

libximc

3.0.2

Документация по libximc. Последние изменения: Вт 16 Дек 2025 15:52:48. Создано  
системой Doxygen 1.14.0

Вт 16 Дек 2025 15:52:48

<b>1 Библиотека libximc</b>	<b>1</b>
1.1 Что делает контроллер 8SMC4-USB и 8SMC5-USB	1
1.2 Что умеет библиотека libximc	1
1.3 Содействие	2
<b>2 Введение</b>	<b>3</b>
2.1 О библиотеке	3
2.1.1 Поддерживаемые операционные системы и требования к окружению:	3
<b>3 Как использовать с...</b>	<b>4</b>
3.0.1 Visual C++	4
3.0.2 CodeBlocks	4
3.0.3 MinGW	5
3.0.4 C++ Builder	5
3.0.5 XCode	5
3.0.6 GCC	5
3.0.7 .NET	6
3.0.8 Java	6
3.0.9 Python	6
3.0.10 MATLAB	7
3.1 Логирование в файл	7
3.2 Требуемые права доступа	7
3.3 Си-профили	7
3.4 Python-профили	7
<b>4 Работа с пользовательскими единицами</b>	<b>8</b>
4.1 Структура пересчета единиц calibration_t	8
4.2 Функции дублиеры для работы с пользовательскими единицами и структуры данных для них	8
4.3 Таблица коррекции координат для более точного позиционирования	9
<b>5 Структуры данных</b>	<b>10</b>
5.1 Структура accessories_settings_t	10
5.1.1 Подробное описание	11
5.1.2 Поля	11
5.1.2.1 LimitSwitchesSettings	11
5.1.2.2 MagneticBrakeInfo	11
5.1.2.3 MBRatedCurrent	11
5.1.2.4 MBRatedVoltage	11
5.1.2.5 MBSettings	11
5.1.2.6 MBTorque	12
5.1.2.7 TemperatureSensorInfo	12
5.1.2.8 TSGrad	12

5.1.2.9 TSMaх . . . . .	12
5.1.2.10 TSMiн . . . . .	12
5.1.2.11 TSSettings . . . . .	12
5.2 Структура analog_data_t . . . . .	12
5.2.1 Подробное описание . . . . .	14
5.2.2 Поля . . . . .	14
5.2.2.1 A1Voltage . . . . .	14
5.2.2.2 A1Voltage_ADC . . . . .	14
5.2.2.3 A2Voltage . . . . .	14
5.2.2.4 A2Voltage_ADC . . . . .	14
5.2.2.5 ACurrent . . . . .	15
5.2.2.6 ACurrent_ADC . . . . .	15
5.2.2.7 B1Voltage . . . . .	15
5.2.2.8 B1Voltage_ADC . . . . .	15
5.2.2.9 B2Voltage . . . . .	15
5.2.2.10 B2Voltage_ADC . . . . .	15
5.2.2.11 BCurrent . . . . .	15
5.2.2.12 BCurrent_ADC . . . . .	16
5.2.2.13 Enc_Check . . . . .	16
5.2.2.14 Enc_Check_ADC . . . . .	16
5.2.2.15 FullCurrent . . . . .	16
5.2.2.16 FullCurrent_ADC . . . . .	16
5.2.2.17 Joy . . . . .	16
5.2.2.18 Joy_ADC . . . . .	16
5.2.2.19 Pot . . . . .	17
5.2.2.20 SupVoltage . . . . .	17
5.2.2.21 SupVoltage_ADC . . . . .	17
5.2.2.22 Temp . . . . .	17
5.2.2.23 Temp_ADC . . . . .	17
5.3 Структура brake_settings_t . . . . .	17
5.3.1 Подробное описание . . . . .	18
5.3.2 Поля . . . . .	18
5.3.2.1 BrakeFlags . . . . .	18
5.3.2.2 t1 . . . . .	18
5.3.2.3 t2 . . . . .	18
5.3.2.4 t3 . . . . .	18
5.3.2.5 t4 . . . . .	18
5.4 Структура calibration_settings_t . . . . .	19
5.4.1 Подробное описание . . . . .	19
5.4.2 Поля . . . . .	19
5.4.2.1 CSS1_A . . . . .	19

5.4.2.2 CSS1_B . . . . .	20
5.4.2.3 CSS2_A . . . . .	20
5.4.2.4 CSS2_B . . . . .	20
5.4.2.5 FullCurrent_A . . . . .	20
5.4.2.6 FullCurrent_B . . . . .	20
5.5 Структура calibration_t . . . . .	20
5.5.1 Подробное описание . . . . .	21
5.5.2 Поля . . . . .	21
5.5.2.1 A . . . . .	21
5.5.2.2 MicrostepMode . . . . .	22
5.6 Структура chart_data_t . . . . .	22
5.6.1 Подробное описание . . . . .	23
5.6.2 Поля . . . . .	23
5.6.2.1 AveragedPowerRatio . . . . .	23
5.6.2.2 Joy . . . . .	23
5.6.2.3 Pot . . . . .	23
5.6.2.4 WindingCurrentA . . . . .	23
5.6.2.5 WindingCurrentB . . . . .	24
5.6.2.6 WindingCurrentC . . . . .	24
5.6.2.7 WindingVoltageA . . . . .	24
5.6.2.8 WindingVoltageB . . . . .	24
5.6.2.9 WindingVoltageC . . . . .	24
5.7 Структура control_settings_calb_t . . . . .	24
5.7.1 Подробное описание . . . . .	25
5.7.2 Поля . . . . .	25
5.7.2.1 Flags . . . . .	25
5.7.2.2 MaxClickTime . . . . .	25
5.7.2.3 MaxSpeed . . . . .	26
5.7.2.4 Timeout . . . . .	26
5.8 Структура control_settings_t . . . . .	26
5.8.1 Подробное описание . . . . .	26
5.8.2 Поля . . . . .	27
5.8.2.1 Flags . . . . .	27
5.8.2.2 MaxClickTime . . . . .	27
5.8.2.3 MaxSpeed . . . . .	27
5.8.2.4 Timeout . . . . .	27
5.8.2.5 uDeltaPosition . . . . .	27
5.8.2.6 uMaxSpeed . . . . .	27
5.9 Структура controller_name_t . . . . .	28
5.9.1 Подробное описание . . . . .	28
5.9.2 Поля . . . . .	28

5.9.2.1 ControllerName . . . . .	28
5.9.2.2 CtrlFlags . . . . .	28
5.10 Структура <code>ctp_settings_t</code> . . . . .	28
5.10.1 Подробное описание . . . . .	29
5.10.2 Поля . . . . .	29
5.10.2.1 CTPFlags . . . . .	29
5.10.2.2 CTPMinError . . . . .	29
5.11 Структура <code>debug_read_t</code> . . . . .	30
5.11.1 Подробное описание . . . . .	30
5.11.2 Поля . . . . .	30
5.11.2.1 DebugData . . . . .	30
5.12 Структура <code>debug_write_t</code> . . . . .	30
5.12.1 Подробное описание . . . . .	31
5.12.2 Поля . . . . .	31
5.12.2.1 DebugData . . . . .	31
5.13 Структура <code>device_information_t</code> . . . . .	31
5.13.1 Подробное описание . . . . .	31
5.13.2 Поля . . . . .	32
5.13.2.1 Major . . . . .	32
5.13.2.2 Minor . . . . .	32
5.13.2.3 Release . . . . .	32
5.14 Структура <code>device_network_information_t</code> . . . . .	32
5.14.1 Подробное описание . . . . .	32
5.15 Структура <code>edges_settings_calb_t</code> . . . . .	33
5.15.1 Подробное описание . . . . .	33
5.15.2 Поля . . . . .	33
5.15.2.1 BorderFlags . . . . .	33
5.15.2.2 EnderFlags . . . . .	33
5.15.2.3 LeftBorder . . . . .	34
5.15.2.4 RightBorder . . . . .	34
5.16 Структура <code>edges_settings_t</code> . . . . .	34
5.16.1 Подробное описание . . . . .	34
5.16.2 Поля . . . . .	35
5.16.2.1 BorderFlags . . . . .	35
5.16.2.2 EnderFlags . . . . .	35
5.16.2.3 LeftBorder . . . . .	35
5.16.2.4 RightBorder . . . . .	35
5.16.2.5 uLeftBorder . . . . .	35
5.16.2.6 uRightBorder . . . . .	35
5.17 Структура <code>emf_settings_t</code> . . . . .	36
5.17.1 Подробное описание . . . . .	36

5.17.2 Поля	36
5.17.2.1 BackEMFFlags	36
5.17.2.2 Km	36
5.17.2.3 L	37
5.17.2.4 R	37
5.18 Структура encoder_information_t	37
5.18.1 Подробное описание	37
5.18.2 Поля	37
5.18.2.1 Manufacturer	37
5.18.2.2 PartNumber	38
5.19 Структура encoder_settings_t	38
5.19.1 Подробное описание	38
5.19.2 Поля	38
5.19.2.1 EncoderSettings	38
5.19.2.2 MaxCurrentConsumption	39
5.19.2.3 MaxOperatingFrequency	39
5.19.2.4 SupplyVoltageMax	39
5.19.2.5 SupplyVoltageMin	39
5.20 Структура engine_advanced_setup_t	39
5.20.1 Подробное описание	40
5.20.2 Поля	40
5.20.2.1 stepcloseloop_Kp_high	40
5.20.2.2 stepcloseloop_Kp_low	40
5.20.2.3 stepcloseloop_Kw	40
5.21 Структура engine_settings_calb_t	40
5.21.1 Подробное описание	41
5.21.2 Поля	41
5.21.2.1 Antiplay	41
5.21.2.2 EngineFlags	41
5.21.2.3 MicrostepMode	42
5.21.2.4 NomCurrent	42
5.21.2.5 NomSpeed	42
5.21.2.6 NomVoltage	42
5.21.2.7 StepsPerRev	42
5.22 Структура engine_settings_t	42
5.22.1 Подробное описание	43
5.22.2 Поля	43
5.22.2.1 Antiplay	43
5.22.2.2 EngineFlags	44
5.22.2.3 MicrostepMode	44
5.22.2.4 NomCurrent	44

5.22.2.5 NomSpeed . . . . .	44
5.22.2.6 NomVoltage . . . . .	44
5.22.2.7 StepsPerRev . . . . .	44
5.22.2.8 uNomSpeed . . . . .	45
5.23 Структура entype_settings_t . . . . .	45
5.23.1 Подробное описание . . . . .	45
5.23.2 Поля . . . . .	45
5.23.2.1 DriverType . . . . .	45
5.23.2.2 EngineType . . . . .	46
5.24 Структура extended_settings_t . . . . .	46
5.24.1 Подробное описание . . . . .	46
5.25 Структура extio_settings_t . . . . .	46
5.25.1 Подробное описание . . . . .	47
5.25.2 Поля . . . . .	47
5.25.2.1 EXTIOModeFlags . . . . .	47
5.25.2.2 EXTIOSetupFlags . . . . .	47
5.26 Структура feedback_settings_t . . . . .	47
5.26.1 Подробное описание . . . . .	48
5.26.2 Поля . . . . .	48
5.26.2.1 CountsPerTurn . . . . .	48
5.26.2.2 FeedbackFlags . . . . .	48
5.26.2.3 FeedbackType . . . . .	48
5.26.2.4 IPS . . . . .	48
5.27 Структура gear_information_t . . . . .	48
5.27.1 Подробное описание . . . . .	49
5.27.2 Поля . . . . .	49
5.27.2.1 Manufacturer . . . . .	49
5.27.2.2 PartNumber . . . . .	49
5.28 Структура gear_settings_t . . . . .	49
5.28.1 Подробное описание . . . . .	50
5.28.2 Поля . . . . .	50
5.28.2.1 Efficiency . . . . .	50
5.28.2.2 InputInertia . . . . .	50
5.28.2.3 MaxOutputBacklash . . . . .	51
5.28.2.4 RatedInputSpeed . . . . .	51
5.28.2.5 RatedInputTorque . . . . .	51
5.28.2.6 ReductionIn . . . . .	51
5.28.2.7 ReductionOut . . . . .	51
5.29 Структура get_position_calb_t . . . . .	51
5.29.1 Подробное описание . . . . .	52
5.29.2 Поля . . . . .	52

5.29.2.1 EncPosition	52
5.29.2.2 Position	52
5.30 Структура get_position_t	52
5.30.1 Подробное описание	53
5.30.2 Поля	53
5.30.2.1 EncPosition	53
5.30.2.2 uPosition	53
5.31 Структура globally_unique_identifier_t	53
5.31.1 Подробное описание	54
5.31.2 Поля	54
5.31.2.1 UniqueID0	54
5.31.2.2 UniqueID1	54
5.31.2.3 UniqueID2	54
5.31.2.4 UniqueID3	54
5.32 Структура hallsensor_information_t	54
5.32.1 Подробное описание	55
5.32.2 Поля	55
5.32.2.1 Manufacturer	55
5.32.2.2 PartNumber	55
5.33 Структура hallsensor_settings_t	55
5.33.1 Подробное описание	56
5.33.2 Поля	56
5.33.2.1 MaxCurrentConsumption	56
5.33.2.2 MaxOperatingFrequency	56
5.33.2.3 SupplyVoltageMax	56
5.33.2.4 SupplyVoltageMin	57
5.34 Структура home_settings_calb_t	57
5.34.1 Подробное описание	57
5.34.2 Поля	57
5.34.2.1 FastHome	57
5.34.2.2 HomeDelta	58
5.34.2.3 HomeFlags	58
5.34.2.4 SlowHome	58
5.35 Структура home_settings_t	58
5.35.1 Подробное описание	59
5.35.2 Поля	59
5.35.2.1 FastHome	59
5.35.2.2 HomeDelta	59
5.35.2.3 HomeFlags	59
5.35.2.4 SlowHome	59
5.35.2.5 uFastHome	60



5.35.2.6 uHomeDelta . . . . .	60
5.35.2.7 uSlowHome . . . . .	60
5.36 Структура init_random_t . . . . .	60
5.36.1 Подробное описание . . . . .	60
5.36.2 Поля . . . . .	61
5.36.2.1 key . . . . .	61
5.37 Структура joystick_settings_t . . . . .	61
5.37.1 Подробное описание . . . . .	61
5.37.2 Поля . . . . .	62
5.37.2.1 DeadZone . . . . .	62
5.37.2.2 ExpFactor . . . . .	62
5.37.2.3 JoyCenter . . . . .	62
5.37.2.4 JoyFlags . . . . .	62
5.37.2.5 JoyHighEnd . . . . .	62
5.37.2.6 JoyLowEnd . . . . .	62
5.38 Структура measurements_t . . . . .	63
5.38.1 Подробное описание . . . . .	63
5.38.2 Поля . . . . .	63
5.38.2.1 Length . . . . .	63
5.39 Структура motor_information_t . . . . .	63
5.39.1 Подробное описание . . . . .	64
5.39.2 Поля . . . . .	64
5.39.2.1 Manufacturer . . . . .	64
5.39.2.2 PartNumber . . . . .	64
5.40 Структура motor_settings_t . . . . .	64
5.40.1 Подробное описание . . . . .	66
5.40.2 Поля . . . . .	66
5.40.2.1 DetentTorque . . . . .	66
5.40.2.2 MaxCurrent . . . . .	66
5.40.2.3 MaxCurrentTime . . . . .	66
5.40.2.4 MaxSpeed . . . . .	66
5.40.2.5 MechanicalTimeConstant . . . . .	67
5.40.2.6 MotorType . . . . .	67
5.40.2.7 NoLoadCurrent . . . . .	67
5.40.2.8 NoLoadSpeed . . . . .	67
5.40.2.9 NominalCurrent . . . . .	67
5.40.2.10 NominalPower . . . . .	67
5.40.2.11 NominalSpeed . . . . .	68
5.40.2.12 NominalTorque . . . . .	68
5.40.2.13 NominalVoltage . . . . .	68
5.40.2.14 Phases . . . . .	68

5.40.2.15 Poles	68
5.40.2.16 RotorInertia	68
5.40.2.17 SpeedConstant	69
5.40.2.18 SpeedTorqueGradient	69
5.40.2.19 StallTorque	69
5.40.2.20 TorqueConstant	69
5.40.2.21 WindingInductance	69
5.40.2.22 WindingResistance	70
5.41 Структура move_settings_calb_t	70
5.41.1 Подробное описание	70
5.41.2 Поля	70
5.41.2.1 Accel	70
5.41.2.2 AntiplaySpeed	71
5.41.2.3 Decel	71
5.41.2.4 MoveFlags	71
5.41.2.5 Speed	71
5.42 Структура move_settings_t	71
5.42.1 Подробное описание	72
5.42.2 Поля	72
5.42.2.1 Accel	72
5.42.2.2 AntiplaySpeed	72
5.42.2.3 Decel	73
5.42.2.4 MoveFlags	73
5.42.2.5 Speed	73
5.42.2.6 uAntiplaySpeed	73
5.42.2.7 uSpeed	73
5.43 Структура network_settings_t	73
5.43.1 Подробное описание	74
5.43.2 Поля	74
5.43.2.1 DefaultGateway	74
5.43.2.2 DHCPEnabled	74
5.43.2.3 IPv4Address	74
5.43.2.4 SubnetMask	75
5.44 Структура nonvolatile_memory_t	75
5.44.1 Подробное описание	75
5.44.2 Поля	75
5.44.2.1 UserData	75
5.45 Структура password_settings_t	75
5.45.1 Подробное описание	76
5.45.2 Поля	76
5.45.2.1 UserPassword	76

5.46 Структура <code>pid_settings_t</code> . . . . .	76
5.46.1 Подробное описание . . . . .	77
5.47 Структура <code>power_settings_t</code> . . . . .	77
5.47.1 Подробное описание . . . . .	77
5.47.2 Поля . . . . .	78
5.47.2.1 <code>CurrentSetTime</code> . . . . .	78
5.47.2.2 <code>CurrReductDelay</code> . . . . .	78
5.47.2.3 <code>HoldCurrent</code> . . . . .	78
5.47.2.4 <code>PowerFlags</code> . . . . .	78
5.47.2.5 <code>PowerOffDelay</code> . . . . .	78
5.48 Структура <code>secure_settings_t</code> . . . . .	78
5.48.1 Подробное описание . . . . .	79
5.48.2 Поля . . . . .	79
5.48.2.1 <code>CriticalIpwr</code> . . . . .	79
5.48.2.2 <code>CriticalUsb</code> . . . . .	79
5.48.2.3 <code>CriticalT</code> . . . . .	80
5.48.2.4 <code>CriticalUpwr</code> . . . . .	80
5.48.2.5 <code>CriticalUusb</code> . . . . .	80
5.48.2.6 <code>Flags</code> . . . . .	80
5.48.2.7 <code>LowUpwrOff</code> . . . . .	80
5.48.2.8 <code>MinimumUusb</code> . . . . .	80
5.49 Структура <code>serial_number_t</code> . . . . .	80
5.49.1 Подробное описание . . . . .	81
5.49.2 Поля . . . . .	81
5.49.2.1 <code>Key</code> . . . . .	81
5.49.2.2 <code>Major</code> . . . . .	81
5.49.2.3 <code>Minor</code> . . . . .	81
5.49.2.4 <code>Release</code> . . . . .	82
5.49.2.5 <code>SN</code> . . . . .	82
5.50 Структура <code>set_position_calb_t</code> . . . . .	82
5.50.1 Подробное описание . . . . .	82
5.50.2 Поля . . . . .	82
5.50.2.1 <code>EncPosition</code> . . . . .	82
5.50.2.2 <code>PosFlags</code> . . . . .	83
5.50.2.3 <code>Position</code> . . . . .	83
5.51 Структура <code>set_position_t</code> . . . . .	83
5.51.1 Подробное описание . . . . .	83
5.51.2 Поля . . . . .	83
5.51.2.1 <code>EncPosition</code> . . . . .	83
5.51.2.2 <code>PosFlags</code> . . . . .	84
5.51.2.3 <code>uPosition</code> . . . . .	84

5.52 Структура stage_information_t . . . . .	84
5.52.1 Подробное описание . . . . .	84
5.52.2 Поля . . . . .	84
5.52.2.1 Manufacturer . . . . .	84
5.52.2.2 PartNumber . . . . .	85
5.53 Структура stage_name_t . . . . .	85
5.53.1 Подробное описание . . . . .	85
5.53.2 Поля . . . . .	85
5.53.2.1 PositionerName . . . . .	85
5.54 Структура stage_settings_t . . . . .	85
5.54.1 Подробное описание . . . . .	86
5.54.2 Поля . . . . .	86
5.54.2.1 HorizontalLoadCapacity . . . . .	86
5.54.2.2 LeadScrewPitch . . . . .	86
5.54.2.3 MaxCurrentConsumption . . . . .	87
5.54.2.4 MaxSpeed . . . . .	87
5.54.2.5 SupplyVoltageMax . . . . .	87
5.54.2.6 SupplyVoltageMin . . . . .	87
5.54.2.7 TravelRange . . . . .	87
5.54.2.8 Units . . . . .	87
5.54.2.9 VerticalLoadCapacity . . . . .	88
5.55 Структура status_calb_t . . . . .	88
5.55.1 Подробное описание . . . . .	89
5.55.2 Поля . . . . .	89
5.55.2.1 CmdBufFreeSpace . . . . .	89
5.55.2.2 CurPosition . . . . .	89
5.55.2.3 CurSpeed . . . . .	89
5.55.2.4 CurT . . . . .	89
5.55.2.5 EncPosition . . . . .	90
5.55.2.6 EncSts . . . . .	90
5.55.2.7 Flags . . . . .	90
5.55.2.8 GPIOFlags . . . . .	90
5.55.2.9 Ipwr . . . . .	90
5.55.2.10 Iusb . . . . .	90
5.55.2.11 MoveSts . . . . .	90
5.55.2.12 MvCmdSts . . . . .	91
5.55.2.13 PWRSts . . . . .	91
5.55.2.14 Upwr . . . . .	91
5.55.2.15 Uusb . . . . .	91
5.55.2.16 WindSts . . . . .	91
5.56 Структура status_t . . . . .	91

5.56.1	Подробное описание	92
5.56.2	Поля	93
5.56.2.1	CmdBufFreeSpace	93
5.56.2.2	CurPosition	93
5.56.2.3	CurSpeed	93
5.56.2.4	CurT	93
5.56.2.5	EncPosition	93
5.56.2.6	EncSts	93
5.56.2.7	Flags	94
5.56.2.8	GPIOFlags	94
5.56.2.9	Ipwr	94
5.56.2.10	Iusb	94
5.56.2.11	MoveSts	94
5.56.2.12	MvCmdSts	94
5.56.2.13	PWRSts	94
5.56.2.14	uCurPosition	95
5.56.2.15	uCurSpeed	95
5.56.2.16	Upwr	95
5.56.2.17	Uusb	95
5.56.2.18	WindSts	95
5.57	Структура sync_in_settings_calb_t	95
5.57.1	Подробное описание	96
5.57.2	Поля	96
5.57.2.1	ClutterTime	96
5.57.2.2	Position	96
5.57.2.3	Speed	96
5.57.2.4	SyncInFlags	97
5.58	Структура sync_in_settings_t	97
5.58.1	Подробное описание	97
5.58.2	Поля	97
5.58.2.1	ClutterTime	97
5.58.2.2	Speed	98
5.58.2.3	SyncInFlags	98
5.58.2.4	uPosition	98
5.58.2.5	uSpeed	98
5.59	Структура sync_out_settings_calb_t	98
5.59.1	Подробное описание	99
5.59.2	Поля	99
5.59.2.1	Accuracy	99
5.59.2.2	SyncOutFlags	99
5.59.2.3	SyncOutPeriod	99

5.59.2.4 SyncOutPulseSteps . . . . .	99
5.60 Структура sync_out_settings_t . . . . .	99
5.60.1 Подробное описание . . . . .	100
5.60.2 Поля . . . . .	100
5.60.2.1 Accuracy . . . . .	100
5.60.2.2 SyncOutFlags . . . . .	100
5.60.2.3 SyncOutPeriod . . . . .	101
5.60.2.4 SyncOutPulseSteps . . . . .	101
5.60.2.5 uAccuracy . . . . .	101
5.61 Структура uart_settings_t . . . . .	101
5.61.1 Подробное описание . . . . .	101
5.61.2 Поля . . . . .	101
5.61.2.1 UARTSetupFlags . . . . .	101
<b>6 Файлы</b>	<b>102</b>
6.1 Файл ximc.h . . . . .	102
6.1.1 Подробное описание . . . . .	128
6.1.2 Макросы . . . . .	128
6.1.2.1 ALARM_FLAGS_STICKING . . . . .	128
6.1.2.2 ALARM_ON_BORDERS_SWAP_MISSET . . . . .	128
6.1.2.3 ALARM_ON_DRIVER_OVERHEATING . . . . .	128
6.1.2.4 BACK_EMF_INDUCTANCE_AUTO . . . . .	129
6.1.2.5 BACK_EMF_KM_AUTO . . . . .	129
6.1.2.6 BACK_EMF_RESISTANCE_AUTO . . . . .	129
6.1.2.7 BORDER_IS_ENCODER . . . . .	129
6.1.2.8 BORDER_STOP_LEFT . . . . .	129
6.1.2.9 BORDER_STOP_RIGHT . . . . .	129
6.1.2.10 BORDERS_SWAP_MISSET_DETECTION . . . . .	129
6.1.2.11 BRAKE_ENABLED . . . . .	130
6.1.2.12 BRAKE_ENG_PWROFF . . . . .	130
6.1.2.13 BRAKING_OVERVOLTAGE_PROTECTION . . . . .	130
6.1.2.14 CONTROL_BTN_LEFT_PUSHED_OPEN . . . . .	130
6.1.2.15 CONTROL_BTN_RIGHT_PUSHED_OPEN . . . . .	130
6.1.2.16 CONTROL_MODE_BITS . . . . .	130
6.1.2.17 CONTROL_MODE_JOY . . . . .	130
6.1.2.18 CONTROL_MODE_LR . . . . .	131
6.1.2.19 CONTROL_MODE_OFF . . . . .	131
6.1.2.20 CTP_ALARM_ON_ERROR . . . . .	131
6.1.2.21 CTP_BASE . . . . .	131
6.1.2.22 CTP_ENABLED . . . . .	131
6.1.2.23 CTP_ERROR_CORRECTION . . . . .	131

6.1.2.24 DRIVER_TYPE_EXTERNAL . . . . .	131
6.1.2.25 DRIVER_TYPE_INTEGRATE . . . . .	132
6.1.2.26 EEPROM_PRECEDENCE . . . . .	132
6.1.2.27 ENC_STATE_ABSENT . . . . .	132
6.1.2.28 ENC_STATE_MALFUNC . . . . .	132
6.1.2.29 ENC_STATE_OK . . . . .	132
6.1.2.30 ENC_STATE_REVERS . . . . .	132
6.1.2.31 ENC_STATE_UNKNOWN . . . . .	132
6.1.2.32 ENDER_SW1_ACTIVE_LOW . . . . .	133
6.1.2.33 ENDER_SW2_ACTIVE_LOW . . . . .	133
6.1.2.34 ENDER_SWAP . . . . .	133
6.1.2.35 ENGINE_ACCEL_ON . . . . .	133
6.1.2.36 ENGINE_ANTIPLAY . . . . .	133
6.1.2.37 ENGINE_CURRENT_AS_RMS . . . . .	133
6.1.2.38 ENGINE_LIMIT_CURR . . . . .	134
6.1.2.39 ENGINE_LIMIT_RPM . . . . .	134
6.1.2.40 ENGINE_LIMIT_VOLT . . . . .	134
6.1.2.41 ENGINE_MAX_SPEED . . . . .	134
6.1.2.42 ENGINE_REVERSE . . . . .	134
6.1.2.43 ENGINE_TYPE_2DC . . . . .	135
6.1.2.44 ENGINE_TYPE_BRUSHLESS . . . . .	135
6.1.2.45 ENGINE_TYPE_DC . . . . .	135
6.1.2.46 ENGINE_TYPE_NONE . . . . .	135
6.1.2.47 ENGINE_TYPE_STEP . . . . .	135
6.1.2.48 ENGINE_TYPE_TEST . . . . .	135
6.1.2.49 ENUMERATE_PROBE . . . . .	135
6.1.2.50 EXTIO_SETUP_INVERT . . . . .	136
6.1.2.51 EXTIO_SETUP_MODE_IN_ALARM . . . . .	136
6.1.2.52 EXTIO_SETUP_MODE_IN_BITS . . . . .	136
6.1.2.53 EXTIO_SETUP_MODE_IN_HOME . . . . .	136
6.1.2.54 EXTIO_SETUP_MODE_IN_MOVR . . . . .	136
6.1.2.55 EXTIO_SETUP_MODE_IN_NOP . . . . .	136
6.1.2.56 EXTIO_SETUP_MODE_IN_PWOF . . . . .	136
6.1.2.57 EXTIO_SETUP_MODE_IN_STOP . . . . .	137
6.1.2.58 EXTIO_SETUP_MODE_OUT_ALARM . . . . .	137
6.1.2.59 EXTIO_SETUP_MODE_OUT_BITS . . . . .	137
6.1.2.60 EXTIO_SETUP_MODE_OUT_MOTOR_ON . . . . .	137
6.1.2.61 EXTIO_SETUP_MODE_OUT_MOVING . . . . .	137
6.1.2.62 EXTIO_SETUP_MODE_OUT_OFF . . . . .	137
6.1.2.63 EXTIO_SETUP_MODE_OUT_ON . . . . .	137
6.1.2.64 EXTIO_SETUP_OUTPUT . . . . .	138

6.1.2.65 FEEDBACK_EMF	138
6.1.2.66 FEEDBACK_ENC_ADAPTIVE_HOLDING	138
6.1.2.67 FEEDBACK_ENC_FILTER_BITS	138
6.1.2.68 FEEDBACK_ENC_FILTER_MEDIUM	138
6.1.2.69 FEEDBACK_ENC_FILTER_NONE	138
6.1.2.70 FEEDBACK_ENC_FILTER_STRONG	138
6.1.2.71 FEEDBACK_ENC_FILTER_WEAK	139
6.1.2.72 FEEDBACK_ENC_REVERSE	139
6.1.2.73 FEEDBACK_ENC_TYPE_AUTO	139
6.1.2.74 FEEDBACK_ENC_TYPE_BITS	139
6.1.2.75 FEEDBACK_ENC_TYPE_DIFFERENTIAL	139
6.1.2.76 FEEDBACK_ENC_TYPE_SINGLE_ENDED	139
6.1.2.77 FEEDBACK_ENCODER	139
6.1.2.78 FEEDBACK_ENCODER_MEDIATED	140
6.1.2.79 FEEDBACK_NONE	140
6.1.2.80 H_BRIDGE_ALERT	140
6.1.2.81 HOME_DIR_FIRST	140
6.1.2.82 HOME_DIR_SECOND	140
6.1.2.83 HOME_HALF_MV	140
6.1.2.84 HOME_MV_SEC_EN	141
6.1.2.85 HOME_STOP_FIRST_BITS	141
6.1.2.86 HOME_STOP_FIRST_LIM	141
6.1.2.87 HOME_STOP_FIRST_REV	141
6.1.2.88 HOME_STOP_FIRST_SYN	141
6.1.2.89 HOME_STOP_SECOND_BITS	141
6.1.2.90 HOME_STOP_SECOND_LIM	141
6.1.2.91 HOME_STOP_SECOND_REV	142
6.1.2.92 HOME_STOP_SECOND_SYN	142
6.1.2.93 HOME_USE_FAST	142
6.1.2.94 JOY_REVERSE	142
6.1.2.95 LOW_UPWR_PROTECTION	142
6.1.2.96 LS_SHORTED	142
6.1.2.97 MICROSTEP_MODE_FRAC_128	142
6.1.2.98 MICROSTEP_MODE_FRAC_16	143
6.1.2.99 MICROSTEP_MODE_FRAC_2	143
6.1.2.100 MICROSTEP_MODE_FRAC_256	143
6.1.2.101 MICROSTEP_MODE_FRAC_32	143
6.1.2.102 MICROSTEP_MODE_FRAC_4	143
6.1.2.103 MICROSTEP_MODE_FRAC_64	143
6.1.2.104 MICROSTEP_MODE_FRAC_8	143
6.1.2.105 MICROSTEP_MODE_FULL	144



6.1.2.106 MOVE_STATE_ANTIPLAY	144
6.1.2.107 MOVE_STATE_MOVING	144
6.1.2.108 MOVE_STATE_TARGET_SPEED	144
6.1.2.109 MVCMD_ERROR	144
6.1.2.110 MVCMD_HOME	144
6.1.2.111 MVCMD_LEFT	144
6.1.2.112 MVCMD_LOFT	145
6.1.2.113 MVCMD_MOVE	145
6.1.2.114 MVCMD_MOVR	145
6.1.2.115 MVCMD_NAME_BITS	145
6.1.2.116 MVCMD_RIGHT	145
6.1.2.117 MVCMD_RUNNING	145
6.1.2.118 MVCMD_SSTP	145
6.1.2.119 MVCMD_STOP	146
6.1.2.120 MVCMD_UKNWN	146
6.1.2.121 POWER_OFF_ENABLED	146
6.1.2.122 POWER_REDUCT_ENABLED	146
6.1.2.123 POWER_SMOOTH_CURRENT	146
6.1.2.124 PWR_STATE_MAX	146
6.1.2.125 PWR_STATE_NORM	146
6.1.2.126 PWR_STATE_OFF	147
6.1.2.127 PWR_STATE_REDUCT	147
6.1.2.128 PWR_STATE_UNKNOWN	147
6.1.2.129 REV_SENS_INV	147
6.1.2.130 RPM_DIV_1000	147
6.1.2.131 SETPOS_IGNORE_ENCODER	147
6.1.2.132 SETPOS_IGNORE_POSITION	147
6.1.2.133 STATE_ALARM	148
6.1.2.134 STATE_BORDERS_SWAP_MISSET	148
6.1.2.135 STATE_BRAKE	148
6.1.2.136 STATE_BUTTON_LEFT	148
6.1.2.137 STATE_BUTTON_RIGHT	148
6.1.2.138 STATE_CONTR	148
6.1.2.139 STATE_CONTROLLER_OVERHEAT	148
6.1.2.140 STATE_CTP_ERROR	149
6.1.2.141 STATE_DIG_SIGNAL	149
6.1.2.142 STATE_EEPROM_CONNECTED	149
6.1.2.143 STATE_ENC_A	149
6.1.2.144 STATE_ENC_B	149
6.1.2.145 STATE_ENGINE_RESPONSE_ERROR	149
6.1.2.146 STATE_ERRC	150

6.1.2.147 STATE_ERRD . . . . .	150
6.1.2.148 STATE_ERRV . . . . .	150
6.1.2.149 STATE_EXTIO_ALARM . . . . .	150
6.1.2.150 STATE_GPIO_LEVEL . . . . .	150
6.1.2.151 STATE_GPIO_PINOUT . . . . .	150
6.1.2.152 STATE_IS_HOMED . . . . .	151
6.1.2.153 STATE_LEFT_EDGE . . . . .	151
6.1.2.154 STATE_LOW_USB_VOLTAGE . . . . .	151
6.1.2.155 STATE_OVERLOAD_POWER_CURRENT . . . . .	151
6.1.2.156 STATE_OVERLOAD_POWER_VOLTAGE . . . . .	151
6.1.2.157 STATE_OVERLOAD_USB_CURRENT . . . . .	151
6.1.2.158 STATE_OVERLOAD_USB_VOLTAGE . . . . .	152
6.1.2.159 STATE_POWER_OVERHEAT . . . . .	152
6.1.2.160 STATE_REV_SENSOR . . . . .	152
6.1.2.161 STATE_RIGHT_EDGE . . . . .	152
6.1.2.162 STATE_SECUR . . . . .	152
6.1.2.163 STATE_SYNC_INPUT . . . . .	152
6.1.2.164 STATE_SYNC_OUTPUT . . . . .	152
6.1.2.165 STATE_WINDING_RES_MISMATCH . . . . .	153
6.1.2.166 SYNCIN_ENABLED . . . . .	153
6.1.2.167 SYNCIN_INVERT . . . . .	153
6.1.2.168 SYNCOUT_ENABLED . . . . .	153
6.1.2.169 SYNCOUT_IN_STEPS . . . . .	153
6.1.2.170 SYNCOUT_INVERT . . . . .	153
6.1.2.171 SYNCOUT_ONPERIOD . . . . .	154
6.1.2.172 SYNCOUT_ONSTART . . . . .	154
6.1.2.173 SYNCOUT_ONSTOP . . . . .	154
6.1.2.174 SYNCOUT_STATE . . . . .	154
6.1.2.175 TS_TYPE_BITS . . . . .	154
6.1.2.176 UART_PARITY_BITS . . . . .	154
6.1.2.177 WIND_A_STATE_ABSENT . . . . .	154
6.1.2.178 WIND_A_STATE_MALFUNC . . . . .	155
6.1.2.179 WIND_A_STATE_OK . . . . .	155
6.1.2.180 WIND_A_STATE_UNKNOWN . . . . .	155
6.1.2.181 WIND_B_STATE_ABSENT . . . . .	155
6.1.2.182 WIND_B_STATE_MALFUNC . . . . .	155
6.1.2.183 WIND_B_STATE_OK . . . . .	155
6.1.2.184 WIND_B_STATE_UNKNOWN . . . . .	155
6.1.2.185 XIMC_API . . . . .	156
6.1.2.186 XIMC_CALLCONV . . . . .	156
6.1.2.187 XIMC_RETTYPE . . . . .	156

6.1.3 Типы	156
6.1.3.1 calibration_t	156
6.1.3.2 device_enumeration_t	157
6.1.3.3 device_network_information_t	157
6.1.3.4 logging_callback_t	157
6.1.3.5 result_t	157
6.1.4 Функции	157
6.1.4.1 close_device()	157
6.1.4.2 command_clear_fram()	158
6.1.4.3 command_eeread_settings()	158
6.1.4.4 command_eesave_settings()	158
6.1.4.5 command_home()	159
6.1.4.6 command_homezero()	159
6.1.4.7 command_left()	160
6.1.4.8 command_loft()	160
6.1.4.9 command_move()	160
6.1.4.10 command_move_calb()	161
6.1.4.11 command_movr()	161
6.1.4.12 command_movr_calb()	162
6.1.4.13 command_power_off()	163
6.1.4.14 command_read_robust_settings()	163
6.1.4.15 command_read_settings()	163
6.1.4.16 command_reset()	164
6.1.4.17 command_right()	164
6.1.4.18 command_save_robust_settings()	164
6.1.4.19 command_save_settings()	164
6.1.4.20 command_sstp()	165
6.1.4.21 command_start_measurements()	165
6.1.4.22 command_stop()	165
6.1.4.23 command_update_firmware()	165
6.1.4.24 command_wait_for_stop()	166
6.1.4.25 command_zero()	166
6.1.4.26 enumerate_devices()	167
6.1.4.27 free_enumerate_devices()	169
6.1.4.28 get_accessories_settings()	169
6.1.4.29 get_analog_data()	169
6.1.4.30 get_bootloader_version()	169
6.1.4.31 get_brake_settings()	170
6.1.4.32 get_calibration_settings()	170
6.1.4.33 get_chart_data()	171
6.1.4.34 get_control_settings()	171

6.1.4.35 get_control_settings_calb()	172
6.1.4.36 get_controller_name()	173
6.1.4.37 get_ctp_settings()	173
6.1.4.38 get_debug_read()	173
6.1.4.39 get_device_count()	174
6.1.4.40 get_device_information()	174
6.1.4.41 get_device_name()	174
6.1.4.42 get_edges_settings()	175
6.1.4.43 get_edges_settings_calb()	175
6.1.4.44 get_emf_settings()	176
6.1.4.45 get_encoder_information()	176
6.1.4.46 get_encoder_settings()	176
6.1.4.47 get_engine_advanced_setup()	177
6.1.4.48 get_engine_settings()	177
6.1.4.49 get_engine_settings_calb()	177
6.1.4.50 get_entype_settings()	178
6.1.4.51 get_enumerate_device_controller_name()	178
6.1.4.52 get_enumerate_device_information()	178
6.1.4.53 get_enumerate_device_network_information()	179
6.1.4.54 get_enumerate_device_serial()	179
6.1.4.55 get_enumerate_device_stage_name()	179
6.1.4.56 get_extended_settings()	180
6.1.4.57 get_extio_settings()	180
6.1.4.58 get_feedback_settings()	180
6.1.4.59 get_firmware_version()	181
6.1.4.60 get_gear_information()	181
6.1.4.61 get_gear_settings()	181
6.1.4.62 get_globally_unique_identifier()	182
6.1.4.63 get_hallsensor_information()	182
6.1.4.64 get_hallsensor_settings()	182
6.1.4.65 get_home_settings()	183
6.1.4.66 get_home_settings_calb()	183
6.1.4.67 get_init_random()	183
6.1.4.68 get_joystick_settings()	184
6.1.4.69 get_measurements()	184
6.1.4.70 get_motor_information()	185
6.1.4.71 get_motor_settings()	185
6.1.4.72 get_move_settings()	185
6.1.4.73 get_move_settings_calb()	185
6.1.4.74 get_network_settings()	186
6.1.4.75 get_nonvolatile_memory()	186

6.1.4.76 <code>get_password_settings()</code> . . . . .	186
6.1.4.77 <code>get_pid_settings()</code> . . . . .	187
6.1.4.78 <code>get_position()</code> . . . . .	187
6.1.4.79 <code>get_position_calb()</code> . . . . .	187
6.1.4.80 <code>get_power_settings()</code> . . . . .	188
6.1.4.81 <code>get_secure_settings()</code> . . . . .	188
6.1.4.82 <code>get_serial_number()</code> . . . . .	188
6.1.4.83 <code>get_stage_information()</code> . . . . .	189
6.1.4.84 <code>get_stage_name()</code> . . . . .	189
6.1.4.85 <code>get_stage_settings()</code> . . . . .	189
6.1.4.86 <code>get_status()</code> . . . . .	189
6.1.4.87 <code>get_status_calb()</code> . . . . .	190
6.1.4.88 <code>get_sync_in_settings()</code> . . . . .	190
6.1.4.89 <code>get_sync_in_settings_calb()</code> . . . . .	191
6.1.4.90 <code>get_sync_out_settings()</code> . . . . .	191
6.1.4.91 <code>get_sync_out_settings_calb()</code> . . . . .	191
6.1.4.92 <code>get_uart_settings()</code> . . . . .	192
6.1.4.93 <code>goto_firmware()</code> . . . . .	192
6.1.4.94 <code>has_firmware()</code> . . . . .	192
6.1.4.95 <code>logging_callback_stderr_narrow()</code> . . . . .	193
6.1.4.96 <code>logging_callback_stderr_wide()</code> . . . . .	193
6.1.4.97 <code>msec_sleep()</code> . . . . .	193
6.1.4.98 <code>open_device()</code> . . . . .	193
6.1.4.99 <code>probe_device()</code> . . . . .	194
6.1.4.100 <code>reset_locks()</code> . . . . .	194
6.1.4.101 <code>service_command_updf()</code> . . . . .	194
6.1.4.102 <code>set_accessories_settings()</code> . . . . .	194
6.1.4.103 <code>set_brake_settings()</code> . . . . .	195
6.1.4.104 <code>set_calibration_settings()</code> . . . . .	195
6.1.4.105 <code>set_control_settings()</code> . . . . .	196
6.1.4.106 <code>set_control_settings_calb()</code> . . . . .	196
6.1.4.107 <code>set_controller_name()</code> . . . . .	196
6.1.4.108 <code>set_correction_table()</code> . . . . .	197
6.1.4.109 <code>set_ctp_settings()</code> . . . . .	197
6.1.4.110 <code>set_debug_write()</code> . . . . .	198
6.1.4.111 <code>set_edges_settings()</code> . . . . .	198
6.1.4.112 <code>set_edges_settings_calb()</code> . . . . .	199
6.1.4.113 <code>set_emf_settings()</code> . . . . .	199
6.1.4.114 <code>set_encoder_information()</code> . . . . .	199
6.1.4.115 <code>set_encoder_settings()</code> . . . . .	200
6.1.4.116 <code>set_engine_advanced_setup()</code> . . . . .	200

6.1.4.117	set_engine_settings()	200
6.1.4.118	set_engine_settings_calb()	201
6.1.4.119	set_entype_settings()	201
6.1.4.120	set_extended_settings()	201
6.1.4.121	set_extio_settings()	202
6.1.4.122	set_feedback_settings()	202
6.1.4.123	set_gear_information()	203
6.1.4.124	set_gear_settings()	203
6.1.4.125	set_hallsensor_information()	203
6.1.4.126	set_hallsensor_settings()	203
6.1.4.127	set_home_settings()	204
6.1.4.128	set_home_settings_calb()	204
6.1.4.129	set_joystick_settings()	205
6.1.4.130	set_logging_callback()	205
6.1.4.131	set_motor_information()	205
6.1.4.132	set_motor_settings()	206
6.1.4.133	set_move_settings()	206
6.1.4.134	set_move_settings_calb()	206
6.1.4.135	set_network_settings()	207
6.1.4.136	set_nonvolatile_memory()	207
6.1.4.137	set_password_settings()	207
6.1.4.138	set_pid_settings()	208
6.1.4.139	set_position()	208
6.1.4.140	set_position_calb()	208
6.1.4.141	set_power_settings()	209
6.1.4.142	set_secure_settings()	209
6.1.4.143	set_serial_number()	209
6.1.4.144	set_stage_information()	210
6.1.4.145	set_stage_name()	210
6.1.4.146	set_stage_settings()	210
6.1.4.147	set_sync_in_settings()	211
6.1.4.148	set_sync_in_settings_calb()	211
6.1.4.149	set_sync_out_settings()	211
6.1.4.150	set_sync_out_settings_calb()	212
6.1.4.151	set_uart_settings()	212
6.1.4.152	write_key()	212
6.1.4.153	ximc_version()	213
6.2	ximc.h	213

# Глава 1

## Библиотека libximc

Документация для библиотеки libximc.

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

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

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

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

```
pip install libximc
```

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

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

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

### 1.2 Что умеет библиотека libximc

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

#### Предупреждения

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

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

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

## 1.3 Содействие

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



# Глава 2

## Введение

### 2.1 О библиотеке

Этот документ содержит всю информацию о библиотеке libximc, за исключением инструкций по сборке, которые вы можете найти в файле README.md в корневом каталоге репозитория GitHub: <https://github.com/Standa-Optomechanics/libximc>. Библиотека libximc использует распространенный и проверенный интерфейс виртуального последовательного порта, поэтому вы можете работать с модулями управления моторами через эту библиотеку практически под всеми ОС: Windows, Linux, macOS для Intel и Apple Silicon (с использованием Rosetta 2), в том числе с 64-битными версиями. Библиотека поддерживает подключение и отключение устройств "на лету".

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

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

- macOS 10.15 или новее
- Windows 7 или новее
- Linux на основе Debian

## Глава 3

# Как использовать с...

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

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

### Заметки

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

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

```
sudo dpkg -i <имя_пакета>.deb
```

### 3.0.1 Visual C++

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

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

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

### 3.0.2 CodeBlocks

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

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

### 3.0.3 MinGW

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

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

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

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

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

### 3.0.4 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 -DDEBUG -D_WINDOWS testapp.c libximc.lib
```

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

Также существует [пример использования библиотеки libximc](#) в проекте C++ Builder, **но он не поддерживается.**

### 3.0.5 XCode

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

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

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

### 3.0.6 GCC

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

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

testapp может быть собран следующим образом с установленной библиотекой:

```
make
```

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

Затем запустите приложение с помощью:

```
make run
```

Примечание: make run на macOS копирует библиотеку в текущую директорию. Если вы хотите использовать библиотеку из другой директории, пожалуйста укажите в LD\_LIBRARY\_PATH или DYLD\_LIBRARY\_PATH путь к директории с библиотекой.

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

### 3.0.7 .NET

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

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

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

### 3.0.8 Java

Как запустить пример на Linux. Перейдите в `examples/test_Java/compiled-winX/` и выполните

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

Как запустить пример на Windows. Перейдите в `examples/test_Java/compiled-winX/`. Запустите:

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

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

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

Затем пересоберите исходные тексты:

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

или для Windows или macOS:

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

Затем соберите jar:

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

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

### 3.0.9 Python

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

Для запуска:

```
python xxxx.py
```

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

### 3.0.10 MATLAB

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

Перед запуском:

На macOS: скопируйте `ximc/macosx/libximc.framework`, `ximc/macosx/wrappers/ximcm.h`, `ximc/ximc.h` в директорию `examples/test_MATLAB`. Установите XCode, совместимый с Matlab

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

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

На Windows: перед запуском ничего делать не нужно

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

```
testximc
```

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

## 3.1 Логирование в файл

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

## 3.2 Требуемые права доступа

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

## 3.3 Си-профили

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

## 3.4 Python-профили

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

Пример использования python-профилей вы можете посмотреть в директории примеров `"examples/test_Python/profiletest/testpythonprofile.py"`.

## Глава 4

# Работа с пользовательскими единицами

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

- Структура пересчета единиц [calibration\\_t](#)
- Функции дублиеры для работы с пользовательскими единицами и структуры данных для них
- Таблица коррекции координат для более точного позиционирования

### 4.1 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)
```

### 4.2 Функции дублиеры для работы с пользовательскими единицами и структуры данных для них

Структуры и функции для работы с пользовательскими единицами имеют постфикс `_calb`. Пользователь используя данные функции может выполнять все действия в собственных единицах не беспокоясь о том, что и как считает контроллер. 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.

## 4.3 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 `set_correction_table()`. В ее описании описаны функции и их данные поддерживающие коррекцию движения.

### Заметки

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

### Формат файла:

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

### Пример файла:

X	dX
0	0
5.0	0.005
10.0	-0.01

## Глава 5

# Структуры данных

### 5.1 Структура `accessories_settings_t`

Информация о дополнительных аксессуарах.

```
#include <ximc.h>
```

#### 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 [TSMaх](#)  
*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.*



### 5.1.1 Подробное описание

Информация о дополнительных аксессуарах.

См. также

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

### 5.1.2 Поля

#### 5.1.2.1 LimitSwitchesSettings

`unsigned int LimitSwitchesSettings`

[Флаги настроек температурного датчика.](#)

#### 5.1.2.2 MagneticBrakeInfo

`char MagneticBrakeInfo[25]`

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

#### 5.1.2.3 MBRatedCurrent

`float MBRatedCurrent`

Номинальный ток для управления магнитным тормозом (А).

Тип данных: float.

#### 5.1.2.4 MBRatedVoltage

`float MBRatedVoltage`

Номинальное напряжение для управления магнитным тормозом (В).

Тип данных: float.

#### 5.1.2.5 MBSettings

`unsigned int MBSettings`

[Флаги настроек энкодера.](#)

#### 5.1.2.6 MBTorque

`float MBTorque`

Удерживающий момент (мН \* м).

Тип данных: `float`.

#### 5.1.2.7 TemperatureSensorInfo

`char TemperatureSensorInfo[25]`

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

#### 5.1.2.8 TSGrad

`float TSGrad`

Температурный градиент (В/град Цельсия).

Тип данных: `float`.

#### 5.1.2.9 TSMax

`float TSMax`

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

#### 5.1.2.10 TSMin

`float TSMin`

Минимальная измеряемая температура (град Цельсия).

Тип данных: `float`.

#### 5.1.2.11 TSSettings

`unsigned int TSSettings`

Флаги настроек температурного датчика.

## 5.2 Структура `analog_data_t`

Аналоговые данные.

```
#include <ximc.h>
```

## Поля данных

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

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

- unsigned int **deprecated1** [2]
- int **R**

*Сопротивление обмоток двигателя(для шагового двигателя), в мОм*

- int **L**

*Псевдоиндуктивность обмоток двигателя(для шагового двигателя), в мкГн*

### 5.2.1 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](#)

### 5.2.2 Поля

#### 5.2.2.1 A1Voltage

int A1Voltage

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

#### 5.2.2.2 A1Voltage\_ADC

unsigned int A1Voltage\_ADC

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

#### 5.2.2.3 A2Voltage

int A2Voltage

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

#### 5.2.2.4 A2Voltage\_ADC

unsigned int A2Voltage\_ADC

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

#### 5.2.2.5 ACurrent

```
int ACurrent
```

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

#### 5.2.2.6 ACurrent\_ADC

```
unsigned int ACurrent_ADC
```

"Ток через обмотку A" необработанные данные с АЦП.

#### 5.2.2.7 B1Voltage

```
int B1Voltage
```

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

#### 5.2.2.8 B1Voltage\_ADC

```
unsigned int B1Voltage_ADC
```

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

#### 5.2.2.9 B2Voltage

```
int B2Voltage
```

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

#### 5.2.2.10 B2Voltage\_ADC

```
unsigned int B2Voltage_ADC
```

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

#### 5.2.2.11 BCurrent

```
int BCurrent
```

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

#### 5.2.2.12 `BCurrent_ADC`

```
unsigned int BCurrent_ADC
```

"Ток через обмотку В" необработанные данные с АЦП.

#### 5.2.2.13 `Enc_Check`

```
int Enc_Check
```

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

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

#### 5.2.2.14 `Enc_Check_ADC`

```
unsigned int Enc_Check_ADC
```

Напряжение на линии проверки типа энкодера, необработанные данные АЦП.

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

#### 5.2.2.15 `FullCurrent`

```
int FullCurrent
```

"Полный ток" откалиброванные данные (в мА).

#### 5.2.2.16 `FullCurrent_ADC`

```
unsigned int FullCurrent_ADC
```

"Полный ток" необработанные данные с АЦП.

#### 5.2.2.17 `Joy`

```
int Joy
```

Джойстик во внутренних единицах.

Диапазон: 0..10000

#### 5.2.2.18 `Joy_ADC`

```
unsigned int Joy_ADC
```

Джойстик, необработанные данные с АЦП.

#### 5.2.2.19 Pot

`int Pot`

Аналоговый вход во внутренних единицах.

Диапазон: 0..10000

#### 5.2.2.20 SupVoltage

`int SupVoltage`

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

#### 5.2.2.21 SupVoltage\_ADC

`unsigned int SupVoltage_ADC`

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

#### 5.2.2.22 Temp

`int Temp`

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

#### 5.2.2.23 Temp\_ADC

`unsigned int Temp_ADC`

Напряжение с датчика температуры, необработанные данные с АЦП.

## 5.3 Структура `brake_settings_t`

Настройки тормоза.

```
#include <ximc.h>
```

### 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.*

### 5.3.1 Подробное описание

Настройки тормоза.

Эта структура содержит параметры управления тормозом.

См. также

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

### 5.3.2 Поля

#### 5.3.2.1 BrakeFlags

`unsigned int BrakeFlags`

[Флаги настроек тормоза.](#)

#### 5.3.2.2 t1

`unsigned int t1`

Время в мс между включением питания мотора и отключением тормоза.

#### 5.3.2.3 t2

`unsigned int t2`

Время в мс между отключением тормоза и готовностью к движению.

Все команды движения начинают выполняться только по истечении этого времени.

#### 5.3.2.4 t3

`unsigned int t3`

Время в мс между остановкой мотора и включением тормоза.

#### 5.3.2.5 t4

`unsigned int t4`

Время в мс между включением тормоза и отключением питания мотора.



## 5.4 Структура `calibration_settings_t`

Калибровочные коэффициенты.

```
#include <ximc.h>
```

### 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.*

### 5.4.1 Подробное описание

Калибровочные коэффициенты.

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

См. также

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

### 5.4.2 Поля

#### 5.4.2.1 `CSS1_A`

```
float CSS1_A
```

Коэффициент масштабирования для аналоговых измерений тока в обмотке A.

#### 5.4.2.2 CSS1\_B

```
float CSS1_B
```

Коэффициент сдвига для аналоговых измерений тока в обмотке A.

#### 5.4.2.3 CSS2\_A

```
float CSS2_A
```

Коэффициент масштабирования для аналоговых измерений тока в обмотке B.

#### 5.4.2.4 CSS2\_B

```
float CSS2_B
```

Коэффициент сдвига для аналоговых измерений тока в обмотке B.

#### 5.4.2.5 FullCurrent\_A

```
float FullCurrent_A
```

Коэффициент масштабирования для аналоговых измерений полного тока.

#### 5.4.2.6 FullCurrent\_B

```
float FullCurrent_B
```

Коэффициент сдвига для аналоговых измерений полного тока.

## 5.5 Структура `calibration_t`

Структура калибровок

```
#include <ximc.h>
```

### Поля данных

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

### 5.5.1 Подробное описание

Структура калибровок

Где найти все значения для расчета?

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

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

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

Ускорение, замедление и скорость в режиме антилюфта считаются аналогично

### 5.5.2 Поля

#### 5.5.2.1 A

double A

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

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

- Для шагового двигателя без энкодера используется только один коэффициент  $A$ . Формула пересчёта:  $[user\_unit/steps]$ . Пример: 800 шагов = 1 мм, тогда  $A = 1/800 = 0.00125$
- При использовании шагового двигателя с энкодером или двигателей DC/BLDC позиция задаётся в counts, но скорость, ускорение/замедление и скорость в режиме антилюфта задаются в RPM, поэтому требуются два коэффициента:
  - А. Формула пересчёта для позиции:  $[user\_unit/counts]$ . Пример: 16000 counts = 1 мм, тогда  $A = 1/16000 = 0.0000625$
  - В. Формула пересчёта для скорости, ускорения/замедления и скорости в режиме антилюфта:  $[60/ENCODER\_COUNTS\_PER\_TURN * A]$ . Пример:  $60 / 4000 * 0.0000625 = 0.000009375$

### 5.5.2.2 MicrostepMode

`unsigned int MicrostepMode`

Настройка контроллера, определяющая режим пошагового деления.

## 5.6 Структура `chart_data_t`

Дополнительное состояние устройства.

```
#include <ximc.h>
```

### 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 десятичных долях.*
- `int AveragedPowerRatio`  
*Отношение подаваемой на мотор мощности к номинальной мощности, в процентах.*

### 5.6.1 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](#)

### 5.6.2 Поля

#### 5.6.2.1 AveragedPowerRatio

```
int AveragedPowerRatio
```

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

#### 5.6.2.2 Joy

```
unsigned int Joy
```

Положение джойстика в десяти тысячных долях.

Диапазон: 0..10000

#### 5.6.2.3 Pot

```
unsigned int Pot
```

Значение на аналоговом входе.

Диапазон: 0..10000

#### 5.6.2.4 WindingCurrentA

```
int WindingCurrentA
```

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

#### 5.6.2.5 WindingCurrentB

```
int WindingCurrentB
```

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

#### 5.6.2.6 WindingCurrentC

```
int WindingCurrentC
```

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

#### 5.6.2.7 WindingVoltageA

```
int WindingVoltageA
```

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

#### 5.6.2.8 WindingVoltageB

```
int WindingVoltageB
```

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

#### 5.6.2.9 WindingVoltageC

```
int WindingVoltageC
```

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

### 5.7 Структура control\_settings\_calb\_t

Настройки управления с использованием пользовательских единиц.

```
#include <ximc.h>
```

## 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*

### 5.7.1 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
```

### 5.7.2 Поля

#### 5.7.2.1 Flags

```
unsigned int Flags
```

Флаги управления.

#### 5.7.2.2 MaxClickTime

```
unsigned int MaxClickTime
```

Максимальное время клика (в мс).

До истечения этого времени первая скорость не включается.

### 5.7.2.3 MaxSpeed

```
float MaxSpeed[10]
```

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

### 5.7.2.4 Timeout

```
unsigned int Timeout[9]
```

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

## 5.8 Структура `control_settings_t`

Настройки управления.

```
#include <ximc.h>
```

### 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.*

### 5.8.1 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
```



## 5.8.2 Поля

### 5.8.2.1 Flags

unsigned int Flags

Флаги управления.

### 5.8.2.2 MaxClickTime

unsigned int MaxClickTime

Максимальное время клика (в мс).

До истечения этого времени первая скорость не включается.

### 5.8.2.3 MaxSpeed

unsigned int MaxSpeed[10]

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

Диапазон: 0..100000.

### 5.8.2.4 Timeout

unsigned int Timeout[9]

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

### 5.8.2.5 uDeltaPosition

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).

### 5.8.2.6 uMaxSpeed

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).

## 5.9 Структура `controller_name_t`

Пользовательское имя контроллера и флаги настройки.

```
#include <ximc.h>
```

### Поля данных

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

### 5.9.1 Подробное описание

Пользовательское имя контроллера и флаги настройки.

См. также

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

### 5.9.2 Поля

#### 5.9.2.1 `ControllerName`

```
char ControllerName[17]
```

Пользовательское имя контроллера.

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

#### 5.9.2.2 `CtrlFlags`

```
unsigned int CtrlFlags
```

[Флаги настроек контроллера.](#)

## 5.10 Структура `ctp_settings_t`

Настройки контроля позиции(для шагового двигателя).

```
#include <ximc.h>
```

## Поля данных

- unsigned int `CTPMinError`

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

- unsigned int `CTPFlags`

Флаги контроля позиции.

### 5.10.1 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](#)

### 5.10.2 Поля

#### 5.10.2.1 CTPFlags

unsigned int `CTPFlags`

Флаги контроля позиции.

#### 5.10.2.2 CTPMinError

unsigned int `CTPMinError`

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

Измеряется в шагах ШД.

## 5.11 Структура `debug_read_t`

Отладочные данные.

```
#include <ximc.h>
```

### Поля данных

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

*Отладочные данные.*

### 5.11.1 Подробное описание

Отладочные данные.

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

См. также

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

### 5.11.2 Поля

#### 5.11.2.1 `DebugData`

```
uint8_t DebugData[128]
```

Отладочные данные.

## 5.12 Структура `debug_write_t`

Отладочные данные.

```
#include <ximc.h>
```

### Поля данных

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

*Отладочные данные.*

### 5.12.1 Подробное описание

Отладочные данные.

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

См. также

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

### 5.12.2 Поля

#### 5.12.2.1 DebugData

`uint8_t DebugData[128]`

Отладочные данные.

## 5.13 Структура `device_information_t`

Информации о контроллере.

```
#include <ximc.h>
```

### 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.*

### 5.13.1 Подробное описание

Информации о контроллере.

См. также

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

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

### 5.13.2 Поля

#### 5.13.2.1 Major

`unsigned int Major`

Основной номер версии железа.

#### 5.13.2.2 Minor

`unsigned int Minor`

Второстепенный номер версии железа.

#### 5.13.2.3 Release

`unsigned int Release`

Номер правок этой версии железа.

## 5.14 Структура `device_network_information_t`

Структура данных с информацией о сетевом устройстве.

```
#include <ximc.h>
```

### Поля данных

- `uint32_t ipv4`  
*IPv4-адрес, передаваемый в сетевом байтовом порядке (big-endian byte order)*
- `char nodename [16]`  
*имя узла Bindu, на котором размещено устройство*
- `uint32_t axis_state`  
*флаги, представляющие состояние устройства*
- `char locker_username [16]`  
*имя пользователя, заблокировавшего устройство (если таковое имеется)*
- `char locker_nodename [16]`  
*имя узла Bindu, которое использовалось для блокировки устройства (если таковое имеется)*
- `time_t locked_time`  
*время, в которое была получена блокировка (UTC, микросекунды с момента начала эпохи)*

### 5.14.1 Подробное описание

Структура данных с информацией о сетевом устройстве.

## 5.15 Структура `edges_settings_calb_t`

Настройки границ с использованием пользовательских единиц.

```
#include <ximc.h>
```

### 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`.*

### 5.15.1 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
```

### 5.15.2 Поля

#### 5.15.2.1 `BorderFlags`

```
unsigned int BorderFlags
```

*Флаги границ.*

#### 5.15.2.2 `EnderFlags`

```
unsigned int EnderFlags
```

*Флаги концевых выключателей.*

### 5.15.2.3 LeftBorder

`float LeftBorder`

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

Корректируется таблицей.

### 5.15.2.4 RightBorder

`float RightBorder`

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

Корректируется таблицей.

## 5.16 Структура `edges_settings_t`

Настройки границ.

```
#include <ximc.h>
```

### 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).*

### 5.16.1 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](#)



## 5.16.2 Поля

### 5.16.2.1 BorderFlags

```
unsigned int BorderFlags
```

Флаги границ.

### 5.16.2.2 EnderFlags

```
unsigned int EnderFlags
```

Флаги концевых выключателей.

### 5.16.2.3 LeftBorder

```
int LeftBorder
```

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

### 5.16.2.4 RightBorder

```
int RightBorder
```

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

### 5.16.2.5 uLeftBorder

```
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`).

### 5.16.2.6 uRightBorder

```
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`).

## 5.17 Структура `emf_settings_t`

Настройки EMF.

```
#include <ximc.h>
```

### 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.*

### 5.17.1 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
```

### 5.17.2 Поля

#### 5.17.2.1 `BackEMFFlags`

```
unsigned int BackEMFFlags
```

[Флаги автоопределения характеристик обмоток двигателя.](#)

#### 5.17.2.2 `Km`

```
float Km
```

Электромеханический коэффициент двигателя.

### 5.17.2.3 L

`float L`

Индуктивность обмоток двигателя.

### 5.17.2.4 R

`float R`

Сопротивление обмоток двигателя.

## 5.18 Структура `encoder_information_t`

Информация об энкодере.

```
#include <ximc.h>
```

### Поля данных

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

### 5.18.1 Подробное описание

Информация об энкодере.

См. также

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

### 5.18.2 Поля

#### 5.18.2.1 Manufacturer

`char Manufacturer[17]`

Производитель.

Максимальная длина строки: 16 символов.

### 5.18.2.2 PartNumber

```
char PartNumber[25]
```

Серия и номер модели.

Максимальная длина строки: 24 символа.

## 5.19 Структура `encoder_settings_t`

Настройки энкодера.

```
#include <ximc.h>
```

### 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.*

### 5.19.1 Подробное описание

Настройки энкодера.

См. также

```
set_encoder_settings  
get_encoder_settings  
get_encoder_settings, set_encoder_settings
```

### 5.19.2 Поля

#### 5.19.2.1 EncoderSettings

```
unsigned int EncoderSettings
```

[Флаги настроек энкодера.](#)

### 5.19.2.2 MaxCurrentConsumption

float MaxCurrentConsumption

Максимальное потребление тока (мА).

Тип данных: float.

### 5.19.2.3 MaxOperatingFrequency

float MaxOperatingFrequency

Максимальная частота (кГц).

Тип данных: float.

### 5.19.2.4 SupplyVoltageMax

float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

### 5.19.2.5 SupplyVoltageMin

float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

## 5.20 Структура engine\_advanced\_setup\_t

Настройки EAS.

```
#include <ximc.h>
```

### Поля данных

- unsigned int [stepcloseloop\\_Kw](#)  
*Только для производителя.*
- unsigned int [stepcloseloop\\_Kp\\_low](#)  
*Только для производителя.*
- unsigned int [stepcloseloop\\_Kp\\_high](#)  
*Только для производителя.*

### 5.20.1 Подробное описание

Настройки EAS.

Только для производителя. 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 контроллера.

См. также

```
set_engine_advanced_setup  
get_engine_advanced_setup  
get_engine_advanced_setup, set_engine_advanced_setup
```

### 5.20.2 Поля

#### 5.20.2.1 stepcloseloop\_Kp\_high

```
unsigned int stepcloseloop_Kp_high
```

Только для производителя.

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

#### 5.20.2.2 stepcloseloop\_Kp\_low

```
unsigned int stepcloseloop_Kp_low
```

Только для производителя.

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

#### 5.20.2.3 stepcloseloop\_Kw

```
unsigned int stepcloseloop_Kw
```

Только для производителя.

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

## 5.21 Структура engine\_settings\_calb\_t

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

```
#include <ximc.h>
```

### 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).*

#### 5.21.1 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](#)

#### 5.21.2 Поля

##### 5.21.2.1 Antiplay

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`.

##### 5.21.2.2 EngineFlags

unsigned int EngineFlags

[Флаги параметров мотора.](#)

### 5.21.2.3 MicrostepMode

```
unsigned int MicrostepMode
```

Флаги параметров микрошагового режима.

### 5.21.2.4 NomCurrent

```
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

### 5.21.2.5 NomSpeed

```
float NomSpeed
```

Номинальная скорость.

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

### 5.21.2.6 NomVoltage

```
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).

### 5.21.2.7 StepsPerRev

```
unsigned int StepsPerRev
```

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

Диапазон: 1..65535.

## 5.22 Структура engine\_settings\_t

Ограничения и настройки движения, связанные с двигателем.

```
#include <ximc.h>
```



### 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).*

#### 5.22.1 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](#)

#### 5.22.2 Поля

##### 5.22.2.1 Antiplay

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`.

### 5.22.2.2 EngineFlags

unsigned int EngineFlags

Флаги параметров мотора.

### 5.22.2.3 MicrostepMode

unsigned int MicrostepMode

Флаги параметров микрошагового режима.

### 5.22.2.4 NomCurrent

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

### 5.22.2.5 NomSpeed

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.

### 5.22.2.6 NomVoltage

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).

### 5.22.2.7 StepsPerRev

unsigned int StepsPerRev

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

Диапазон: 1..65535.

### 5.22.2.8 uNomSpeed

`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`).

## 5.23 Структура `entype_settings_t`

Настройки типа мотора и типа силового драйвера.

```
#include <ximc.h>
```

### Поля данных

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

### 5.23.1 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](#)

### 5.23.2 Поля

#### 5.23.2.1 DriverType

`unsigned int DriverType`

*Флаги, определяющие тип силового драйвера.*

### 5.23.2.2 EngineType

`unsigned int EngineType`

Флаги, определяющие тип мотора.

## 5.24 Структура `extended_settings_t`

Настройки EST.

```
#include <ximc.h>
```

### Поля данных

- `unsigned int Param1`

### 5.24.1 Подробное описание

Настройки EST.

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

См. также

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

## 5.25 Структура `extio_settings_t`

Настройки EXTIO.

```
#include <ximc.h>
```

### Поля данных

- `unsigned int EXTIOSetupFlags`  
*Флаги настройки работы внешнего ввода/вывода.*
- `unsigned int EXTIOModeFlags`  
*Флаги настройки режимов внешнего ввода/вывода.*

### 5.25.1 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](#)

### 5.25.2 Поля

#### 5.25.2.1 EXTIOModeFlags

`unsigned int EXTIOModeFlags`

[Флаги настройки режимов внешнего ввода/вывода.](#)

#### 5.25.2.2 EXTIOSetupFlags

`unsigned int EXTIOSetupFlags`

[Флаги настройки работы внешнего ввода/вывода.](#)

## 5.26 Структура `feedback_settings_t`

Настройки обратной связи.

```
#include <ximc.h>
```

### 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.*

### 5.26.1 Подробное описание

Настройки обратной связи.

Эта структура содержит настройки обратной связи.

См. также

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

### 5.26.2 Поля

#### 5.26.2.1 CountsPerTurn

```
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`.

#### 5.26.2.2 FeedbackFlags

```
unsigned int FeedbackFlags
```

[Флаги обратной связи](#).

#### 5.26.2.3 FeedbackType

```
unsigned int FeedbackType
```

[Тип обратной связи](#).

#### 5.26.2.4 IPS

```
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.

## 5.27 Структура `gear_information_t`

Информация о редукторе.

```
#include <ximc.h>
```

### Поля данных

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

### 5.27.1 Подробное описание

Информация о редукторе.

См. также

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

### 5.27.2 Поля

#### 5.27.2.1 Manufacturer

char [Manufacturer](#)[17]

Производитель.

Максимальная длина строки: 16 символов.

#### 5.27.2.2 PartNumber

char [PartNumber](#)[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

## 5.28 Структура gear\_settings\_t

Настройки редуктора.

```
#include <ximc.h>
```

### Поля данных

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

### 5.28.1 Подробное описание

Настройки редуктора.

См. также

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

### 5.28.2 Поля

#### 5.28.2.1 Efficiency

float Efficiency

КПД редуктора (%).

Тип данных: float.

#### 5.28.2.2 InputInertia

float InputInertia

Эквивалентная входная инерция редуктора( $г * см^2$ ).

Тип данных: float.



### 5.28.2.3 MaxOutputBacklash

`float MaxOutputBacklash`

Выходной люфт редуктора (градус).

Тип данных: `float`.

### 5.28.2.4 RatedInputSpeed

`float RatedInputSpeed`

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

Тип данных: `float`.

### 5.28.2.5 RatedInputTorque

`float RatedInputTorque`

Максимальный крутящий момент (Н \* м).

Тип данных: `float`.

### 5.28.2.6 ReductionIn

`float ReductionIn`

Входной коэффициент редуктора.

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

### 5.28.2.7 ReductionOut

`float ReductionOut`

Выходной коэффициент редуктора.

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

## 5.29 Структура `get_position_calb_t`

Данные о позиции.

```
#include <ximc.h>
```

### Поля данных

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

## 5.29.1 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](#)

## 5.29.2 Поля

### 5.29.2.1 EncPosition

long\_t EncPosition

Позиция энкодера.

### 5.29.2.2 Position

float Position

Позиция двигателя.

Корректируется таблицей.

## 5.30 Структура `get_position_t`

Данные о позиции.

```
#include <ximc.h>
```

### Поля данных

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

### 5.30.1 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](#)

### 5.30.2 Поля

#### 5.30.2.1 EncPosition

```
long_t EncPosition
```

Позиция энкодера.

#### 5.30.2.2 uPosition

```
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`).

## 5.31 Структура `globally_unique_identifier_t`

Глобальный уникальный идентификатор.

```
#include <ximc.h>
```

### 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.

### 5.31.1 Подробное описание

Глобальный уникальный идентификатор.

Только для производителя.

См. также

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

### 5.31.2 Поля

#### 5.31.2.1 UniqueID0

`unsigned int UniqueID0`

Уникальный ID 0.

#### 5.31.2.2 UniqueID1

`unsigned int UniqueID1`

Уникальный ID 1.

#### 5.31.2.3 UniqueID2

`unsigned int UniqueID2`

Уникальный ID 2.

#### 5.31.2.4 UniqueID3

`unsigned int UniqueID3`

Уникальный ID 3.

## 5.32 Структура `hallsensor_information_t`

Информация о датчиках Холла.

```
#include <ximc.h>
```

### Поля данных

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

#### 5.32.1 Подробное описание

Информация о датчиках Холла.

См. также

`set_hallsensor_information`  
`get_hallsensor_information`  
`get_hallsensor_information`, `set_hallsensor_information`

#### 5.32.2 Поля

##### 5.32.2.1 `Manufacturer`

`char Manufacturer`[17]

Производитель.

Максимальная длина строки: 16 символов.

##### 5.32.2.2 `PartNumber`

`char PartNumber`[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

### 5.33 Структура `hallsensor_settings_t`

Настройки датчиков Холла.

```
#include <ximc.h>
```

### 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*

### 5.33.1 Подробное описание

Настройки датчиков Холла.

См. также

```
set_hallsensor_settings  
get_hallsensor_settings  
get_hallsensor_settings, set_hallsensor_settings
```

### 5.33.2 Поля

#### 5.33.2.1 `MaxCurrentConsumption`

```
float MaxCurrentConsumption
```

Максимальное потребление тока (мА).

Тип данных: float.

#### 5.33.2.2 `MaxOperatingFrequency`

```
float MaxOperatingFrequency
```

Максимальная частота (кГц).

Тип данных: float.

#### 5.33.2.3 `SupplyVoltageMax`

```
float SupplyVoltageMax
```

Максимальное напряжение питания (В).

Тип данных: float.

#### 5.33.2.4 SupplyVoltageMin

`float SupplyVoltageMin`

Минимальное напряжение питания (В).

Тип данных: `float`.

## 5.34 Структура `home_settings_calb_t`

Настройки калибровки позиции с использованием пользовательских единиц.

```
#include <ximc.h>
```

### 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.*

### 5.34.1 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`

### 5.34.2 Поля

#### 5.34.2.1 FastHome

`float FastHome`

Скорость первого движения.

### 5.34.2.2 HomeDelta

`float HomeDelta`

Расстояние отхода от точки останова.

### 5.34.2.3 HomeFlags

`unsigned int HomeFlags`

Флаги настроек команды `home`.

### 5.34.2.4 SlowHome

`float SlowHome`

Скорость второго движения.

## 5.35 Структура `home_settings_t`

Настройки калибровки позиции.

```
#include <ximc.h>
```

### 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`.*



### 5.35.1 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](#)

### 5.35.2 Поля

#### 5.35.2.1 FastHome

```
unsigned int FastHome
```

Скорость первого движения (в полных шагах).

Диапазон: 0..100000

#### 5.35.2.2 HomeDelta

```
int HomeDelta
```

Расстояние отхода от точки останова (в полных шагах).

#### 5.35.2.3 HomeFlags

```
unsigned int HomeFlags
```

Флаги настроек команды `home`.

#### 5.35.2.4 SlowHome

```
unsigned int SlowHome
```

Скорость второго движения (в полных шагах).

Диапазон: 0..100000.

### 5.35.2.5 uFastHome

`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`).

### 5.35.2.6 uHomeDelta

`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`).

### 5.35.2.7 uSlowHome

`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`).

## 5.36 Структура `init_random_t`

Случайный ключ.

```
#include <ximc.h>
```

### Поля данных

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

### 5.36.1 Подробное описание

Случайный ключ.

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

См. также

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

## 5.36.2 Поля

### 5.36.2.1 key

```
uint8_t key[16]
```

Случайный ключ.

## 5.37 Структура joystick\_settings\_t

Настройки джойстика.

```
#include <ximc.h>
```

### 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.*

### 5.37.1 Подробное описание

Настройки джойстика.

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

См. также

```
set_joystick_settings  
get_joystick_settings  
get_joystick_settings, set_joystick_settings
```

## 5.37.2 Поля

### 5.37.2.1 DeadZone

`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.

### 5.37.2.2 ExpFactor

`unsigned int ExpFactor`

Фактор экспоненциальной нелинейности отклика джойстика.

### 5.37.2.3 JoyCenter

`unsigned int JoyCenter`

Значение в шагах джойстика, соответствующее неотклонённому устройству.

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

### 5.37.2.4 JoyFlags

`unsigned int JoyFlags`

Флаги джойстика.

### 5.37.2.5 JoyHighEnd

`unsigned int JoyHighEnd`

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

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

### 5.37.2.6 JoyLowEnd

`unsigned int JoyLowEnd`

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

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

## 5.38 Структура `measurements_t`

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

```
#include <ximc.h>
```

### Поля данных

- `int Speed` [25]  
*Последовательность измеренных скоростей (в отсчётах энкодера/сек или микрошагах/сек, в зависимости от типа двигателя и режима управления)*
- `int Error` [25]  
*Последовательность измеренных ошибок позиции (в отсчётах энкодера или микрошагах, в зависимости от типа двигателя и режима управления)*
- `unsigned int Length`  
*Фактическая длина последовательности.*

### 5.38.1 Подробное описание

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

Шаг по времени между двумя последовательными измерениями - 1 мс.

См. также

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

### 5.38.2 Поля

#### 5.38.2.1 `Length`

```
unsigned int Length
```

Фактическая длина последовательности.

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

## 5.39 Структура `motor_information_t`

Информация о двигателе.

```
#include <ximc.h>
```

**Поля данных**

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

**5.39.1 Подробное описание**

Информация о двигателе.

См. также

`set_motor_information`  
`get_motor_information`  
`get_motor_information`, `set_motor_information`

**5.39.2 Поля****5.39.2.1 Manufacturer**

`char Manufacturer`[17]

Производитель.

Максимальная длина строки: 16 символов.

**5.39.2.2 PartNumber**

`char PartNumber`[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

**5.40 Структура `motor_settings_t`**

Физический характеристики и ограничения мотора.

```
#include <ximc.h>
```

## 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 ( $\text{мН} \cdot \text{м}$ ).*
- 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 ( $\text{г} \cdot \text{см}^2$ ).*
- float `StallTorque`  
*Крутящий момент удержания позиции для шагового двигателя или крутящий момент при неподвижном роторе для других типов двигателей ( $\text{мН} \cdot \text{м}$ ).*
- float `DetentTorque`  
*Момент удержания позиции с незапитанными обмотками ( $\text{мН} \cdot \text{м}$ ).*
- float `TorqueConstant`  
*Константа крутящего момента, определяющая коэффициент пропорциональности максимального момента силы ротора от протекающего в обмотке тока ( $\text{мН} \cdot \text{м}/\text{А}$ ).*
- 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 /  $\text{мН} \cdot \text{м}$ ).*
- 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).*

### 5.40.1 Подробное описание

Физический характеристики и ограничения мотора.

См. также

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

## 5.40.2 Поля

### 5.40.2.1 DetentTorque

float `DetentTorque`

Момент удержания позиции с незапитанными обмотками (мН·м).

Тип данных: float.

### 5.40.2.2 MaxCurrent

float `MaxCurrent`

Максимальный ток в обмотке (А).

Тип данных: float.

### 5.40.2.3 MaxCurrentTime

float `MaxCurrentTime`

Безопасная длительность максимального тока в обмотке (мс).

Тип данных: float.

### 5.40.2.4 MaxSpeed

float `MaxSpeed`

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

Тип данных: float.



#### 5.40.2.5 MechanicalTimeConstant

`float MechanicalTimeConstant`

Механическая постоянная времени (мс).

Тип данных: `float`.

#### 5.40.2.6 MotorType

`unsigned int MotorType`

Флаги типа двигателя.

#### 5.40.2.7 NoLoadCurrent

`float NoLoadCurrent`

Ток потребления в холостом режиме (А).

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

#### 5.40.2.8 NoLoadSpeed

`float NoLoadSpeed`

Скорость в холостом режиме (об/мин).

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

#### 5.40.2.9 NominalCurrent

`float NominalCurrent`

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

Тип данных: `float`.

#### 5.40.2.10 NominalPower

`float NominalPower`

Номинальная мощность(Вт).

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

#### 5.40.2.11 NominalSpeed

float NominalSpeed

Не используется.

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

#### 5.40.2.12 NominalTorque

float NominalTorque

Номинальный крутящий момент (мН \* м).

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

#### 5.40.2.13 NominalVoltage

float NominalVoltage

Номинальное напряжение на обмотке (В).

Тип данных: float.

#### 5.40.2.14 Phases

unsigned int Phases

Кол-во фаз у BLDC двигателя.

#### 5.40.2.15 Poles

unsigned int Poles

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

#### 5.40.2.16 RotorInertia

float RotorInertia

Инерция ротора (г см<sup>2</sup>).

Тип данных: float.

#### 5.40.2.17 SpeedConstant

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.

#### 5.40.2.18 SpeedTorqueGradient

float SpeedTorqueGradient

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

Тип данных: float.

#### 5.40.2.19 StallTorque

float StallTorque

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

Тип данных: float.

#### 5.40.2.20 TorqueConstant

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.

#### 5.40.2.21 WindingInductance

float WindingInductance

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

Тип данных: float.

#### 5.40.2.22 WindingResistance

`float WindingResistance`

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

Тип данных: `float`.

## 5.41 Структура `move_settings_calb_t`

Настройки движения с использованием пользовательских единиц.

```
#include <ximc.h>
```

### Поля данных

- `float Speed`  
*Скорость*
- `float Accel`  
*Ускорение*
- `float Decel`  
*Торможение*
- `float AntiplaySpeed`  
*Скорость в режиме антилюфта*
- `unsigned int MoveFlags`  
*Флаги параметров движения.*

### 5.41.1 Подробное описание

Настройки движения с использованием пользовательских единиц.

См. также

`set_move_settings_calb`  
`get_move_settings_calb`  
`get_move_settings`, `set_move_settings`

### 5.41.2 Поля

#### 5.41.2.1 Accel

`float Accel`

Ускорение

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

### 5.41.2.2 AntiplaySpeed

`float AntiplaySpeed`

Скорость в режиме антилюфта

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

### 5.41.2.3 Decel

`float Decel`

Торможение

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

### 5.41.2.4 MoveFlags

`unsigned int MoveFlags`

Флаги параметров движения.

### 5.41.2.5 Speed

`float Speed`

Скорость

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

## 5.42 Структура `move_settings_t`

Настройки движения.

```
#include <ximc.h>
```

### 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.*

#### 5.42.1 Подробное описание

Настройки движения.

См. также

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

#### 5.42.2 Поля

##### 5.42.2.1 Accel

unsigned int Accel

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

Диапазон: 1..65535.

##### 5.42.2.2 AntiplaySpeed

unsigned int AntiplaySpeed

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

Диапазон: 0..100000.

#### 5.42.2.3 Decel

`unsigned int Decel`

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

Диапазон: 1..65535.

#### 5.42.2.4 MoveFlags

`unsigned int MoveFlags`

Флаги параметров движения.

#### 5.42.2.5 Speed

`unsigned int Speed`

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

Диапазон: 0..100000.

#### 5.42.2.6 uAntiplaySpeed

`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.

#### 5.42.2.7 uSpeed

`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.

### 5.43 Структура `network_settings_t`

Настройки сети.

```
#include <ximc.h>
```

### Поля данных

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

## 5.43.1 Подробное описание

Настройки сети.

Только для производителя. Эта структура содержит настройки сети.

См. также

`get_network_settings`  
`set_network_settings`  
`get_network_settings, set_network_settings`

## 5.43.2 Поля

### 5.43.2.1 DefaultGateway

`unsigned int DefaultGateway[4]`

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

### 5.43.2.2 DHCPEnabled

`unsigned int DHCPEnabled`

Определяет способ получения IP-адреса каналов.

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

### 5.43.2.3 IPv4Address

`unsigned int IPv4Address[4]`

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



#### 5.43.2.4 SubnetMask

```
unsigned int SubnetMask[4]
```

Маска подсети в формате x.x.x.x.

## 5.44 Структура `nonvolatile_memory_t`

Пользовательские данные для сохранения во FRAM.

```
#include <ximc.h>
```

### Поля данных

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

### 5.44.1 Подробное описание

Пользовательские данные для сохранения во FRAM.

См. также

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

### 5.44.2 Поля

#### 5.44.2.1 UserData

```
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.

## 5.45 Структура `password_settings_t`

Пароль.

```
#include <ximc.h>
```

### Поля данных

- char `UserPassword` [21]

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

## 5.45.1 Подробное описание

Пароль.

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

См. также

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

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

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

## 5.45.2 Поля

### 5.45.2.1 UserPassword

```
char UserPassword[21]
```

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

## 5.46 Структура pid\_settings\_t

Настройки ПИД.

```
#include <ximc.h>
```

### 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*

### 5.46.1 Подробное описание

Настройки ПИД.

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

См. также

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

## 5.47 Структура `power_settings_t`

Настройки питания шагового мотора.

```
#include <ximc.h>
```

### 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.*

### 5.47.1 Подробное описание

Настройки питания шагового мотора.

См. также

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

## 5.47.2 Поля

### 5.47.2.1 CurrentSetTime

```
unsigned int CurrentSetTime
```

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

### 5.47.2.2 CurrReductDelay

```
unsigned int CurrReductDelay
```

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

### 5.47.2.3 HoldCurrent

```
unsigned int HoldCurrent
```

Ток мотора в режиме удержания, в процентах от номинального.

Диапазон: 0..100.

### 5.47.2.4 PowerFlags

```
unsigned int PowerFlags
```

Флаги параметров питания шагового мотора.

### 5.47.2.5 PowerOffDelay

```
unsigned int PowerOffDelay
```

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

## 5.48 Структура `secure_settings_t`

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

```
#include <ximc.h>
```

### 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`  
*Устарело.*
- unsigned int `CriticalUusb`  
*Устарело.*
- unsigned int `MinimumUusb`  
*Устарело.*
- unsigned int `Flags`  
*Флаги критических параметров.*

### 5.48.1 Подробное описание

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

См. также

`get_secure_settings`  
`set_secure_settings`  
`get_secure_settings, set_secure_settings`

### 5.48.2 Поля

#### 5.48.2.1 `Criticalpwr`

`unsigned int CriticalIpwr`

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

#### 5.48.2.2 `Criticalusb`

`unsigned int CriticalIusb`

Устарело.

Максимальный ток USB, вызывающий состояние ALARM, в мА.

#### 5.48.2.3 CriticalT

`unsigned int CriticalT`

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

#### 5.48.2.4 CriticalUpwr

`unsigned int CriticalUpwr`

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

#### 5.48.2.5 CriticalUusb

`unsigned int CriticalUusb`

Устарело.

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

#### 5.48.2.6 Flags

`unsigned int Flags`

Флаги критических параметров.

#### 5.48.2.7 LowUpwrOff

`unsigned int LowUpwrOff`

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

#### 5.48.2.8 MinimumUusb

`unsigned int MinimumUusb`

Устарело.

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

### 5.49 Структура `serial_number_t`

Структура с серийным номером и версией железа.

```
#include <ximc.h>
```

### 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.*

### 5.49.1 Подробное описание

Структура с серийным номером и версией железа.

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

См. также

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

### 5.49.2 Поля

#### 5.49.2.1 Key

uint8\_t Key[32]

Ключ защиты для установки серийного номера (256 бит).

#### 5.49.2.2 Major

unsigned int Major

Основной номер версии железа.

#### 5.49.2.3 Minor

unsigned int Minor

Второстепенный номер версии железа.

#### 5.49.2.4 Release

`unsigned int Release`

Номер правок этой версии железа.

#### 5.49.2.5 SN

`unsigned int SN`

Новый серийный номер платы.

## 5.50 Структура `set_position_calb_t`

Данные о позиции с использованием пользовательских единиц.

```
#include <ximc.h>
```

### Поля данных

- `float` [Position](#)  
*Позиция двигателя.*
- `long_t` [EncPosition](#)  
*Позиция энкодера.*
- `unsigned int` [PosFlags](#)  
*Флаги установки положения.*

### 5.50.1 Подробное описание

Данные о позиции с использованием пользовательских единиц.

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

См. также

[set\\_position](#)

### 5.50.2 Поля

#### 5.50.2.1 EncPosition

`long_t EncPosition`

Позиция энкодера.



### 5.50.2.2 PosFlags

`unsigned int PosFlags`

Флаги установки положения.

### 5.50.2.3 Position

`float Position`

Позиция двигателя.

## 5.51 Структура `set_position_t`

Данные о позиции.

```
#include <ximc.h>
```

### 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.*

### 5.51.1 Подробное описание

Данные о позиции.

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

См. также

[set\\_position](#)

### 5.51.2 Поля

#### 5.51.2.1 EncPosition

`long_t EncPosition`

Позиция энкодера.

### 5.51.2.2 PosFlags

`unsigned int PosFlags`

Флаги установки положения.

### 5.51.2.3 uPosition

`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`).

## 5.52 Структура `stage_information_t`

Информация о позиционере.

```
#include <ximc.h>
```

### Поля данных

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

### 5.52.1 Подробное описание

Информация о позиционере.

См. также

`set_stage_information`  
`get_stage_information`  
`get_stage_information, set_stage_information`

### 5.52.2 Поля

#### 5.52.2.1 Manufacturer

`char Manufacturer[17]`

Производитель.

Максимальная длина строки: 16 символов.

### 5.52.2.2 PartNumber

```
char PartNumber[25]
```

Серия и номер модели.

Максимальная длина строки: 24 символа.

## 5.53 Структура stage\_name\_t

Пользовательское имя подвижки.

```
#include <ximc.h>
```

### Поля данных

- char [PositionerName](#) [17]

*Пользовательское имя подвижки.*

### 5.53.1 Подробное описание

Пользовательское имя подвижки.

См. также

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

### 5.53.2 Поля

#### 5.53.2.1 PositionerName

```
char PositionerName[17]
```

Пользовательское имя подвижки.

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

## 5.54 Структура stage\_settings\_t

Настройки позиционера.

```
#include <ximc.h>
```

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

## 5.54.1 Подробное описание

Настройки позиционера.

См. также

`set_stage_settings`  
`get_stage_settings`  
`get_stage_settings, set_stage_settings`

## 5.54.2 Поля

### 5.54.2.1 `HorizontalLoadCapacity`

`float HorizontalLoadCapacity`

Горизонтальная грузоподъемность (`кг`).

Тип данных: `float`.

### 5.54.2.2 `LeadScrewPitch`

`float LeadScrewPitch`

Шаг ходового винта в мм.

Тип данных: `float`.

#### 5.54.2.3 MaxCurrentConsumption

float MaxCurrentConsumption

Максимальный ток потребления (A).

Тип данных: float.

#### 5.54.2.4 MaxSpeed

float MaxSpeed

Максимальная скорость (Units/c).

Тип данных: float.

#### 5.54.2.5 SupplyVoltageMax

float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

#### 5.54.2.6 SupplyVoltageMin

float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

#### 5.54.2.7 TravelRange

float TravelRange

Диапазон перемещения (Units).

Тип данных: float.

#### 5.54.2.8 Units

char Units[9]

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

### 5.54.2.9 VerticalLoadCapacity

float VerticalLoadCapacity

Вертикальная грузоподъемность (кг).

Тип данных: float.

## 5.55 Структура status\_calb\_t

Состояние устройства с использованием пользовательских единиц.

```
#include <ximc.h>
```

### Поля данных

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

### 5.55.1 Подробное описание

Состояние устройства с использованием пользовательских единиц.

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

См. также

`get_status_impl`

### 5.55.2 Поля

#### 5.55.2.1 `CmdBufFreeSpace`

```
unsigned int CmdBufFreeSpace
```

Данное поле служебное.

Оно показывает количество свободных ячеек буфера цепочки синхронизации.

#### 5.55.2.2 `CurPosition`

```
float CurPosition
```

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

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

#### 5.55.2.3 `CurSpeed`

```
float CurSpeed
```

Текущая скорость.

#### 5.55.2.4 `CurT`

```
int CurT
```

Температура процессора в десятых долях градусов Цельсия.

#### 5.55.2.5 EncPosition

`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.

#### 5.55.2.6 EncSts

`unsigned int EncSts`

[Состояние энкодера.](#)

#### 5.55.2.7 Flags

`unsigned int Flags`

[Флаги состояния.](#)

#### 5.55.2.8 GPIOFlags

`unsigned int GPIOFlags`

[Флаги состояния GPIO входов.](#)

#### 5.55.2.9 Ipwr

`int Ipwr`

Ток потребления силовой части, мА.

#### 5.55.2.10 Iusb

`int Iusb`

Ток потребления по USB, мА.

#### 5.55.2.11 MoveSts

`unsigned int MoveSts`

[Флаги состояния движения.](#)



#### 5.55.2.12 MvCmdSts

unsigned int MvCmdSts

Состояние команды движения.

#### 5.55.2.13 PWRSts

unsigned int PWRSts

Флаги состояния питания шагового мотора.

#### 5.55.2.14 Upwr

int Upwr

Напряжение на силовой части, десятки мВ.

#### 5.55.2.15 Uusb

int Uusb

Напряжение на USB, десятки мВ.

#### 5.55.2.16 WindSts

unsigned int WindSts

Состояние обмоток.

### 5.56 Структура status\_t

Состояние устройства.

```
#include <ximc.h>
```

### Поля данных

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

#### 5.56.1 Подробное описание

Состояние устройства.

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

См. также

`get_status_impl`

## 5.56.2 Поля

### 5.56.2.1 CmdBufFreeSpace

```
unsigned int CmdBufFreeSpace
```

Данное поле служебное.

Оно показывает количество свободных ячеек буфера цепочки синхронизации.

### 5.56.2.2 CurPosition

```
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.

### 5.56.2.3 CurSpeed

```
int CurSpeed
```

Текущая скорость.

### 5.56.2.4 CurT

```
int CurT
```

Температура процессора в десятых долях градусов Цельсия.

### 5.56.2.5 EncPosition

```
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.

### 5.56.2.6 EncSts

```
unsigned int EncSts
```

Состояние энкодера.

### 5.56.2.7 Flags

unsigned int Flags

Флаги состояния.

### 5.56.2.8 GPIOFlags

unsigned int GPIOFlags

Флаги состояния GPIO входов.

### 5.56.2.9 Ipwr

int Ipwr

Ток потребления силовой части, мА.

### 5.56.2.10 Iusb

int Iusb

Ток потребления по USB, мА.

### 5.56.2.11 MoveSts

unsigned int MoveSts

Флаги состояния движения.

### 5.56.2.12 MvCmdSts

unsigned int MvCmdSts

Состояние команды движения.

### 5.56.2.13 PWRSts

unsigned int PWRSts

Флаги состояния питания шагового мотора.

#### 5.56.2.14 `uCurPosition`

```
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.

#### 5.56.2.15 `uCurSpeed`

```
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.

#### 5.56.2.16 `Upwr`

```
int Upwr
```

Напряжение на силовой части, десятки мВ.

#### 5.56.2.17 `Uusb`

```
int Uusb
```

Напряжение на USB, десятки мВ.

#### 5.56.2.18 `WindSts`

```
unsigned int WindSts
```

Состояние обмоток.

### 5.57 Структура `sync_in_settings_calb_t`

Настройки входной синхронизации с использованием пользовательских единиц.

```
#include <ximc.h>
```

## 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.*
- unsigned int `reserved0`

### 5.57.1 Подробное описание

Настройки входной синхронизации с использованием пользовательских единиц.

Эта структура содержит все настройки, определяющие поведение входа синхронизации.

См. также

`get_sync_in_settings_calb`  
`set_sync_in_settings_calb`  
`get_sync_in_settings`, `set_sync_in_settings`

### 5.57.2 Поля

#### 5.57.2.1 `ClutterTime`

unsigned int `ClutterTime`

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

#### 5.57.2.2 `Position`

float `Position`

Желаемая позиция или смещение.

#### 5.57.2.3 `Speed`

float `Speed`

Заданная скорость.

#### 5.57.2.4 SyncInFlags

`unsigned int SyncInFlags`

Флаги настроек синхронизации входа.

## 5.58 Структура `sync_in_settings_t`

Настройки входной синхронизации.

```
#include <ximc.h>
```

### 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.*
- `unsigned int reserved0`

### 5.58.1 Подробное описание

Настройки входной синхронизации.

Эта структура содержит все настройки, определяющие поведение входа синхронизации.

См. также

```
get_sync_in_settings  
set_sync_in_settings  
get_sync_in_settings, set_sync_in_settings
```

### 5.58.2 Поля

#### 5.58.2.1 ClutterTime

`unsigned int ClutterTime`

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

### 5.58.2.2 Speed

`unsigned int Speed`

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

Диапазон: 0..100000.

### 5.58.2.3 SyncInFlags

`unsigned int SyncInFlags`

Флаги настроек синхронизации входа.

### 5.58.2.4 uPosition

`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`).

### 5.58.2.5 uSpeed

`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.

## 5.59 Структура `sync_out_settings_calb_t`

Настройки выходной синхронизации с использованием пользовательских единиц.

```
#include <ximc.h>
```

### 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.



### 5.59.1 Подробное описание

Настройки выходной синхронизации с использованием пользовательских единиц.

Эта структура содержит все настройки, определяющие поведение выхода синхронизации.

См. также

```
get_sync_out_settings_calb  
set_sync_out_settings_calb  
get_sync_out_settings, set_sync_out_settings
```

### 5.59.2 Поля

#### 5.59.2.1 Accuracy

```
float Accuracy
```

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

#### 5.59.2.2 SyncOutFlags

```
unsigned int SyncOutFlags
```

Флаги настроек синхронизации выхода.

#### 5.59.2.3 SyncOutPeriod

```
unsigned int SyncOutPeriod
```

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

#### 5.59.2.4 SyncOutPulseSteps

```
unsigned int SyncOutPulseSteps
```

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

## 5.60 Структура `sync_out_settings_t`

Настройки выходной синхронизации.

```
#include <ximc.h>
```

### 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).*

## 5.60.1 Подробное описание

Настройки выходной синхронизации.

Эта структура содержит все настройки, определяющие поведение выхода синхронизации.

См. также

```
get_sync_out_settings
set_sync_out_settings
get_sync_out_settings, set_sync_out_settings
```

## 5.60.2 Поля

### 5.60.2.1 Accuracy

unsigned int Accuracy

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

### 5.60.2.2 SyncOutFlags

unsigned int SyncOutFlags

Флаги настроек синхронизации выхода.

### 5.60.2.3 SyncOutPeriod

`unsigned int SyncOutPeriod`

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

### 5.60.2.4 SyncOutPulseSteps

`unsigned int SyncOutPulseSteps`

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

### 5.60.2.5 uAccuracy

`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`).

## 5.61 Структура `uart_settings_t`

Настройки UART.

```
#include <ximc.h>
```

### Поля данных

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

### 5.61.1 Подробное описание

Настройки UART.

Эта структура содержит настройки UART.

См. также

```
get_uart_settings  
set_uart_settings  
get_uart_settings, set_uart_settings
```

### 5.61.2 Поля

#### 5.61.2.1 UARTSetupFlags

`unsigned int UARTSetupFlags`

*Флаги настроек четности команды UART.*

# Глава 6

## Файлы

### 6.1 Файл `ximc.h`

#### Структуры данных

- struct `calibration_t`  
*Структура калибровок*
- struct `device_network_information_t`  
*Структура данных с информацией о сетевом устройстве.*
- struct `feedback_settings_t`  
*Настройки обратной связи.*
- struct `home_settings_t`  
*Настройки калибровки позиции.*
- struct `home_settings_calb_t`  
*Настройки калибровки позиции с использованием пользовательских единиц.*
- struct `move_settings_t`  
*Настройки движения.*
- struct `move_settings_calb_t`  
*Настройки движения с использованием пользовательских единиц.*
- struct `engine_settings_t`  
*Ограничения и настройки движения, связанные с двигателем.*
- struct `engine_settings_calb_t`  
*Ограничения и настройки движения, связанные с двигателем, с использованием пользовательских единиц.*
- struct `entype_settings_t`  
*Настройки типа мотора и типа силового драйвера.*
- struct `power_settings_t`  
*Настройки питания шагового мотора.*
- struct `secure_settings_t`  
*Структура содержит параметры защиты: критические значения электрических характеристик, флаги управления алгоритмами защиты.*
- struct `edges_settings_t`  
*Настройки границ.*
- struct `edges_settings_calb_t`  
*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 `network_settings_t`  
*Настройки сети.*
- struct `password_settings_t`  
*Пароль.*
- struct `calibration_settings_t`  
*Калибровочные коэффициенты.*
- struct `controller_name_t`  
*Пользовательское имя контроллера и флаги настройки.*
- struct `nonvolatile_memory_t`  
*Пользовательские данные для сохранения во FRAM.*
- struct `emf_settings_t`  
*Настройки EMF.*
- struct `engine_advanced_setup_t`  
*Настройки EAS.*
- struct `extended_settings_t`  
*Настройки EST.*
- struct `get_position_t`  
*Данные о позиции.*
- struct `get_position_calb_t`  
*Данные о позиции.*
- struct `set_position_t`  
*Данные о позиции.*
- struct `set_position_calb_t`  
*Данные о позиции с использованием пользовательских единиц.*
- struct `status_t`

*Состояние устройства.*

- struct `status_calb_t`

*Состояние устройства с использованием пользовательских единиц.*

- struct `measurements_t`

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

- struct `chart_data_t`

*Дополнительное состояние устройства.*

- struct `device_information_t`

*Информации о контроллере.*

- struct `serial_number_t`

*Структура с серийным номером и версией железа.*

- struct `analog_data_t`

*Аналоговые данные.*

- struct `debug_read_t`

*Отладочные данные.*

- struct `debug_write_t`

*Отладочные данные.*

- struct `stage_name_t`

*Пользовательское имя подвижки.*

- struct `stage_information_t`

*Информация о позиционере.*

- struct `stage_settings_t`

*Настройки позиционера.*

- struct `motor_information_t`

*Информация о двигателе.*

- struct `motor_settings_t`

*Физический характеристики и ограничения мотора.*

- struct `encoder_information_t`

*Информация об энкодере.*

- struct `encoder_settings_t`

*Настройки энкодера.*

- struct `hallsensor_information_t`

*Информация о датчиках Холла.*

- struct `hallsensor_settings_t`

*Настройки датчиков Холла.*

- struct `gear_information_t`

*Информация о редукторе.*

- struct `gear_settings_t`

*Настройки редуктора.*

- struct `accessories_settings_t`

*Информация о дополнительных аксессуарах.*

- struct `init_random_t`

*Случайный ключ.*

- struct `globally_unique_identifier_t`

*Глобальный уникальный идентификатор.*

## Макросы

- `#define XIMC_API`
- `#define XIMC_CALLCONV`
- `#define XIMC_RETTYPE void*`
- `#define device_undefined -1`  
Макрос, означающий неопределенное устройство

## Результаты выполнения команд

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

## Флаги поиска устройств

Это битовая маска для побитовых операций.

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

## Флаги состояния движения

Это битовая маска для побитовых операций.

Возвращаются командой `get_status`.

См. также

`get_status`  
`status_t::MoveSts, get_status_impl`

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

## Флаги настроек контроллера

Это битовая маска для побитовых операций.

См. также

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

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

### Флаги состояния питания шагового мотора

Это битовая маска для побитовых операций.

Возвращаются командой `get_status`.

См. также

`get_status`  
`status_t::PWRSts, get_status_impl`

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

### Флаги состояния

Это битовая маска для побитовых операций.

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

См. также

`get_status`  
`status_t::Flags, get_status_impl`

- `#define STATE_CONTR 0x0000003F`  
*Флаги состояния контроллера.*
- `#define STATE_ERRC 0x00000001`  
*Недопустимая команда.*
- `#define STATE_ERRD 0x00000002`  
*Обнаружена ошибка целостности данных.*
- `#define STATE_ERRV 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 0x0000080
  - Контроль позиции нарушен(используется только с шаговым двигателем).
- #define STATE\_POWER\_OVERHEAT 0x0000100
  - Перегрев силового драйвера.
- #define STATE\_CONTROLLER\_OVERHEAT 0x0000200
  - Перегрелась микросхема контроллера.
- #define STATE\_OVERLOAD\_POWER\_VOLTAGE 0x0000400
  - Превышено напряжение на силовой части.
- #define STATE\_OVERLOAD\_POWER\_CURRENT 0x0000800
  - Превышен максимальный ток потребления силовой части.
- #define STATE\_OVERLOAD\_USB\_VOLTAGE 0x0001000
  - Устарело.
- #define STATE\_LOW\_USB\_VOLTAGE 0x0002000
  - Устарело.
- #define STATE\_OVERLOAD\_USB\_CURRENT 0x0004000
  - Устарело.
- #define STATE\_BORDERS\_SWAP\_MISSET 0x0008000
  - Достижение неверной границы.
- #define STATE\_LOW\_POWER\_VOLTAGE 0x0010000
  - Напряжение на силовой части ниже чем напряжение Low Voltage Protection
- #define STATE\_H\_BRIDGE\_FAULT 0x0020000
  - Получен сигнал от драйвера о неисправности
- #define STATE\_WINDING\_RES\_MISMATCH 0x0100000
  - Сопротивления обмоток слишком сильно отличаются друг от друга.
- #define STATE\_ENCODER\_FAULT 0x0200000
  - Получен сигнал от энкодера о неисправности
- #define STATE\_ENGINE\_RESPONSE\_ERROR 0x0800000
  - Ошибка реакции двигателя на управляющее воздействие.
- #define STATE\_EXTIO\_ALARM 0x1000000
  - Ошибка вызвана внешним входным сигналом EXTIO.

### Флаги состояния GPIO входов

Это битовая маска для побитовых операций.

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

См. также

[get\\_status](#)

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

- #define STATE\_DIG\_SIGNAL 0xFFFF
  - Флаги цифровых сигналов.
- #define STATE\_RIGHT\_EDGE 0x0001
  - Достижение правой границы.
- #define STATE\_LEFT\_EDGE 0x0002
  - Достижение левой границы.
- #define STATE\_BUTTON\_RIGHT 0x0004
  - Состояние кнопки "вправо" (1, если нажата).
- #define STATE\_BUTTON\_LEFT 0x0008
  - Состояние кнопки "влево" (1, если нажата).
- #define STATE\_GPIO\_PINOUT 0x0010
  - Если флаг установлен, ввод/вывод общего назначения работает как выход; если флаг сброшен, ввод/вывод работает как вход.
- #define STATE\_GPIO\_LEVEL 0x0020
  - Состояние ввода/вывода общего назначения.

- `#define STATE_BRAKE 0x0200`  
Состояние вывода управления тормозом.
- `#define STATE_REV_SENSOR 0x0400`  
Состояние вывода датчика оборотов(флаг "1", если датчик активен).
- `#define STATE_SYNC_INPUT 0x0800`  
Состояние входа синхронизации(1, если вход синхронизации активен).
- `#define STATE_SYNC_OUTPUT 0x1000`  
Состояние выхода синхронизации(1, если выход синхронизации активен).
- `#define STATE_ENC_A 0x2000`  
Состояние ножки A энкодера(флаг "1", если энкодер активен).
- `#define STATE_ENC_B 0x4000`  
Состояние ножки B энкодера(флаг "1", если энкодер активен).

### Состояние энкодера

Это битовая маска для побитовых операций.

Состояние энкодера, подключенного к контроллеру.

Заметки

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

См. также

`get_status`  
`status_t::EncSts, get_status_impl`

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

### Состояние обмоток

Это битовая маска для побитовых операций.

Состояние обмоток двигателя, подключенного к контроллеру.

См. также

`get_status`  
`status_t::WindSts, get_status_impl`

- `#define WIND_A_STATE_ABSENT 0x00`  
Обмотка A не подключена.
- `#define WIND_A_STATE_UNKNOWN 0x01`  
Состояние обмотки A неизвестно.
- `#define WIND_A_STATE_MALFUNC 0x02`  
Короткое замыкание на обмотке A.
- `#define WIND_A_STATE_OK 0x03`

- Обмотка A работает адекватно.  
• #define WIND\_B\_STATE\_ABSENT 0x00
- Обмотка B не подключена.  
• #define WIND\_B\_STATE\_UNKNOWN 0x10
- Состояние обмотки B неизвестно.  
• #define WIND\_B\_STATE\_MALFUNC 0x20
- Короткое замыкание на обмотке B.  
• #define WIND\_B\_STATE\_OK 0x30
- Обмотка B работает адекватно.

### Состояние команды движения

Это битовая маска для побитовых операций.

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

См. также

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

- #define MVCMD\_NAME\_BITS 0x3F  
Битовая маска активной команды.
- #define MVCMD\_UKNWN 0x00  
Неизвестная команда.
- #define MVCMD\_MOVE 0x01  
Команда move.
- #define MVCMD\_MOVR 0x02  
Команда movr.
- #define MVCMD\_LEFT 0x03  
Команда left.
- #define MVCMD\_RIGHT 0x04  
Команда 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`

- `#define MICROSTEP_MODE_FRAC_16 0x05`  
*Деление шага 1/8.*
- `#define MICROSTEP_MODE_FRAC_32 0x06`  
*Деление шага 1/16.*
- `#define MICROSTEP_MODE_FRAC_64 0x07`  
*Деление шага 1/32.*
- `#define MICROSTEP_MODE_FRAC_128 0x08`  
*Деление шага 1/64.*
- `#define MICROSTEP_MODE_FRAC_256 0x09`  
*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_INTEGRATE 0x02`  
*Силовой драйвер с использованием ключей, интегрированных в микросхему.*
  - `#define DRIVER_TYPE_EXTERNAL 0x03`  
*Внешний силовой драйвер.*

### Флаги параметров питания шагового мотора

Это битовая маска для побитовых операций.

Возвращаются командой `get_power_settings`.

См. также

[get\\_power\\_settings](#)  
[set\\_power\\_settings](#)  
[power\\_settings\\_t::PowerFlags, get\\_power\\_settings, set\\_power\\_settings](#)

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

#### Флаги критических параметров.

Это битовая маска для побитовых операций.

Возвращаются командой `get_secure_settings`.

См. также

[get\\_secure\\_settings](#)  
[set\\_secure\\_settings](#)  
[secure\\_settings\\_t::Flags, get\\_secure\\_settings, set\\_secure\\_settings](#)

- #define **ALARM\_ON\_DRIVER\_OVERHEATING** 0x01  
*Если флаг установлен, то войти в состояние Alarm при получении сигнала подступающего перегрева с драйвера.*
- #define **LOW\_UPWR\_PROTECTION** 0x02  
*Если установлен, то выключать силовую часть при напряжении меньшем LowUpwrOff.*
- #define **H\_BRIDGE\_ALERT** 0x04  
*Если установлен, то выключать силовую часть при сигнале неполадки в одном из транзисторных мостов.*
- #define **ALARM\_ON\_BORDERS\_SWAP\_MISSET** 0x08  
*Если флаг установлен, то войти в состояние Alarm при получении сигнала с противоположного концевого выключателя.*
- #define **ALARM\_FLAGS\_STICKING** 0x10  
*Если флаг установлен, то только по команде STOP возможен сброс всех флагов ALARM.*
- #define **BRAKING\_OVERVOLTAGE\_PROTECTION** 0x20  
*Если флаг установлен, то микропрограмма контроллера будет замыкать нижние ключи H-моста, отсоединяя мотор от цепи питания, при перенапряжении.*
- #define **ALARM\_WINDING\_MISMATCH** 0x40  
*Если флаг установлен, то войти в состояние Alarm при получении сигнала рассогласования обмоток*
- #define **ALARM\_ENGINE\_RESPONSE** 0x80  
*Если флаг установлен, то войти в состояние Alarm при получении сигнала ошибки реакции двигателя на управляющее воздействие*

#### Флаги установки положения

Это битовая маска для побитовых операций.

Возвращаются командой `get_position`.

См. также

[get\\_position](#)  
[set\\_position](#)  
[set\\_position\\_t::PosFlags, set\\_position](#)

- #define SETPOS\_IGNORE\_POSITION 0x01  
*Если установлен, то позиция в шагах и микрошагах не обновляется.*
- #define SETPOS\_IGNORE\_ENCODER 0x02  
*Если установлен, то счётчик энкодера не обновляется.*

### Тип обратной связи.

Это битовая маска для побитовых операций.

См. также

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

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

### Флаги обратной связи.

Это битовая маска для побитовых операций.

См. также

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

- #define FEEDBACK\_ENC\_REVERSE 0x01  
*Обратный счет у энкодера.*
- #define FEEDBACK\_ENC\_ADAPTIVE\_HOLDING 0x02  
*Включает алгоритм адаптивного удержания.*
- #define FEEDBACK\_ENC\_FILTER\_NONE 0x00  
*Выключает внутренний фильтр сигнала энкодера.*
- #define FEEDBACK\_ENC\_FILTER\_WEAK 0x10  
*Слабая фильтрация шумов: максимальная частота сигнала энкодера 3 МГц.*
- #define FEEDBACK\_ENC\_FILTER\_MEDIUM 0x20  
*Средняя фильтрация шумов: максимальная частота сигнала энкодера 1 МГц.*
- #define FEEDBACK\_ENC\_FILTER\_STRONG 0x30  
*Сильная фильтрация шумов: максимальная частота сигнала энкодера 300 кГц.*
- #define FEEDBACK\_ENC\_FILTER\_BITS 0x30  
*Биты, отвечающие за настройку внутреннего фильтра энкодерного сигнала.*
- #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.*
- #define `FEEDBACK_ENC_TYPE_BITS` 0xC0  
*Биты, отвечающие за тип энкодера.*

#### Флаги настроек синхронизации входа

Это битовая маска для побитовых операций.

См. также

`sync_in_settings_t::SyncInFlags, get_sync_in_settings, set_sync_in_settings`

- #define `SYNCIN_ENABLED` 0x01  
*Включение необходимости импульса синхронизации для начала движения.*
- #define `SYNCIN_INVERT` 0x02  
*Если установлен - срабатывает по переходу из 1 в 0.*
- #define `SYNCIN_GOTOPOSITION` 0x04  
*Если флаг установлен, то двигатель смещается к позиции, установленной в `Position` и `uPosition`, иначе двигатель смещается на `Position` и `uPosition`*

#### Флаги настроек синхронизации выхода

Это битовая маска для побитовых операций.

См. также

`sync_out_settings_t::SyncOutFlags, get_sync_out_settings, set_sync_out_settings`

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

#### Флаги настройки работы внешнего ввода/вывода

Это битовая маска для побитовых операций.

См. также

`get_extio_settings`  
`set_extio_settings`  
`extio_settings_t::EXTIOSetupFlags, get_extio_settings, set_extio_settings`

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

#### Флаги настройки режимов внешнего ввода/вывода

Это битовая маска для побитовых операций.



См. также

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

- #define EXTIO\_SETUP\_MODE\_IN\_BITS 0x0F  
*Биты, отвечающие за поведение при переходе сигнала в активное состояние.*
- #define EXTIO\_SETUP\_MODE\_IN\_NOP 0x00  
*Ничего не делать.*
- #define EXTIO\_SETUP\_MODE\_IN\_STOP 0x01  
*По переднему фронту входного сигнала делается остановка двигателя (эквивалент команды STOP).*
- #define EXTIO\_SETUP\_MODE\_IN\_PWOF 0x02  
*Выполняет команду PWOF, обесточивая обмотки двигателя.*
- #define EXTIO\_SETUP\_MODE\_IN\_MOVR 0x03  
*Выполняется команда MOVR с последними настройками.*
- #define EXTIO\_SETUP\_MODE\_IN\_HOME 0x04  
*Выполняется команда HOME.*
- #define EXTIO\_SETUP\_MODE\_IN\_ALARM 0x05  
*Войти в состояние ALARM при переходе сигнала в активное состояние.*
- #define EXTIO\_SETUP\_MODE\_OUT\_BITS 0xF0  
*Биты выбора поведения на выходе.*
- #define EXTIO\_SETUP\_MODE\_OUT\_OFF 0x00  
*Ножка всегда в неактивном состоянии.*
- #define EXTIO\_SETUP\_MODE\_OUT\_ON 0x10  
*Ножка всегда в активном состоянии.*
- #define EXTIO\_SETUP\_MODE\_OUT\_MOVING 0x20  
*Ножка находится в активном состоянии при движении.*
- #define EXTIO\_SETUP\_MODE\_OUT\_ALARM 0x30  
*Ножка находится в активном состоянии при нахождении в состоянии ALARM.*
- #define EXTIO\_SETUP\_MODE\_OUT\_MOTOR\_ON 0x40  
*Ножка находится в активном состоянии при подаче питания на обмотки.*

### Флаги границ

Это битовая маска для побитовых операций.

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

См. также

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

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

### Флаги 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 - Концевой переключатель, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте.
- #define [ENDER\\_SW2\\_ACTIVE\\_LOW](#) 0x04  
 1 - Концевой переключатель, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте.

### Флаги настроек тормоза

Это битовая маска для побитовых операций.

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

См. также

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

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

### Флаги управления

Это битовая маска для побитовых операций.

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

См. также

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

- #define [CONTROL\\_MODE\\_BITS](#) 0x03  
 Биты управления мотором с помощью джойстика или кнопок влево/вправо.
- #define [CONTROL\\_MODE\\_OFF](#) 0x00  
 Управление отключено.
- #define [CONTROL\\_MODE\\_JOY](#) 0x01  
 Управление с помощью джойстика.
- #define [CONTROL\\_MODE\\_LR](#) 0x02  
 Управление с помощью кнопок влево/вправо.
- #define [CONTROL\\_BTN\\_LEFT\\_PUSHED\\_OPEN](#) 0x04  
 Нажатая левая кнопка соответствует открытому контакту, если этот флаг установлен.
- #define [CONTROL\\_BTN\\_RIGHT\\_PUSHED\\_OPEN](#) 0x08  
 Нажатая правая кнопка соответствует открытому контакту, если этот флаг установлен.

### Флаги джойстика

Это битовая маска для побитовых операций.

Управляют состояниями джойстика.

См. также

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

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

### Флаги контроля позиции

Это битовая маска для побитовых операций.

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

См. также

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

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

### Флаги настроек команды home

Это битовая маска для побитовых операций.

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

См. также

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

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

- `#define HOME_STOP_FIRST_SYN 0x020`  
*Первое движение завершается по сигналу со входа синхронизации.*
- `#define HOME_STOP_FIRST_LIM 0x030`  
*Первое движение завершается по сигналу с концевого переключателя.*
- `#define HOME_STOP_SECOND_BITS 0x0C0`  
*Биты, отвечающие за выбор сигнала завершения второго движения.*
- `#define HOME_STOP_SECOND_REV 0x040`  
*Второе движение завершается по сигналу с Revolution sensor.*
- `#define HOME_STOP_SECOND_SYN 0x080`  
*Второе движение завершается по сигналу со входа синхронизации.*
- `#define HOME_STOP_SECOND_LIM 0x0C0`  
*Второе движение завершается по сигналу с концевого переключателя.*
- `#define HOME_USE_FAST 0x100`  
*Если флаг установлен, используется быстрый поиск домашней позиции; иначе - традиционный.*

### Флаги настроек четности команды UART

Это битовая маска для побитовых операций.

См. также

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

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

### Флаги типа двигателя

Это битовая маска для побитовых операций.

См. также

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

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

### Флаги настроек энкодера

Это битовая маска для побитовых операций.

См. также

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

- **#define ENCSET\_DIFFERENTIAL\_OUTPUT** 0x001  
*Если флаг установлен, то энкодер имеет дифференциальный выход, иначе - несимметричный выход*
- **#define ENCSET\_PUSHPULL\_OUTPUT** 0x004  
*Если флаг установлен, то энкодер имеет двухтактный выход, иначе - выход с открытым коллектором*
- **#define ENCSET\_INDEXCHANNEL\_PRESENT** 0x010  
*Если флаг установлен, то энкодер имеет дополнительный индексный канал, иначе - он отсутствует*
- **#define ENCSET\_REVOLUTIONSENSOR\_PRESENT** 0x040  
*Если флаг установлен, то энкодер имеет датчик оборотов, иначе - он отсутствует*
- **#define ENCSET\_REVOLUTIONSENSOR\_ACTIVE\_HIGH** 0x100  
*Если флаг установлен, то активное состояние датчика оборотов соответствует логической 1, иначе - логическому 0*
- **#define MB\_AVAILABLE** 0x01  
*Если флаг установлен, то магнитный тормоз доступен*
- **#define MB\_POWERED\_HOLD** 0x02  
*Если флаг установлен, то магнитный тормоз находится в режиме удержания (активен) при подаче питания*

#### Флаги настроек температурного датчика

Это битовая маска для побитовых операций.

См. также

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

- **#define TS\_TYPE\_BITS** 0x07  
*Биты, отвечающие за тип температурного датчика.*
- **#define TS\_TYPE\_UNKNOWN** 0x00  
*Неизвестный сенсор*
- **#define TS\_TYPE\_THERMOCOUPLE** 0x01  
*Термопара*
- **#define TS\_TYPE\_SEMICONDUCTOR** 0x02  
*Полупроводниковый температурный датчик*
- **#define TS\_AVAILABLE** 0x08  
*Если флаг установлен, то датчик температуры доступен*
- **#define LS\_ON\_SW1\_AVAILABLE** 0x01  
*Если флаг установлен, то концевой переключатель, подключенный к ножке SW1, доступен*
- **#define LS\_ON\_SW2\_AVAILABLE** 0x02  
*Если флаг установлен, то концевой переключатель, подключенный к ножке SW2, доступен*
- **#define LS\_SW1\_ACTIVE\_LOW** 0x04  
*Если флаг установлен, то концевой переключатель, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте*
- **#define LS\_SW2\_ACTIVE\_LOW** 0x08  
*Если флаг установлен, то концевой переключатель, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте*
- **#define LS\_SHORTED** 0x10  
*Если флаг установлен, то концевые переключатели замкнуты.*

#### Флаги автоопределения характеристик обмоток двигателя.

Это битовая маска для побитовых операций.

См. также

[set\\_emf\\_settings](#)  
[get\\_emf\\_settings](#)  
[emf\\_settings\\_t::BackEMFFlags](#), [get\\_emf\\_settings](#), [set\\_emf\\_settings](#)

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

## Определения типов

- `typedef unsigned long long ulong_t`
- `typedef long long long_t`
- `typedef int device_t`  
 Тип идентификатора устройства
- `typedef int result_t`  
 Тип, определяющий результат выполнения команды.
- `typedef uint32_t device_enumeration_t`  
 Тип, определяющий структуру данных о всех контроллерах, обнаруженных при опросе устройств.

## Функции

### Группа команд настройки контроллера

Функции для чтения/записи большинства настроек контроллера.

- `result_t XIMC_API set_feedback_settings (device_t id, const feedback_settings_t *feedback_↔  
 _settings)`  
 Запись настроек обратной связи.
- `result_t XIMC_API get_feedback_settings (device_t id, feedback_settings_t *feedback_↔  
 settings)`  
 Чтение настроек обратной связи
- `result_t XIMC_API set_home_settings (device_t id, const home_settings_t *home_settings)`  
 Команда записи настроек для подхода в *home position*.
- `result_t XIMC_API set_home_settings_calb (device_t id, const home_settings_calb_t *home_settings_calb, const calibration_t *calibration)`  
 Команда записи настроек для подхода в *home position* с использованием пользовательских единиц.
- `result_t XIMC_API get_home_settings (device_t id, home_settings_t *home_settings)`  
 Команда чтения настроек для подхода в *home position*.
- `result_t XIMC_API get_home_settings_calb (device_t id, home_settings_calb_t *home_↔  
 settings_calb, const calibration_t *calibration)`  
 Команда чтения настроек для подхода в *home position* с использованием пользовательских единиц.
- `result_t XIMC_API set_move_settings (device_t id, const move_settings_t *move_settings)`  
 Команда записи настроек перемещения (скорость, ускорение, *threshold* и скорость в режиме антилюфта).
- `result_t XIMC_API set_move_settings_calb (device_t id, const move_settings_calb_t *move_settings_calb, const calibration_t *calibration)`  
 Команда записи настроек перемещения, с использованием пользовательских единиц (скорость, ускорение, *threshold* и скорость в режиме антилюфта).
- `result_t XIMC_API get_move_settings (device_t id, move_settings_t *move_settings)`

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

- `result_t XIMC_API get_move_settings_calb (device_t id, move_settings_calb_t *move_↔ settings_calb, const calibration_t *calibration)`

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

- `result_t XIMC_API set_engine_settings (device_t id, const engine_settings_t *engine_↔ settings)`

Запись настроек мотора.

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

Запись настроек мотора с использованием пользовательских единиц.

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

Чтение настроек мотора.

- `result_t XIMC_API get_engine_settings_calb (device_t id, engine_settings_calb_t *engine_↔ settings_calb, const calibration_t *calibration)`

Чтение настроек мотора с использованием пользовательских единиц.

- `result_t XIMC_API set_entype_settings (device_t id, const entype_settings_t *entype_↔ settings)`

Запись информации о типе мотора и типе силового драйвера.

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

Возвращает информацию о типе мотора и силового драйвера.

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

Команда записи параметров питания мотора.

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

Команда чтения параметров питания мотора.

- `result_t XIMC_API set_secure_settings (device_t id, const secure_settings_t *secure_↔ settings)`

Команда записи установок защит.

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

Команда записи установок защит.

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

Запись настроек границ и концевых выключателей.

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

Запись настроек границ и концевых выключателей с использованием пользовательских единиц.

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

Чтение настроек границ и концевых выключателей.

- `result_t XIMC_API get_edges_settings_calb (device_t id, edges_settings_calb_t *edges_↔ settings_calb, const calibration_t *calibration)`

Чтение настроек границ и концевых выключателей с использованием пользовательских единиц.

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

Запись ПИД коэффициентов.

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

Чтение ПИД коэффициентов.

- `result_t XIMC_API set_sync_in_settings (device_t id, const sync_in_settings_t *sync_in_↔ settings)`

Запись настроек для входного импульса синхронизации.

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

Запись настроек для входного импульса синхронизации с использованием пользовательских единиц.

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

Чтение настроек для входного импульса синхронизации.

- `result_t XIMC_API get_sync_in_settings_calb (device_t id, sync_in_settings_calb_t *sync_↔ in_settings_calb, const calibration_t *calibration)`

Чтение настроек для входного импульса синхронизации с использованием пользовательских единиц.

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



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

### Группа команд управления движением

- `result_t XIMC_API command_stop (device_t id)`  
*Немедленная остановка двигателя, переход в состояние STOP, ключи в режиме BREAK (обмотки накоротко замкнуты), режим "удержания" деактивируется для DC двигателей, удержание тока в обмотках для шаговых двигателей (с учётом Power management настроек).*
- `result_t XIMC_API command_power_off (device_t id)`  
*Немедленное отключение питания двигателя вне зависимости от его состояния.*
- `result_t XIMC_API command_move (device_t id, int Position, int uPosition)`  
*При получении команды "move" двигатель начинает перемещаться (если не используется режим "ТТЛСинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), к точке указанной в полях Position, uPosition.*
- `result_t XIMC_API command_move_calb (device_t id, float Position, const calibration_t *calibration)`  
*Перемещение в позицию с использованием пользовательских единиц.*
- `result_t XIMC_API command_movr (device_t id, int DeltaPosition, int uDeltaPosition)`  
*Перемещение на заданное смещение.*
- `result_t XIMC_API command_movr_calb (device_t id, float DeltaPosition, const calibration_t *calibration)`  
*Перемещение на заданное смещение с использованием пользовательских единиц.*
- `result_t XIMC_API command_home (device_t id)`  
*Движение в домашнюю позицию.*
- `result_t XIMC_API command_left (device_t id)`

- При получении команды "left" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), влево.
- `result_t XIMC_API command_right (device_t id)`  
При получении команды "right" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), вправо.
  - `result_t XIMC_API command_loft (device_t id)`  
При получении команды "loft" двигатель смещается из текущей точки на расстояние *Antiplay*, заданное в настройках мотора (*engine\_settings*), затем двигается в ту же точку.
  - `result_t XIMC_API command_sstp (device_t id)`  
Плавная остановка.
  - `result_t XIMC_API get_position (device_t id, get_position_t *the_get_position)`  
Считывает значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.
  - `result_t XIMC_API get_position_calb (device_t id, get_position_calb_t *the_get_position_calb, const calibration_t *calibration)`  
Считывает значение положения в пользовательских единицах для шагового двигателя и в шагах энкодера всех двигателей.
  - `result_t XIMC_API set_position (device_t id, const set_position_t *the_set_position)`  
Устанавливает произвольное значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера для всех двигателей.
  - `result_t XIMC_API set_position_calb (device_t id, const set_position_calb_t *the_set_position_calb, const calibration_t *calibration)`  
Устанавливает произвольное значение положения и значение энкодера всех двигателей с использованием пользовательских единиц.
  - `result_t XIMC_API command_zero (device_t id)`  
Устанавливает текущую позицию равной 0.

### Группа команд сохранения и загрузки настроек

- `result_t XIMC_API command_save_settings (device_t id)`  
При получении команды контроллер выполняет операцию сохранения текущих настроек во встроенную энергонезависимую память контроллера.
- `result_t XIMC_API command_read_settings (device_t id)`  
Чтение всех настроек контроллера из flash памяти в оперативную, заменяя текущие настройки.
- `result_t XIMC_API command_save_robust_settings (device_t id)`  
При получении команды контроллер выполняет операцию сохранения важных настроек (калибровочные коэффициенты и т.
- `result_t XIMC_API command_read_robust_settings (device_t id)`  
Чтение важных настроек (калибровочные коэффициенты и т.
- `result_t XIMC_API command_eesave_settings (device_t id)`  
Запись настроек контроллера в EEPROM памяти позиционера. Функция должна использоваться только производителем.
- `result_t XIMC_API command_eeread_settings (device_t id)`  
Чтение настроек контроллера из EEPROM памяти позиционера.
- `result_t XIMC_API command_start_measurements (device_t id)`  
Начать измерения и буферизацию скорости, ошибки следования.
- `result_t XIMC_API get_measurements (device_t id, measurements_t *measurements)`  
Команда чтения буфера данных для построения графиков скорости и ошибки следования.
- `result_t XIMC_API get_chart_data (device_t id, chart_data_t *chart_data)`  
Команда чтения состояния обмоток и других не часто используемых данных.
- `result_t XIMC_API get_serial_number (device_t id, unsigned int *SerialNumber)`  
Чтение серийного номера контроллера.
- `result_t XIMC_API get_firmware_version (device_t id, unsigned int *Major, unsigned int *Minor, unsigned int *Release)`  
Чтение номера версии прошивки контроллера.
- `result_t XIMC_API service_command_updf (device_t id)`  
Команда переводит контроллер в режим обновления прошивки.

**Группа сервисных команд**

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

**Группа команд работы с EEPROM подвижки**

- `result_t XIMC_API set_stage_name (device_t id, const stage_name_t *stage_name)`  
*Запись пользовательского имени подвижки в EEPROM.*
- `result_t XIMC_API get_stage_name (device_t id, stage_name_t *stage_name)`  
*Чтение пользовательского имени подвижки из EEPROM.*
- `result_t XIMC_API set_stage_information (device_t id, const stage_information_t *stage_↔  
information)`  
*Запись информации о позиционере в EEPROM.*
- `result_t XIMC_API get_stage_information (device_t id, stage_information_t *stage_↔  
information)`  
*Чтение информации о позиционере из EEPROM.*
- `result_t XIMC_API set_stage_settings (device_t id, const stage_settings_t *stage_settings)`  
*Запись настроек позиционера в EEPROM.*
- `result_t XIMC_API get_stage_settings (device_t id, stage_settings_t *stage_settings)`  
*Чтение настроек позиционера из EEPROM.*
- `result_t XIMC_API set_motor_information (device_t id, const motor_information_t *motor_↔  
_information)`  
*Запись информации о двигателе в EEPROM.*
- `result_t XIMC_API get_motor_information (device_t id, motor_information_t *motor_↔  
information)`  
*Чтение информации о двигателе из EEPROM.*
- `result_t XIMC_API set_motor_settings (device_t id, const motor_settings_t *motor_settings)`  
*Запись настроек двигателя в EEPROM.*
- `result_t XIMC_API get_motor_settings (device_t id, motor_settings_t *motor_settings)`  
*Чтение настроек двигателя из EEPROM.*
- `result_t XIMC_API set_encoder_information (device_t id, const encoder_information_t  
*encoder_information)`  
*Запись информации об энкодере в EEPROM.*
- `result_t XIMC_API get_encoder_information (device_t id, encoder_information_t *encoder_↔  
_information)`  
*Чтение информации об энкодере из EEPROM.*
- `result_t XIMC_API set_encoder_settings (device_t id, const encoder_settings_t *encoder_↔  
settings)`  
*Запись настроек энкодера в EEPROM.*
- `result_t XIMC_API get_encoder_settings (device_t id, encoder_settings_t *encoder_settings)`  
*Чтение настроек энкодера из EEPROM.*
- `result_t XIMC_API set_hallsensor_information (device_t id, const hallsensor_information_t  
*hallsensor_information)`  
*Запись информации о датчиках Холла в EEPROM.*
- `result_t XIMC_API get_hallsensor_information (device_t id, hallsensor_information_t  
*hallsensor_information)`  
*Чтение информации о датчиках Холла из EEPROM.*
- `result_t XIMC_API set_hallsensor_settings (device_t id, const hallsensor_settings_t  
*hallsensor_settings)`  
*Запись настроек датчиков Холла в EEPROM.*

- `result_t XIMC_API get_hallsensor_settings (device_t id, hallsensor_settings_t *hallsensor_settings)`  
Чтение настроек датчиков Холла из EEPROM.
- `result_t XIMC_API set_gear_information (device_t id, const gear_information_t *gear_information)`  
Запись информации о редукторе в EEPROM.
- `result_t XIMC_API get_gear_information (device_t id, gear_information_t *gear_information)`  
Чтение информации о редукторе из EEPROM.
- `result_t XIMC_API set_gear_settings (device_t id, const gear_settings_t *gear_settings)`  
Запись настроек редуктора в EEPROM.
- `result_t XIMC_API get_gear_settings (device_t id, gear_settings_t *gear_settings)`  
Чтение настроек редуктора из EEPROM.
- `result_t XIMC_API set_accessories_settings (device_t id, const accessories_settings_t *accessories_settings)`  
Запись информации о дополнительных аксессуарах в EEPROM.
- `result_t XIMC_API get_accessories_settings (device_t id, accessories_settings_t *accessories_settings)`  
Чтение информации о дополнительных аксессуарах из EEPROM.
- `result_t XIMC_API get_bootloader_version (device_t id, unsigned int *Major, unsigned int *Minor, unsigned int *Release)`  
Чтение номера версии загрузчика контроллера.
- `result_t XIMC_API get_init_random (device_t id, init_random_t *init_random)`  
Чтение случайного числа из контроллера.
- `result_t XIMC_API get_globally_unique_identifier (device_t id, globally_unique_identifier_t *globally_unique_identifier)`  
Считывает уникальный идентификатор каждого чипа, это значение не является случайным.
- `result_t XIMC_API goto_firmware (device_t id, uint8_t *ret)`  
Перезагрузка в прошивку в контроллере
- `result_t XIMC_API has_firmware (const char *uri, uint8_t *ret)`  
Проверка наличия прошивки в контроллере
- `result_t XIMC_API command_update_firmware (const char *uri, const uint8_t *data, uint32_t data_size)`  
Обновление прошивки.
- `result_t XIMC_API write_key (const char *uri, uint8_t *key)`  
Запись ключа защиты. Функция используется только производителем.
- `result_t XIMC_API command_reset (device_t id)`  
Перезагрузка контроллера.
- `result_t XIMC_API command_clear_fram (device_t id)`  
Очистка FRAM памяти контроллера.

### 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
- `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.

## 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 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.*

- device\_enumeration\_t XIMC\_API enumerate\_devices (int enumerate\_flags, const char \*hints)

*Поиск и составление списка доступных устройств.*

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

*Освобождает память, выделенную enumerate\_devices.*

- int XIMC\_API get\_device\_count (device\_enumeration\_t device\_enumeration)

*Возвращает количество подключенных устройств.*

- pchar XIMC\_API get\_device\_name (device\_enumeration\_t device\_enumeration, int device\_index)

*Возвращает имя подключенного устройства из перечисления устройств.*

- result\_t XIMC\_API get\_enumerate\_device\_serial (device\_enumeration\_t device\_enumeration, int device\_index, uint32\_t \*serial)

*Возвращает серийный номер подключенного устройства из перечисления устройств.*

- result\_t XIMC\_API get\_enumerate\_device\_information (device\_enumeration\_t device\_enumeration, int device\_index, device\_information\_t \*device\_information)

*Возвращает информацию о подключенном устройстве из перечисления устройств.*

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

*Возвращает имя подключенного устройства из перечисления устройств.*

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

*Возвращает имя подвижки для подключенного устройства из перечисления устройств.*

- result\_t XIMC\_API get\_enumerate\_device\_network\_information (device\_enumeration\_t device\_enumeration, int device\_index, device\_network\_information\_t \*device\_network\_information)

*Возвращает сетевую информацию о подключенном устройстве из перечисления устройств.*

- result\_t XIMC\_API reset\_locks ()

*Сбрасывает ошибку неправильной передачи данных.*

- 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.*

### 6.1.1 Подробное описание

Заголовочный файл для библиотеки libximc

### 6.1.2 Макросы

#### 6.1.2.1 ALARM\_FLAGS\_STICKING

```
#define ALARM_FLAGS_STICKING 0x10
```

Если флаг установлен, то только по команде STOP возможен сброс всех флагов ALARM.

#### 6.1.2.2 ALARM\_ON\_BORDERS\_SWAP\_MISSET

```
#define ALARM_ON_BORDERS_SWAP_MISSET 0x08
```

Если флаг установлен, то войти в состояние Alarm при получении сигнала с противоположного концевого выключателя.

#### 6.1.2.3 ALARM\_ON\_DRIVER\_OVERHEATING

```
#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.

#### 6.1.2.4 BACK\_EMF\_INDUCTANCE\_AUTO

```
#define BACK_EMF_INDUCTANCE_AUTO 0x01
```

Флаг автоопределения индуктивности обмоток двигателя.

#### 6.1.2.5 BACK\_EMF\_KM\_AUTO

```
#define BACK_EMF_KM_AUTO 0x04
```

Флаг автоопределения электромеханического коэффициента двигателя.

#### 6.1.2.6 BACK\_EMF\_RESISTANCE\_AUTO

```
#define BACK_EMF_RESISTANCE_AUTO 0x02
```

Флаг автоопределения сопротивления обмоток двигателя.

#### 6.1.2.7 BORDER\_IS\_ENCODER

```
#define BORDER_IS_ENCODER 0x01
```

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

Если флаг сброшен, границы определяются концевыми выключателями.

#### 6.1.2.8 BORDER\_STOP\_LEFT

```
#define BORDER_STOP_LEFT 0x02
```

Если флаг установлен, мотор останавливается при достижении левой границы.

#### 6.1.2.9 BORDER\_STOP\_RIGHT

```
#define BORDER_STOP_RIGHT 0x04
```

Если флаг установлен, мотор останавливается при достижении правой границы.

#### 6.1.2.10 BORDERS\_SWAP\_MISSET\_DETECTION

```
#define BORDERS_SWAP_MISSET_DETECTION 0x08
```

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

Нужен для предотвращения поломки двигателя при неправильных настройках концевых выключателей

#### 6.1.2.11 BRAKE\_ENABLED

```
#define BRAKE_ENABLED 0x01
```

Управление тормозом включено, если флаг установлен.

#### 6.1.2.12 BRAKE\_ENG\_PWROFF

```
#define BRAKE_ENG_PWROFF 0x02
```

Тормоз отключает питание шагового мотора, если флаг установлен.

#### 6.1.2.13 BRAKING\_OVERVOLTAGE\_PROTECTION

```
#define BRAKING_OVERVOLTAGE_PROTECTION 0x20
```

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

#### 6.1.2.14 CONTROL\_BTN\_LEFT\_PUSHED\_OPEN

```
#define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04
```

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

#### 6.1.2.15 CONTROL\_BTN\_RIGHT\_PUSHED\_OPEN

```
#define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08
```

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

#### 6.1.2.16 CONTROL\_MODE\_BITS

```
#define CONTROL_MODE_BITS 0x03
```

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

#### 6.1.2.17 CONTROL\_MODE\_JOY

```
#define CONTROL_MODE_JOY 0x01
```

Управление с помощью джойстика.



#### 6.1.2.18 `CONTROL_MODE_LR`

```
#define CONTROL_MODE_LR 0x02
```

Управление с помощью кнопок влево/вправо.

#### 6.1.2.19 `CONTROL_MODE_OFF`

```
#define CONTROL_MODE_OFF 0x00
```

Управление отключено.

#### 6.1.2.20 `CTP_ALARM_ON_ERROR`

```
#define CTP_ALARM_ON_ERROR 0x04
```

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

#### 6.1.2.21 `CTP_BASE`

```
#define CTP_BASE 0x02
```

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

#### 6.1.2.22 `CTP_ENABLED`

```
#define CTP_ENABLED 0x01
```

Контроль позиции включен, если флаг установлен.

#### 6.1.2.23 `CTP_ERROR_CORRECTION`

```
#define CTP_ERROR_CORRECTION 0x10
```

Корректировать ошибки, возникающие при проскальзывании, если флаг установлен.

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

#### 6.1.2.24 `DRIVER_TYPE_EXTERNAL`

```
#define DRIVER_TYPE_EXTERNAL 0x03
```

Внешний силовой драйвер.

#### 6.1.2.25 DRIVER\_TYPE\_INTEGRATE

```
#define DRIVER_TYPE_INTEGRATE 0x02
```

Силовой драйвер с использованием ключей, интегрированных в микросхему.

#### 6.1.2.26 EEPROM\_PRECEDENCE

```
#define EEPROM_PRECEDENCE 0x01
```

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

#### 6.1.2.27 ENC\_STATE\_ABSENT

```
#define ENC_STATE_ABSENT 0x00
```

Энкодер не подключен.

#### 6.1.2.28 ENC\_STATE\_MALFUNC

```
#define ENC_STATE_MALFUNC 0x02
```

Энкодер подключен и неисправен.

#### 6.1.2.29 ENC\_STATE\_OK

```
#define ENC_STATE_OK 0x04
```

Энкодер подключен и работает должным образом.

#### 6.1.2.30 ENC\_STATE\_REVERS

```
#define ENC_STATE_REVERS 0x03
```

Энкодер подключен и исправен, но считает в другую сторону.

#### 6.1.2.31 ENC\_STATE\_UNKNOWN

```
#define ENC_STATE_UNKNOWN 0x01
```

Состояние энкодера неизвестно.

### 6.1.2.32 `ENDER_SW1_ACTIVE_LOW`

```
#define ENDER_SW1_ACTIVE_LOW 0x02
```

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

### 6.1.2.33 `ENDER_SW2_ACTIVE_LOW`

```
#define ENDER_SW2_ACTIVE_LOW 0x04
```

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

### 6.1.2.34 `ENDER_SWAP`

```
#define ENDER_SWAP 0x01
```

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

### 6.1.2.35 `ENGINE_ACCEL_ON`

```
#define ENGINE_ACCEL_ON 0x10
```

Ускорение.

Если флаг установлен, движение происходит с ускорением.

### 6.1.2.36 `ENGINE_ANTIPLAY`

```
#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.

### 6.1.2.37 `ENGINE_CURRENT_AS_RMS`

```
#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).

### 6.1.2.38 `ENGINE_LIMIT_CURR`

```
#define ENGINE_LIMIT_CURR 0x40
```

Номинальный ток мотора.

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

### 6.1.2.39 `ENGINE_LIMIT_RPM`

```
#define ENGINE_LIMIT_RPM 0x80
```

Номинальная частота вращения мотора.

Если флаг установлен, частота вращения ограничивается заданным номинальным значением.

### 6.1.2.40 `ENGINE_LIMIT_VOLT`

```
#define ENGINE_LIMIT_VOLT 0x20
```

Номинальное напряжение мотора.

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

### 6.1.2.41 `ENGINE_MAX_SPEED`

```
#define ENGINE_MAX_SPEED 0x04
```

Флаг максимальной скорости.

Если флаг установлен, движение происходит на максимальной скорости.

### 6.1.2.42 `ENGINE_REVERSE`

```
#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.

#### 6.1.2.43 ENGINE\_TYPE\_2DC

```
#define ENGINE_TYPE_2DC 0x02
```

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

#### 6.1.2.44 ENGINE\_TYPE\_BRUSHLESS

```
#define ENGINE_TYPE_BRUSHLESS 0x05
```

Бесщеточный мотор.

#### 6.1.2.45 ENGINE\_TYPE\_DC

```
#define ENGINE_TYPE_DC 0x01
```

Мотор постоянного тока.

#### 6.1.2.46 ENGINE\_TYPE\_NONE

```
#define ENGINE_TYPE_NONE 0x00
```

Это значение не нужно использовать.

#### 6.1.2.47 ENGINE\_TYPE\_STEP

```
#define ENGINE_TYPE_STEP 0x03
```

Шаговый мотор.

#### 6.1.2.48 ENGINE\_TYPE\_TEST

```
#define ENGINE_TYPE_TEST 0x04
```

Продолжительность включения фиксирована.

Используется только производителем.

#### 6.1.2.49 ENUMERATE\_PROBE

```
#define ENUMERATE_PROBE 0x01
```

Проверять, является ли устройство XIMC-совместимым.

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

#### 6.1.2.50 `EXTIO_SETUP_INVERT`

```
#define EXTIO_SETUP_INVERT 0x02
```

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

#### 6.1.2.51 `EXTIO_SETUP_MODE_IN_ALARM`

```
#define EXTIO_SETUP_MODE_IN_ALARM 0x05
```

Войти в состояние ALARM при переходе сигнала в активное состояние.

#### 6.1.2.52 `EXTIO_SETUP_MODE_IN_BITS`

```
#define EXTIO_SETUP_MODE_IN_BITS 0x0F
```

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

#### 6.1.2.53 `EXTIO_SETUP_MODE_IN_HOME`

```
#define EXTIO_SETUP_MODE_IN_HOME 0x04
```

Выполняется команда HOME.

#### 6.1.2.54 `EXTIO_SETUP_MODE_IN_MOVR`

```
#define EXTIO_SETUP_MODE_IN_MOVR 0x03
```

Выполняется команда MOVR с последними настройками.

#### 6.1.2.55 `EXTIO_SETUP_MODE_IN_NOP`

```
#define EXTIO_SETUP_MODE_IN_NOP 0x00
```

Ничего не делать.

#### 6.1.2.56 `EXTIO_SETUP_MODE_IN_PWOF`

```
#define EXTIO_SETUP_MODE_IN_PWOF 0x02
```

Выполняет команду PWOF, обесточивая обмотки двигателя.

#### 6.1.2.57 EXTIO\_SETUP\_MODE\_IN\_STOP

```
#define EXTIO_SETUP_MODE_IN_STOP 0x01
```

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

#### 6.1.2.58 EXTIO\_SETUP\_MODE\_OUT\_ALARM

```
#define EXTIO_SETUP_MODE_OUT_ALARM 0x30
```

Ножка находится в активном состоянии при нахождении в состоянии ALARM.

#### 6.1.2.59 EXTIO\_SETUP\_MODE\_OUT\_BITS

```
#define EXTIO_SETUP_MODE_OUT_BITS 0xF0
```

Биты выбора поведения на выходе.

#### 6.1.2.60 EXTIO\_SETUP\_MODE\_OUT\_MOTOR\_ON

```
#define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40
```

Ножка находится в активном состоянии при подаче питания на обмотки.

#### 6.1.2.61 EXTIO\_SETUP\_MODE\_OUT\_MOVING

```
#define EXTIO_SETUP_MODE_OUT_MOVING 0x20
```

Ножка находится в активном состоянии при движении.

#### 6.1.2.62 EXTIO\_SETUP\_MODE\_OUT\_OFF

```
#define EXTIO_SETUP_MODE_OUT_OFF 0x00
```

Ножка всегда в неактивном состоянии.

#### 6.1.2.63 EXTIO\_SETUP\_MODE\_OUT\_ON

```
#define EXTIO_SETUP_MODE_OUT_ON 0x10
```

Ножка всегда в активном состоянии.

#### 6.1.2.64 EXTIO\_SETUP\_OUTPUT

```
#define EXTIO_SETUP_OUTPUT 0x01
```

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

#### 6.1.2.65 FEEDBACK\_EMF

```
#define FEEDBACK_EMF 0x04
```

Обратная связь по ЭДС.

#### 6.1.2.66 FEEDBACK\_ENC\_ADAPTIVE\_HOLDING

```
#define FEEDBACK_ENC_ADAPTIVE_HOLDING 0x02
```

Включает алгоритм адаптивного удержания.

#### 6.1.2.67 FEEDBACK\_ENC\_FILTER\_BITS

```
#define FEEDBACK_ENC_FILTER_BITS 0x30
```

Биты, отвечающие за настройку внутреннего фильтра энкодерного сигнала.

#### 6.1.2.68 FEEDBACK\_ENC\_FILTER\_MEDIUM

```
#define FEEDBACK_ENC_FILTER_MEDIUM 0x20
```

Средняя фильтрация шумов: максимальная частота сигнала энкодера 1 МГц.

#### 6.1.2.69 FEEDBACK\_ENC\_FILTER\_NONE

```
#define FEEDBACK_ENC_FILTER_NONE 0x00
```

Выключает внутренний фильтр сигнала энкодера.

#### 6.1.2.70 FEEDBACK\_ENC\_FILTER\_STRONG

```
#define FEEDBACK_ENC_FILTER_STRONG 0x30
```

Сильная фильтрация шумов: максимальная частота сигнала энкодера 300 кГц.



#### 6.1.2.71 `FEEDBACK_ENC_FILTER_WEAK`

```
#define FEEDBACK_ENC_FILTER_WEAK 0x10
```

Слабая фильтрация шумов: максимальная частота сигнала энкодера 3 МГц.

#### 6.1.2.72 `FEEDBACK_ENC_REVERSE`

```
#define FEEDBACK_ENC_REVERSE 0x01
```

Обратный счет у энкодера.

#### 6.1.2.73 `FEEDBACK_ENC_TYPE_AUTO`

```
#define FEEDBACK_ENC_TYPE_AUTO 0x00
```

Определяет тип энкодера автоматически.

#### 6.1.2.74 `FEEDBACK_ENC_TYPE_BITS`

```
#define FEEDBACK_ENC_TYPE_BITS 0xC0
```

Биты, отвечающие за тип энкодера.

#### 6.1.2.75 `FEEDBACK_ENC_TYPE_DIFFERENTIAL`

```
#define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80
```

Дифференциальный энкодер.

#### 6.1.2.76 `FEEDBACK_ENC_TYPE_SINGLE_ENDED`

```
#define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40
```

Недифференциальный энкодер.

#### 6.1.2.77 `FEEDBACK_ENCODER`

```
#define FEEDBACK_ENCODER 0x01
```

Обратная связь с помощью энкодера.

#### 6.1.2.78 `FEEDBACK_ENCODER_MEDIATED`

```
#define FEEDBACK_ENCODER_MEDIATED 0x06
```

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

#### 6.1.2.79 `FEEDBACK_NONE`

```
#define FEEDBACK_NONE 0x05
```

Обратная связь отсутствует.

#### 6.1.2.80 `H_BRIDGE_ALERT`

```
#define H_BRIDGE_ALERT 0x04
```

Если установлен, то выключать силовую часть при сигнале неполадки в одном из транзисторных мостов.

#### 6.1.2.81 `HOME_DIR_FIRST`

```
#define HOME_DIR_FIRST 0x001
```

Определяет направление первоначального движения мотора после поступления команды `HOME`.

Если флаг установлен - вправо; иначе - влево.

#### 6.1.2.82 `HOME_DIR_SECOND`

```
#define HOME_DIR_SECOND 0x002
```

Определяет направление второго движения мотора.

Если флаг установлен - вправо; иначе - влево.

#### 6.1.2.83 `HOME_HALF_MV`

```
#define HOME_HALF_MV 0x008
```

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

#### 6.1.2.84 HOME\_MV\_SEC\_EN

```
#define HOME_MV_SEC_EN 0x004
```

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

#### 6.1.2.85 HOME\_STOP\_FIRST\_BITS

```
#define HOME_STOP_FIRST_BITS 0x030
```

Биты, отвечающие за выбор сигнала завершения первого движения.

#### 6.1.2.86 HOME\_STOP\_FIRST\_LIM

```
#define HOME_STOP_FIRST_LIM 0x030
```

Первое движение завершается по сигналу с концевого переключателя.

#### 6.1.2.87 HOME\_STOP\_FIRST\_REV

```
#define HOME_STOP_FIRST_REV 0x010
```

Первое движение завершается по сигналу с Revolution sensor.

#### 6.1.2.88 HOME\_STOP\_FIRST\_SYN

```
#define HOME_STOP_FIRST_SYN 0x020
```

Первое движение завершается по сигналу со входа синхронизации.

#### 6.1.2.89 HOME\_STOP\_SECOND\_BITS

```
#define HOME_STOP_SECOND_BITS 0x0C0
```

Биты, отвечающие за выбор сигнала завершения второго движения.

#### 6.1.2.90 HOME\_STOP\_SECOND\_LIM

```
#define HOME_STOP_SECOND_LIM 0x0C0
```

Второе движение завершается по сигналу с концевого переключателя.

#### 6.1.2.91 HOME\_STOP\_SECOND\_REV

```
#define HOME_STOP_SECOND_REV 0x040
```

Второе движение завершается по сигналу с Revolution sensor.

#### 6.1.2.92 HOME\_STOP\_SECOND\_SYN

```
#define HOME_STOP_SECOND_SYN 0x080
```

Второе движение завершается по сигналу со входа синхронизации.

#### 6.1.2.93 HOME\_USE\_FAST

```
#define HOME_USE_FAST 0x100
```

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

#### 6.1.2.94 JOY\_REVERSE

```
#define JOY_REVERSE 0x01
```

Реверс воздействия джойстика.

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

#### 6.1.2.95 LOW\_UPWR\_PROTECTION

```
#define LOW_UPWR_PROTECTION 0x02
```

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

#### 6.1.2.96 LS\_SHORTED

```
#define LS_SHORTED 0x10
```

Если флаг установлен, то концевые переключатели замкнуты.

#### 6.1.2.97 MICROSTEP\_MODE\_FRAC\_128

```
#define MICROSTEP_MODE_FRAC_128 0x08
```

Деление шага 1/128.

**6.1.2.98 MICROSTEP\_MODE\_FRAC\_16**

```
#define MICROSTEP_MODE_FRAC_16 0x05
```

Деление шага 1/16.

**6.1.2.99 MICROSTEP\_MODE\_FRAC\_2**

```
#define MICROSTEP_MODE_FRAC_2 0x02
```

Деление шага 1/2.

**6.1.2.100 MICROSTEP\_MODE\_FRAC\_256**

```
#define MICROSTEP_MODE_FRAC_256 0x09
```

Деление шага 1/256.

**6.1.2.101 MICROSTEP\_MODE\_FRAC\_32**

```
#define MICROSTEP_MODE_FRAC_32 0x06
```

Деление шага 1/32.

**6.1.2.102 MICROSTEP\_MODE\_FRAC\_4**

```
#define MICROSTEP_MODE_FRAC_4 0x03
```

Деление шага 1/4.

**6.1.2.103 MICROSTEP\_MODE\_FRAC\_64**

```
#define MICROSTEP_MODE_FRAC_64 0x07
```

Деление шага 1/64.

**6.1.2.104 MICROSTEP\_MODE\_FRAC\_8**

```
#define MICROSTEP_MODE_FRAC_8 0x04
```

Деление шага 1/8.

**6.1.2.105 MICROSTEP\_MODE\_FULL**

```
#define MICROSTEP_MODE_FULL 0x01
```

Полношаговый режим.

**6.1.2.106 MOVE\_STATE\_ANTIPLAY**

```
#define MOVE_STATE_ANTIPLAY 0x04
```

Выполняется компенсация люфта, если флаг установлен.

**6.1.2.107 MOVE\_STATE\_MOVING**

```
#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.

**6.1.2.108 MOVE\_STATE\_TARGET\_SPEED**

```
#define MOVE_STATE_TARGET_SPEED 0x02
```

Флаг устанавливается при достижении заданной скорости.

**6.1.2.109 MVCMD\_ERROR**

```
#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.

**6.1.2.110 MVCMD\_HOME**

```
#define MVCMD_HOME 0x06
```

Команда home.

**6.1.2.111 MVCMD\_LEFT**

```
#define MVCMD_LEFT 0x03
```

Команда left.

**6.1.2.112 MVCMD\_LOFT**

```
#define MVCMD_LOFT 0x07
```

Команда loft.

**6.1.2.113 MVCMD\_MOVE**

```
#define MVCMD_MOVE 0x01
```

Команда move.

**6.1.2.114 MVCMD\_MOVR**

```
#define MVCMD_MOVR 0x02
```

Команда movr.

**6.1.2.115 MVCMD\_NAME\_BITS**

```
#define MVCMD_NAME_BITS 0x3F
```

Битовая маска активной команды.

**6.1.2.116 MVCMD\_RIGHT**

```
#define MVCMD_RIGHT 0x04
```

Команда rigt.

**6.1.2.117 MVCMD\_RUNNING**

```
#define MVCMD_RUNNING 0x80
```

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

**6.1.2.118 MVCMD\_SSTP**

```
#define MVCMD_SSTP 0x08
```

Команда плавной остановки(SSTP).

**6.1.2.119 MVCMD\_STOP**

```
#define MVCMD_STOP 0x05
```

Команда stop.

**6.1.2.120 MVCMD\_UKNWN**

```
#define MVCMD_UKNWN 0x00
```

Неизвестная команда.

**6.1.2.121 POWER\_OFF\_ENABLED**

```
#define POWER_OFF_ENABLED 0x02
```

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

Иначе - не снимать.

**6.1.2.122 POWER\_REDUCT\_ENABLED**

```
#define POWER_REDUCT_ENABLED 0x01
```

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

Иначе - не уменьшать.

**6.1.2.123 POWER\_SMOOTH\_CURRENT**

```
#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.

**6.1.2.124 PWR\_STATE\_MAX**

```
#define PWR_STATE_MAX 0x05
```

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

**6.1.2.125 PWR\_STATE\_NORM**

```
#define PWR_STATE_NORM 0x03
```

Обмотки запитаны номинальным током.



**6.1.2.126 PWR\_STATE\_OFF**

```
#define PWR_STATE_OFF 0x01
```

Обмотки мотора разомкнуты и не управляются драйвером.

**6.1.2.127 PWR\_STATE\_REDUCT**

```
#define PWR_STATE_REDUCT 0x04
```

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

**6.1.2.128 PWR\_STATE\_UNKNOWN**

```
#define PWR_STATE_UNKNOWN 0x00
```

Неизвестное состояние, которое не должно никогда реализовываться.

**6.1.2.129 REV\_SENS\_INV**

```
#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.

**6.1.2.130 RPM\_DIV\_1000**

```
#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.

**6.1.2.131 SETPOS\_IGNORE\_ENCODER**

```
#define SETPOS_IGNORE_ENCODER 0x02
```

Если установлен, то счётчик энкодера не обновляется.

**6.1.2.132 SETPOS\_IGNORE\_POSITION**

```
#define SETPOS_IGNORE_POSITION 0x01
```

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

**6.1.2.133 STATE\_ALARM**

```
#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.

**6.1.2.134 STATE\_BORDERS\_SWAP\_MISSET**

```
#define STATE_BORDERS_SWAP_MISSET 0x0008000
```

Достижение неверной границы.

**6.1.2.135 STATE\_BRAKE**

```
#define STATE_BRAKE 0x0200
```

Состояние вывода управления тормозом.

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

**6.1.2.136 STATE\_BUTTON\_LEFT**

```
#define STATE_BUTTON_LEFT 0x0008
```

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

**6.1.2.137 STATE\_BUTTON\_RIGHT**

```
#define STATE_BUTTON_RIGHT 0x0004
```

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

**6.1.2.138 STATE\_CONTR**

```
#define STATE_CONTR 0x000003F
```

Флаги состояния контроллера.

**6.1.2.139 STATE\_CONTROLLER\_OVERHEAT**

```
#define STATE_CONTROLLER_OVERHEAT 0x0000200
```

Перегрелась микросхема контроллера.

#### 6.1.2.140 `STATE_CTP_ERROR`

```
#define STATE_CTP_ERROR 0x00000080
```

Контроль позиции нарушен(используется только с шаговым двигателем).

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

#### 6.1.2.141 `STATE_DIG_SIGNAL`

```
#define STATE_DIG_SIGNAL 0xFFFF
```

Флаги цифровых сигналов.

#### 6.1.2.142 `STATE_EEPROM_CONNECTED`

```
#define STATE_EEPROM_CONNECTED 0x00000010
```

Подключена память EEPROM с настройками.

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

#### 6.1.2.143 `STATE_ENC_A`

```
#define STATE_ENC_A 0x2000
```

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

#### 6.1.2.144 `STATE_ENC_B`

```
#define STATE_ENC_B 0x4000
```

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

#### 6.1.2.145 `STATE_ENGINE_RESPONSE_ERROR`

```
#define STATE_ENGINE_RESPONSE_ERROR 0x08000000
```

Ошибка реакции двигателя на управляющее воздействие.

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

**6.1.2.146 STATE\_ERRC**

```
#define STATE_ERRC 0x0000001
```

Недопустимая команда.

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

**6.1.2.147 STATE\_ERRD**

```
#define STATE_ERRD 0x0000002
```

Обнаружена ошибка целостности данных.

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

**6.1.2.148 STATE\_ERRV**

```
#define STATE_ERRV 0x0000004
```

Недопустимое значение данных.

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

**6.1.2.149 STATE\_EXTIO\_ALARM**

```
#define STATE_EXTIO_ALARM 0x1000000
```

Ошибка вызвана внешним входным сигналом EXTIO.

**6.1.2.150 STATE\_GPIO\_LEVEL**

```
#define STATE_GPIO_LEVEL 0x0020
```

Состояние ввода/вывода общего назначения.

**6.1.2.151 STATE\_GPIO\_PINOUT**

```
#define STATE_GPIO_PINOUT 0x0010
```

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

**6.1.2.152 STATE\_IS\_HOMED**

```
#define STATE_IS_HOMED 0x0000020
```

Калибровка выполнена.

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

**6.1.2.153 STATE\_LEFT\_EDGE**

```
#define STATE_LEFT_EDGE 0x0002
```

Достижение левой границы.

**6.1.2.154 STATE\_LOW\_USB\_VOLTAGE**

```
#define STATE_LOW_USB_VOLTAGE 0x0002000
```

Устарело.

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

**6.1.2.155 STATE\_OVERLOAD\_POWER\_CURRENT**

```
#define STATE_OVERLOAD_POWER_CURRENT 0x0000800
```

Превышен максимальный ток потребления силовой части.

**6.1.2.156 STATE\_OVERLOAD\_POWER\_VOLTAGE**

```
#define STATE_OVERLOAD_POWER_VOLTAGE 0x0000400
```

Превышено напряжение на силовой части.

**6.1.2.157 STATE\_OVERLOAD\_USB\_CURRENT**

```
#define STATE_OVERLOAD_USB_CURRENT 0x0004000
```

Устарело.

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

**6.1.2.158 STATE\_OVERLOAD\_USB\_VOLTAGE**

```
#define STATE_OVERLOAD_USB_VOLTAGE 0x0001000
```

Устарело.

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

**6.1.2.159 STATE\_POWER\_OVERHEAT**

```
#define STATE_POWER_OVERHEAT 0x0000100
```

Перегрев силового драйвера.

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

**6.1.2.160 STATE\_REV\_SENSOR**

```
#define STATE_REV_SENSOR 0x0400
```

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

**6.1.2.161 STATE\_RIGHT\_EDGE**

```
#define STATE_RIGHT_EDGE 0x0001
```

Достижение правой границы.

**6.1.2.162 STATE\_SECUR**

```
#define STATE_SECUR 0x1B3FFC0
```

Флаги опасности.

**6.1.2.163 STATE\_SYNC\_INPUT**

```
#define STATE_SYNC_INPUT 0x0800
```

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

**6.1.2.164 STATE\_SYNC\_OUTPUT**

```
#define STATE_SYNC_OUTPUT 0x1000
```

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

**6.1.2.165 STATE\_WINDING\_RES\_MISMATCH**

```
#define STATE_WINDING_RES_MISMATCH 0x0100000
```

Сопротивления обмоток слишком сильно отличаются друг от друга.

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

**6.1.2.166 SYNCIN\_ENABLED**

```
#define SYNCIN_ENABLED 0x01
```

Включение необходимости импульса синхронизации для начала движения.

**6.1.2.167 SYNCIN\_INVERT**

```
#define SYNCIN_INVERT 0x02
```

Если установлен - срабатывает по переходу из 1 в 0.

Иначе - из 0 в 1.

**6.1.2.168 SYNCOUT\_ENABLED**

```
#define SYNCOUT_ENABLED 0x01
```

Синхронизация выхода работает согласно настройкам, если флаг установлен.

В ином случае значение выхода фиксировано и подчиняется SYNCOUT\_STATE.

**6.1.2.169 SYNCOUT\_IN\_STEPS**

```
#define SYNCOUT_IN_STEPS 0x08
```

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

**6.1.2.170 SYNCOUT\_INVERT**

```
#define SYNCOUT_INVERT 0x04
```

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

**6.1.2.171 SYNCOUT\_ONPERIOD**

```
#define SYNCOUT_ONPERIOD 0x40
```

Выдает импульс синхронизации после прохождения SyncOutPeriod отсчётов.

**6.1.2.172 SYNCOUT\_ONSTART**

```
#define SYNCOUT_ONSTART 0x10
```

Генерация синхронизирующего импульса при начале движения.

**6.1.2.173 SYNCOUT\_ONSTOP**

```
#define SYNCOUT_ONSTOP 0x20
```

Генерация синхронизирующего импульса при остановке.

**6.1.2.174 SYNCOUT\_STATE**

```
#define SYNCOUT_STATE 0x02
```

Когда значение выхода управляется напрямую (см.

флаг SYNCOUT\_ENABLED), значение на выходе соответствует значению этого флага.

**6.1.2.175 TS\_TYPE\_BITS**

```
#define TS_TYPE_BITS 0x07
```

Биты, отвечающие за тип температурного датчика.

**6.1.2.176 UART\_PARITY\_BITS**

```
#define UART_PARITY_BITS 0x03
```

Биты, отвечающие за выбор четности.

**6.1.2.177 WIND\_A\_STATE\_ABSENT**

```
#define WIND_A_STATE_ABSENT 0x00
```

Обмотка A не подключена.



**6.1.2.178 WIND\_A\_STATE\_MALFUNC**

```
#define WIND_A_STATE_MALFUNC 0x02
```

Короткое замыкание на обмотке A.

**6.1.2.179 WIND\_A\_STATE\_OK**

```
#define WIND_A_STATE_OK 0x03
```

Обмотка A работает адекватно.

**6.1.2.180 WIND\_A\_STATE\_UNKNOWN**

```
#define WIND_A_STATE_UNKNOWN 0x01
```

Состояние обмотки A неизвестно.

**6.1.2.181 WIND\_B\_STATE\_ABSENT**

```
#define WIND_B_STATE_ABSENT 0x00
```

Обмотка B не подключена.

**6.1.2.182 WIND\_B\_STATE\_MALFUNC**

```
#define WIND_B_STATE_MALFUNC 0x20
```

Короткое замыкание на обмотке B.

**6.1.2.183 WIND\_B\_STATE\_OK**

```
#define WIND_B_STATE_OK 0x30
```

Обмотка B работает адекватно.

**6.1.2.184 WIND\_B\_STATE\_UNKNOWN**

```
#define WIND_B_STATE_UNKNOWN 0x10
```

Состояние обмотки B неизвестно.

### 6.1.2.185 XIMC\_API

```
#define XIMC_API
```

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

### 6.1.2.186 XIMC\_CALLCONV

```
#define XIMC_CALLCONV
```

Библиотека вызывающая условные макросы.

### 6.1.2.187 XIMC\_RETTYPE

```
#define XIMC_RETTYPE void*
```

Возвращаемый тип потока.

## 6.1.3 Типы

### 6.1.3.1 calibration\_t

```
typedef struct calibration_t calibration_t
```

Структура калибровок

Где найти все значения для расчета?

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

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

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

Ускорение, замедление и скорость в режиме антилюфта считаются аналогично

### 6.1.3.2 device\_enumeration\_t

```
typedef uint32_t device_enumeration_t
```

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

### 6.1.3.3 device\_network\_information\_t

```
typedef struct device_network_information_t device_network_information_t
```

Структура данных с информацией о сетевом устройстве.

### 6.1.3.4 logging\_callback\_t

```
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

### 6.1.3.5 result\_t

```
typedef int result_t
```

Тип, определяющий результат выполнения команды.

## 6.1.4 Функции

### 6.1.4.1 close\_device()

```
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

#### 6.1.4.2 `command_clear_fram()`

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

Очистка FRAM памяти контроллера.

Функция используется только производителем.

Аргументы

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

#### 6.1.4.3 `command_eeread_settings()`

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

Чтение настроек контроллера из EEPROM памяти позиционера.

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

Аргументы

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

#### 6.1.4.4 `command_eesave_settings()`

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

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

Аргументы

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

#### 6.1.4.5 `command_home()`

```
result_t XIMC_API command_home (  
    device_t id)
```

Движение в домашнюю позицию.

Алгоритм движения:

- 1) Двигает мотор согласно скоростям FastHome, uFastHome и флагу HOME\_DIR\_FAST до достижения концевого выключателя, если флаг HOME\_STOP\_ENDS установлен. Или двигает до достижения сигнала с входа синхронизации, если установлен флаг HOME\_STOP\_SYNC. Или до поступления сигнала с датчика оборотов, если установлен флаг HOME\_STOP\_REV\_SN
- 2) далее двигает согласно скоростям SlowHome, uSlowHome и флагу HOME\_DIR\_SLOW до достижения сигнала с входа синхронизации, если установлен флаг HOME\_MV\_SEC. Если флаг HOME\_MV\_SEC сброшен, пропускаем этот пункт.
- 3) далее двигает мотор согласно скоростям FastHome, uFastHome и флагу HOME\_DIR\_SLOW на расстояние HomeDelta, uHomeDelta.

Описание флагов и переменных см. описание команд GHOM/SHOM

Аргументы

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

См. также

```
home_settings_t  
get_home_settings  
set_home_settings
```

#### 6.1.4.6 `command_homezero()`

```
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
out	<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.

#### 6.1.4.7 command\_left()

```
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
-----------	--------------------------

#### 6.1.4.8 command\_loft()

```
result_t XIMC_API command_loft (
    device_t id)
```

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

Аргументы

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

#### 6.1.4.9 command\_move()

```
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).

#### 6.1.4.10 command\_move\_calb()

```
result_t XIMC_API command_move_calb (
    device_t id,
    float Position,
    const calibration_t * calibration)
```

Перемещение в позицию с использованием пользовательских единиц.

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

Аргументы

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

Заметки

Параметр Position корректируется таблицей коррекции.

#### 6.1.4.11 command\_movr()

```
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 DC 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

#### 6.1.4.12 command\_movr\_calb()

```
result_t XIMC_API command_movr_calb (  
    device_t id,  
    float DeltaPosition,  
    const calibration_t * calibration)
```

Перемещение на заданное смещение с использованием пользовательских единиц.

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



## Аргументы

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

## Заметки

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

**6.1.4.13 `command_power_off()`**

```
result_t XIMC_API command_power_off (
    device_t id)
```

Немедленное отключение питания двигателя вне зависимости от его состояния.

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

## Аргументы

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

См. также

```
get_power_settings
set_power_settings
```

**6.1.4.14 `command_read_robust_settings()`**

```
result_t XIMC_API command_read_robust_settings (
    device_t id)
```

Чтение важных настроек (калибровочные коэффициенты и т.

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

## Аргументы

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

**6.1.4.15 `command_read_settings()`**

```
result_t XIMC_API command_read_settings (
    device_t id)
```

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

Аргументы

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

#### 6.1.4.16 `command_reset()`

```
result_t XIMC_API command_reset (  
    device_t id)
```

Перезагрузка контроллера.

Функция используется только производителем.

Аргументы

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

#### 6.1.4.17 `command_right()`

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

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

Аргументы

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

#### 6.1.4.18 `command_save_robust_settings()`

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

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

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

Аргументы

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

#### 6.1.4.19 `command_save_settings()`

```
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
-----------	--------------------------

#### 6.1.4.20 command\_sstp()

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

Плавная остановка.

Двигатель останавливается с ускорением замедления.

Аргументы

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

#### 6.1.4.21 command\_start\_measurements()

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

Начать измерения и буферизацию скорости, ошибки следования.

Аргументы

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

#### 6.1.4.22 command\_stop()

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

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

При вызове этой команды сбрасывается флаг ALARM.

Аргументы

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

#### 6.1.4.23 command\_update\_firmware()

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

Обновление прошивки.

Команда только для производителя.

Аргументы

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

#### 6.1.4.24 `command_wait_for_stop()`

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

Ожидание остановки контроллера

Аргументы

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

#### 6.1.4.25 `command_zero()`

```
result_t XIMC_API command_zero (
    device_t id)
```

Устанавливает текущую позицию равной 0.

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

Аргументы

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

## 6.1.4.26 enumerate\_devices()

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

Поиск и составление списка доступных устройств.

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

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

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

Аргументы

in	<i>enumerate_flags</i>	<p>набор флагов, задающих режимы поиска. Флаги могут применяться совместно через побитовое "ИЛИ".</p> <ul style="list-style-type: none"> <li>• <b>ENUMERATE_NETWORK</b> - включает поиск сетевых устройств. Если флаг установлен, сетевые устройства будут добавлены в общий список. Если флаг не установлен, в списке будут только устройства, подключенные к данному компьютеру.</li> <li>• <b>ENUMERATE_ALL_COM</b> - при включенной опции опрашивает все устройства типа COM-порт в системе. При отключенной опции опрашивает только устройства, имена которых соответствуют маске устройств XIMC ("XIMC Motor Controller" в Windows, /dev/ximc/ и /dev/ttyACM/ на Linux/Mac).</li> <li>• <b>ENUMERATE_PROBE</b> - включает проверку устройств и сбор дополнительной информации (серийный номер, версию, модель, имя...). Если данный флаг установлен, в список будут добавлены только устройства, которые гарантированно можно открыть, но могут не попадать устройства, подключенные через RS232-преобразователи. Если флаг не установлен, то в списке будет больше устройств (в частности, попадут устройства, явным образом перечисленные в <i>hints</i>), но доступность и совместимость с данной библиотекой не гарантируется.</li> </ul>
----	------------------------	---

## Аргументы

in	<i>hints</i>	<p>дополнительная информация для повышения эффективности поиска. Имеет смысл использовать в случае сложной сетевой конфигурации, когда автоматический поиск может находить не всё. Формат - строка "ключ1=значение1\ключ2=значение2". Неизвестные ключи игнорируются. Один ключ может иметь несколько значений, которые перечисляются через запятую: ключ=значение1, значение2, значение3. Допустимые ключи:</p> <ul style="list-style-type: none"> <li>• <code>addr</code> - список URЛОВ сетевых контроллеров или серверов с подключенными контроллерами. Поле применяется совместно с флагом <code>ENUMERATE_NETWORK</code>. Протоколы и форматы адресов: <ul style="list-style-type: none"> <li>– <code>xi-tcp://&lt;ip-адрес&gt;</code> - сетевые контроллеры и контроллеры, подключенные через Ethernet-RS232 преобразователи. Если флаг <code>ENUMERATE_PROBE</code> не установлен, все перечисленные устройства попадут в список.</li> <li>– <code>xi-net://&lt;ip-адрес&gt;</code> - сетевые многоосевые системы, <code>xi-net</code> сервера. Запрос информации о наличии контроллеров по этим адресам будет сделан независимо от результатов автоматической процедуры сетевого поиска.</li> </ul> </li> <li>• <code>adapter_addr</code> - список IP-адресов локальных сетевых адаптеров, через который должен осуществляться поиск. Если ключ отсутствует или ни одного адаптера не указано, то поиск производится на всех адаптерах.</li> </ul>
----	--------------	---

## Возвращает

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

## Примеры использования:

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

**6.1.4.27 free\_enumerate\_devices()**

```
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
----	---------------------------	--

**6.1.4.28 get\_accessories\_settings()**

```
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

**6.1.4.29 get\_analog\_data()**

```
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

**6.1.4.30 get\_bootloader\_version()**

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

Чтение номера версии загрузчика контроллера.

Аргументы

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

#### 6.1.4.31 get\_brake\_settings()

```
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

#### 6.1.4.32 get\_calibration\_settings()

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

Команда чтения калибровочных коэффициентов.

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

См. также

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

Аргументы

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



#### 6.1.4.33 get\_chart\_data()

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

Команда чтения состояния обмоток и других не часто используемых данных.

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

См. также

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

Аргументы

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

#### 6.1.4.34 get\_control\_settings()

```
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.

#### 6.1.4.35 get\_control\_settings\_calb()

```
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

#### 6.1.4.36 get\_controller\_name()

```
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

#### 6.1.4.37 get\_ctp\_settings()

```
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

#### 6.1.4.38 get\_debug\_read()

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

Чтение данных из прошивки для отладки и поиска неисправностей.

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

Аргументы

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

#### 6.1.4.39 get\_device\_count()

```
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
----	---------------------------	--

#### 6.1.4.40 get\_device\_information()

```
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](#)

#### 6.1.4.41 get\_device\_name()

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

Возвращает имя подключенного устройства из перечисления устройств.

Возвращает имя устройства с номером *device\_index*.

Аргументы

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

#### 6.1.4.42 `get_edges_settings()`

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

Чтение настроек границ и концевых выключателей.

См. также

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

Аргументы

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

#### 6.1.4.43 `get_edges_settings_calb()`

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

Чтение настроек границ и концевых выключателей с использованием пользовательских единиц.

См. также

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

Аргументы

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

Заметки

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

#### 6.1.4.44 get\_emf\_settings()

```
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

#### 6.1.4.45 get\_encoder\_information()

```
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

#### 6.1.4.46 get\_encoder\_settings()

```
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

#### 6.1.4.47 get\_engine\_advanced\_setup()

```
result_t XIMC_API get_engine_advanced_setup (  
    device_t id,  
    engine_advanced_setup_t * engine_advanced_setup)
```

Чтение расширенных настроек.

Только для производителя.

См. также

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

Аргументы

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

#### 6.1.4.48 get\_engine\_settings()

```
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

#### 6.1.4.49 get\_engine\_settings\_calb()

```
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

#### 6.1.4.50 get\_entype\_settings()

```
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

#### 6.1.4.51 get\_enumerate\_device\_controller\_name()

```
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 устройства

#### 6.1.4.52 get\_enumerate\_device\_information()

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

Возвращает информацию о подключенном устройстве из перечисления устройств.

Возвращает информацию о устройстве с номером *device\_index*.



Аргументы

in	<code>device__enumeration</code>	закрытый указатель на данные о перечисленных устройствах
in	<code>device__index</code>	номер устройства
out	<code>device__information</code>	информация об устройстве

#### 6.1.4.53 `get__enumerate__device__network__information()`

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

Возвращает сетевую информацию о подключенном устройстве из перечисления устройств.

Возвращает сетевую информацию о устройстве с номером `device__index`.

Аргументы

in	<code>device__enumeration</code>	закрытый указатель на данные о перечисленных устройствах
in	<code>device__index</code>	номер устройства
out	<code>device__network__information</code>	сетевая информация об устройстве

#### 6.1.4.54 `get__enumerate__device__serial()`

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

Возвращает серийный номер подключенного устройства из перечисления устройств.

Возвращает серийный номер устройства с номером `device__index`.

Аргументы

in	<code>device__enumeration</code>	закрытый указатель на данные о перечисленных устройствах
in	<code>device__index</code>	номер устройства
in	<code>serial</code>	серийный номер устройства

#### 6.1.4.55 `get__enumerate__device__stage__name()`

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

Возвращает имя подвигки для подключенного устройства из перечисления устройств.

Возвращает имя подвигки устройства с номером `device__index`.

Аргументы

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

#### 6.1.4.56 `get_extended_settings()`

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

Чтение расширенных настроек.

В настоящее время не используется.

См. также

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

Аргументы

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

#### 6.1.4.57 `get_extio_settings()`

```
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

#### 6.1.4.58 `get_feedback_settings()`

```
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.

#### 6.1.4.59 get\_firmware\_version()

```
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

#### 6.1.4.60 get\_gear\_information()

```
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

#### 6.1.4.61 get\_gear\_settings()

```
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

#### 6.1.4.62 get\_globally\_unique\_identifier()

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

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

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

Аргументы

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

#### 6.1.4.63 get\_hallsensor\_information()

```
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

#### 6.1.4.64 get\_hallsensor\_settings()

```
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

**6.1.4.65 get\_home\_settings()**

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

Команда чтения настроек для подхода в home position.

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

См. также

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

Аргументы

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

**6.1.4.66 get\_home\_settings\_calb()**

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

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

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

См. также

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

Аргументы

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

**6.1.4.67 get\_init\_random()**

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

Чтение случайного числа из контроллера.

Только для производителя.

Аргументы

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

#### 6.1.4.68 get\_joystick\_settings()

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

Чтение настроек джойстика.

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

Аргументы

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

#### 6.1.4.69 get\_measurements()

```
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.

#### 6.1.4.70 `get_motor_information()`

```
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

#### 6.1.4.71 `get_motor_settings()`

```
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

#### 6.1.4.72 `get_move_settings()`

```
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.

#### 6.1.4.73 `get_move_settings_calb()`

```
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

#### 6.1.4.74 get\_network\_settings()

```
result_t XIMC_API get_network_settings (
    device_t id,
    network_settings_t * network_settings)
```

Команда чтения сетевых настроек.

Только для производителя. Эта функция возвращает текущие сетевые настройки.

Аргументы

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

#### 6.1.4.75 get\_nonvolatile\_memory()

```
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

#### 6.1.4.76 get\_password\_settings()

```
result_t XIMC_API get_password_settings (
    device_t id,
    password_settings_t * password_settings)
```

Команда чтения пароля к веб-странице.

Только для производителя. Эта функция позволяет прочитать пользовательский пароль к веб-странице из памяти контроллера.



Аргументы

<i>UserPassword[20]</i>	Строчка-пароль для доступа к веб-странице
-------------------------	---

#### 6.1.4.77 get\_pid\_settings()

```
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

#### 6.1.4.78 get\_position()

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

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>the_get_position</i>	структура, содержащая позицию мотора.

#### 6.1.4.79 get\_position\_calb()

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

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>the_get_position_calb</i>	структура, содержащая позицию мотора.
	<i>calibration</i>	настройки пользовательских единиц

Заметки

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

#### 6.1.4.80 `get_power_settings()`

```
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

#### 6.1.4.81 `get_secure_settings()`

```
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`

#### 6.1.4.82 `get_serial_number()`

```
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

#### 6.1.4.83 get\_stage\_information()

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

Чтение информации о позиционере из EEPROM.

Не поддерживается.

Аргументы

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

#### 6.1.4.84 get\_stage\_name()

```
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

#### 6.1.4.85 get\_stage\_settings()

```
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

#### 6.1.4.86 get\_status()

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

Возвращает информацию о текущем состоянии устройства.

Аргументы

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

См. также

[get\\_status](#)

#### 6.1.4.87 get\_status\_calb()

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

Возвращает информацию о текущем состоянии устройства.

Аргументы

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

См. также

[get\\_status](#)

#### 6.1.4.88 get\_sync\_in\_settings()

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

Чтение настроек для входного импульса синхронизации.

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

См. также

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

Аргументы

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

#### 6.1.4.89 `get_sync_in_settings_calb()`

```
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

#### 6.1.4.90 `get_sync_out_settings()`

```
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.

#### 6.1.4.91 `get_sync_out_settings_calb()`

```
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

#### 6.1.4.92 get\_uart\_settings()

```
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

#### 6.1.4.93 goto\_firmware()

```
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.

#### 6.1.4.94 has\_firmware()

```
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

#### 6.1.4.95 logging\_callback\_stderr\_narrow()

```
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

#### 6.1.4.96 logging\_callback\_stderr\_wide()

```
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

#### 6.1.4.97 msec\_sleep()

```
void XIMC_API msec_sleep (  
    unsigned int msec)
```

Приостанавливает работу на указанное время

Аргументы

<i>msec</i>	время в миллисекундах
-------------	-----------------------

#### 6.1.4.98 open\_device()

```
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:///abs_path_to_file". На POSIX системах допускается пропуск "рутовского" слэша; например, "xi-emu:///home/user/virt_controller.bin". Для USB-COM устройства "port" это URI устройства в ОС. Например, "xi-com:\\\\\\\\\\\\\\\\COM3" в Windows (с учётом экранирования двойные обратные слэши преобразуются в одинарные) или "xi-com:///dev/ttyACM0" в Linux/Mac. Для сетевого устройства "host" это IPv4 адрес или полностью определённое имя домена, "serial" это серийный номер устройства в шестнадцатеричной системе. Например, "xi-net://192.168.0.1/00001234" или "xi-net://hostname.com/89ABCDEF". Для работы по TCP протоколу используйте "xi-tcp://<ip/host>:<port>". Например, "xi-tcp://192.168.0.1:1818". Для виртуального устройства "abs_file_to_file" это путь к файлу с сохранённым состоянием устройства. Если файл не существует, он будет создан и инициализирован значениями по умолчанию. Например, "xi-emu:///C:/dir/file.bin" в Windows или "xi-emu:///home/user/file.bin" в Linux/Mac.
-----------------	------------------	---

**6.1.4.99 probe\_device()**

```
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
-----------------	------------------	---------------------------------------

**6.1.4.100 reset\_locks()**

```
result_t XIMC_API reset_locks ()
```

Сбрасывает ошибку неправильной передачи данных.

**6.1.4.101 service\_command\_updf()**

```
result_t XIMC_API service_command_updf (
    device_t id)
```

Команда переводит контроллер в режим обновления прошивки.

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

**6.1.4.102 set\_accessories\_settings()**

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

Запись информации о дополнительных аксессуарах в EEPROM.

Функция должна использоваться только производителем.



Аргументы

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

#### 6.1.4.103 set\_brake\_settings()

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

Запись настроек управления тормозом.

Аргументы

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

#### 6.1.4.104 set\_calibration\_settings()

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

Команда записи калибровочных коэффициентов.

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

См. также

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

Аргументы

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

**6.1.4.105 set\_control\_settings()**

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

Запись настроек управления мотором.

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

Аргументы

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

**6.1.4.106 set\_control\_settings\_calb()**

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

Запись настроек управления мотором с использованием пользовательских единиц.

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

Аргументы

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

**6.1.4.107 set\_controller\_name()**

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

Запись пользовательского имени контроллера и настроек в FRAM.

Аргументы

	<i>id</i>	идентификатор устройства
in	<i>controller_information</i>	структура, содержащая информацию о контроллере

#### 6.1.4.108 set\_correction\_table()

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

Команда загрузки корректирующей таблицы из текстового файла.

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

Аргументы

	<i>id</i>	- идентификатор устройства
in	<i>namefile</i>	- путь до файла должен быть полным или относительным. Если параметр равен NULL, таблица коррекции будет очищена. Формат файла: два столбца, разделенных табуляцией. Заголовки столбцов строковые. Данные действительные, разделитель точка. Первый столбец - координата. Второй - отклонение, вызванное ошибкой механики. Максимальная длина таблицы 100 строк. Координаты должны быть отсортированы по возрастанию.

См. также

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

#### 6.1.4.109 set\_ctp\_settings()

```
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
in	<i>ctp_settings</i>	структура, содержащая настройки контроля позиции

#### 6.1.4.110 `set_debug_write()`

```
result_t XIMC_API set_debug_write (
    device_t id,
    const debug_write_t * debug_write)
```

Запись данных в прошивку для отладки и поиска неисправностей.

Команда только для производителя.

Аргументы

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

#### 6.1.4.111 `set_edges_settings()`

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

**6.1.4.112 set\_edges\_settings\_calb()**

```
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
in	<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.

**6.1.4.113 set\_emf\_settings()**

```
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
in	<i>emf_settings</i>	настройки EMF

**6.1.4.114 set\_encoder\_information()**

```
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
in	<i>encoder_information</i>	структура, содержащая информацию об энкодере

#### 6.1.4.115 set\_encoder\_settings()

```
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
in	<i>encoder_settings</i>	структура, содержащая настройки энкодера

#### 6.1.4.116 set\_engine\_advanced\_setup()

```
result_t XIMC_API set_engine_advanced_setup (
    device_t id,
    const engine_advanced_setup_t * engine_advanced_setup)
```

Запись расширенных настроек.

Только для производителя.

См. также

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

Аргументы

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

#### 6.1.4.117 set\_engine\_settings()

```
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
in	<i>engine_settings</i>	структура с настройками мотора

#### 6.1.4.118 `set_engine_settings_calb()`

```
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
in	<i>engine_settings_calb</i>	структура с настройками мотора
	<i>calibration</i>	настройки пользовательских единиц

#### 6.1.4.119 `set_entype_settings()`

```
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
in	<i>entype_settings</i>	структура, содержащая настройки типа мотора и типа силового драйвера

#### 6.1.4.120 `set_extended_settings()`

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

Запись расширенных настроек.

В настоящее время не используется.

См. также

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

Аргументы

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

#### 6.1.4.121 set\_extio\_settings()

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

Команда записи параметров настройки режимов внешнего ввода/вывода.

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

См. также

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

Аргументы

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

#### 6.1.4.122 set\_feedback\_settings()

```
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
in	<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.
in	<i>FeedbackType</i>	тип обратной связи
in	<i>FeedbackFlags</i>	флаги обратной связи
in	<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.



**6.1.4.123 set\_gear\_information()**

```
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
in	<i>gear_information</i>	структура, содержащая информацию о редукторе

**6.1.4.124 set\_gear\_settings()**

```
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
in	<i>gear_settings</i>	структура, содержащая настройки редуктора

**6.1.4.125 set\_hallsensor\_information()**

```
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
in	<i>hallsensor_information</i>	структура, содержащая информацию о датчиках Холла

**6.1.4.126 set\_hallsensor\_settings()**

```
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
in	<i>hallsensor_settings</i>	структура, содержащая настройки датчиков Холла

#### 6.1.4.127 set\_home\_settings()

```
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, используемых для калибровки позиции, в память контроллера.

См. также

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

Аргументы

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

#### 6.1.4.128 set\_home\_settings\_calb()

```
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, используемых для калибровки позиции, в память контроллера.

См. также

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

Аргументы

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

**6.1.4.129 set\_joystick\_settings()**

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

Запись настроек джойстика.

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

Аргументы

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

**6.1.4.130 set\_logging\_callback()**

```
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
-------------------------	---------------------------------------

**6.1.4.131 set\_motor\_information()**

```
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
in	<i>motor_information</i>	структура, содержащая информацию о двигателе

#### 6.1.4.132 set\_motor\_settings()

```
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
in	<i>motor_settings</i>	структура, содержащая настройки двигателя

#### 6.1.4.133 set\_move\_settings()

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

#### 6.1.4.134 set\_move\_settings\_calb()

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

#### 6.1.4.135 `set_network_settings()`

```
result_t XIMC_API set_network_settings (
    device_t id,
    const network_settings_t * network_settings)
```

Команда записи сетевых настроек.

Только для производителя. Эта функция меняет сетевые настройки на заданные.

Аргументы

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

#### 6.1.4.136 `set_nonvolatile_memory()`

```
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
in	<i>nonvolatile_memory</i>	структура, содержащая установленные пользовательские данные

#### 6.1.4.137 `set_password_settings()`

```
result_t XIMC_API set_password_settings (
    device_t id,
    const password_settings_t * password_settings)
```

Команда записи пароля к веб-странице.

Только для производителя. Эта функция меняет пользовательский пароль к веб-странице.

Аргументы

<i>UserPassword[20]</i>	Строчка-пароль для доступа к веб-странице
-------------------------	---

#### 6.1.4.138 set\_pid\_settings()

```
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
in	<i>pid_settings</i>	настройки ПИД

#### 6.1.4.139 set\_position()

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

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

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>the_set_position</i>	структура, содержащая позицию мотора.

#### 6.1.4.140 set\_position\_calb()

```
result_t XIMC_API set_position_calb (
    device_t id,
    const set_position_calb_t * the_set_position_calb,
    const calibration_t * calibration)
```

Устанавливает произвольное значение положения и значение энкодера всех двигателей с использованием пользовательских единиц.

Аргументы

	<i>id</i>	идентификатор устройства
out	<i>the_set_position_calb</i>	структура, содержащая позицию мотора.
	<i>calibration</i>	настройки пользовательских единиц

#### 6.1.4.141 set\_power\_settings()

```
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
in	<i>power_settings</i>	структура, содержащая настройки питания шагового мотора

#### 6.1.4.142 set\_secure\_settings()

```
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

#### 6.1.4.143 set\_serial\_number()

```
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. Используется только загрузчиком.

Аргументы

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

#### 6.1.4.144 set\_stage\_information()

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

Запись информации о позиционере в EEPROM.

Не поддерживается. Функция должна использоваться только производителем.

Аргументы

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

#### 6.1.4.145 set\_stage\_name()

```
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
in	<i>stage_name</i>	структура, содержащая установленное пользовательское имя позиционера

#### 6.1.4.146 set\_stage\_settings()

```
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
in	<i>stage_settings</i>	структура, содержащая настройки позиционера



**6.1.4.147 set\_sync\_in\_settings()**

```
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
in	<i>sync_in_settings</i>	настройки синхронизации

**6.1.4.148 set\_sync\_in\_settings\_calb()**

```
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
in	<i>sync_in_settings_calb</i>	настройки синхронизации
	<i>calibration</i>	настройки пользовательских единиц

**6.1.4.149 set\_sync\_out\_settings()**

```
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
in	<i>sync_out_settings</i>	настройки синхронизации

#### 6.1.4.150 set\_sync\_out\_settings\_calb()

```
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
in	<i>sync_out_settings_calb</i>	настройки синхронизации
	<i>calibration</i>	настройки пользовательских единиц

#### 6.1.4.151 set\_uart\_settings()

```
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
in	<i>uart_settings</i>	настройки UART

#### 6.1.4.152 write\_key()

```
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
in	<i>key</i>	ключ защиты. Диапазон: 0..4294967295

#### 6.1.4.153 ximc\_version()

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

Возвращает версию библиотеки

Аргументы

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

## 6.2 ximc.h

[См. документацию.](#)

```
00001 #ifndef INC_XIMC_H
00002 #define INC_XIMC_H
00003
00012
00025 #if defined(_WIN32) || defined(LABVIEW64_IMPORT) || defined(LABVIEW32_IMPORT) || defined(MATLAB_IMPORT)
00026     #define XIMC_API __stdcall
00027 #else
00028     #ifdef LIBXIMC_EXPORTS
00029         #define XIMC_API __attribute__((visibility("default")))
00030     #else
00031         #define XIMC_API
00032     #endif
00033 #endif
00034
00043 #if defined(_WIN32) || defined(LABVIEW64_IMPORT) || defined(LABVIEW32_IMPORT) || defined(MATLAB_IMPORT)
00044     #define XIMC_CALLCONV __stdcall
00045 #else
00046     #define XIMC_CALLCONV
00047 #endif
00048
00057 #if defined(_WIN32) || defined(LABVIEW64_IMPORT) || defined(LABVIEW32_IMPORT) || defined(MATLAB_IMPORT)
00058 #define XIMC_RETTYPE unsigned int
00059 #else
00060 #define XIMC_RETTYPE void*
00061 #endif
00062
00063
00064 #if !defined(XIMC_NO_STDINT)
00065
00066 #if ( (defined(_MSC_VER) && (_MSC_VER < 1600)) || defined(LABVIEW64_IMPORT) || defined(LABVIEW32_IMPORT)) &&
!defined(MATLAB_IMPORT)
00067 // msvc types burden
00068 typedef __int8 int8_t;
00069 typedef __int16 int16_t;
00070 typedef __int32 int32_t;
00071 typedef __int64 int64_t;
00072 typedef unsigned __int8 uint8_t;
00073 typedef unsigned __int16 uint16_t;
00074 typedef unsigned __int32 uint32_t;
00075 typedef unsigned __int64 uint64_t;
00076 #else
00077 #include <stdint.h>
00078 #endif
00079
00080 /* labview doesn't speak C99 */
00081 #if defined(LABVIEW64_IMPORT) || defined(LABVIEW32_IMPORT)
00082 typedef unsigned __int64 ulong_t;
```

```

00083 typedef __int64 long_t;
00084 #else
00085 typedef unsigned long long ulong_t;
00086 typedef long long long_t;
00087 #endif
00088
00089 #endif
00090
00091 #include <time.h>
00092
00093 #if defined(__cplusplus)
00094 extern "C"
00095 {
00096 #endif
00097
00098
00107     typedef int device_t;
00108
00117     typedef int result_t;
00118
00127     #if defined(_WIN64) || defined(_LP64_) || defined(LABVIEW64_IMPORT)
00128     typedef uint64_t device_enumeration_t;
00129     #else
00130     typedef uint32_t device_enumeration_t;
00131     #endif
00132     //typedef device_enumeration_t* pdevice_enumeration_t;
00133
00142 #define device_undefined -1
00143
00152
00161 #define result_ok 0
00162
00171 #define result_error -1
00172
00181 #define result_not_implemented -2
00182
00191 #define result_value_error -3
00192
00201 #define result_noddevice -4
00202
00204
00213
00222 #define LOGLEVEL_ERROR      0x01
00231 #define LOGLEVEL_WARNING    0x02
00240 #define LOGLEVEL_INFO       0x03
00249 #define LOGLEVEL_DEBUG      0x04
00251
00252
00296     typedef struct calibration_t
00297     {
00298         double A;
00326         unsigned int MicrostepMode;
00327     } calibration_t;
00328
00336     typedef struct device_network_information_t
00337     {
00338         uint32_t ipv4;
00339         char nodename[16];
00340
00341         uint32_t axis_state;
00342         char locker_username[16];
00343         char locker_nodename[16];
00344         time_t locked_time;
00345     } device_network_information_t;
00346
00347
00348
00350 #define LIBXIMC_VERSION 3.0.2
00352
00353
00355 #define LIBXIMC_PROTOCOL_VERSION 20.14
00357
00358
00359 /*
00360 -----
00361 BEGIN OF GENERATED struct declarations
00362 -----
00363 */
00364
00376 #define ENUMERATE_PROBE      0x01
00377 #define ENUMERATE_ALL_COM    0x02
00378 #define ENUMERATE_NETWORK    0x04
00380
00381
00398 #define MOVE_STATE_MOVING      0x01
00399 #define MOVE_STATE_TARGET_SPEED 0x02
00400 #define MOVE_STATE_ANTIPLAY    0x04

```

```

00402
00403
00419 #define EEPROM_PRECEDENCE    0x01
00421
00422
00439 #define PWR_STATE_UNKNOWN    0x00
00440 #define PWR_STATE_OFF        0x01
00441 #define PWR_STATE_NORM       0x03
00442 #define PWR_STATE_REDUCT     0x04
00443 #define PWR_STATE_MAX        0x05
00445
00446
00465 #define STATE_CONTR            0x000003F
00466 #define STATE_ERRC            0x0000001
00467 #define STATE_ERRD            0x0000002
00468 #define STATE_ERRV            0x0000004
00469 #define STATE_EEPROM_CONNECTED 0x0000010
00470 #define STATE_IS_HOMED        0x0000020
00471 #define STATE_SECUR           0x1B3FFC0
00472 #define STATE_ALARM           0x0000040
00473 #define STATE_CTP_ERROR       0x0000080
00474 #define STATE_POWER_OVERHEAT  0x0000100
00475 #define STATE_CONTROLLER_OVERHEAT 0x0000200
00476 #define STATE_OVERLOAD_POWER_VOLTAGE 0x0000400
00477 #define STATE_OVERLOAD_POWER_CURRENT 0x0000800
00478 #define STATE_OVERLOAD_USB_VOLTAGE 0x0001000
00479 #define STATE_LOW_USB_VOLTAGE 0x0002000
00480 #define STATE_OVERLOAD_USB_CURRENT 0x0004000
00481 #define STATE_BORDERS_SWAP_MISSET 0x0008000
00482 #define STATE_LOW_POWER_VOLTAGE 0x0010000
00483 #define STATE_H_BRIDGE_FAULT  0x0020000
00484 #define STATE_WINDING_RES_MISMATCH 0x0100000
00485 #define STATE_ENCODER_FAULT   0x0200000
00486 #define STATE_ENGINE_RESPONSE_ERROR 0x0800000
00487 #define STATE_EXTIO_ALARM     0x1000000
00489
00490
00509 #define STATE_DIG_SIGNAL      0xFFFF
00510 #define STATE_RIGHT_EDGE      0x0001
00511 #define STATE_LEFT_EDGE       0x0002
00512 #define STATE_BUTTON_RIGHT    0x0004
00513 #define STATE_BUTTON_LEFT     0x0008
00514 #define STATE_GPIO_PINOUT     0x0010
00515 #define STATE_GPIO_LEVEL      0x0020
00516 #define STATE_BRAKE           0x0200
00517 #define STATE_REV_SENSOR      0x0400
00518 #define STATE_SYNC_INPUT      0x0800
00519 #define STATE_SYNC_OUTPUT     0x1000
00520 #define STATE_ENC_A           0x2000
00521 #define STATE_ENC_B           0x4000
00523
00524
00553 #define ENC_STATE_ABSENT      0x00
00554 #define ENC_STATE_UNKNOWN     0x01
00555 #define ENC_STATE_MALFUNC     0x02
00556 #define ENC_STATE_REVERS      0x03
00557 #define ENC_STATE_OK          0x04
00559
00560
00577 #define WIND_A_STATE_ABSENT    0x00
00578 #define WIND_A_STATE_UNKNOWN   0x01
00579 #define WIND_A_STATE_MALFUNC   0x02
00580 #define WIND_A_STATE_OK        0x03
00581 #define WIND_B_STATE_ABSENT    0x00
00582 #define WIND_B_STATE_UNKNOWN   0x10
00583 #define WIND_B_STATE_MALFUNC   0x20
00584 #define WIND_B_STATE_OK        0x30
00586
00587
00606 #define MVCMD_NAME_BITS      0x3F
00607 #define MVCMD_UKNWN          0x00
00608 #define MVCMD_MOVE            0x01
00609 #define MVCMD_MOVR            0x02
00610 #define MVCMD_LEFT            0x03
00611 #define MVCMD_RIGHT           0x04
00612 #define MVCMD_STOP            0x05
00613 #define MVCMD_HOME            0x06
00614 #define MVCMD_LOFT            0x07
00615 #define MVCMD_SSTP            0x08
00616 #define MVCMD_ERROR           0x40
00617 #define MVCMD_RUNNING         0x80
00619
00620
00640 #define RPM_DIV_1000          0x01
00642
00643
00663 #define ENGINE_REVERSE        0x01

```

```

00664 #define ENGINE_CURRENT_AS_RMS    0x02
00665 #define ENGINE_MAX_SPEED           0x04
00666 #define ENGINE_ANTIPLAY             0x08
00667 #define ENGINE_ACCEL_ON              0x10
00668 #define ENGINE_LIMIT_VOLT           0x20
00669 #define ENGINE_LIMIT_CURR           0x40
00670 #define ENGINE_LIMIT_RPM            0x80
00672
00673
00694 #define MICROSTEP_MODE_FULL         0x01
00695 #define MICROSTEP_MODE_FRAC_2       0x02
00696 #define MICROSTEP_MODE_FRAC_4       0x03
00697 #define MICROSTEP_MODE_FRAC_8       0x04
00698 #define MICROSTEP_MODE_FRAC_16      0x05
00699 #define MICROSTEP_MODE_FRAC_32      0x06
00700 #define MICROSTEP_MODE_FRAC_64      0x07
00701 #define MICROSTEP_MODE_FRAC_128     0x08
00702 #define MICROSTEP_MODE_FRAC_256     0x09
00704
00705
00726 #define ENGINE_TYPE_NONE            0x00
00727 #define ENGINE_TYPE_DC               0x01
00728 #define ENGINE_TYPE_2DC              0x02
00729 #define ENGINE_TYPE_STEP             0x03
00730 #define ENGINE_TYPE_TEST             0x04
00731 #define ENGINE_TYPE_BRUSHLESS        0x05
00733
00734
00755 #define DRIVER_TYPE_INTEGRATE        0x02
00756 #define DRIVER_TYPE_EXTERNAL         0x03
00758
00759
00778 #define POWER_REDUCT_ENABLED         0x01
00779 #define POWER_OFF_ENABLED            0x02
00780 #define POWER_SMOOTH_CURRENT         0x04
00782
00783
00802 #define ALARM_ON_DRIVER_OVERHEATING 0x01
00803 #define LOW_UPWR_PROTECTION          0x02
00804 #define H_BRIDGE_ALERT               0x04
00805 #define ALARM_ON_BORDERS_SWAP_MISSET 0x08
00806 #define ALARM_FLAGS_STICKING         0x10
00807 #define BRAKING_OVERVOLTAGE_PROTECTION 0x20
00808 #define ALARM_WINDING_MISMATCH        0x40
00809 #define ALARM_ENGINE_RESPONSE        0x80
00811
00812
00830 #define SETPOS_IGNORE_POSITION       0x01
00831 #define SETPOS_IGNORE_ENCODER        0x02
00833
00834
00850 #define FEEDBACK_ENCODER             0x01
00851 #define FEEDBACK_EMF                 0x04
00852 #define FEEDBACK_NONE                0x05
00853 #define FEEDBACK_ENCODER_MEDIATED    0x06
00855
00856
00872 #define FEEDBACK_ENC_REVERSE         0x01
00873 #define FEEDBACK_ENC_ADAPTIVE_HOLDING 0x02
00874 #define FEEDBACK_ENC_FILTER_NONE     0x00
00875 #define FEEDBACK_ENC_FILTER_WEAK     0x10
00876 #define FEEDBACK_ENC_FILTER_MEDIUM   0x20
00877 #define FEEDBACK_ENC_FILTER_STRONG    0x30
00878 #define FEEDBACK_ENC_FILTER_BITS     0x30
00879 #define FEEDBACK_ENC_TYPE_AUTO        0x00
00880 #define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40
00881 #define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80
00882 #define FEEDBACK_ENC_TYPE_BITS       0xC0
00884
00885
00899 #define SYNCIN_ENABLED               0x01
00900 #define SYNCIN_INVERT                 0x02
00901 #define SYNCIN_GOTOPOSITION           0x04
00903
00904
00918 #define SYNCOUT_ENABLED              0x01
00919 #define SYNCOUT_STATE                 0x02
00920 #define SYNCOUT_INVERT                0x04
00921 #define SYNCOUT_IN_STEPS              0x08
00922 #define SYNCOUT_ONSTART               0x10
00923 #define SYNCOUT_ONSTOP                0x20
00924 #define SYNCOUT_ONPERIOD              0x40
00926
00927
00943 #define EXTIO_SETUP_OUTPUT            0x01
00944 #define EXTIO_SETUP_INVERT           0x02
00946

```

```

00947
00964 #define EXTIO_SETUP_MODE_IN_BITS      0x0F
00965 #define EXTIO_SETUP_MODE_IN_NOP         0x00
00966 #define EXTIO_SETUP_MODE_IN_STOP        0x01
00967 #define EXTIO_SETUP_MODE_IN_PWOF       0x02
00968 #define EXTIO_SETUP_MODE_IN_MOVR       0x03
00969 #define EXTIO_SETUP_MODE_IN_HOME        0x04
00970 #define EXTIO_SETUP_MODE_IN_ALARM      0x05
00971 #define EXTIO_SETUP_MODE_OUT_BITS      0xF0
00972 #define EXTIO_SETUP_MODE_OUT_OFF       0x00
00973 #define EXTIO_SETUP_MODE_OUT_ON        0x10
00974 #define EXTIO_SETUP_MODE_OUT_MOVING    0x20
00975 #define EXTIO_SETUP_MODE_OUT_ALARM     0x30
00976 #define EXTIO_SETUP_MODE_OUT_MOTOR_ON  0x40
00978
00979
00999 #define BORDER_IS_ENCODER              0x01
01000 #define BORDER_STOP_LEFT               0x02
01001 #define BORDER_STOP_RIGHT            0x04
01002 #define BORDERS_SWAP_MISSET_DETECTION  0x08
01004
01005
01025 #define ENDER_SWAP                     0x01
01026 #define ENDER_SW1_ACTIVE_LOW           0x02
01027 #define ENDER_SW2_ACTIVE_LOW           0x04
01029
01030
01050 #define BRAKE_ENABLED                   0x01
01051 #define BRAKE_ENG_PWROFF               0x02
01053
01054
01074 #define CONTROL_MODE_BITS              0x03
01075 #define CONTROL_MODE_OFF                0x00
01076 #define CONTROL_MODE_JOY                0x01
01077 #define CONTROL_MODE_LR                 0x02
01078 #define CONTROL_BTN_LEFT_PUSHED_OPEN    0x04
01079 #define CONTROL_BTN_RIGHT_PUSHED_OPEN   0x08
01081
01082
01100 #define JOY_REVERSE                     0x01
01102
01103
01123 #define CTP_ENABLED                     0x01
01124 #define CTP_BASE                         0x02
01125 #define CTP_ALARM_ON_ERROR              0x04
01126 #define REV_SENS_INV                    0x08
01127 #define CTP_ERROR_CORRECTION             0x10
01129
01130
01151 #define HOME_DIR_FIRST                   0x001
01152 #define HOME_DIR_SECOND                  0x002
01153 #define HOME_MV_SEC_EN                   0x004
01154 #define HOME_HALF_MV                     0x008
01155 #define HOME_STOP_FIRST_BITS             0x030
01156 #define HOME_STOP_FIRST_REV              0x010
01157 #define HOME_STOP_FIRST_SYN              0x020
01158 #define HOME_STOP_FIRST_LIM              0x030
01159 #define HOME_STOP_SECOND_BITS            0x0C0
01160 #define HOME_STOP_SECOND_REV              0x040
01161 #define HOME_STOP_SECOND_SYN              0x080
01162 #define HOME_STOP_SECOND_LIM              0x0C0
01163 #define HOME_USE_FAST                     0x100
01165
01166
01180 #define UART_PARITY_BITS                 0x03
01181 #define UART_PARITY_BIT_EVEN              0x00
01182 #define UART_PARITY_BIT_ODD               0x01
01183 #define UART_PARITY_BIT_SPACE             0x02
01184 #define UART_PARITY_BIT_MARK              0x03
01185 #define UART_PARITY_BIT_USE               0x04
01186 #define UART_STOP_BIT                     0x08
01188
01189
01203 #define MOTOR_TYPE_UNKNOWN               0x00
01204 #define MOTOR_TYPE_STEP                   0x01
01205 #define MOTOR_TYPE_DC                     0x02
01206 #define MOTOR_TYPE_BLDC                   0x03
01208
01209
01223 #define ENCSET_DIFFERENTIAL_OUTPUT       0x001
01224 #define ENCSET_PUSHPULL_OUTPUT            0x004
01225 #define ENCSET_INDEXCHANNEL_PRESENT       0x010
01226 #define ENCSET_REOLUTIONSENSOR_PRESENT    0x040
01227 #define ENCSET_REOLUTIONSENSOR_ACTIVE_HIGH 0x100
01229
01230
01244 #define MB_AVAILABLE                     0x01

```

```

01245 #define MB_POWERED_HOLD    0x02
01247
01248
01262 #define TS_TYPE_BITS        0x07
01263 #define TS_TYPE_UNKNOWN      0x00
01264 #define TS_TYPE_THERMOCOUPLE 0x01
01265 #define TS_TYPE_SEMICONDUCTOR 0x02
01266 #define TS_AVAILABLE        0x08
01268
01269
01283 #define LS_ON_SW1_AVAILABLE 0x01
01284 #define LS_ON_SW2_AVAILABLE 0x02
01285 #define LS_SW1_ACTIVE_LOW   0x04
01286 #define LS_SW2_ACTIVE_LOW   0x08
01287 #define LS_SHORTED          0x10
01289
01290
01306 #define BACK_EMF_INDUCTANCE_AUTO 0x01
01307 #define BACK_EMF_RESISTANCE_AUTO 0x02
01308 #define BACK_EMF_KM_AUTO         0x04
01310
01311
01323 typedef struct
01324 {
01325     unsigned int IPS;
01326     unsigned int FeedbackType;
01327     unsigned int FeedbackFlags;
01328     unsigned int CountsPerTurn;
01329 } feedback_settings_t;
01330
01345 typedef struct
01346 {
01347     unsigned int FastHome;
01348     unsigned int uFastHome;
01349     unsigned int SlowHome;
01350     unsigned int uSlowHome;
01351     int HomeDelta;
01352     int uHomeDelta;
01353     unsigned int HomeFlags;
01354 } home_settings_t;
01355
01371 typedef struct
01372 {
01373     float FastHome;
01374     float SlowHome;
01375     float HomeDelta;
01376     unsigned int HomeFlags;
01377 } home_settings_calb_t;
01378
01390 typedef struct
01391 {
01392     unsigned int Speed;
01393     unsigned int uSpeed;
01394     unsigned int Accel;
01395     unsigned int Decel;
01396     unsigned int AntiplaySpeed;
01397     unsigned int uAntiplaySpeed;
01398     unsigned int MoveFlags;
01399 } move_settings_t;
01400
01412 typedef struct
01413 {
01414     float Speed;
01426     float Accel;
01438     float Decel;
01450     float AntiplaySpeed;
01462     unsigned int MoveFlags;
01463 } move_settings_calb_t;
01464
01481 typedef struct
01482 {
01483     unsigned int NomVoltage;
01484     unsigned int NomCurrent;
01485     unsigned int NomSpeed;
01486     unsigned int uNomSpeed;
01487     unsigned int EngineFlags;
01488     int Antiplay;
01489     unsigned int MicrostepMode;
01490     unsigned int StepsPerRev;
01491 } engine_settings_t;
01492
01510 typedef struct
01511 {
01512     unsigned int NomVoltage;
01513     unsigned int NomCurrent;
01514     float NomSpeed;
01515     unsigned int EngineFlags;

```



```

01516         float Antiplay;
01517         unsigned int MicrostepMode;
01518         unsigned int StepsPerRev;
01519     } engine_settings_calb_t;
01520
01537     typedef struct
01538     {
01539         unsigned int EngineType;
01540         unsigned int DriverType;
01541     } entype_settings_t;
01542
01554     typedef struct
01555     {
01556         unsigned int HoldCurrent;
01557         unsigned int CurrReductDelay;
01558         unsigned int PowerOffDelay;
01559         unsigned int CurrentSetTime;
01560         unsigned int PowerFlags;
01561     } power_settings_t;
01562
01574     typedef struct
01575     {
01576         unsigned int LowUpwrOff;
01577         unsigned int CriticalIpwr;
01578         unsigned int CriticalUpwr;
01579         unsigned int CriticalT;
01580         unsigned int CriticalIusb;
01581         unsigned int CriticalUusb;
01582         unsigned int MinimumUusb;
01583         unsigned int Flags;
01584     } secure_settings_t;
01585
01601     typedef struct
01602     {
01603         unsigned int BorderFlags;
01604         unsigned int EnderFlags;
01605         int LeftBorder;
01606         int uLeftBorder;
01607         int RightBorder;
01608         int uRightBorder;
01609     } edges_settings_t;
01610
01626     typedef struct
01627     {
01628         unsigned int BorderFlags;
01629         unsigned int EnderFlags;
01630         float LeftBorder;
01631         float RightBorder;
01632     } edges_settings_calb_t;
01633
01651     typedef struct
01652     {
01653         unsigned int KpU;
01654         unsigned int KiU;
01655         unsigned int KdU;
01656         float Kpf;
01657         float Kif;
01658         float Kdf;
01659     } pid_settings_t;
01660
01675     typedef struct
01676     {
01677         unsigned int SyncInFlags;
01678         unsigned int ClutterTime;
01679         int Position;
01680         int uPosition;
01681         unsigned int Speed;
01682         unsigned int uSpeed;
01683         unsigned int reserved0;
01684     } sync_in_settings_t;
01685
01699     typedef struct
01700     {
01701         unsigned int SyncInFlags;
01702         unsigned int ClutterTime;
01703         float Position;
01704         float Speed;
01705         unsigned int reserved0;
01706     } sync_in_settings_calb_t;
01707
01721     typedef struct
01722     {
01723         unsigned int SyncOutFlags;
01724         unsigned int SyncOutPulseSteps;
01725         unsigned int SyncOutPeriod;
01726         unsigned int Accuracy;
01727         unsigned int uAccuracy;

```

```

01728     } sync_out_settings_t;
01729
01744     typedef struct
01745     {
01746         unsigned int SyncOutFlags;
01747         unsigned int SyncOutPulseSteps;
01748         unsigned int SyncOutPeriod;
01749         float Accuracy;
01750     } sync_out_settings_calb_t;
01751
01768     typedef struct
01769     {
01770         unsigned int EXTIOSetupFlags;
01771         unsigned int EXTIOModeFlags;
01772     } extio_settings_t;
01773
01788     typedef struct
01789     {
01790         unsigned int t1;
01791         unsigned int t2;
01792         unsigned int t3;
01793         unsigned int t4;
01794         unsigned int BrakeFlags;
01795     } brake_settings_t;
01796
01822     typedef struct
01823     {
01824         unsigned int MaxSpeed[10];
01825         unsigned int uMaxSpeed[10];
01826         unsigned int Timeout[9];
01827         unsigned int MaxClickTime;
01828         unsigned int Flags;
01829         int DeltaPosition;
01830         int uDeltaPosition;
01831     } control_settings_t;
01832
01858     typedef struct
01859     {
01860         float MaxSpeed[10];
01861         unsigned int Timeout[9];
01862         unsigned int MaxClickTime;
01863         unsigned int Flags;
01864         float DeltaPosition;
01865     } control_settings_calb_t;
01866
01899     typedef struct
01900     {
01901         unsigned int JoyLowEnd;
01902         unsigned int JoyCenter;
01903         unsigned int JoyHighEnd;
01904         unsigned int ExpFactor;
01905         unsigned int DeadZone;
01906         unsigned int JoyFlags;
01907     } joystick_settings_t;
01908
01927     typedef struct
01928     {
01929         unsigned int CTPMinError;
01930         unsigned int CTPFlags;
01931     } ctp_settings_t;
01932
01947     typedef struct
01948     {
01949         unsigned int Speed;
01950         unsigned int UARTSetupFlags;
01951     } uart_settings_t;
01952
01966     typedef struct
01967     {
01968         unsigned int DHCPEnabled;
01969         unsigned int IPv4Address[4];
01970         unsigned int SubnetMask[4];
01971         unsigned int DefaultGateway[4];
01972     } network_settings_t;
01973
01987     typedef struct
01988     {
01989         char UserPassword[21];
01990     } password_settings_t;
01991
02006     typedef struct
02007     {
02008         float CSS1_A;
02009         float CSS1_B;
02010         float CSS2_A;
02011         float CSS2_B;
02012         float FullCurrent_A;

```

```

02013         float FullCurrent_B;
02014     } calibration_settings_t;
02015
02025     typedef struct
02026     {
02027         char ControllerName[17];
02028         unsigned int CtrlFlags;
02029     } controller_name_t;
02030
02040     typedef struct
02041     {
02042         unsigned int UserData[7];
02043     } nonvolatile_memory_t;
02044
02063     typedef struct
02064     {
02065         float L;
02066         float R;
02067         float Km;
02068         unsigned int BackEMFFlags;
02069     } emf_settings_t;
02070
02087     typedef struct
02088     {
02089         unsigned int stepcloseloop_Kw;
02090         unsigned int stepcloseloop_Kp_low;
02091         unsigned int stepcloseloop_Kp_high;
02092     } engine_advanced_setup_t;
02093
02109     typedef struct
02110     {
02111         unsigned int Param1;
02112     } extended_settings_t;
02113
02128     typedef struct
02129     {
02130         int Position;
02131         int uPosition;
02132         long_t EncPosition;
02133     } get_position_t;
02134
02148     typedef struct
02149     {
02150         float Position;
02151         long_t EncPosition;
02152     } get_position_calb_t;
02153
02166     typedef struct
02167     {
02168         int Position;
02169         int uPosition;
02170         long_t EncPosition;
02171         unsigned int PosFlags;
02172     } set_position_t;
02173
02186     typedef struct
02187     {
02188         float Position;
02189         long_t EncPosition;
02190         unsigned int PosFlags;
02191     } set_position_calb_t;
02192
02205     typedef struct
02206     {
02207         unsigned int MoveSts;
02208         unsigned int MvCmdSts;
02209         unsigned int PWRSts;
02210         unsigned int EncSts;
02211         unsigned int WindSts;
02212         int CurPosition;
02213         int uCurPosition;
02214         long_t EncPosition;
02215         int CurSpeed;
02216         int uCurSpeed;
02217         int Ipwr;
02218         int Upwr;
02219         int Iusb;
02220         int Uusb;
02221         int CurT;
02222         unsigned int Flags;
02223         unsigned int GPIOFlags;
02224         unsigned int CmdBufFreeSpace;
02225     } status_t;
02226
02239     typedef struct
02240     {
02241         unsigned int MoveSts;

```

```

02242         unsigned int MvCmdSts;
02243         unsigned int PWRSts;
02244         unsigned int EncSts;
02245         unsigned int WindSts;
02246         float CurPosition;
02247         long_t EncPosition;
02248         float CurSpeed;
02249         int Ipwr;
02250         int Upwr;
02251         int Iusb;
02252         int Uusb;
02253         int CurT;
02254         unsigned int Flags;
02255         unsigned int GPIOFlags;
02256         unsigned int CmdBufFreeSpace;
02257     } status_calb_t;
02258
02269     typedef struct
02270     {
02271         int Speed[25];
02272         int Error[25];
02273         unsigned int Length;
02274     } measurements_t;
02275
02289     typedef struct
02290     {
02291         int WindingVoltageA;
02292         int WindingVoltageB;
02293         int WindingVoltageC;
02294         int WindingCurrentA;
02295         int WindingCurrentB;
02296         int WindingCurrentC;
02297         unsigned int Pot;
02298         unsigned int Joy;
02299         int AveragedPowerRatio;
02300     } chart_data_t;
02301
02312     typedef struct
02313     {
02314         char Manufacturer[5];
02315         char ManufacturerId[3];
02316         char ProductDescription[9];
02317         unsigned int Major;
02318         unsigned int Minor;
02319         unsigned int Release;
02320     } device_information_t;
02321
02332     typedef struct
02333     {
02334         unsigned int SN;
02335         uint8_t Key[32];
02336         unsigned int Major;
02337         unsigned int Minor;
02338         unsigned int Release;
02339     } serial_number_t;
02340
02354     typedef struct
02355     {
02356         unsigned int A1Voltage_ADC;
02357         unsigned int A2Voltage_ADC;
02358         unsigned int B1Voltage_ADC;
02359         unsigned int B2Voltage_ADC;
02360         unsigned int SupVoltage_ADC;
02361         unsigned int ACurrent_ADC;
02362         unsigned int BCurrent_ADC;
02363         unsigned int FullCurrent_ADC;
02364         unsigned int Temp_ADC;
02365         unsigned int Joy_ADC;
02366         unsigned int Pot_ADC;
02367         unsigned int Enc_Check_ADC;
02368         unsigned int deprecated0;
02369         int A1Voltage;
02370         int A2Voltage;
02371         int B1Voltage;
02372         int B2Voltage;
02373         int SupVoltage;
02374         int ACurrent;
02375         int BCurrent;
02376         int FullCurrent;
02377         int Temp;
02378         int Joy;
02379         int Pot;
02380         int Enc_Check;
02381         unsigned int deprecated1[2];
02382         int R;
02383         int L;
02384     } analog_data_t;

```

```

02385
02397     typedef struct
02398     {
02399         uint8_t DebugData[128];
02400     } debug_read_t;
02401
02413     typedef struct
02414     {
02415         uint8_t DebugData[128];
02416     } debug_write_t;
02417
02427     typedef struct
02428     {
02429         char PositionerName[17];
02430     } stage_name_t;
02431
02443     typedef struct
02444     {
02445         char Manufacturer[17];
02446         char PartNumber[25];
02447     } stage_information_t;
02448
02460     typedef struct
02461     {
02462         float LeadScrewPitch;
02463         char Units[9];
02464         float MaxSpeed;
02465         float TravelRange;
02466         float SupplyVoltageMin;
02467         float SupplyVoltageMax;
02468         float MaxCurrentConsumption;
02469         float HorizontalLoadCapacity;
02470         float VerticalLoadCapacity;
02471     } stage_settings_t;
02472
02484     typedef struct
02485     {
02486         char Manufacturer[17];
02487         char PartNumber[25];
02488     } motor_information_t;
02489
02501     typedef struct
02502     {
02503         unsigned int MotorType;
02504         unsigned int ReservedField;
02505         unsigned int Poles;
02506         unsigned int Phases;
02507         float NominalVoltage;
02508         float NominalCurrent;
02509         float NominalSpeed;
02510         float NominalTorque;
02511         float NominalPower;
02512         float WindingResistance;
02513         float WindingInductance;
02514         float RotorInertia;
02515         float StallTorque;
02516         float DetentTorque;
02517         float TorqueConstant;
02518         float SpeedConstant;
02519         float SpeedTorqueGradient;
02520         float MechanicalTimeConstant;
02521         float MaxSpeed;
02522         float MaxCurrent;
02523         float MaxCurrentTime;
02524         float NoLoadCurrent;
02525         float NoLoadSpeed;
02526     } motor_settings_t;
02527
02539     typedef struct
02540     {
02541         char Manufacturer[17];
02542         char PartNumber[25];
02543     } encoder_information_t;
02544
02556     typedef struct
02557     {
02558         float MaxOperatingFrequency;
02559         float SupplyVoltageMin;
02560         float SupplyVoltageMax;
02561         float MaxCurrentConsumption;
02562         unsigned int PPR;
02563         unsigned int EncoderSettings;
02564     } encoder_settings_t;
02565
02577     typedef struct
02578     {
02579         char Manufacturer[17];

```

```

02580         char PartNumber[25];
02581     } hallsensor_information_t;
02582
02594     typedef struct
02595     {
02596         float MaxOperatingFrequency;
02597         float SupplyVoltageMin;
02598         float SupplyVoltageMax;
02599         float MaxCurrentConsumption;
02600         unsigned int PPR;
02601     } hallsensor_settings_t;
02602
02614     typedef struct
02615     {
02616         char Manufacturer[17];
02617         char PartNumber[25];
02618     } gear_information_t;
02619
02631     typedef struct
02632     {
02633         float ReductionIn;
02634         float ReductionOut;
02635         float RatedInputTorque;
02636         float RatedInputSpeed;
02637         float MaxOutputBacklash;
02638         float InputInertia;
02639         float Efficiency;
02640     } gear_settings_t;
02641
02653     typedef struct
02654     {
02655         char MagneticBrakeInfo[25];
02656         float MBRatedVoltage;
02657         float MBRatedCurrent;
02658         float MBTorque;
02659         unsigned int MBSettings;
02660         char TemperatureSensorInfo[25];
02661         float TSMIn;
02662         float TSMMax;
02663         float TSGrad;
02664         unsigned int TSSettings;
02665         unsigned int LimitSwitchesSettings;
02666     } accessories_settings_t;
02667
02680     typedef struct
02681     {
02682         uint8_t key[16];
02683     } init_random_t;
02684
02694     typedef struct
02695     {
02696         unsigned int UniqueID0;
02697         unsigned int UniqueID1;
02698         unsigned int UniqueID2;
02699         unsigned int UniqueID3;
02700     } globally_unique_identifier_t;
02701
02702 /*
02703 -----
02704 BEGIN OF GENERATED function declarations
02705 -----
02706 */
02707
02718
02720
02739     result_t XIMC_API set_feedback_settings (device_t id, const feedback_settings_t* feedback_settings);
02740
02759     result_t XIMC_API get_feedback_settings (device_t id, feedback_settings_t* feedback_settings);
02760
02777     result_t XIMC_API set_home_settings (device_t id, const home_settings_t* home_settings);
02778
02797     result_t XIMC_API set_home_settings_calb (device_t id, const home_settings_calb_t* home_settings_calb, const
calibration_t* calibration);
02798
02815     result_t XIMC_API get_home_settings (device_t id, home_settings_t* home_settings);
02816
02835     result_t XIMC_API get_home_settings_calb (device_t id, home_settings_calb_t* home_settings_calb, const
calibration_t* calibration);
02836
02849     result_t XIMC_API set_move_settings (device_t id, const move_settings_t* move_settings);
02850
02865     result_t XIMC_API set_move_settings_calb (device_t id, const move_settings_calb_t* move_settings_calb, const
calibration_t* calibration);
02866
02879     result_t XIMC_API get_move_settings (device_t id, move_settings_t* move_settings);
02880
02895     result_t XIMC_API get_move_settings_calb (device_t id, move_settings_calb_t* move_settings_calb, const

```

```

        calibration_t* calibration);
02896
02915         result_t XIMC_API set_engine_settings (device_t id, const engine_settings_t* engine_settings);
02916
02937         result_t XIMC_API set_engine_settings_calb (device_t id, const engine_settings_calb_t* engine_settings_calb,
const calibration_t* calibration);
02938
02957         result_t XIMC_API get_engine_settings (device_t id, engine_settings_t* engine_settings);
02958
02979         result_t XIMC_API get_engine_settings_calb (device_t id, engine_settings_calb_t* engine_settings_calb, const
calibration_t* calibration);
02980
02993         result_t XIMC_API set_entype_settings (device_t id, const entype_settings_t* entype_settings);
02994
03007         result_t XIMC_API get_entype_settings (device_t id, entype_settings_t* entype_settings);
03008
03022         result_t XIMC_API set_power_settings (device_t id, const power_settings_t* power_settings);
03023
03038         result_t XIMC_API get_power_settings (device_t id, power_settings_t* power_settings);
03039
03053         result_t XIMC_API set_secure_settings (device_t id, const secure_settings_t* secure_settings);
03054
03068         result_t XIMC_API get_secure_settings (device_t id, secure_settings_t* secure_settings);
03069
03084         result_t XIMC_API set_edges_settings (device_t id, const edges_settings_t* edges_settings);
03085
03110         result_t XIMC_API set_edges_settings_calb (device_t id, const edges_settings_calb_t* edges_settings_calb,
const calibration_t* calibration);
03111
03126         result_t XIMC_API get_edges_settings (device_t id, edges_settings_t* edges_settings);
03127
03152         result_t XIMC_API get_edges_settings_calb (device_t id, edges_settings_calb_t* edges_settings_calb, const
calibration_t* calibration);
03153
03172         result_t XIMC_API set_pid_settings (device_t id, const pid_settings_t* pid_settings);
03173
03191         result_t XIMC_API get_pid_settings (device_t id, pid_settings_t* pid_settings);
03192
03209         result_t XIMC_API set_sync_in_settings (device_t id, const sync_in_settings_t* sync_in_settings);
03210
03229         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);
03230
03247         result_t XIMC_API get_sync_in_settings (device_t id, sync_in_settings_t* sync_in_settings);
03248
03267         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);
03268
03285         result_t XIMC_API set_sync_out_settings (device_t id, const sync_out_settings_t* sync_out_settings);
03286
03305         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);
03306
03321         result_t XIMC_API get_sync_out_settings (device_t id, sync_out_settings_t* sync_out_settings);
03322
03341         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);
03342
03360         result_t XIMC_API set_extio_settings (device_t id, const extio_settings_t* extio_settings);
03361
03377         result_t XIMC_API get_extio_settings (device_t id, extio_settings_t* extio_settings);
03378
03391         result_t XIMC_API set_brake_settings (device_t id, const brake_settings_t* brake_settings);
03392
03405         result_t XIMC_API get_brake_settings (device_t id, brake_settings_t* brake_settings);
03406
03431         result_t XIMC_API set_control_settings (device_t id, const control_settings_t* control_settings);
03432
03459         result_t XIMC_API set_control_settings_calb (device_t id, const control_settings_calb_t*
control_settings_calb, const calibration_t* calibration);
03460
03485         result_t XIMC_API get_control_settings (device_t id, control_settings_t* control_settings);
03486
03513         result_t XIMC_API get_control_settings_calb (device_t id, control_settings_calb_t* control_settings_calb,
const calibration_t* calibration);
03514
03543         result_t XIMC_API set_joystick_settings (device_t id, const joystick_settings_t* joystick_settings);
03544
03573         result_t XIMC_API get_joystick_settings (device_t id, joystick_settings_t* joystick_settings);
03574
03594         result_t XIMC_API set_ctp_settings (device_t id, const ctp_settings_t* ctp_settings);
03595
03613         result_t XIMC_API get_ctp_settings (device_t id, ctp_settings_t* ctp_settings);
03614
03631         result_t XIMC_API set_uart_settings (device_t id, const uart_settings_t* uart_settings);
03632
03649         result_t XIMC_API get_uart_settings (device_t id, uart_settings_t* uart_settings);

```

```

03650
03670     result_t XIMC_API set_network_settings (device_t id, const network_settings_t* network_settings);
03671
03691     result_t XIMC_API get_network_settings (device_t id, network_settings_t* network_settings);
03692
03706     result_t XIMC_API set_password_settings (device_t id, const password_settings_t* password_settings);
03707
03721     result_t XIMC_API get_password_settings (device_t id, password_settings_t* password_settings);
03722
03739     result_t XIMC_API set_calibration_settings (device_t id, const calibration_settings_t* calibration_settings);
03740
03757     result_t XIMC_API get_calibration_settings (device_t id, calibration_settings_t* calibration_settings);
03758
03771     result_t XIMC_API set_controller_name (device_t id, const controller_name_t* controller_name);
03772
03785     result_t XIMC_API get_controller_name (device_t id, controller_name_t* controller_name);
03786
03799     result_t XIMC_API set_nonvolatile_memory (device_t id, const nonvolatile_memory_t* nonvolatile_memory);
03800
03813     result_t XIMC_API get_nonvolatile_memory (device_t id, nonvolatile_memory_t* nonvolatile_memory);
03814
03832     result_t XIMC_API set_emf_settings (device_t id, const emf_settings_t* emf_settings);
03833
03850     result_t XIMC_API get_emf_settings (device_t id, emf_settings_t* emf_settings);
03851
03870     result_t XIMC_API set_engine_advanced_setup (device_t id, const engine_advanced_setup_t*
engine_advanced_setup);
03871
03890     result_t XIMC_API get_engine_advanced_setup (device_t id, engine_advanced_setup_t* engine_advanced_setup);
03891
03906     result_t XIMC_API set_extended_settings (device_t id, const extended_settings_t* extended_settings);
03907
03922     result_t XIMC_API get_extended_settings (device_t id, extended_settings_t* extended_settings);
03923
03924
03926
03937
03939
03957     result_t XIMC_API command_stop (device_t id);
03958
03973     result_t XIMC_API command_power_off (device_t id);
03974
03993     result_t XIMC_API command_move (device_t id, int Position, int uPosition);
03994
04021     result_t XIMC_API command_move_calb (device_t id, float Position, const calibration_t* calibration);
04022
04039     result_t XIMC_API command_movr (device_t id, int DeltaPosition, int uDeltaPosition);
04040
04065     result_t XIMC_API command_movr_calb (device_t id, float DeltaPosition, const calibration_t* calibration);
04066
04102     result_t XIMC_API command_home (device_t id);
04103
04114     result_t XIMC_API command_left (device_t id);
04115
04126     result_t XIMC_API command_right (device_t id);
04127
04139     result_t XIMC_API command_loft (device_t id);
04140
04151     result_t XIMC_API command_sstp (device_t id);
04152
04166     result_t XIMC_API get_position (device_t id, get_position_t* the_get_position);
04167
04190     result_t XIMC_API get_position_calb (device_t id, get_position_calb_t* the_get_position_calb, const
calibration_t* calibration);
04191
04206     result_t XIMC_API set_position (device_t id, const set_position_t* the_set_position);
04207
04222     result_t XIMC_API set_position_calb (device_t id, const set_position_calb_t* the_set_position_calb, const
calibration_t* calibration);
04223
04234     result_t XIMC_API command_zero (device_t id);
04235
04236
04238
04249
04251
04262     result_t XIMC_API command_save_settings (device_t id);
04263
04274     result_t XIMC_API command_read_settings (device_t id);
04275
04286     result_t XIMC_API command_save_robust_settings (device_t id);
04287
04298     result_t XIMC_API command_read_robust_settings (device_t id);
04299
04312     result_t XIMC_API command_eesave_settings (device_t id);
04313
04324     result_t XIMC_API command_eeread_settings (device_t id);

```



```

04325
04336     result_t XIMC_API command_start_measurements (device_t id);
04337
04356     result_t XIMC_API get_measurements (device_t id, measurements_t* measurements);
04357
04374     result_t XIMC_API get_chart_data (device_t id, chart_data_t* chart_data);
04375
04388     result_t XIMC_API get_serial_number (device_t id, unsigned int* SerialNumber);
04389
04406     result_t XIMC_API get_firmware_version (device_t id, unsigned int* Major, unsigned int* Minor, unsigned int*
Release);
04407
04418     result_t XIMC_API service_command_updf (device_t id);
04419
04420
04422
04433
04435
04454     result_t XIMC_API set_serial_number (device_t id, const serial_number_t* serial_number);
04455
04468     result_t XIMC_API get_analog_data (device_t id, analog_data_t* analog_data);
04469
04484     result_t XIMC_API get_debug_read (device_t id, debug_read_t* debug_read);
04485
04498     result_t XIMC_API set_debug_write (device_t id, const debug_write_t* debug_write);
04499
04500
04502
04513
04515
04528     result_t XIMC_API set_stage_name (device_t id, const stage_name_t* stage_name);
04529
04542     result_t XIMC_API get_stage_name (device_t id, stage_name_t* stage_name);
04543
04558     result_t XIMC_API set_stage_information (device_t id, const stage_information_t* stage_information);
04559
04572     result_t XIMC_API get_stage_information (device_t id, stage_information_t* stage_information);
04573
04588     result_t XIMC_API set_stage_settings (device_t id, const stage_settings_t* stage_settings);
04589
04602     result_t XIMC_API get_stage_settings (device_t id, stage_settings_t* stage_settings);
04603
04618     result_t XIMC_API set_motor_information (device_t id, const motor_information_t* motor_information);
04619
04632     result_t XIMC_API get_motor_information (device_t id, motor_information_t* motor_information);
04633
04648     result_t XIMC_API set_motor_settings (device_t id, const motor_settings_t* motor_settings);
04649
04662     result_t XIMC_API get_motor_settings (device_t id, motor_settings_t* motor_settings);
04663
04678     result_t XIMC_API set_encoder_information (device_t id, const encoder_information_t* encoder_information);
04679
04692     result_t XIMC_API get_encoder_information (device_t id, encoder_information_t* encoder_information);
04693
04708     result_t XIMC_API set_encoder_settings (device_t id, const encoder_settings_t* encoder_settings);
04709
04722     result_t XIMC_API get_encoder_settings (device_t id, encoder_settings_t* encoder_settings);
04723
04738     result_t XIMC_API set_hallsensor_information (device_t id, const hallsensor_information_t*
hallsensor_information);
04739
04752     result_t XIMC_API get_hallsensor_information (device_t id, hallsensor_information_t* hallsensor_information);
04753
04768     result_t XIMC_API set_hallsensor_settings (device_t id, const hallsensor_settings_t* hallsensor_settings);
04769
04782     result_t XIMC_API get_hallsensor_settings (device_t id, hallsensor_settings_t* hallsensor_settings);
04783
04798     result_t XIMC_API set_gear_information (device_t id, const gear_information_t* gear_information);
04799
04812     result_t XIMC_API get_gear_information (device_t id, gear_information_t* gear_information);
04813
04828     result_t XIMC_API set_gear_settings (device_t id, const gear_settings_t* gear_settings);
04829
04842     result_t XIMC_API get_gear_settings (device_t id, gear_settings_t* gear_settings);
04843
04858     result_t XIMC_API set_accessories_settings (device_t id, const accessories_settings_t* accessories_settings);
04859
04872     result_t XIMC_API get_accessories_settings (device_t id, accessories_settings_t* accessories_settings);
04873
04890     result_t XIMC_API get_bootloader_version (device_t id, unsigned int* Major, unsigned int* Minor, unsigned
int* Release);
04891
04904     result_t XIMC_API get_init_random (device_t id, init_random_t* init_random);
04905
04921     result_t XIMC_API get_globally_unique_identifier (device_t id, globally_unique_identifier_t*
globally_unique_identifier);
04922

```

```

04923
04924 /*
04925 -----
04926     END OF GENERATED CODE
04927 -----
04928 */
04929
04930 /* hand-crafted functions begin */
04931
04944     result_t XIMC_API goto_firmware(device_t id, uint8_t* ret);
04945
04958     result_t XIMC_API has_firmware(const char* uri, uint8_t* ret);
04959
04975     result_t XIMC_API command_update_firmware(const char* uri, const uint8_t* data, uint32_t data_size);
04976
04991     result_t XIMC_API write_key (const char* uri, uint8_t* key);
04992
05005     result_t XIMC_API command_reset(device_t id);
05006
05019     result_t XIMC_API command_clear_fram(device_t id);
05020
05022
05023     // -----
05024
05036
05064     device_t XIMC_API open_device (const char* uri);
05065
05080     result_t XIMC_API close_device (device_t* id);
05081
05126     result_t XIMC_API set_correction_table(device_t id, const char* namefile);
05127
05140     result_t XIMC_API probe_device (const char* uri);
05141
05252     device_enumeration_t XIMC_API enumerate_devices(int enumerate_flags, const char *hints);
05253
05264     result_t XIMC_API free_enumerate_devices(device_enumeration_t device_enumeration);
05265
05276     int XIMC_API get_device_count(device_enumeration_t device_enumeration);
05277
05286     typedef char* pchar;
05287
05302     pchar XIMC_API get_device_name(device_enumeration_t device_enumeration, int device_index);
05303
05304
05321     result_t XIMC_API get_enumerate_device_serial(device_enumeration_t device_enumeration, int device_index,
uint32_t* serial);
05322
05339     result_t XIMC_API get_enumerate_device_information(device_enumeration_t device_enumeration, int device_index,
device_information_t* device_information);
05340
05357     result_t XIMC_API get_enumerate_device_controller_name(device_enumeration_t device_enumeration, int
device_index, controller_name_t* controller_name);
05358
05375     result_t XIMC_API get_enumerate_device_stage_name(device_enumeration_t device_enumeration, int device_index,
stage_name_t* stage_name);
05376
05393     result_t XIMC_API get_enumerate_device_network_information(device_enumeration_t device_enumeration, int
device_index, device_network_information_t* device_network_information);
05394
05402     result_t XIMC_API reset_locks ();
05403
05413     void XIMC_API msec_sleep (unsigned int msec);
05414
05424     void XIMC_API ximc_version (char* version);
05425
05426 #if !defined(MATLAB_IMPORT) && !defined(LABVIEW64_IMPORT) && !defined(LABVIEW32_IMPORT)
05427
05439     typedef void (XIMC_CALLCONV *logging_callback_t)(int loglevel, const wchar_t* message, void* user_data);
05440
05452     void XIMC_API logging_callback_stderr_wide(int loglevel, const wchar_t* message, void* user_data);
05453
05465     void XIMC_API logging_callback_stderr_narrow(int loglevel, const wchar_t* message, void* user_data);
05466
05479     void XIMC_API set_logging_callback(logging_callback_t logging_callback, void* user_data);
05480
05481 #endif
05482
05506     result_t XIMC_API get_status (device_t id, status_t* status);
05507
05533     result_t XIMC_API get_status_calb (device_t id, status_calb_t* status, const calibration_t* calibration);
05534
05560     result_t XIMC_API get_device_information (device_t id, device_information_t* device_information);
05561
05580     result_t XIMC_API command_wait_for_stop(device_t id, uint32_t refresh_interval_ms);
05581
05594     result_t XIMC_API command_homezero(device_t id);
05596

```

```
05597 #if defined(__cplusplus)
05598 };
05599 #endif
05600
05601 #endif
05602
05603 // vim: ts=4 shiftwidth=4
05604
```

# Предметный указатель

## A

- calibration\_t, [21](#)
- A1Voltage
  - analog\_data\_t, [14](#)
- A1Voltage\_ADC
  - analog\_data\_t, [14](#)
- A2Voltage
  - analog\_data\_t, [14](#)
- A2Voltage\_ADC
  - analog\_data\_t, [14](#)
- Accel
  - move\_settings\_calb\_t, [70](#)
  - move\_settings\_t, [72](#)
- accessories\_settings\_t, [10](#)
  - LimitSwitchesSettings, [11](#)
  - MagneticBrakeInfo, [11](#)
  - MBRatedCurrent, [11](#)
  - MBRatedVoltage, [11](#)
  - MBSettings, [11](#)
  - MBTorque, [11](#)
  - TemperatureSensorInfo, [12](#)
  - TSGrad, [12](#)
  - TSMaх, [12](#)
  - TSMin, [12](#)
  - TSSettings, [12](#)
- Accuracy
  - sync\_out\_settings\_calb\_t, [99](#)
  - sync\_out\_settings\_t, [100](#)
- ACurrent
  - analog\_data\_t, [14](#)
- ACurrent\_ADC
  - analog\_data\_t, [15](#)
- ALARM\_FLAGS\_STICKING
  - ximc.h, [128](#)
- ALARM\_ON\_BORDERS\_SWAP\_MISSET
  - ximc.h, [128](#)
- ALARM\_ON\_DRIVER\_OVERHEATING
  - ximc.h, [128](#)
- analog\_data\_t, [12](#)
  - A1Voltage, [14](#)
  - A1Voltage\_ADC, [14](#)
  - A2Voltage, [14](#)
  - A2Voltage\_ADC, [14](#)
  - ACurrent, [14](#)
  - ACurrent\_ADC, [15](#)
  - B1Voltage, [15](#)
  - B1Voltage\_ADC, [15](#)
  - B2Voltage, [15](#)
  - B2Voltage\_ADC, [15](#)
  - BCurrent, [15](#)
  - BCurrent\_ADC, [15](#)
  - Enc\_Check, [16](#)
  - Enc\_Check\_ADC, [16](#)
  - FullCurrent, [16](#)
  - FullCurrent\_ADC, [16](#)
  - Joy, [16](#)
  - Joy\_ADC, [16](#)
  - Pot, [16](#)
  - SupVoltage, [17](#)
  - SupVoltage\_ADC, [17](#)
  - Temp, [17](#)
  - Temp\_ADC, [17](#)
- Antiplay
  - engine\_settings\_calb\_t, [41](#)
  - engine\_settings\_t, [43](#)
- AntiplaySpeed
  - move\_settings\_calb\_t, [70](#)
  - move\_settings\_t, [72](#)
- AveragedPowerRatio
  - chart\_data\_t, [23](#)
- B1Voltage
  - analog\_data\_t, [15](#)
- B1Voltage\_ADC
  - analog\_data\_t, [15](#)
- B2Voltage
  - analog\_data\_t, [15](#)
- B2Voltage\_ADC
  - analog\_data\_t, [15](#)
- BACK\_EMF\_INDUCTANCE\_AUTO
  - ximc.h, [128](#)
- BACK\_EMF\_KM\_AUTO
  - ximc.h, [129](#)
- BACK\_EMF\_RESISTANCE\_AUTO
  - ximc.h, [129](#)
- BackEMFFlags
  - emf\_settings\_t, [36](#)
- BCurrent
  - analog\_data\_t, [15](#)
- BCurrent\_ADC
  - analog\_data\_t, [15](#)
- BORDER\_IS\_ENCODER
  - ximc.h, [129](#)
- BORDER\_STOP\_LEFT

- ximc.h, [129](#)
- BORDER\_STOP\_RIGHT
  - ximc.h, [129](#)
- BorderFlags
  - edges\_settings\_calb\_t, [33](#)
  - edges\_settings\_t, [35](#)
- BORDERS\_SWAP\_MISSET\_DETECTION
  - ximc.h, [129](#)
- BRAKE\_ENABLED
  - ximc.h, [129](#)
- BRAKE\_ENG\_PWROFF
  - ximc.h, [130](#)
- brake\_settings\_t, [17](#)
  - BrakeFlags, [18](#)
  - t1, [18](#)
  - t2, [18](#)
  - t3, [18](#)
  - t4, [18](#)
- BrakeFlags
  - brake\_settings\_t, [18](#)
- BRAKING\_OVERVOLTAGE\_PROTECTION
  - ximc.h, [130](#)
- calibration\_settings\_t, [19](#)
  - CSS1\_A, [19](#)
  - CSS1\_B, [19](#)
  - CSS2\_A, [20](#)
  - CSS2\_B, [20](#)
  - FullCurrent\_A, [20](#)
  - FullCurrent\_B, [20](#)
- calibration\_t, [20](#)
  - A, [21](#)
  - MicrostepMode, [22](#)
  - ximc.h, [156](#)
- chart\_data\_t, [22](#)
  - AveragedPowerRatio, [23](#)
  - Joy, [23](#)
  - Pot, [23](#)
  - WindingCurrentA, [23](#)
  - WindingCurrentB, [23](#)
  - WindingCurrentC, [24](#)
  - WindingVoltageA, [24](#)
  - WindingVoltageB, [24](#)
  - WindingVoltageC, [24](#)
- close\_device
  - ximc.h, [157](#)
- ClutterTime
  - sync\_in\_settings\_calb\_t, [96](#)
  - sync\_in\_settings\_t, [97](#)
- CmdBufFreeSpace
  - status\_calb\_t, [89](#)
  - status\_t, [93](#)
- command\_clear\_fram
  - ximc.h, [158](#)
- command\_eeread\_settings
  - ximc.h, [158](#)
- command\_eesave\_settings
  - ximc.h, [158](#)
- command\_home
  - ximc.h, [159](#)
- command\_homezero
  - ximc.h, [159](#)
- command\_left
  - ximc.h, [160](#)
- command\_loft
  - ximc.h, [160](#)
- command\_move
  - ximc.h, [160](#)
- command\_move\_calb
  - ximc.h, [160](#)
- command\_movr
  - ximc.h, [161](#)
- command\_movr\_calb
  - ximc.h, [161](#)
- command\_power\_off
  - ximc.h, [163](#)
- command\_read\_robust\_settings
  - ximc.h, [163](#)
- command\_read\_settings
  - ximc.h, [163](#)
- command\_reset
  - ximc.h, [164](#)
- command\_right
  - ximc.h, [164](#)
- command\_save\_robust\_settings
  - ximc.h, [164](#)
- command\_save\_settings
  - ximc.h, [164](#)
- command\_sstp
  - ximc.h, [165](#)
- command\_start\_measurements
  - ximc.h, [165](#)
- command\_stop
  - ximc.h, [165](#)
- command\_update\_firmware
  - ximc.h, [165](#)
- command\_wait\_for\_stop
  - ximc.h, [166](#)
- command\_zero
  - ximc.h, [166](#)
- CONTROL\_BTN\_LEFT\_PUSHED\_OPEN
  - ximc.h, [130](#)
- CONTROL\_BTN\_RIGHT\_PUSHED\_OPEN
  - ximc.h, [130](#)
- CONTROL\_MODE\_BITS
  - ximc.h, [130](#)
- CONTROL\_MODE\_JOY
  - ximc.h, [130](#)
- CONTROL\_MODE\_LR
  - ximc.h, [130](#)
- CONTROL\_MODE\_OFF
  - ximc.h, [131](#)

- control\_settings\_calb\_t, 24
  - Flags, 25
  - MaxClickTime, 25
  - MaxSpeed, 25
  - Timeout, 26
- control\_settings\_t, 26
  - Flags, 27
  - MaxClickTime, 27
  - MaxSpeed, 27
  - Timeout, 27
  - uDeltaPosition, 27
  - uMaxSpeed, 27
- controller\_name\_t, 28
  - ControllerName, 28
  - CtrlFlags, 28
- ControllerName
  - controller\_name\_t, 28
- CountsPerTurn
  - feedback\_settings\_t, 48
- Criticalpwr
  - secure\_settings\_t, 79
- Criticalusb
  - secure\_settings\_t, 79
- CriticalT
  - secure\_settings\_t, 79
- CriticalUpwr
  - secure\_settings\_t, 80
- CriticalUusb
  - secure\_settings\_t, 80
- CSS1\_A
  - calibration\_settings\_t, 19
- CSS1\_B
  - calibration\_settings\_t, 19
- CSS2\_A
  - calibration\_settings\_t, 20
- CSS2\_B
  - calibration\_settings\_t, 20
- CTP\_ALARM\_ON\_ERROR
  - ximc.h, 131
- CTP\_BASE
  - ximc.h, 131
- CTP\_ENABLED
  - ximc.h, 131
- CTP\_ERROR\_CORRECTION
  - ximc.h, 131
- ctp\_settings\_t, 28
  - CTPFlags, 29
  - CTPMinError, 29
- CTPFlags
  - ctp\_settings\_t, 29
- CTPMinError
  - ctp\_settings\_t, 29
- CtrlFlags
  - controller\_name\_t, 28
- CurPosition
  - status\_calb\_t, 89
- status\_t, 93
- CurrentSetTime
  - power\_settings\_t, 78
- CurrReductDelay
  - power\_settings\_t, 78
- CurSpeed
  - status\_calb\_t, 89
  - status\_t, 93
- CurT
  - status\_calb\_t, 89
  - status\_t, 93
- DeadZone
  - joystick\_settings\_t, 62
- debug\_read\_t, 30
  - DebugData, 30
- debug\_write\_t, 30
  - DebugData, 31
- DebugData
  - debug\_read\_t, 30
  - debug\_write\_t, 31
- Decel
  - move\_settings\_calb\_t, 71
  - move\_settings\_t, 72
- DefaultGateway
  - network\_settings\_t, 74
- DetentTorque
  - motor\_settings\_t, 66
- device\_enumeration\_t
  - ximc.h, 157
- device\_information\_t, 31
  - Major, 32
  - Minor, 32
  - Release, 32
- device\_network\_information\_t, 32
  - ximc.h, 157
- DHCPEnabled
  - network\_settings\_t, 74
- DRIVER\_TYPE\_EXTERNAL
  - ximc.h, 131
- DRIVER\_TYPE\_INTEGRATE
  - ximc.h, 131
- DriverType
  - entype\_settings\_t, 45
- edges\_settings\_calb\_t, 33
  - BorderFlags, 33
  - EnderFlags, 33
  - LeftBorder, 33
  - RightBorder, 34
- edges\_settings\_t, 34
  - BorderFlags, 35
  - EnderFlags, 35
  - LeftBorder, 35
  - RightBorder, 35
  - uLeftBorder, 35
  - uRightBorder, 35

- EEPROM\_PRECEDENCE
  - ximc.h, [132](#)
- Efficiency
  - gear\_settings\_t, [50](#)
- emf\_settings\_t, [36](#)
  - BackEMFFlags, [36](#)
  - Km, [36](#)
  - L, [36](#)
  - R, [37](#)
- Enc\_Check
  - analog\_data\_t, [16](#)
- Enc\_Check\_ADC
  - analog\_data\_t, [16](#)
- ENC\_STATE\_ABSENT
  - ximc.h, [132](#)
- ENC\_STATE\_MALFUNC
  - ximc.h, [132](#)
- ENC\_STATE\_OK
  - ximc.h, [132](#)
- ENC\_STATE\_REVERS
  - ximc.h, [132](#)
- ENC\_STATE\_UNKNOWN
  - ximc.h, [132](#)
- encoder\_information\_t, [37](#)
  - Manufacturer, [37](#)
  - PartNumber, [37](#)
- encoder\_settings\_t, [38](#)
  - EncoderSettings, [38](#)
  - MaxCurrentConsumption, [38](#)
  - MaxOperatingFrequency, [39](#)
  - SupplyVoltageMax, [39](#)
  - SupplyVoltageMin, [39](#)
- EncoderSettings
  - encoder\_settings\_t, [38](#)
- EncPosition
  - get\_position\_calb\_t, [52](#)
  - get\_position\_t, [53](#)
  - set\_position\_calb\_t, [82](#)
  - set\_position\_t, [83](#)
  - status\_calb\_t, [89](#)
  - status\_t, [93](#)
- EncSts
  - status\_calb\_t, [90](#)
  - status\_t, [93](#)
- ENDER\_SW1\_ACTIVE\_LOW
  - ximc.h, [132](#)
- ENDER\_SW2\_ACTIVE\_LOW
  - ximc.h, [133](#)
- ENDER\_SWAP
  - ximc.h, [133](#)
- EnderFlags
  - edges\_settings\_calb\_t, [33](#)
  - edges\_settings\_t, [35](#)
- ENGINE\_ACCEL\_ON
  - ximc.h, [133](#)
- engine\_advanced\_setup\_t, [39](#)
  - stepcloseloop\_Kp\_high, [40](#)
  - stepcloseloop\_Kp\_low, [40](#)
  - stepcloseloop\_Kw, [40](#)
- ENGINE\_ANTIPLAY
  - ximc.h, [133](#)
- ENGINE\_CURRENT\_AS\_RMS
  - ximc.h, [133](#)
- ENGINE\_LIMIT\_CURR
  - ximc.h, [133](#)
- ENGINE\_LIMIT\_RPM
  - ximc.h, [134](#)
- ENGINE\_LIMIT\_VOLT
  - ximc.h, [134](#)
- ENGINE\_MAX\_SPEED
  - ximc.h, [134](#)
- ENGINE\_REVERSE
  - ximc.h, [134](#)
- engine\_settings\_calb\_t, [40](#)
  - Antiplay, [41](#)
  - EngineFlags, [41](#)
  - MicrostepMode, [41](#)
  - NomCurrent, [42](#)
  - NomSpeed, [42](#)
  - NomVoltage, [42](#)
  - StepsPerRev, [42](#)
- engine\_settings\_t, [42](#)
  - Antiplay, [43](#)
  - EngineFlags, [43](#)
  - MicrostepMode, [44](#)
  - NomCurrent, [44](#)
  - NomSpeed, [44](#)
  - NomVoltage, [44](#)
  - StepsPerRev, [44](#)
  - uNomSpeed, [44](#)
- ENGINE\_TYPE\_2DC
  - ximc.h, [134](#)
- ENGINE\_TYPE\_BRUSHLESS
  - ximc.h, [135](#)
- ENGINE\_TYPE\_DC
  - ximc.h, [135](#)
- ENGINE\_TYPE\_NONE
  - ximc.h, [135](#)
- ENGINE\_TYPE\_STEP
  - ximc.h, [135](#)
- ENGINE\_TYPE\_TEST
  - ximc.h, [135](#)
- EngineFlags
  - engine\_settings\_calb\_t, [41](#)
  - engine\_settings\_t, [43](#)
- EngineType
  - entype\_settings\_t, [45](#)
- entype\_settings\_t, [45](#)
  - DriverType, [45](#)
  - EngineType, [45](#)
- enumerate\_devices
  - ximc.h, [166](#)

- ENUMERATE\_PROBE
  - ximc.h, [135](#)
- ExpFactor
  - joystick\_settings\_t, [62](#)
- extended\_settings\_t, [46](#)
- extio\_settings\_t, [46](#)
  - EXTIOModeFlags, [47](#)
  - EXTIOSetupFlags, [47](#)
- EXTIO\_SETUP\_INVERT
  - ximc.h, [135](#)
- EXTIO\_SETUP\_MODE\_IN\_ALARM
  - ximc.h, [136](#)
- EXTIO\_SETUP\_MODE\_IN\_BITS
  - ximc.h, [136](#)
- EXTIO\_SETUP\_MODE\_IN\_HOME
  - ximc.h, [136](#)
- EXTIO\_SETUP\_MODE\_IN\_MOVR
  - ximc.h, [136](#)
- EXTIO\_SETUP\_MODE\_IN\_NOP
  - ximc.h, [136](#)
- EXTIO\_SETUP\_MODE\_IN\_PWOF
  - ximc.h, [136](#)
- EXTIO\_SETUP\_MODE\_IN\_STOP
  - ximc.h, [136](#)
- EXTIO\_SETUP\_MODE\_OUT\_ALARM
  - ximc.h, [137](#)
- EXTIO\_SETUP\_MODE\_OUT\_BITS
  - ximc.h, [137](#)
- EXTIO\_SETUP\_MODE\_OUT\_MOTOR\_ON
  - ximc.h, [137](#)
- EXTIO\_SETUP\_MODE\_OUT\_MOVING
  - ximc.h, [137](#)
- EXTIO\_SETUP\_MODE\_OUT\_OFF
  - ximc.h, [137](#)
- EXTIO\_SETUP\_MODE\_OUT\_ON
  - ximc.h, [137](#)
- EXTIO\_SETUP\_OUTPUT
  - ximc.h, [137](#)
- EXTIOModeFlags
  - extio\_settings\_t, [47](#)
- EXTIOSetupFlags
  - extio\_settings\_t, [47](#)
- FastHome
  - home\_settings\_calb\_t, [57](#)
  - home\_settings\_t, [59](#)
- FEEDBACK\_EMF
  - ximc.h, [138](#)
- FEEDBACK\_ENC\_ADAPTIVE\_HOLDING
  - ximc.h, [138](#)
- FEEDBACK\_ENC\_FILTER\_BITS
  - ximc.h, [138](#)
- FEEDBACK\_ENC\_FILTER\_MEDIUM
  - ximc.h, [138](#)
- FEEDBACK\_ENC\_FILTER\_NONE
  - ximc.h, [138](#)
- FEEDBACK\_ENC\_FILTER\_STRONG
  - ximc.h, [138](#)
- FEEDBACK\_ENC\_FILTER\_WEAK
  - ximc.h, [138](#)
- FEEDBACK\_ENC\_REVERSE
  - ximc.h, [139](#)
- FEEDBACK\_ENC\_TYPE\_AUTO
  - ximc.h, [139](#)
- FEEDBACK\_ENC\_TYPE\_BITS
  - ximc.h, [139](#)
- FEEDBACK\_ENC\_TYPE\_DIFFERENTIAL
  - ximc.h, [139](#)
- FEEDBACK\_ENC\_TYPE\_SINGLE\_ENDED
  - ximc.h, [139](#)
- FEEDBACK\_ENCODER
  - ximc.h, [139](#)
- FEEDBACK\_ENCODER\_MEDIATED
  - ximc.h, [139](#)
- FEEDBACK\_NONE
  - ximc.h, [140](#)
- feedback\_settings\_t, [47](#)
  - CountsPerTurn, [48](#)
  - FeedbackFlags, [48](#)
  - FeedbackType, [48](#)
  - IPS, [48](#)
- FeedbackFlags
  - feedback\_settings\_t, [48](#)
- FeedbackType
  - feedback\_settings\_t, [48](#)
- Flags
  - control\_settings\_calb\_t, [25](#)
  - control\_settings\_t, [27](#)
  - secure\_settings\_t, [80](#)
  - status\_calb\_t, [90](#)
  - status\_t, [93](#)
- free\_enumerate\_devices
  - ximc.h, [168](#)
- FullCurrent
  - analog\_data\_t, [16](#)
- FullCurrent\_A
  - calibration\_settings\_t, [20](#)
- FullCurrent\_ADC
  - analog\_data\_t, [16](#)
- FullCurrent\_B
  - calibration\_settings\_t, [20](#)
- gear\_information\_t, [48](#)
  - Manufacturer, [49](#)
  - PartNumber, [49](#)
- gear\_settings\_t, [49](#)
  - Efficiency, [50](#)
  - InputInertia, [50](#)
  - MaxOutputBacklash, [50](#)
  - RatedInputSpeed, [51](#)
  - RatedInputTorque, [51](#)
  - ReductionIn, [51](#)



- ReductionOut, [51](#)
- get\_accessories\_settings
  - ximc.h, [169](#)
- get\_analog\_data
  - ximc.h, [169](#)
- get\_bootloader\_version
  - ximc.h, [169](#)
- get\_brake\_settings
  - ximc.h, [170](#)
- get\_calibration\_settings
  - ximc.h, [170](#)
- get\_chart\_data
  - ximc.h, [170](#)
- get\_control\_settings
  - ximc.h, [171](#)
- get\_control\_settings\_calb
  - ximc.h, [171](#)
- get\_controller\_name
  - ximc.h, [173](#)
- get\_ctp\_settings
  - ximc.h, [173](#)
- get\_debug\_read
  - ximc.h, [173](#)
- get\_device\_count
  - ximc.h, [174](#)
- get\_device\_information
  - ximc.h, [174](#)
- get\_device\_name
  - ximc.h, [174](#)
- get\_edges\_settings
  - ximc.h, [175](#)
- get\_edges\_settings\_calb
  - ximc.h, [175](#)
- get\_emf\_settings
  - ximc.h, [175](#)
- get\_encoder\_information
  - ximc.h, [176](#)
- get\_encoder\_settings
  - ximc.h, [176](#)
- get\_engine\_advanced\_setup
  - ximc.h, [176](#)
- get\_engine\_settings
  - ximc.h, [177](#)
- get\_engine\_settings\_calb
  - ximc.h, [177](#)
- get\_entype\_settings
  - ximc.h, [178](#)
- get\_enumerate\_device\_controller\_name
  - ximc.h, [178](#)
- get\_enumerate\_device\_information
  - ximc.h, [178](#)
- get\_enumerate\_device\_network\_information
  - ximc.h, [179](#)
- get\_enumerate\_device\_serial
  - ximc.h, [179](#)
- get\_enumerate\_device\_stage\_name
  - ximc.h, [179](#)
- get\_extended\_settings
  - ximc.h, [180](#)
- get\_extio\_settings
  - ximc.h, [180](#)
- get\_feedback\_settings
  - ximc.h, [180](#)
- get\_firmware\_version
  - ximc.h, [181](#)
- get\_gear\_information
  - ximc.h, [181](#)
- get\_gear\_settings
  - ximc.h, [181](#)
- get\_globally\_unique\_identifier
  - ximc.h, [182](#)
- get\_hallsensor\_information
  - ximc.h, [182](#)
- get\_hallsensor\_settings
  - ximc.h, [182](#)
- get\_home\_settings
  - ximc.h, [182](#)
- get\_home\_settings\_calb
  - ximc.h, [183](#)
- get\_init\_random
  - ximc.h, [183](#)
- get\_joystick\_settings
  - ximc.h, [184](#)
- get\_measurements
  - ximc.h, [184](#)
- get\_motor\_information
  - ximc.h, [184](#)
- get\_motor\_settings
  - ximc.h, [185](#)
- get\_move\_settings
  - ximc.h, [185](#)
- get\_move\_settings\_calb
  - ximc.h, [185](#)
- get\_network\_settings
  - ximc.h, [186](#)
- get\_nonvolatile\_memory
  - ximc.h, [186](#)
- get\_password\_settings
  - ximc.h, [186](#)
- get\_pid\_settings
  - ximc.h, [187](#)
- get\_position
  - ximc.h, [187](#)
- get\_position\_calb
  - ximc.h, [187](#)
- get\_position\_calb\_t, [51](#)
  - EncPosition, [52](#)
  - Position, [52](#)
- get\_position\_t, [52](#)
  - EncPosition, [53](#)
  - uPosition, [53](#)
- get\_power\_settings

- ximc.h, 188
- get\_secure\_settings
  - ximc.h, 188
- get\_serial\_number
  - ximc.h, 188
- get\_stage\_information
  - ximc.h, 189
- get\_stage\_name
  - ximc.h, 189
- get\_stage\_settings
  - ximc.h, 189
- get\_status
  - ximc.h, 189
- get\_status\_calb
  - ximc.h, 190
- get\_sync\_in\_settings
  - ximc.h, 190
- get\_sync\_in\_settings\_calb
  - ximc.h, 191
- get\_sync\_out\_settings
  - ximc.h, 191
- get\_sync\_out\_settings\_calb
  - ximc.h, 191
- get\_uart\_settings
  - ximc.h, 192
- globally\_unique\_identifier\_t, 53
  - UniqueID0, 54
  - UniqueID1, 54
  - UniqueID2, 54
  - UniqueID3, 54
- goto\_firmware
  - ximc.h, 192
- GPIOFlags
  - status\_calb\_t, 90
  - status\_t, 94
- H\_BRIDGE\_ALERT
  - ximc.h, 140
- hallsensor\_information\_t, 54
  - Manufacturer, 55
  - PartNumber, 55
- hallsensor\_settings\_t, 55
  - MaxCurrentConsumption, 56
  - MaxOperatingFrequency, 56
  - SupplyVoltageMax, 56
  - SupplyVoltageMin, 56
- has\_firmware
  - ximc.h, 192
- HoldCurrent
  - power\_settings\_t, 78
- HOME\_DIR\_FIRST
  - ximc.h, 140
- HOME\_DIR\_SECOND
  - ximc.h, 140
- HOME\_HALF\_MV
  - ximc.h, 140
- HOME\_MV\_SEC\_EN
  - ximc.h, 140
- home\_settings\_calb\_t, 57
  - FastHome, 57
  - HomeDelta, 57
  - HomeFlags, 58
  - SlowHome, 58
- home\_settings\_t, 58
  - FastHome, 59
  - HomeDelta, 59
  - HomeFlags, 59
  - SlowHome, 59
  - uFastHome, 59
  - uHomeDelta, 60
  - uSlowHome, 60
- HOME\_STOP\_FIRST\_BITS
  - ximc.h, 141
- HOME\_STOP\_FIRST\_LIM
  - ximc.h, 141
- HOME\_STOP\_FIRST\_REV
  - ximc.h, 141
- HOME\_STOP\_FIRST\_SYN
  - ximc.h, 141
- HOME\_STOP\_SECOND\_BITS
  - ximc.h, 141
- HOME\_STOP\_SECOND\_LIM
  - ximc.h, 141
- HOME\_STOP\_SECOND\_REV
  - ximc.h, 141
- HOME\_STOP\_SECOND\_SYN
  - ximc.h, 142
- HOME\_USE\_FAST
  - ximc.h, 142
- HomeDelta
  - home\_settings\_calb\_t, 57
  - home\_settings\_t, 59
- HomeFlags
  - home\_settings\_calb\_t, 58
  - home\_settings\_t, 59
- HorizontalLoadCapacity
  - stage\_settings\_t, 86
- init\_random\_t, 60
  - key, 61
- InputInertia
  - gear\_settings\_t, 50
- IPS
  - feedback\_settings\_t, 48
- IPv4Address
  - network\_settings\_t, 74
- Ipwr
  - status\_calb\_t, 90
  - status\_t, 94
- lusb
  - status\_calb\_t, 90
  - status\_t, 94

- Joy
  - analog\_data\_t, 16
  - chart\_data\_t, 23
- Joy\_ADC
  - analog\_data\_t, 16
- JOY\_REVERSE
  - ximc.h, 142
- JoyCenter
  - joystick\_settings\_t, 62
- JoyFlags
  - joystick\_settings\_t, 62
- JoyHighEnd
  - joystick\_settings\_t, 62
- JoyLowEnd
  - joystick\_settings\_t, 62
- joystick\_settings\_t, 61
  - DeadZone, 62
  - ExpFactor, 62
  - JoyCenter, 62
  - JoyFlags, 62
  - JoyHighEnd, 62
  - JoyLowEnd, 62
- Key
  - serial\_number\_t, 81
- key
  - init\_random\_t, 61
- Km
  - emf\_settings\_t, 36
- L
  - emf\_settings\_t, 36
- LeadScrewPitch
  - stage\_settings\_t, 86
- LeftBorder
  - edges\_settings\_calb\_t, 33
  - edges\_settings\_t, 35
- Length
  - measurements\_t, 63
- LimitSwitchesSettings
  - accessories\_settings\_t, 11
- logging\_callback\_stderr\_narrow
  - ximc.h, 193
- logging\_callback\_stderr\_wide
  - ximc.h, 193
- logging\_callback\_t
  - ximc.h, 157
- LOW\_UPWR\_PROTECTION
  - ximc.h, 142
- LowUpwrOff
  - secure\_settings\_t, 80
- LS\_SHORTED
  - ximc.h, 142
- MagneticBrakeInfo
  - accessories\_settings\_t, 11
- Major
  - device\_information\_t, 32
  - serial\_number\_t, 81
- Manufacturer
  - encoder\_information\_t, 37
  - gear\_information\_t, 49
  - hallsensor\_information\_t, 55
  - motor\_information\_t, 64
  - stage\_information\_t, 84
- MaxClickTime
  - control\_settings\_calb\_t, 25
  - control\_settings\_t, 27
- MaxCurrent
  - motor\_settings\_t, 66
- MaxCurrentConsumption
  - encoder\_settings\_t, 38
  - hallsensor\_settings\_t, 56
  - stage\_settings\_t, 86
- MaxCurrentTime
  - motor\_settings\_t, 66
- MaxOperatingFrequency
  - encoder\_settings\_t, 39
  - hallsensor\_settings\_t, 56
- MaxOutputBacklash
  - gear\_settings\_t, 50
- MaxSpeed
  - control\_settings\_calb\_t, 25
  - control\_settings\_t, 27
  - motor\_settings\_t, 66
  - stage\_settings\_t, 87
- MBRatedCurrent
  - accessories\_settings\_t, 11
- MBRatedVoltage
  - accessories\_settings\_t, 11
- MBSettings
  - accessories\_settings\_t, 11
- MBTorque
  - accessories\_settings\_t, 11
- measurements\_t, 63
  - Length, 63
- MechanicalTimeConstant
  - motor\_settings\_t, 66
- MICROSTEP\_MODE\_FRAC\_128
  - ximc.h, 142
- MICROSTEP\_MODE\_FRAC\_16
  - ximc.h, 142
- MICROSTEP\_MODE\_FRAC\_2
  - ximc.h, 143
- MICROSTEP\_MODE\_FRAC\_256
  - ximc.h, 143
- MICROSTEP\_MODE\_FRAC\_32
  - ximc.h, 143
- MICROSTEP\_MODE\_FRAC\_4
  - ximc.h, 143
- MICROSTEP\_MODE\_FRAC\_64
  - ximc.h, 143
- MICROSTEP\_MODE\_FRAC\_8

- ximc.h, [143](#)
- MICROSTEP\_MODE\_FULL
  - ximc.h, [143](#)
- MicrostepMode
  - calibration\_t, [22](#)
  - engine\_settings\_calb\_t, [41](#)
  - engine\_settings\_t, [44](#)
- MinimumUusb
  - secure\_settings\_t, [80](#)
- Minor
  - device\_information\_t, [32](#)
  - serial\_number\_t, [81](#)
- motor\_information\_t, [63](#)
  - Manufacturer, [64](#)
  - PartNumber, [64](#)
- motor\_settings\_t, [64](#)
  - DetentTorque, [66](#)
  - MaxCurrent, [66](#)
  - MaxCurrentTime, [66](#)
  - MaxSpeed, [66](#)
  - MechanicalTimeConstant, [66](#)
  - MotorType, [67](#)
  - NoLoadCurrent, [67](#)
  - NoLoadSpeed, [67](#)
  - NominalCurrent, [67](#)
  - NominalPower, [67](#)
  - NominalSpeed, [67](#)
  - NominalTorque, [68](#)
  - NominalVoltage, [68](#)
  - Phases, [68](#)
  - Poles, [68](#)
  - RotorInertia, [68](#)
  - SpeedConstant, [68](#)
  - SpeedTorqueGradient, [69](#)
  - StallTorque, [69](#)
  - TorqueConstant, [69](#)
  - WindingInductance, [69](#)
  - WindingResistance, [69](#)
- MotorType
  - motor\_settings\_t, [67](#)
- move\_settings\_calb\_t, [70](#)
  - Accel, [70](#)
  - AntiplaySpeed, [70](#)
  - Decel, [71](#)
  - MoveFlags, [71](#)
  - Speed, [71](#)
- move\_settings\_t, [71](#)
  - Accel, [72](#)
  - AntiplaySpeed, [72](#)
  - Decel, [72](#)
  - MoveFlags, [73](#)
  - Speed, [73](#)
  - uAntiplaySpeed, [73](#)
  - uSpeed, [73](#)
- MOVE\_STATE\_ANTIPLAY
  - ximc.h, [144](#)
- MOVE\_STATE\_MOVING
  - ximc.h, [144](#)
- MOVE\_STATE\_TARGET\_SPEED
  - ximc.h, [144](#)
- MoveFlags
  - move\_settings\_calb\_t, [71](#)
  - move\_settings\_t, [73](#)
- MoveSts
  - status\_calb\_t, [90](#)
  - status\_t, [94](#)
- msec\_sleep
  - ximc.h, [193](#)
- MVCMD\_ERROR
  - ximc.h, [144](#)
- MVCMD\_HOME
  - ximc.h, [144](#)
- MVCMD\_LEFT
  - ximc.h, [144](#)
- MVCMD\_LOFT
  - ximc.h, [144](#)
- MVCMD\_MOVE
  - ximc.h, [145](#)
- MVCMD\_MOVR
  - ximc.h, [145](#)
- MVCMD\_NAME\_BITS
  - ximc.h, [145](#)
- MVCMD\_RIGHT
  - ximc.h, [145](#)
- MVCMD\_RUNNING
  - ximc.h, [145](#)
- MVCMD\_SSTP
  - ximc.h, [145](#)
- MVCMD\_STOP
  - ximc.h, [145](#)
- MVCMD\_UKNWN
  - ximc.h, [146](#)
- MvCmdSts
  - status\_calb\_t, [90](#)
  - status\_t, [94](#)
- network\_settings\_t, [73](#)
  - DefaultGateway, [74](#)
  - DHCPEnabled, [74](#)
  - IPv4Address, [74](#)
  - SubnetMask, [74](#)
- NoLoadCurrent
  - motor\_settings\_t, [67](#)
- NoLoadSpeed
  - motor\_settings\_t, [67](#)
- NomCurrent
  - engine\_settings\_calb\_t, [42](#)
  - engine\_settings\_t, [44](#)
- NominalCurrent
  - motor\_settings\_t, [67](#)
- NominalPower
  - motor\_settings\_t, [67](#)

- NominalSpeed
  - motor\_settings\_t, 67
- NominalTorque
  - motor\_settings\_t, 68
- NominalVoltage
  - motor\_settings\_t, 68
- NomSpeed
  - engine\_settings\_calb\_t, 42
  - engine\_settings\_t, 44
- NomVoltage
  - engine\_settings\_calb\_t, 42
  - engine\_settings\_t, 44
- nonvolatile\_memory\_t, 75
  - UserData, 75
- open\_device
  - ximc.h, 193
- PartNumber
  - encoder\_information\_t, 37
  - gear\_information\_t, 49
  - hallsensor\_information\_t, 55
  - motor\_information\_t, 64
  - stage\_information\_t, 84
- password\_settings\_t, 75
  - UserPassword, 76
- Phases
  - motor\_settings\_t, 68
- pid\_settings\_t, 76
- Poles
  - motor\_settings\_t, 68
- PosFlags
  - set\_position\_calb\_t, 82
  - set\_position\_t, 83
- Position
  - get\_position\_calb\_t, 52
  - set\_position\_calb\_t, 83
  - sync\_in\_settings\_calb\_t, 96
- PositionerName
  - stage\_name\_t, 85
- Pot
  - analog\_data\_t, 16
  - chart\_data\_t, 23
- POWER\_OFF\_ENABLED
  - ximc.h, 146
- POWER\_REDUCT\_ENABLED
  - ximc.h, 146
- power\_settings\_t, 77
  - CurrentSetTime, 78
  - CurrReductDelay, 78
  - HoldCurrent, 78
  - PowerFlags, 78
  - PowerOffDelay, 78
- POWER\_SMOOTH\_CURRENT
  - ximc.h, 146
- PowerFlags
  - power\_settings\_t, 78
- PowerOffDelay
  - power\_settings\_t, 78
- probe\_device
  - ximc.h, 194
- PWR\_STATE\_MAX
  - ximc.h, 146
- PWR\_STATE\_NORM
  - ximc.h, 146
- PWR\_STATE\_OFF
  - ximc.h, 146
- PWR\_STATE\_REDUCT
  - ximc.h, 147
- PWR\_STATE\_UNKNOWN
  - ximc.h, 147
- PWRSts
  - status\_calb\_t, 91
  - status\_t, 94
- R
  - emf\_settings\_t, 37
- RatedInputSpeed
  - gear\_settings\_t, 51
- RatedInputTorque
  - gear\_settings\_t, 51
- ReductionIn
  - gear\_settings\_t, 51
- ReductionOut
  - gear\_settings\_t, 51
- Release
  - device\_information\_t, 32
  - serial\_number\_t, 81
- reset\_locks
  - ximc.h, 194
- result\_t
  - ximc.h, 157
- REV\_SENS\_INV
  - ximc.h, 147
- RightBorder
  - edges\_settings\_calb\_t, 34
  - edges\_settings\_t, 35
- RotorInertia
  - motor\_settings\_t, 68
- RPM\_DIV\_1000
  - ximc.h, 147
- secure\_settings\_t, 78
  - CriticalIpwr, 79
  - Criticalusb, 79
  - CriticalT, 79
  - CriticalUpwr, 80
  - CriticalUusb, 80
  - Flags, 80
  - LowUpwrOff, 80
  - MinimumUusb, 80
- serial\_number\_t, 80
  - Key, 81
  - Major, 81

- Minor, [81](#)
- Release, [81](#)
- SN, [82](#)
- service\_command\_updf
  - ximc.h, [194](#)
- set\_accessories\_settings
  - ximc.h, [194](#)
- set\_brake\_settings
  - ximc.h, [195](#)
- set\_calibration\_settings
  - ximc.h, [195](#)
- set\_control\_settings
  - ximc.h, [195](#)
- set\_control\_settings\_calb
  - ximc.h, [196](#)
- set\_controller\_name
  - ximc.h, [196](#)
- set\_correction\_table
  - ximc.h, [197](#)
- set\_ctp\_settings
  - ximc.h, [197](#)
- set\_debug\_write
  - ximc.h, [198](#)
- set\_edges\_settings
  - ximc.h, [198](#)
- set\_edges\_settings\_calb
  - ximc.h, [198](#)
- set\_emf\_settings
  - ximc.h, [199](#)
- set\_encoder\_information
  - ximc.h, [199](#)
- set\_encoder\_settings
  - ximc.h, [200](#)
- set\_engine\_advanced\_setup
  - ximc.h, [200](#)
- set\_engine\_settings
  - ximc.h, [200](#)
- set\_engine\_settings\_calb
  - ximc.h, [201](#)
- set\_entype\_settings
  - ximc.h, [201](#)
- set\_extended\_settings
  - ximc.h, [201](#)
- set\_extio\_settings
  - ximc.h, [202](#)
- set\_feedback\_settings
  - ximc.h, [202](#)
- set\_gear\_information
  - ximc.h, [202](#)
- set\_gear\_settings
  - ximc.h, [203](#)
- set\_hallsensor\_information
  - ximc.h, [203](#)
- set\_hallsensor\_settings
  - ximc.h, [203](#)
- set\_home\_settings
  - ximc.h, [204](#)
- set\_home\_settings\_calb
  - ximc.h, [204](#)
- set\_joystick\_settings
  - ximc.h, [204](#)
- set\_logging\_callback
  - ximc.h, [205](#)
- set\_motor\_information
  - ximc.h, [205](#)
- set\_motor\_settings
  - ximc.h, [206](#)
- set\_move\_settings
  - ximc.h, [206](#)
- set\_move\_settings\_calb
  - ximc.h, [206](#)
- set\_network\_settings
  - ximc.h, [207](#)
- set\_nonvolatile\_memory
  - ximc.h, [207](#)
- set\_password\_settings
  - ximc.h, [207](#)
- set\_pid\_settings
  - ximc.h, [208](#)
- set\_position
  - ximc.h, [208](#)
- set\_position\_calb
  - ximc.h, [208](#)
- set\_position\_calb\_t, [82](#)
  - EncPosition, [82](#)
  - PosFlags, [82](#)
  - Position, [83](#)
- set\_position\_t, [83](#)
  - EncPosition, [83](#)
  - PosFlags, [83](#)
  - uPosition, [84](#)
- set\_power\_settings
  - ximc.h, [209](#)
- set\_secure\_settings
  - ximc.h, [209](#)
- set\_serial\_number
  - ximc.h, [209](#)
- set\_stage\_information
  - ximc.h, [210](#)
- set\_stage\_name
  - ximc.h, [210](#)
- set\_stage\_settings
  - ximc.h, [210](#)
- set\_sync\_in\_settings
  - ximc.h, [210](#)
- set\_sync\_in\_settings\_calb
  - ximc.h, [211](#)
- set\_sync\_out\_settings
  - ximc.h, [211](#)
- set\_sync\_out\_settings\_calb
  - ximc.h, [212](#)
- set\_uart\_settings

- ximc.h, [212](#)
- SETPOS\_IGNORE\_ENCODER
  - ximc.h, [147](#)
- SETPOS\_IGNORE\_POSITION
  - ximc.h, [147](#)
- SlowHome
  - home\_settings\_calb\_t, [58](#)
  - home\_settings\_t, [59](#)
- SN
  - serial\_number\_t, [82](#)
- Speed
  - move\_settings\_calb\_t, [71](#)
  - move\_settings\_t, [73](#)
  - sync\_in\_settings\_calb\_t, [96](#)
  - sync\_in\_settings\_t, [97](#)
- SpeedConstant
  - motor\_settings\_t, [68](#)
- SpeedTorqueGradient
  - motor\_settings\_t, [69](#)
- stage\_information\_t, [84](#)
  - Manufacturer, [84](#)
  - PartNumber, [84](#)
- stage\_name\_t, [85](#)
  - PositionerName, [85](#)
- stage\_settings\_t, [85](#)
  - HorizontalLoadCapacity, [86](#)
  - LeadScrewPitch, [86](#)
  - MaxCurrentConsumption, [86](#)
  - MaxSpeed, [87](#)
  - SupplyVoltageMax, [87](#)
  - SupplyVoltageMin, [87](#)
  - TravelRange, [87](#)
  - Units, [87](#)
  - VerticalLoadCapacity, [87](#)
- StallTorque
  - motor\_settings\_t, [69](#)
- STATE\_ALARM
  - ximc.h, [147](#)
- STATE\_BORDERS\_SWAP\_MISSET
  - ximc.h, [148](#)
- STATE\_BRAKE
  - ximc.h, [148](#)
- STATE\_BUTTON\_LEFT
  - ximc.h, [148](#)
- STATE\_BUTTON\_RIGHT
  - ximc.h, [148](#)
- STATE\_CONTR
  - ximc.h, [148](#)
- STATE\_CONTROLLER\_OVERHEAT
  - ximc.h, [148](#)
- STATE\_CTP\_ERROR
  - ximc.h, [148](#)
- STATE\_DIG\_SIGNAL
  - ximc.h, [149](#)
- STATE\_EEPROM\_CONNECTED
  - ximc.h, [149](#)
- STATE\_ENC\_A
  - ximc.h, [149](#)
- STATE\_ENC\_B
  - ximc.h, [149](#)
- STATE\_ENGINE\_RESPONSE\_ERROR
  - ximc.h, [149](#)
- STATE\_ERRC
  - ximc.h, [149](#)
- STATE\_ERRD
  - ximc.h, [150](#)
- STATE\_ERRV
  - ximc.h, [150](#)
- STATE\_EXTIO\_ALARM
  - ximc.h, [150](#)
- STATE\_GPIO\_LEVEL
  - ximc.h, [150](#)
- STATE\_GPIO\_PINOUT
  - ximc.h, [150](#)
- STATE\_IS\_HOMED
  - ximc.h, [150](#)
- STATE\_LEFT\_EDGE
  - ximc.h, [151](#)
- STATE\_LOW\_USB\_VOLTAGE
  - ximc.h, [151](#)
- STATE\_OVERLOAD\_POWER\_CURRENT
  - ximc.h, [151](#)
- STATE\_OVERLOAD\_POWER\_VOLTAGE
  - ximc.h, [151](#)
- STATE\_OVERLOAD\_USB\_CURRENT
  - ximc.h, [151](#)
- STATE\_OVERLOAD\_USB\_VOLTAGE
  - ximc.h, [151](#)
- STATE\_POWER\_OVERHEAT
  - ximc.h, [152](#)
- STATE\_REV\_SENSOR
  - ximc.h, [152](#)
- STATE\_RIGHT\_EDGE
  - ximc.h, [152](#)
- STATE\_SECUR
  - ximc.h, [152](#)
- STATE\_SYNC\_INPUT
  - ximc.h, [152](#)
- STATE\_SYNC\_OUTPUT
  - ximc.h, [152](#)
- STATE\_WINDING\_RES\_MISMATCH
  - ximc.h, [152](#)
- status\_calb\_t, [88](#)
  - CmdBufFreeSpace, [89](#)
  - CurPosition, [89](#)
  - CurSpeed, [89](#)
  - CurT, [89](#)
  - EncPosition, [89](#)
  - EncSts, [90](#)
  - Flags, [90](#)
  - GPIOFlags, [90](#)
  - Ipwr, [90](#)

- lusb, 90
- MoveSts, 90
- MvCmdSts, 90
- PWRSts, 91
- Upwr, 91
- Uusb, 91
- WindSts, 91
- status\_t, 91
  - CmdBufFreeSpace, 93
  - CurPosition, 93
  - CurSpeed, 93
  - CurT, 93
  - EncPosition, 93
  - EncSts, 93
  - Flags, 93
  - GPIOFlags, 94
  - Ipwr, 94
  - lusb, 94
  - MoveSts, 94
  - MvCmdSts, 94
  - PWRSts, 94
  - uCurPosition, 94
  - uCurSpeed, 95
  - Upwr, 95
  - Uusb, 95
  - WindSts, 95
- stepcloseloop\_Kp\_high
  - engine\_advanced\_setup\_t, 40
- stepcloseloop\_Kp\_low
  - engine\_advanced\_setup\_t, 40
- stepcloseloop\_Kw
  - engine\_advanced\_setup\_t, 40
- StepsPerRev
  - engine\_settings\_calb\_t, 42
  - engine\_settings\_t, 44
- SubnetMask
  - network\_settings\_t, 74
- SupplyVoltageMax
  - encoder\_settings\_t, 39
  - hallsensor\_settings\_t, 56
  - stage\_settings\_t, 87
- SupplyVoltageMin
  - encoder\_settings\_t, 39
  - hallsensor\_settings\_t, 56
  - stage\_settings\_t, 87
- SupVoltage
  - analog\_data\_t, 17
- SupVoltage\_ADC
  - analog\_data\_t, 17
- sync\_in\_settings\_calb\_t, 95
  - ClutterTime, 96
  - Position, 96
  - Speed, 96
  - SyncInFlags, 96
- sync\_in\_settings\_t, 97
  - ClutterTime, 97
  - Speed, 97
  - SyncInFlags, 98
  - uPosition, 98
  - uSpeed, 98
- sync\_out\_settings\_calb\_t, 98
  - Accuracy, 99
  - SyncOutFlags, 99
  - SyncOutPeriod, 99
  - SyncOutPulseSteps, 99
- sync\_out\_settings\_t, 99
  - Accuracy, 100
  - SyncOutFlags, 100
  - SyncOutPeriod, 100
  - SyncOutPulseSteps, 101
  - uAccuracy, 101
- SYNCIN\_ENABLED
  - ximc.h, 153
- SYNCIN\_INVERT
  - ximc.h, 153
- SyncInFlags
  - sync\_in\_settings\_calb\_t, 96
  - sync\_in\_settings\_t, 98
- SYNCOUT\_ENABLED
  - ximc.h, 153
- SYNCOUT\_IN\_STEPS
  - ximc.h, 153
- SYNCOUT\_INVERT
  - ximc.h, 153
- SYNCOUT\_ONPERIOD
  - ximc.h, 153
- SYNCOUT\_ONSTART
  - ximc.h, 154
- SYNCOUT\_ONSTOP
  - ximc.h, 154
- SYNCOUT\_STATE
  - ximc.h, 154
- SyncOutFlags
  - sync\_out\_settings\_calb\_t, 99
  - sync\_out\_settings\_t, 100
- SyncOutPeriod
  - sync\_out\_settings\_calb\_t, 99
  - sync\_out\_settings\_t, 100
- SyncOutPulseSteps
  - sync\_out\_settings\_calb\_t, 99
  - sync\_out\_settings\_t, 101
- t1
  - brake\_settings\_t, 18
- t2
  - brake\_settings\_t, 18
- t3
  - brake\_settings\_t, 18
- t4
  - brake\_settings\_t, 18
- Temp
  - analog\_data\_t, 17



- Temp\_ADC
  - analog\_data\_t, 17
- TemperatureSensorInfo
  - accessories\_settings\_t, 12
- Timeout
  - control\_settings\_calb\_t, 26
  - control\_settings\_t, 27
- TorqueConstant
  - motor\_settings\_t, 69
- TravelRange
  - stage\_settings\_t, 87
- TS\_TYPE\_BITS
  - ximc.h, 154
- TSGrad
  - accessories\_settings\_t, 12
- TSMaх
  - accessories\_settings\_t, 12
- TSMin
  - accessories\_settings\_t, 12
- TSSettings
  - accessories\_settings\_t, 12
- uAccuracy
  - sync\_out\_settings\_t, 101
- uAntiplaySpeed
  - move\_settings\_t, 73
- UART\_PARITY\_BITS
  - ximc.h, 154
- uart\_settings\_t, 101
  - UARTSetupFlags, 101
- UARTSetupFlags
  - uart\_settings\_t, 101
- uCurPosition
  - status\_t, 94
- uCurSpeed
  - status\_t, 95
- uDeltaPosition
  - control\_settings\_t, 27
- uFastHome
  - home\_settings\_t, 59
- uHomeDelta
  - home\_settings\_t, 60
- uLeftBorder
  - edges\_settings\_t, 35
- uMaxSpeed
  - control\_settings\_t, 27
- UniquelD0
  - globally\_unique\_identifier\_t, 54
- UniquelD1
  - globally\_unique\_identifier\_t, 54
- UniquelD2
  - globally\_unique\_identifier\_t, 54
- UniquelD3
  - globally\_unique\_identifier\_t, 54
- Units
  - stage\_settings\_t, 87
- uNomSpeed
  - engine\_settings\_t, 44
- uPosition
  - get\_position\_t, 53
  - set\_position\_t, 84
  - sync\_in\_settings\_t, 98
- Upwr
  - status\_calb\_t, 91
  - status\_t, 95
- uRightBorder
  - edges\_settings\_t, 35
- UserData
  - nonvolatile\_memory\_t, 75
- UserPassword
  - password\_settings\_t, 76
- uSlowHome
  - home\_settings\_t, 60
- uSpeed
  - move\_settings\_t, 73
  - sync\_in\_settings\_t, 98
- Uusb
  - status\_calb\_t, 91
  - status\_t, 95
- VerticalLoadCapacity
  - stage\_settings\_t, 87
- WIND\_A\_STATE\_ABSENT
  - ximc.h, 154
- WIND\_A\_STATE\_MALFUNC
  - ximc.h, 154
- WIND\_A\_STATE\_OK
  - ximc.h, 155
- WIND\_A\_STATE\_UNKNOWN
  - ximc.h, 155
- WIND\_B\_STATE\_ABSENT
  - ximc.h, 155
- WIND\_B\_STATE\_MALFUNC
  - ximc.h, 155
- WIND\_B\_STATE\_OK
  - ximc.h, 155
- WIND\_B\_STATE\_UNKNOWN
  - ximc.h, 155
- WindingCurrentA
  - chart\_data\_t, 23
- WindingCurrentB
  - chart\_data\_t, 23
- WindingCurrentC
  - chart\_data\_t, 24
- WindingInductance
  - motor\_settings\_t, 69
- WindingResistance
  - motor\_settings\_t, 69
- WindingVoltageA
  - chart\_data\_t, 24
- WindingVoltageB
  - chart\_data\_t, 24

- WindingVoltageC
  - chart\_data\_t, 24
- WindSts
  - status\_calb\_t, 91
  - status\_t, 95
- write\_key
  - ximc.h, 212
- ximc.h, 102, 213
  - ALARM\_FLAGS\_STICKING, 128
  - ALARM\_ON\_BORDERS\_SWAP\_MISSET, 128
  - ALARM\_ON\_DRIVER\_OVERHEATING, 128
  - BACK\_EMF\_INDUCTANCE\_AUTO, 128
  - BACK\_EMF\_KM\_AUTO, 129
  - BACK\_EMF\_RESISTANCE\_AUTO, 129
  - BORDER\_IS\_ENCODER, 129
  - BORDER\_STOP\_LEFT, 129
  - BORDER\_STOP\_RIGHT, 129
  - BORDERS\_SWAP\_MISSET\_DETECTION, 129
  - BRAKE\_ENABLED, 129
  - BRAKE\_ENG\_PWROFF, 130
  - BRAKING\_OVERVOLTAGE\_PROTECTION, 130
  - calibration\_t, 156
  - close\_device, 157
  - command\_clear\_fram, 158
  - command\_eeread\_settings, 158
  - command\_eesave\_settings, 158
  - command\_home, 159
  - command\_homezero, 159
  - command\_left, 160
  - command\_loft, 160
  - command\_move, 160
  - command\_move\_calb, 160
  - command\_movr, 161
  - command\_movr\_calb, 161
  - command\_power\_off, 163
  - command\_read\_robust\_settings, 163
  - command\_read\_settings, 163
  - command\_reset, 164
  - command\_right, 164
  - command\_save\_robust\_settings, 164
  - command\_save\_settings, 164
  - command\_sstp, 165
  - command\_start\_measurements, 165
  - command\_stop, 165
  - command\_update\_firmware, 165
  - command\_wait\_for\_stop, 166
  - command\_zero, 166
  - CONTROL\_BTN\_LEFT\_PUSHED\_OPEN, 130
  - CONTROL\_BTN\_RIGHT\_PUSHED\_OPEN, 130
  - CONTROL\_MODE\_BITS, 130
  - CONTROL\_MODE\_JOY, 130
  - CONTROL\_MODE\_LR, 130
  - CONTROL\_MODE\_OFF, 131
  - CTP\_ALARM\_ON\_ERROR, 131
  - CTP\_BASE, 131
  - CTP\_ENABLED, 131
  - CTP\_ERROR\_CORRECTION, 131
  - device\_enumeration\_t, 157
  - device\_network\_information\_t, 157
  - DRIVER\_TYPE\_EXTERNAL, 131
  - DRIVER\_TYPE\_INTEGRATE, 131
  - EEPROM\_PRECEDENCE, 132
  - ENC\_STATE\_ABSENT, 132
  - ENC\_STATE\_MALFUNC, 132
  - ENC\_STATE\_OK, 132
  - ENC\_STATE\_REVERS, 132
  - ENC\_STATE\_UNKNOWN, 132
  - ENDER\_SW1\_ACTIVE\_LOW, 132
  - ENDER\_SW2\_ACTIVE\_LOW, 133
  - ENDER\_SWAP, 133
  - ENGINE\_ACCEL\_ON, 133
  - ENGINE\_ANTIPLAY, 133
  - ENGINE\_CURRENT\_AS\_RMS, 133
  - ENGINE\_LIMIT\_CURR, 133
  - ENGINE\_LIMIT\_RPM, 134
  - ENGINE\_LIMIT\_VOLT, 134
  - ENGINE\_MAX\_SPEED, 134
  - ENGINE\_REVERSE, 134
  - ENGINE\_TYPE\_2DC, 134
  - ENGINE\_TYPE\_BRUSHLESS, 135
  - ENGINE\_TYPE\_DC, 135
  - ENGINE\_TYPE\_NONE, 135
  - ENGINE\_TYPE\_STEP, 135
  - ENGINE\_TYPE\_TEST, 135
  - enumerate\_devices, 166
  - ENUMERATE\_PROBE, 135
  - EXTIO\_SETUP\_INVERT, 135
  - EXTIO\_SETUP\_MODE\_IN\_ALARM, 136
  - EXTIO\_SETUP\_MODE\_IN\_BITS, 136
  - EXTIO\_SETUP\_MODE\_IN\_HOME, 136
  - EXTIO\_SETUP\_MODE\_IN\_MOVR, 136
  - EXTIO\_SETUP\_MODE\_IN\_NOP, 136
  - EXTIO\_SETUP\_MODE\_IN\_PWOF, 136
  - EXTIO\_SETUP\_MODE\_IN\_STOP, 136
  - EXTIO\_SETUP\_MODE\_OUT\_ALARM, 137
  - EXTIO\_SETUP\_MODE\_OUT\_BITS, 137
  - EXTIO\_SETUP\_MODE\_OUT\_MOTOR\_ON, 137
  - EXTIO\_SETUP\_MODE\_OUT\_MOVING, 137
  - EXTIO\_SETUP\_MODE\_OUT\_OFF, 137
  - EXTIO\_SETUP\_MODE\_OUT\_ON, 137
  - EXTIO\_SETUP\_OUTPUT, 137
  - FEEDBACK\_EMF, 138

- FEEDBACK\_ENC\_ADAPTIVE\_HOLDING, 138
- FEEDBACK\_ENC\_FILTER\_BITS, 138
- FEEDBACK\_ENC\_FILTER\_MEDIUM, 138
- FEEDBACK\_ENC\_FILTER\_NONE, 138
- FEEDBACK\_ENC\_FILTER\_STRONG, 138
- FEEDBACK\_ENC\_FILTER\_WEAK, 138
- FEEDBACK\_ENC\_REVERSE, 139
- FEEDBACK\_ENC\_TYPE\_AUTO, 139
- FEEDBACK\_ENC\_TYPE\_BITS, 139
- FEEDBACK\_ENC\_TYPE\_DIFFERENTIAL, 139
- FEEDBACK\_ENC\_TYPE\_SINGLE\_ENDED, 139
- FEEDBACK\_ENCODER, 139
- FEEDBACK\_ENCODER\_MEDIATED, 139
- FEEDBACK\_NONE, 140
- free\_enumerate\_devices, 168
- get\_accessories\_settings, 169
- get\_analog\_data, 169
- get\_bootloader\_version, 169
- get\_brake\_settings, 170
- get\_calibration\_settings, 170
- get\_chart\_data, 170
- get\_control\_settings, 171
- get\_control\_settings\_calb, 171
- get\_controller\_name, 173
- get\_ctp\_settings, 173
- get\_debug\_read, 173
- get\_device\_count, 174
- get\_device\_information, 174
- get\_device\_name, 174
- get\_edges\_settings, 175
- get\_edges\_settings\_calb, 175
- get\_emf\_settings, 175
- get\_encoder\_information, 176
- get\_encoder\_settings, 176
- get\_engine\_advanced\_setup, 176
- get\_engine\_settings, 177
- get\_engine\_settings\_calb, 177
- get\_entype\_settings, 178
- get\_enumerate\_device\_controller\_name, 178
- get\_enumerate\_device\_information, 178
- get\_enumerate\_device\_network\_information, 179
- get\_enumerate\_device\_serial, 179
- get\_enumerate\_device\_stage\_name, 179
- get\_extended\_settings, 180
- get\_extio\_settings, 180
- get\_feedback\_settings, 180
- get\_firmware\_version, 181
- get\_gear\_information, 181
- get\_gear\_settings, 181
- get\_globally\_unique\_identifier, 182
- get\_hallsensor\_information, 182
- get\_hallsensor\_settings, 182
- get\_home\_settings, 182
- get\_home\_settings\_calb, 183
- get\_init\_random, 183
- get\_joystick\_settings, 184
- get\_measurements, 184
- get\_motor\_information, 184
- get\_motor\_settings, 185
- get\_move\_settings, 185
- get\_move\_settings\_calb, 185
- get\_network\_settings, 186
- get\_nonvolatile\_memory, 186
- get\_password\_settings, 186
- get\_pid\_settings, 187
- get\_position, 187
- get\_position\_calb, 187
- get\_power\_settings, 188
- get\_secure\_settings, 188
- get\_serial\_number, 188
- get\_stage\_information, 189
- get\_stage\_name, 189
- get\_stage\_settings, 189
- get\_status, 189
- get\_status\_calb, 190
- get\_sync\_in\_settings, 190
- get\_sync\_in\_settings\_calb, 191
- get\_sync\_out\_settings, 191
- get\_sync\_out\_settings\_calb, 191
- get\_uart\_settings, 192
- goto\_firmware, 192
- H\_BRIDGE\_ALERT, 140
- has\_firmware, 192
- HOME\_DIR\_FIRST, 140
- HOME\_DIR\_SECOND, 140
- HOME\_HALF\_MV, 140
- HOME\_MV\_SEC\_EN, 140
- HOME\_STOP\_FIRST\_BITS, 141
- HOME\_STOP\_FIRST\_LIM, 141
- HOME\_STOP\_FIRST\_REV, 141
- HOME\_STOP\_FIRST\_SYN, 141
- HOME\_STOP\_SECOND\_BITS, 141
- HOME\_STOP\_SECOND\_LIM, 141
- HOME\_STOP\_SECOND\_REV, 141
- HOME\_STOP\_SECOND\_SYN, 142
- HOME\_USE\_FAST, 142
- JOY\_REVERSE, 142
- logging\_callback\_stderr\_narrow, 193
- logging\_callback\_stderr\_wide, 193
- logging\_callback\_t, 157
- LOW\_UPWR\_PROTECTION, 142
- LS\_SHORTED, 142
- MICROSTEP\_MODE\_FRAC\_128, 142
- MICROSTEP\_MODE\_FRAC\_16, 142
- MICROSTEP\_MODE\_FRAC\_2, 143
- MICROSTEP\_MODE\_FRAC\_256, 143
- MICROSTEP\_MODE\_FRAC\_32, 143
- MICROSTEP\_MODE\_FRAC\_4, 143

- MICROSTEP\_MODE\_FRAC\_64, 143
- MICROSTEP\_MODE\_FRAC\_8, 143
- MICROSTEP\_MODE\_FULL, 143
- MOVE\_STATE\_ANTIPLAY, 144
- MOVE\_STATE\_MOVING, 144
- MOVE\_STATE\_TARGET\_SPEED, 144
- msec\_sleep, 193
- MVCMD\_ERROR, 144
- MVCMD\_HOME, 144
- MVCMD\_LEFT, 144
- MVCMD\_LOFT, 144
- MVCMD\_MOVE, 145
- MVCMD\_MOVR, 145
- MVCMD\_NAME\_BITS, 145
- MVCMD\_RIGHT, 145
- MVCMD\_RUNNING, 145
- MVCMD\_SSTP, 145
- MVCMD\_STOP, 145
- MVCMD\_UKNWN, 146
- open\_device, 193
- POWER\_OFF\_ENABLED, 146
- POWER\_REDUCT\_ENABLED, 146
- POWER\_SMOOTH\_CURRENT, 146
- probe\_device, 194
- PWR\_STATE\_MAX, 146
- PWR\_STATE\_NORM, 146
- PWR\_STATE\_OFF, 146
- PWR\_STATE\_REDUCT, 147
- PWR\_STATE\_UNKNOWN, 147
- reset\_locks, 194
- result\_t, 157
- REV\_SENS\_INV, 147
- RPM\_DIV\_1000, 147
- service\_command\_updf, 194
- set\_accessories\_settings, 194
- set\_brake\_settings, 195
- set\_calibration\_settings, 195
- set\_control\_settings, 195
- set\_control\_settings\_calb, 196
- set\_controller\_name, 196
- set\_correction\_table, 197
- set\_ctp\_settings, 197
- set\_debug\_write, 198
- set\_edges\_settings, 198
- set\_edges\_settings\_calb, 198
- set\_emf\_settings, 199
- set\_encoder\_information, 199
- set\_encoder\_settings, 200
- set\_engine\_advanced\_setup, 200
- set\_engine\_settings, 200
- set\_engine\_settings\_calb, 201
- set\_entype\_settings, 201
- set\_extended\_settings, 201
- set\_extio\_settings, 202
- set\_feedback\_settings, 202
- set\_gear\_information, 202
- set\_gear\_settings, 203
- set\_hallsensor\_information, 203
- set\_hallsensor\_settings, 203
- set\_home\_settings, 204
- set\_home\_settings\_calb, 204
- set\_joystick\_settings, 204
- set\_logging\_callback, 205
- set\_motor\_information, 205
- set\_motor\_settings, 206
- set\_move\_settings, 206
- set\_move\_settings\_calb, 206
- set\_network\_settings, 207
- set\_nonvolatile\_memory, 207
- set\_password\_settings, 207
- set\_pid\_settings, 208
- set\_position, 208
- set\_position\_calb, 208
- set\_power\_settings, 209
- set\_secure\_settings, 209
- set\_serial\_number, 209
- set\_stage\_information, 210
- set\_stage\_name, 210
- set\_stage\_settings, 210
- set\_sync\_in\_settings, 210
- set\_sync\_in\_settings\_calb, 211
- set\_sync\_out\_settings, 211
- set\_sync\_out\_settings\_calb, 212
- set\_uart\_settings, 212
- SETPOS\_IGNORE\_ENCODER, 147
- SETPOS\_IGNORE\_POSITION, 147
- STATE\_ALARM, 147
- STATE\_BORDERS\_SWAP\_MISSET, 148
- STATE\_BRAKE, 148
- STATE\_BUTTON\_LEFT, 148
- STATE\_BUTTON\_RIGHT, 148
- STATE\_CONTR, 148
- STATE\_CONTROLLER\_OVERHEAT, 148
- STATE\_CTP\_ERROR, 148
- STATE\_DIG\_SIGNAL, 149
- STATE\_EEPROM\_CONNECTED, 149
- STATE\_ENC\_A, 149
- STATE\_ENC\_B, 149
- STATE\_ENGINE\_RESPONSE\_ERROR, 149
- STATE\_ERRC, 149
- STATE\_ERRD, 150
- STATE\_ERRV, 150
- STATE\_EXTIO\_ALARM, 150
- STATE\_GPIO\_LEVEL, 150
- STATE\_GPIO\_PINOUT, 150
- STATE\_IS\_HOMED, 150
- STATE\_LEFT\_EDGE, 151
- STATE\_LOW\_USB\_VOLTAGE, 151
- STATE\_OVERLOAD\_POWER\_CURRENT, 151
- STATE\_OVERLOAD\_POWER\_VOLTAGE, 151

STATE\_OVERLOAD\_USB\_CURRENT, [151](#)  
STATE\_OVERLOAD\_USB\_VOLTAGE, [151](#)  
STATE\_POWER\_OVERHEAT, [152](#)  
STATE\_REV\_SENSOR, [152](#)  
STATE\_RIGHT\_EDGE, [152](#)  
STATE\_SECUR, [152](#)  
STATE\_SYNC\_INPUT, [152](#)  
STATE\_SYNC\_OUTPUT, [152](#)  
STATE\_WINDING\_RES\_MISMATCH, [152](#)  
SYNCIN\_ENABLED, [153](#)  
SYNCIN\_INVERT, [153](#)  
SYNCOUT\_ENABLED, [153](#)  
SYNCOUT\_IN\_STEPS, [153](#)  
SYNCOUT\_INVERT, [153](#)  
SYNCOUT\_ONPERIOD, [153](#)  
SYNCOUT\_ONSTART, [154](#)  
SYNCOUT\_ONSTOP, [154](#)  
SYNCOUT\_STATE, [154](#)  
TS\_TYPE\_BITS, [154](#)  
UART\_PARITY\_BITS, [154](#)  
WIND\_A\_STATE\_ABSENT, [154](#)  
WIND\_A\_STATE\_MALFUNC, [154](#)  
WIND\_A\_STATE\_OK, [155](#)  
WIND\_A\_STATE\_UNKNOWN, [155](#)  
WIND\_B\_STATE\_ABSENT, [155](#)  
WIND\_B\_STATE\_MALFUNC, [155](#)  
WIND\_B\_STATE\_OK, [155](#)  
WIND\_B\_STATE\_UNKNOWN, [155](#)  
write\_key, [212](#)  
XIMC\_API, [155](#)  
XIMC\_CALLCONV, [156](#)  
XIMC\_RETTYPE, [156](#)  
ximc\_version, [213](#)  
XIMC\_API  
    ximc.h, [155](#)  
XIMC\_CALLCONV  
    ximc.h, [156](#)  
XIMC\_RETTYPE  
    ximc.h, [156](#)  
ximc\_version  
    ximc.h, [213](#)  
  
Библиотека libximc, [1](#)  
Введение, [3](#)  
Как использовать с..., [4](#)  
Работа с пользовательскими единицами, [8](#)