MOVE, 116
- MVCMD\_MOVR, 116

MVCMD\_NAME\_BITS, 116  
MVCMD\_RIGHT, 116  
MVCMD\_RUNNING, 116  
MVCMD\_SSTP, 116  
MVCMD\_STOP, 116  
MVCMD\_UKNWN, 116  
msec\_sleep, 148  
open\_device, 148  
POWER\_OFF\_ENABLED, 116  
PWR\_STATE\_MAX, 117  
PWR\_STATE\_NORM, 117  
PWR\_STATE\_OFF, 117  
PWR\_STATE\_REDUCT, 117  
PWR\_STATE\_UNKNOWN, 117  
probe\_device, 149  
REV\_SENS\_INV, 117  
RPM\_DIV\_1000, 117  
reset\_locks, 149  
STATE\_ALARM, 117  
STATE\_BRAKE, 118  
STATE\_BUTTON\_LEFT, 118  
STATE\_BUTTON\_RIGHT, 118  
STATE\_CONTR, 118  
STATE\_CTP\_ERROR, 118  
STATE\_DIG\_SIGNAL, 118  
STATE\_ENC\_A, 118  
STATE\_ENC\_B, 118  
STATE\_ERRC, 119  
STATE\_ERRD, 119  
STATE\_ERRV, 119  
STATE\_EXTIO\_ALARM, 119  
STATE\_GPIO\_LEVEL, 119  
STATE\_GPIO\_PINOUT, 119  
STATE\_LEFT\_EDGE, 119  
STATE\_REV\_SENSOR, 120  
STATE\_RIGHT\_EDGE, 120  
STATE\_SECUR, 120  
STATE\_SYNC\_INPUT, 120  
STATE\_SYNC\_OUTPUT, 120  
SYNCIN\_ENABLED, 120  
SYNCIN\_INVERT, 120  
SYNCOUT\_ENABLED, 120  
SYNCOUT\_IN\_STEPS, 120  
SYNCOUT\_INVERT, 121  
SYNCOUT\_ONPERIOD, 121  
SYNCOUT\_ONSTART, 121  
SYNCOUT\_ONSTOP, 121  
SYNCOUT\_STATE, 121  
service\_command\_updf, 149  
set\_accessories\_settings, 149  
set\_bindy\_key, 149  
set\_brake\_settings, 150  
set\_calibration\_settings, 150  
set\_control\_settings, 150  
set\_control\_settings\_calb, 151  
set\_controller\_name, 151  
set\_correction\_table, 151  
set\_ctp\_settings, 152  
set\_debug\_write, 152  
set\_edges\_settings, 152  
set\_edges\_settings\_calb, 153  
set\_emf\_settings, 153  
set\_encoder\_information, 153  
set\_encoder\_settings, 154  
set\_engine\_advansed\_setup, 154  
set\_engine\_settings, 154  
set\_engine\_settings\_calb, 154  
set\_entype\_settings, 155  
set\_extended\_settings, 155  
set\_extio\_settings, 155  
set\_feedback\_settings, 156  
set\_gear\_information, 156  
set\_gear\_settings, 156  
set\_hallsensor\_information, 157  
set\_hallsensor\_settings, 157  
set\_home\_settings, 157  
set\_home\_settings\_calb, 157  
set\_joystick\_settings, 158  
set\_logging\_callback, 158  
set\_motor\_information, 158  
set\_motor\_settings, 159  
set\_move\_settings, 159  
set\_move\_settings\_calb, 159  
set\_nonvolatile\_memory, 159  
set\_pid\_settings, 159  
set\_position, 160  
set\_position\_calb, 160  
set\_power\_settings, 160  
set\_secure\_settings, 160  
set\_serial\_number, 161  
set\_stage\_information, 161  
set\_stage\_name, 161  
set\_stage\_settings, 161  
set\_sync\_in\_settings, 162  
set\_sync\_in\_settings\_calb, 162  
set\_sync\_out\_settings, 162  
set\_sync\_out\_settings\_calb, 163  
set\_uart\_settings, 163  
TS\_TYPE\_BITS, 121  
UART\_PARITY\_BITS, 121  
WIND\_A\_STATE\_OK, 121  
WIND\_B\_STATE\_OK, 122  
write\_key, 163  
XIMC\_API, 122  
ximc\_fix\_usbser\_sys, 164  
ximc\_version, 164  
ximc\_fix\_usbser\_sys  
ximc.h, 164  
ximc\_version  
ximc.h, 164