

libxime
2.9.8

Создано системой Doxygen 1.8.2

Чт 9 Мар 2017 16:14:04

Оглавление

1	Введение	1
1.1	О библиотеке	1
1.2	Требования к установленному программному обеспечению	1
1.2.1	Для сборки библиотеки	1
1.2.2	Для использования библиотеки	2
2	Как пересобрать библиотеку	3
2.1	Сборка для UNIX	3
2.2	Сборка для Linux на основе Debian	3
2.3	Сборка для Linux на основе RedHat	3
2.4	Сборка для Mac OS X	4
2.5	Сборка в ОС Windows	4
2.6	Доступ к исходным кодам	4
3	Как использовать с...	5
3.1	Использование на C	5
3.1.1	Visual C++	5
3.1.2	MinGW	5
3.1.3	C++ Builder	5
3.1.4	XCode	6
3.1.5	GCC	6
3.2	.NET	6
3.3	Delphi	6
3.4	Java	7
3.5	Python	7
3.6	MATLAB	8
3.7	Логирование в файл	8
3.8	Требуемые права доступа	8
4	Структуры данных	9
4.1	Структура accessories_settings_t	9
4.1.1	Подробное описание	9

4.1.2	Поля	10
4.1.2.1	LimitSwitchesSettings	10
4.1.2.2	MagneticBrakeInfo	10
4.1.2.3	MBRatedCurrent	10
4.1.2.4	MBRatedVoltage	10
4.1.2.5	MBSettings	10
4.1.2.6	MBTorque	10
4.1.2.7	TemperatureSensorInfo	10
4.1.2.8	TSGrad	10
4.1.2.9	TSMax	10
4.1.2.10	TSMin	10
4.1.2.11	TSSettings	10
4.2	Структура analog_data_t	11
4.2.1	Подробное описание	12
4.2.2	Поля	12
4.2.2.1	A1Voltage	12
4.2.2.2	A1Voltage_ADC	12
4.2.2.3	A2Voltage	12
4.2.2.4	A2Voltage_ADC	12
4.2.2.5	ACurrent	12
4.2.2.6	ACurrent_ADC	12
4.2.2.7	B1Voltage	13
4.2.2.8	B1Voltage_ADC	13
4.2.2.9	B2Voltage	13
4.2.2.10	B2Voltage_ADC	13
4.2.2.11	BCurrent	13
4.2.2.12	BCurrent_ADC	13
4.2.2.13	FullCurrent	13
4.2.2.14	FullCurrent_ADC	13
4.2.2.15	Joy	13
4.2.2.16	Joy_ADC	13
4.2.2.17	L5_ADC	13
4.2.2.18	Pot	13
4.2.2.19	SupVoltage	14
4.2.2.20	SupVoltage_ADC	14
4.2.2.21	Temp	14
4.2.2.22	Temp_ADC	14
4.3	Структура brake_settings_t	14
4.3.1	Подробное описание	14
4.3.2	Поля	14

4.3.2.1	BrakeFlags	14
4.3.2.2	t1	15
4.3.2.3	t2	15
4.3.2.4	t3	15
4.3.2.5	t4	15
4.4	Структура calibration_settings_t	15
4.4.1	Подробное описание	15
4.4.2	Поля	15
4.4.2.1	CSS1_A	15
4.4.2.2	CSS1_B	16
4.4.2.3	CSS2_A	16
4.4.2.4	CSS2_B	16
4.4.2.5	FullCurrent_A	16
4.4.2.6	FullCurrent_B	16
4.5	Структура calibration_t	16
4.5.1	Подробное описание	16
4.6	Структура chart_data_t	16
4.6.1	Подробное описание	17
4.6.2	Поля	17
4.6.2.1	DutyCycle	17
4.6.2.2	Joy	17
4.6.2.3	Pot	17
4.6.2.4	WindingCurrentA	17
4.6.2.5	WindingCurrentB	17
4.6.2.6	WindingCurrentC	18
4.6.2.7	WindingVoltageA	18
4.6.2.8	WindingVoltageB	18
4.6.2.9	WindingVoltageC	18
4.7	Структура command_add_sync_in_action_calb_t	18
4.7.1	Поля	18
4.7.1.1	Position	18
4.7.1.2	Time	18
4.8	Структура command_add_sync_in_action_t	18
4.8.1	Подробное описание	19
4.8.2	Поля	19
4.8.2.1	Time	19
4.8.2.2	uPosition	19
4.9	Структура command_change_motor_t	19
4.10	Структура control_settings_calb_t	19
4.10.1	Поля	19

4.10.1.1	Flags	19
4.10.1.2	MaxClickTime	19
4.10.1.3	MaxSpeed	20
4.10.1.4	Timeout	20
4.11	Структура control_settings_t	20
4.11.1	Подробное описание	20
4.11.2	Поля	21
4.11.2.1	Flags	21
4.11.2.2	MaxClickTime	21
4.11.2.3	MaxSpeed	21
4.11.2.4	Timeout	21
4.11.2.5	uDeltaPosition	21
4.11.2.6	uMaxSpeed	21
4.12	Структура controller_name_t	21
4.12.1	Подробное описание	21
4.12.2	Поля	22
4.12.2.1	ControllerName	22
4.12.2.2	CtrlFlags	22
4.13	Структура ctp_settings_t	22
4.13.1	Подробное описание	22
4.13.2	Поля	22
4.13.2.1	CTPFlags	22
4.13.2.2	CTPMinError	23
4.14	Структура debug_read_t	23
4.14.1	Подробное описание	23
4.14.2	Поля	23
4.14.2.1	DebugData	23
4.15	Структура debug_write_t	23
4.15.1	Подробное описание	23
4.15.2	Поля	24
4.15.2.1	DebugData	24
4.16	Структура device_information_t	24
4.16.1	Подробное описание	24
4.16.2	Поля	24
4.16.2.1	Major	24
4.16.2.2	Minor	24
4.16.2.3	Release	24
4.17	Структура device_network_information_t	25
4.17.1	Подробное описание	25
4.18	Структура edges_settings_calb_t	25

4.18.1	Поля	25
4.18.1.1	BorderFlags	25
4.18.1.2	EnderFlags	25
4.18.1.3	LeftBorder	25
4.18.1.4	RightBorder	26
4.19	Структура edges_settings_t	26
4.19.1	Подробное описание	26
4.19.2	Поля	26
4.19.2.1	BorderFlags	26
4.19.2.2	EnderFlags	26
4.19.2.3	LeftBorder	26
4.19.2.4	RightBorder	27
4.19.2.5	uLeftBorder	27
4.19.2.6	uRightBorder	27
4.20	Структура encoder_information_t	27
4.20.1	Подробное описание	27
4.20.2	Поля	27
4.20.2.1	Manufacturer	27
4.20.2.2	PartNumber	27
4.21	Структура encoder_settings_t	27
4.21.1	Подробное описание	28
4.21.2	Поля	28
4.21.2.1	EncoderSettings	28
4.21.2.2	MaxCurrentConsumption	28
4.21.2.3	MaxOperatingFrequency	28
4.21.2.4	SupplyVoltageMax	28
4.21.2.5	SupplyVoltageMin	28
4.22	Структура engine_settings_calb_t	29
4.22.1	Поля	29
4.22.1.1	Antiplay	29
4.22.1.2	EngineFlags	29
4.22.1.3	MicrostepMode	29
4.22.1.4	NomCurrent	29
4.22.1.5	NomSpeed	29
4.22.1.6	NomVoltage	30
4.22.1.7	StepsPerRev	30
4.23	Структура engine_settings_t	30
4.23.1	Подробное описание	30
4.23.2	Поля	31
4.23.2.1	Antiplay	31

4.23.2.2	EngineFlags	31
4.23.2.3	MicrostepMode	31
4.23.2.4	NomCurrent	31
4.23.2.5	NomSpeed	31
4.23.2.6	NomVoltage	31
4.23.2.7	StepsPerRev	31
4.23.2.8	uNomSpeed	31
4.24	Структура entype_settings_t	31
4.24.1	Подробное описание	32
4.24.2	Поля	32
4.24.2.1	DriverType	32
4.24.2.2	EngineType	32
4.25	Структура extio_settings_t	32
4.25.1	Подробное описание	32
4.25.2	Поля	33
4.25.2.1	EXTIOModeFlags	33
4.25.2.2	EXTIOSetupFlags	33
4.26	Структура feedback_settings_t	33
4.26.1	Подробное описание	33
4.26.2	Поля	33
4.26.2.1	FeedbackFlags	33
4.26.2.2	FeedbackType	33
4.26.2.3	HallShift	34
4.26.2.4	HallSPR	34
4.27	Структура gear_information_t	34
4.27.1	Подробное описание	34
4.27.2	Поля	34
4.27.2.1	Manufacturer	34
4.27.2.2	PartNumber	34
4.28	Структура gear_settings_t	34
4.28.1	Подробное описание	35
4.28.2	Поля	35
4.28.2.1	Efficiency	35
4.28.2.2	InputInertia	35
4.28.2.3	MaxOutputBacklash	35
4.28.2.4	RatedInputSpeed	35
4.28.2.5	RatedInputTorque	36
4.28.2.6	ReductionIn	36
4.28.2.7	ReductionOut	36
4.29	Структура get_position_calb_t	36

4.29.1 Поля	36
4.29.1.1 EncPosition	36
4.29.1.2 Position	36
4.30 Структура get_position_t	36
4.30.1 Подробное описание	37
4.30.2 Поля	37
4.30.2.1 EncPosition	37
4.30.2.2 uPosition	37
4.31 Структура hallsensor_information_t	37
4.31.1 Подробное описание	37
4.31.2 Поля	37
4.31.2.1 Manufacturer	37
4.31.2.2 PartNumber	37
4.32 Структура hallsensor_settings_t	38
4.32.1 Подробное описание	38
4.32.2 Поля	38
4.32.2.1 MaxCurrentConsumption	38
4.32.2.2 MaxOperatingFrequency	38
4.32.2.3 SupplyVoltageMax	38
4.32.2.4 SupplyVoltageMin	38
4.33 Структура home_settings_calb_t	39
4.33.1 Поля	39
4.33.1.1 FastHome	39
4.33.1.2 HomeDelta	39
4.33.1.3 HomeFlags	39
4.33.1.4 SlowHome	39
4.34 Структура home_settings_t	39
4.34.1 Подробное описание	40
4.34.2 Поля	40
4.34.2.1 FastHome	40
4.34.2.2 HomeDelta	40
4.34.2.3 HomeFlags	40
4.34.2.4 SlowHome	40
4.34.2.5 uFastHome	40
4.34.2.6 uHomeDelta	40
4.34.2.7 uSlowHome	40
4.35 Структура init_random_t	41
4.35.1 Подробное описание	41
4.35.2 Поля	41
4.35.2.1 key	41

4.36 Структура joystick_settings_t	41
4.36.1 Подробное описание	41
4.36.2 Поля	42
4.36.2.1 DeadZone	42
4.36.2.2 ExpFactor	42
4.36.2.3 JoyCenter	42
4.36.2.4 JoyFlags	42
4.36.2.5 JoyHighEnd	42
4.36.2.6 JoyLowEnd	42
4.37 Структура motor_information_t	42
4.37.1 Подробное описание	43
4.37.2 Поля	43
4.37.2.1 Manufacturer	43
4.37.2.2 PartNumber	43
4.38 Структура motor_settings_t	43
4.38.1 Подробное описание	44
4.38.2 Поля	44
4.38.2.1 DetentTorque	44
4.38.2.2 MaxCurrent	45
4.38.2.3 MaxCurrentTime	45
4.38.2.4 MaxSpeed	45
4.38.2.5 MechanicalTimeConstant	45
4.38.2.6 MotorType	45
4.38.2.7 NoLoadCurrent	45
4.38.2.8 NoLoadSpeed	45
4.38.2.9 NominalCurrent	45
4.38.2.10 NominalPower	45
4.38.2.11 NominalSpeed	46
4.38.2.12 NominalTorque	46
4.38.2.13 NominalVoltage	46
4.38.2.14 Phases	46
4.38.2.15 Poles	46
4.38.2.16 RotorInertia	46
4.38.2.17 SpeedConstant	46
4.38.2.18 SpeedTorqueGradient	46
4.38.2.19 StallTorque	46
4.38.2.20 TorqueConstant	46
4.38.2.21 WindingInductance	47
4.38.2.22 WindingResistance	47
4.39 Структура move_settings_calb_t	47

4.39.1 Поля	47
4.39.1.1 Accel	47
4.39.1.2 AntiplaySpeed	47
4.39.1.3 Decel	47
4.39.1.4 Speed	47
4.40 Структура move_settings_t	47
4.40.1 Подробное описание	48
4.40.2 Поля	48
4.40.2.1 Accel	48
4.40.2.2 AntiplaySpeed	48
4.40.2.3 Decel	48
4.40.2.4 Speed	48
4.40.2.5 uAntiplaySpeed	48
4.40.2.6 uSpeed	49
4.41 Структура nonvolatile_memory_t	49
4.41.1 Подробное описание	49
4.41.2 Поля	49
4.41.2.1 UserData	49
4.42 Структура pid_settings_t	49
4.42.1 Подробное описание	49
4.43 Структура power_settings_t	50
4.43.1 Подробное описание	50
4.43.2 Поля	50
4.43.2.1 CurrentSetTime	50
4.43.2.2 CurrReductDelay	50
4.43.2.3 HoldCurrent	50
4.43.2.4 PowerFlags	50
4.43.2.5 PowerOffDelay	51
4.44 Структура secure_settings_t	51
4.44.1 Подробное описание	51
4.44.2 Поля	51
4.44.2.1 CriticalIpwr	51
4.44.2.2 CriticalIusb	51
4.44.2.3 CriticalUpwr	51
4.44.2.4 CriticalUusb	52
4.44.2.5 Flags	52
4.44.2.6 LowUpwrOff	52
4.44.2.7 MinimumUusb	52
4.45 Структура serial_number_t	52
4.45.1 Подробное описание	52

4.45.2 Поля	52
4.45.2.1 Key	52
4.45.2.2 Major	53
4.45.2.3 Minor	53
4.45.2.4 Release	53
4.45.2.5 SN	53
4.46 Структура set_position_calb_t	53
4.46.1 Поля	53
4.46.1.1 EncPosition	53
4.46.1.2 PosFlags	53
4.46.1.3 Position	53
4.47 Структура set_position_t	53
4.47.1 Подробное описание	54
4.47.2 Поля	54
4.47.2.1 EncPosition	54
4.47.2.2 PosFlags	54
4.47.2.3 uPosition	54
4.48 Структура stage_information_t	54
4.48.1 Подробное описание	54
4.48.2 Поля	55
4.48.2.1 Manufacturer	55
4.48.2.2 PartNumber	55
4.49 Структура stage_name_t	55
4.49.1 Подробное описание	55
4.49.2 Поля	55
4.49.2.1 PositionerName	55
4.50 Структура stage_settings_t	55
4.50.1 Подробное описание	56
4.50.2 Поля	56
4.50.2.1 HorizontalLoadCapacity	56
4.50.2.2 LeadScrewPitch	56
4.50.2.3 MaxCurrentConsumption	56
4.50.2.4 MaxSpeed	56
4.50.2.5 SupplyVoltageMax	56
4.50.2.6 SupplyVoltageMin	57
4.50.2.7 TravelRange	57
4.50.2.8 Units	57
4.50.2.9 VerticalLoadCapacity	57
4.51 Структура status_calb_t	57
4.51.1 Поля	58

4.51.1.1	CmdBufFreeSpace	58
4.51.1.2	CurPosition	58
4.51.1.3	CurSpeed	58
4.51.1.4	CurT	58
4.51.1.5	EncPosition	58
4.51.1.6	EncSts	58
4.51.1.7	Flags	58
4.51.1.8	GPIOFlags	58
4.51.1.9	Ipwr	58
4.51.1.10	Iusb	59
4.51.1.11	MoveSts	59
4.51.1.12	MvCmdSts	59
4.51.1.13	PWRSts	59
4.51.1.14	Upwr	59
4.51.1.15	Uusb	59
4.51.1.16	WindSts	59
4.52	Структура status_t	59
4.52.1	Подробное описание	60
4.52.2	Поля	60
4.52.2.1	CmdBufFreeSpace	60
4.52.2.2	CurPosition	60
4.52.2.3	CurSpeed	60
4.52.2.4	CurT	61
4.52.2.5	EncPosition	61
4.52.2.6	EncSts	61
4.52.2.7	Flags	61
4.52.2.8	GPIOFlags	61
4.52.2.9	Ipwr	61
4.52.2.10	Iusb	61
4.52.2.11	MoveSts	61
4.52.2.12	MvCmdSts	61
4.52.2.13	PWRSts	61
4.52.2.14	uCurPosition	61
4.52.2.15	uCurSpeed	61
4.52.2.16	Upwr	62
4.52.2.17	Uusb	62
4.52.2.18	WindSts	62
4.53	Структура sync_in_settings_calb_t	62
4.53.1	Поля	62
4.53.1.1	ClutterTime	62

4.53.1.2	Position	62
4.53.1.3	Speed	62
4.53.1.4	SyncInFlags	62
4.54	Структура sync_in_settings_t	62
4.54.1	Подробное описание	63
4.54.2	Поля	63
4.54.2.1	ClutterTime	63
4.54.2.2	Speed	63
4.54.2.3	SyncInFlags	63
4.54.2.4	uPosition	63
4.54.2.5	uSpeed	63
4.55	Структура sync_out_settings_calb_t	64
4.55.1	Поля	64
4.55.1.1	Accuracy	64
4.55.1.2	SyncOutFlags	64
4.55.1.3	SyncOutPeriod	64
4.55.1.4	SyncOutPulseSteps	64
4.56	Структура sync_out_settings_t	64
4.56.1	Подробное описание	65
4.56.2	Поля	65
4.56.2.1	Accuracy	65
4.56.2.2	SyncOutFlags	65
4.56.2.3	SyncOutPeriod	65
4.56.2.4	SyncOutPulseSteps	65
4.56.2.5	uAccuracy	65
4.57	Структура uart_settings_t	65
4.57.1	Подробное описание	66
4.57.2	Поля	66
4.57.2.1	UARTSetupFlags	66
5	Файлы	67
5.1	Файл ximc.h	67
5.1.1	Подробное описание	90
5.1.2	Макросы	90
5.1.2.1	ALARM_ON_DRIVER_OVERHEATING	90
5.1.2.2	BORDER_IS_ENCODER	90
5.1.2.3	BORDER_STOP_LEFT	90
5.1.2.4	BORDER_STOP_RIGHT	90
5.1.2.5	BORDERS_SWAP_MISSET_DETECTION	91
5.1.2.6	BRAKE_ENABLED	91

5.1.2.7	BRAKE_ENG_PWROFF	91
5.1.2.8	CONTROL_BTN_LEFT_PUSHED_OPEN	91
5.1.2.9	CONTROL_BTN_RIGHT_PUSHED_OPEN	91
5.1.2.10	CONTROL_MODE_BITS	91
5.1.2.11	CONTROL_MODE_JOY	91
5.1.2.12	CONTROL_MODE_LR	91
5.1.2.13	CONTROL_MODE_OFF	91
5.1.2.14	CTP_ALARM_ON_ERROR	91
5.1.2.15	CTP_BASE	91
5.1.2.16	CTP_ENABLED	91
5.1.2.17	CTP_ERROR_CORRECTION	92
5.1.2.18	DRIVER_TYPE_DISCRETE_FET	92
5.1.2.19	DRIVER_TYPE_EXTERNAL	92
5.1.2.20	DRIVER_TYPE_INTEGRATE	92
5.1.2.21	EEPROM_PRECEDENCE	92
5.1.2.22	ENC_STATE_ABSENT	92
5.1.2.23	ENC_STATE_MALFUNC	92
5.1.2.24	ENC_STATE_OK	92
5.1.2.25	ENC_STATE_REVERS	92
5.1.2.26	ENC_STATE_UNKNOWN	92
5.1.2.27	ENDER_SW1_ACTIVE_LOW	92
5.1.2.28	ENDER_SW2_ACTIVE_LOW	92
5.1.2.29	ENDER_SWAP	93
5.1.2.30	ENGINE_ACCEL_ON	93
5.1.2.31	ENGINE_ANTIPLAY	93
5.1.2.32	ENGINE_CURRENT_AS_RMS	93
5.1.2.33	ENGINE_LIMIT_CURR	93
5.1.2.34	ENGINE_LIMIT_RPM	93
5.1.2.35	ENGINE_LIMIT_VOLT	93
5.1.2.36	ENGINE_MAX_SPEED	93
5.1.2.37	ENGINE_REVERSE	93
5.1.2.38	ENGINE_TYPE_2DC	94
5.1.2.39	ENGINE_TYPE_BRUSHLESS	94
5.1.2.40	ENGINE_TYPE_DC	94
5.1.2.41	ENGINE_TYPE_NONE	94
5.1.2.42	ENGINE_TYPE_STEP	94
5.1.2.43	ENGINE_TYPE_TEST	94
5.1.2.44	ENUMERATE_PROBE	94
5.1.2.45	EXTIO_SETUP_INVERT	94
5.1.2.46	EXTIO_SETUP_MODE_IN_ALARM	94

5.1.2.47	EXTIO_SETUP_MODE_IN_BITS	94
5.1.2.48	EXTIO_SETUP_MODE_IN_HOME	95
5.1.2.49	EXTIO_SETUP_MODE_IN_MOVR	95
5.1.2.50	EXTIO_SETUP_MODE_IN_NOP	95
5.1.2.51	EXTIO_SETUP_MODE_IN_PWOF	95
5.1.2.52	EXTIO_SETUP_MODE_IN_STOP	95
5.1.2.53	EXTIO_SETUP_MODE_OUT_ALARM	95
5.1.2.54	EXTIO_SETUP_MODE_OUT_BITS	95
5.1.2.55	EXTIO_SETUP_MODE_OUT_MOTOR_FOUND	95
5.1.2.56	EXTIO_SETUP_MODE_OUT_MOTOR_ON	95
5.1.2.57	EXTIO_SETUP_MODE_OUT_MOVING	95
5.1.2.58	EXTIO_SETUP_MODE_OUT_OFF	95
5.1.2.59	EXTIO_SETUP_MODE_OUT_ON	95
5.1.2.60	EXTIO_SETUP_OUTPUT	96
5.1.2.61	FEEDBACK_EMF	96
5.1.2.62	FEEDBACK_ENC_REVERSE	96
5.1.2.63	FEEDBACK_ENC_TYPE_AUTO	96
5.1.2.64	FEEDBACK_ENC_TYPE_BITS	96
5.1.2.65	FEEDBACK_ENC_TYPE_DIFFERENTIAL	96
5.1.2.66	FEEDBACK_ENC_TYPE_SINGLE_ENDED	96
5.1.2.67	FEEDBACK_ENCODER	96
5.1.2.68	FEEDBACK_ENCODERHALL	96
5.1.2.69	FEEDBACK_HALL_REVERSE	96
5.1.2.70	FEEDBACK_NONE	96
5.1.2.71	HOME_DIR_FIRST	96
5.1.2.72	HOME_DIR_SECOND	97
5.1.2.73	HOME_HALF_MV	97
5.1.2.74	HOME_MV_SEC_EN	97
5.1.2.75	HOME_STOP_FIRST_BITS	97
5.1.2.76	HOME_STOP_FIRST_LIM	97
5.1.2.77	HOME_STOP_FIRST_REV	97
5.1.2.78	HOME_STOP_FIRST_SYN	97
5.1.2.79	HOME_STOP_SECOND_BITS	97
5.1.2.80	HOME_STOP_SECOND_LIM	97
5.1.2.81	HOME_STOP_SECOND_REV	97
5.1.2.82	HOME_STOP_SECOND_SYN	97
5.1.2.83	HOME_USE_FAST	97
5.1.2.84	JOY_REVERSE	98
5.1.2.85	LOW_UPWR_PROTECTION	98
5.1.2.86	LS_SHORTED	98

5.1.2.87 MICROSTEP_MODE_FRAC_128	98
5.1.2.88 MICROSTEP_MODE_FRAC_16	98
5.1.2.89 MICROSTEP_MODE_FRAC_2	98
5.1.2.90 MICROSTEP_MODE_FRAC_256	98
5.1.2.91 MICROSTEP_MODE_FRAC_32	98
5.1.2.92 MICROSTEP_MODE_FRAC_4	98
5.1.2.93 MICROSTEP_MODE_FRAC_64	98
5.1.2.94 MICROSTEP_MODE_FRAC_8	98
5.1.2.95 MICROSTEP_MODE_FULL	98
5.1.2.96 MOVE_STATE_ANTIPLAY	99
5.1.2.97 MOVE_STATE_MOVING	99
5.1.2.98 MOVE_STATE_TARGET_SPEED	99
5.1.2.99 MVCMD_ERROR	99
5.1.2.100 MVCMD_HOME	99
5.1.2.101 MVCMD_LEFT	99
5.1.2.102 MVCMD_LOFT	99
5.1.2.103 MVCMD_MOVE	99
5.1.2.104 MVCMD_MOVR	99
5.1.2.105 MVCMD_NAME_BITS	99
5.1.2.106 MVCMD_RIGHT	99
5.1.2.107 MVCMD_RUNNING	100
5.1.2.108 MVCMD_SSTP	100
5.1.2.109 MVCMD_STOP	100
5.1.2.110 MVCMD_UKNWN	100
5.1.2.111 POWER_OFF_ENABLED	100
5.1.2.112 POWER_REDUCT_ENABLED	100
5.1.2.113 POWER_SMOOTH_CURRENT	100
5.1.2.114 PWR_STATE_MAX	100
5.1.2.115 PWR_STATE_NORM	100
5.1.2.116 PWR_STATE_OFF	100
5.1.2.117 PWR_STATE_REDUCT	100
5.1.2.118 PWR_STATE_UNKNOWN	101
5.1.2.119 REV_SENS_INV	101
5.1.2.120 SETPOS_IGNORE_ENCODER	101
5.1.2.121 SETPOS_IGNORE_POSITION	101
5.1.2.122 STATE_ALARM	101
5.1.2.123 STATE_BORDERS_SWAP_MISSET	101
5.1.2.124 STATE_BRAKE	101
5.1.2.125 STATE_BUTTON_LEFT	101
5.1.2.126 STATE_BUTTON_RIGHT	101

5.1.2.127 STATE_CONTR	101
5.1.2.128 STATE_CONTROLLER_OVERHEAT	101
5.1.2.129 STATE_CTP_ERROR	102
5.1.2.130 STATE_CURRENT_MOTOR0	102
5.1.2.131 STATE_CURRENT_MOTOR1	102
5.1.2.132 STATE_CURRENT_MOTOR2	102
5.1.2.133 STATE_CURRENT_MOTOR3	102
5.1.2.134 STATE_CURRENT_MOTOR_BITS	102
5.1.2.135 STATE_DIG_SIGNAL	102
5.1.2.136 STATE_EEPROM_CONNECTED	102
5.1.2.137 STATE_ENC_A	102
5.1.2.138 STATE_ENC_B	102
5.1.2.139 STATE_ERRC	102
5.1.2.140 STATE_ERRD	102
5.1.2.141 STATE_ERRV	102
5.1.2.142 STATE_GPIO_LEVEL	103
5.1.2.143 STATE_GPIO_PINOUT	103
5.1.2.144 STATE_HALL_A	103
5.1.2.145 STATE_HALL_B	103
5.1.2.146 STATE_HALL_C	103
5.1.2.147 STATE_LEFT_EDGE	103
5.1.2.148 STATE_LOW_USB_VOLTAGE	103
5.1.2.149 STATE_OVERLOAD_POWER_CURRENT	103
5.1.2.150 STATE_OVERLOAD_POWER_VOLTAGE	103
5.1.2.151 STATE_OVERLOAD_USB_CURRENT	103
5.1.2.152 STATE_OVERLOAD_USB_VOLTAGE	103
5.1.2.153 STATE_POWER_OVERHEAT	103
5.1.2.154 STATE_REV_SENSOR	104
5.1.2.155 STATE_RIGHT_EDGE	104
5.1.2.156 STATE_SECUR	104
5.1.2.157 STATE_SYNC_INPUT	104
5.1.2.158 STATE_SYNC_OUTPUT	104
5.1.2.159 SYNCIN_ENABLED	104
5.1.2.160 SYNCIN_INVERT	104
5.1.2.161 SYNCOUT_ENABLED	104
5.1.2.162 SYNCOUT_IN_STEPS	104
5.1.2.163 SYNCOUT_INVERT	104
5.1.2.164 SYNCOUT_ONPERIOD	104
5.1.2.165 SYNCOUT_ONSTART	104
5.1.2.166 SYNCOUT_ONSTOP	105

5.1.2.167	SYNCOUT_STATE	105
5.1.2.168	TS_TYPE_BITS	105
5.1.2.169	UART_PARITY_BITS	105
5.1.2.170	WIND_A_STATE_ABSENT	105
5.1.2.171	WIND_A_STATE_MALFUNC	105
5.1.2.172	WIND_A_STATE_OK	105
5.1.2.173	WIND_A_STATE_UNKNOWN	105
5.1.2.174	WIND_B_STATE_ABSENT	105
5.1.2.175	WIND_B_STATE_MALFUNC	105
5.1.2.176	WIND_B_STATE_OK	105
5.1.2.177	WIND_B_STATE_UNKNOWN	105
5.1.2.178	XIMC_API	106
5.1.3	Типы	106
5.1.3.1	logging_callback_t	106
5.1.4	Функции	106
5.1.4.1	close_device	106
5.1.4.2	command_add_sync_in_action	106
5.1.4.3	command_clear_fram	106
5.1.4.4	command_eeread_settings	107
5.1.4.5	command_eesave_settings	107
5.1.4.6	command_home	107
5.1.4.7	command_homezero	107
5.1.4.8	command_left	108
5.1.4.9	command_loft	108
5.1.4.10	command_move	108
5.1.4.11	command_movr	108
5.1.4.12	command_power_off	109
5.1.4.13	command_read_robust_settings	109
5.1.4.14	command_read_settings	109
5.1.4.15	command_reset	109
5.1.4.16	command_right	109
5.1.4.17	command_save_robust_settings	110
5.1.4.18	command_save_settings	110
5.1.4.19	command_sstp	110
5.1.4.20	command_stop	110
5.1.4.21	command_update_firmware	110
5.1.4.22	command_wait_for_stop	111
5.1.4.23	command_zero	111
5.1.4.24	enumerate_devices	111
5.1.4.25	free_enumerate_devices	111

5.1.4.26	<code>get_accessories_settings</code>	112
5.1.4.27	<code>get_analog_data</code>	112
5.1.4.28	<code>get_bootloader_version</code>	112
5.1.4.29	<code>get_brake_settings</code>	112
5.1.4.30	<code>get_calibration_settings</code>	112
5.1.4.31	<code>get_chart_data</code>	113
5.1.4.32	<code>get_control_settings</code>	113
5.1.4.33	<code>get_controller_name</code>	113
5.1.4.34	<code>get_ctp_settings</code>	114
5.1.4.35	<code>get_debug_read</code>	114
5.1.4.36	<code>get_device_count</code>	114
5.1.4.37	<code>get_device_information</code>	114
5.1.4.38	<code>get_device_name</code>	115
5.1.4.39	<code>get_edges_settings</code>	115
5.1.4.40	<code>get_encoder_information</code>	115
5.1.4.41	<code>get_encoder_settings</code>	115
5.1.4.42	<code>get_engine_settings</code>	115
5.1.4.43	<code>get_entype_settings</code>	116
5.1.4.44	<code>get_enumerate_device_controller_name</code>	116
5.1.4.45	<code>get_enumerate_device_information</code>	116
5.1.4.46	<code>get_enumerate_device_network_information</code>	117
5.1.4.47	<code>get_enumerate_device_serial</code>	117
5.1.4.48	<code>get_enumerate_device_stage_name</code>	117
5.1.4.49	<code>get_extio_settings</code>	117
5.1.4.50	<code>get_feedback_settings</code>	118
5.1.4.51	<code>get_firmware_version</code>	118
5.1.4.52	<code>get_gear_information</code>	118
5.1.4.53	<code>get_gear_settings</code>	118
5.1.4.54	<code>get_hallsensor_information</code>	118
5.1.4.55	<code>get_hallsensor_settings</code>	119
5.1.4.56	<code>get_home_settings</code>	119
5.1.4.57	<code>get_init_random</code>	119
5.1.4.58	<code>get_joystick_settings</code>	119
5.1.4.59	<code>get_motor_information</code>	120
5.1.4.60	<code>get_motor_settings</code>	120
5.1.4.61	<code>get_move_settings</code>	120
5.1.4.62	<code>get_nonvolatile_memory</code>	120
5.1.4.63	<code>get_pid_settings</code>	121
5.1.4.64	<code>get_position</code>	121
5.1.4.65	<code>get_power_settings</code>	121

5.1.4.66	<code>get_secure_settings</code>	121
5.1.4.67	<code>get_serial_number</code>	122
5.1.4.68	<code>get_stage_information</code>	122
5.1.4.69	<code>get_stage_name</code>	122
5.1.4.70	<code>get_stage_settings</code>	122
5.1.4.71	<code>get_status</code>	122
5.1.4.72	<code>get_status_calb</code>	123
5.1.4.73	<code>get_sync_in_settings</code>	123
5.1.4.74	<code>get_sync_out_settings</code>	123
5.1.4.75	<code>get_uart_settings</code>	123
5.1.4.76	<code>goto_firmware</code>	124
5.1.4.77	<code>has_firmware</code>	124
5.1.4.78	<code>logging_callback_stderr_narrow</code>	124
5.1.4.79	<code>logging_callback_stderr_wide</code>	124
5.1.4.80	<code>msec_sleep</code>	124
5.1.4.81	<code>open_device</code>	125
5.1.4.82	<code>probe_device</code>	125
5.1.4.83	<code>service_command_updf</code>	125
5.1.4.84	<code>set_accessories_settings</code>	125
5.1.4.85	<code>set_bindy_key</code>	125
5.1.4.86	<code>set_brake_settings</code>	126
5.1.4.87	<code>set_calibration_settings</code>	126
5.1.4.88	<code>set_control_settings</code>	126
5.1.4.89	<code>set_controller_name</code>	127
5.1.4.90	<code>set_ctp_settings</code>	127
5.1.4.91	<code>set_debug_write</code>	127
5.1.4.92	<code>set_edges_settings</code>	127
5.1.4.93	<code>set_encoder_information</code>	128
5.1.4.94	<code>set_encoder_settings</code>	128
5.1.4.95	<code>set_engine_settings</code>	128
5.1.4.96	<code>set_entype_settings</code>	128
5.1.4.97	<code>set_extio_settings</code>	129
5.1.4.98	<code>set_feedback_settings</code>	129
5.1.4.99	<code>set_gear_information</code>	129
5.1.4.100	<code>set_gear_settings</code>	129
5.1.4.101	<code>set_hallsensor_information</code>	130
5.1.4.102	<code>set_hallsensor_settings</code>	130
5.1.4.103	<code>set_home_settings</code>	130
5.1.4.104	<code>set_joystick_settings</code>	130
5.1.4.105	<code>set_logging_callback</code>	131

5.1.4.106 set_motor_information	131
5.1.4.107 set_motor_settings	131
5.1.4.108 set_move_settings	131
5.1.4.109 set_nonvolatile_memory	132
5.1.4.110 set_pid_settings	132
5.1.4.111 set_position	132
5.1.4.112 set_power_settings	132
5.1.4.113 set_secure_settings	133
5.1.4.114 set_serial_number	133
5.1.4.115 set_stage_information	133
5.1.4.116 set_stage_name	133
5.1.4.117 set_stage_settings	134
5.1.4.118 set_sync_in_settings	134
5.1.4.119 set_sync_out_settings	134
5.1.4.120 set_uart_settings	134
5.1.4.121 write_key	135
5.1.4.122 ximc_fix_usbser_sys	135
5.1.4.123 ximc_version	135

Алфавитный указатель	135
--------------------------------	-----

Глава 1

Введение

1.1 О библиотеке

Спасибо, что вы выбрали мультиплатформенную библиотеку XIMC! Этот документ содержит всю необходимую информацию о библиотеке XIMC. Она использует распространенный и проверенный интерфейс виртуального последовательного порта, поэтому вы можете работать с модулями управления моторами через эту библиотеку практически под всеми под ОС, в том числе Windows 7, Windows Vista, Windows XP, Windows Server 2003, Windows 2000, Linux, Mac OS X. Библиотека XIMC поддерживает подключение и отключение устройств "на лету". 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.

1.2 Требования к установленному программному обеспечению

1.2.1 Для сборки библиотеки

Для Windows:

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

Для Linux:

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

- mercurial (для сборки версии для разработки из hg)

Для Mac OS X:

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

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

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

1.2.2 Для использования библиотеки

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

- Mac OS X 10.6
- Windows 2000 или старше
- Autotools-совместимый unix. Библиотека устанавливается из бинарного вида.
- Linux на основе debian 32 и 64 бита. DEB собирается на Debian Squeeze 7
- Linux на основе debian ARM. DEB собирается кросс-компилятором на Ubuntu 14.04
- Linux на основе rpm. RPM собирается на OpenSUSE 12
- Java 7 64 бит или (только Linux) 32 бит
- .NET 2.0 (только 32 бит)
- Delphi (только 32 бит)

Требования сборки:

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

Глава 2

Как пересобрать библиотеку

2.1 Сборка для UNIX

Обобщенная версия собирается обычными autotools.

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

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

2.2 Сборка для Linux на основе Debian

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

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

Для сборки библиотеки и пакета запустите скрипт:

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

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

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

2.3 Сборка для Linux на основе RedHat

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

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

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

Для сборки библиотеки и пакета запустите скрипт:

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

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

2.4 Сборка для Mac OS X

Для сборки библиотеки и пакета запустите скрипт:

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

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

2.5 Сборка в ОС Windows

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

Запустите скрипт:

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

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

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

2.6 Доступ к исходным кодам

Исходные коды XIMC могут быть выданы по отдельному запросу.

Глава 3

Как использовать с...

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

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

3.1 Использование на C

3.1.1 Visual C++

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

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

3.1.2 MinGW

MinGW это вариант GCC для платформы win32. Требует установки пакета MinGW. В данный момент не поддерживается.

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

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

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

3.1.3 C++ Builder

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

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

Затем скомпилируйте тестовое приложение:

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

3.1.4 XCode

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

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

3.1.5 GCC

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

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

testapp может быть собран следующим образом с установленной библиотекой:

```
$ make
```

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

Затем запустите приложение с помощью:

```
$ make run
```

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

3.2 .NET

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

Тестовые приложения на языке C# для Visual Studio 2013 расположены в директориях testcs (для C#) и testvbnet (для VB.NET). Скопируйте ximcnet.dll, libximc.dll, xiwrapper.dll, bindy.dll из ximc-2.x.x/ximc/win64 или ximc-2.x.x/ximc/win32 соответственно вашей системе в текущую директорию. Откройте проекты, скомпилируйте и запустите.

3.3 Delphi

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

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

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

3.4 Java

Как запустить пример на Linux. Перейдите to `ximc-2.x.x/examples/testjava/compiled/` и запустите;

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

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

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

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

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

Затем пересоберите исходные тексты:

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

или для Windows или Mac:

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

Затем соберите jar:

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

3.5 Python

1. Измените текущую директорию на `examples/testpython`.

1. На OS X скопируйте библиотеку `ximc/macosx/libximc.framework` в текущую директорию. На Windows скопируйте библиотеки из `ximc/win32` или `ximc/win64` в текущую директорию. Замечание: архитектура используемой среды Python (32 или 64 бита) должна совпадать с ОС Windows и библиотекой `libximc`.

1. Скопируйте содержимое `ximc/crossplatform/wrappers/python/` в текущую директорию или установите переменную окружения `PYTHONPATH`:

```
export PYTHONPATH=$PYTHONPATH:$(pwd)/../../ximc/crossplatform/wrappers/python
```

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

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$(pwd)
```

1. Затем запустите Python 2 или Python 3:

```
python testpython.py
```

3.6 MATLAB

Тестовая программа на MATLAB `testximc.m` располагается в директории `examples/testmatlab`. Измените текущую директорию на `examples/matlab`.

На windows скопируйте [ximc.h](#), `libximc.dll`, `bindy.dll`, `xiwrapper.dll`, а также содержимое директории `ximc/(win32,win64)/wrappers/matlab/`, в текущую директорию.

На Mac установите "Command Line Tools for XCode" от Apple и затем скопируйте `ximc/macosx/libximc.framework` и содержимое `ximc/macosx/wrappers/matlab` в текущую директорию.

Затем запустите в MATLAB:

```
testximc
```

3.7 Логирование в файл

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

3.8 Требуемые права доступа

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

```
vim: ft=doxygen
```

Глава 4

Структуры данных

4.1 Структура accessories_settings_t

Информация о дополнительных аксессуарах.

Поля данных

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

4.1.1 Подробное описание

Информация о дополнительных аксессуарах.

См. также

```
set_accessories_settings  
get_accessories_settings  
get_accessories_settings, set_accessories_settings
```

4.1.2 Поля

4.1.2.1 unsigned int LimitSwitchesSettings

Флаги настроек температурного датчика.

4.1.2.2 char MagneticBrakeInfo[25]

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

4.1.2.3 float MBRatedCurrent

Номинальный ток для управления магнитным тормозом (А).

Тип данных: float.

4.1.2.4 float MBRatedVoltage

Номинальное напряжение для управления магнитным тормозом (В).

Тип данных: float.

4.1.2.5 unsigned int MBSettings

Флаги настроек энкодера.

4.1.2.6 float MBTorque

Удерживающий момент (мН м).

Тип данных: float.

4.1.2.7 char TemperatureSensorInfo[25]

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

4.1.2.8 float TSGrad

Температурный градиент (В/град Цельсия).

Тип данных: float.

4.1.2.9 float TSMax

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

4.1.2.10 float TSMin

Минимальная измеряемая температура (град Цельсия).

Тип данных: float.

4.1.2.11 unsigned int TSSettings

Флаги настроек температурного датчика.

4.2 Структура analog_data_t

Аналоговые данные.

Поля данных

- unsigned int [A1Voltage_ADC](#)
"Выходное напряжение на 1 выводе обмотки A" необработанные данные с АЦП.
- unsigned int [A2Voltage_ADC](#)
"Выходное напряжение на 2 выводе обмотки A" необработанные данные с АЦП.
- unsigned int [B1Voltage_ADC](#)
"Выходное напряжение на 1 выводе обмотки B" необработанные данные с АЦП.
- unsigned int [B2Voltage_ADC](#)
"Выходное напряжение на 2 выводе обмотки B" необработанные данные с АЦП.
- unsigned int [SupVoltage_ADC](#)
"Напряжение питания ключей H-моста" необработанные данные с АЦП.
- unsigned int [ACurrent_ADC](#)
"Ток через обмотку A" необработанные данные с АЦП.
- unsigned int [BCurrent_ADC](#)
"Ток через обмотку B" необработанные данные с АЦП.
- unsigned int [FullCurrent_ADC](#)
"Полный ток" необработанные данные с АЦП.
- unsigned int [Temp_ADC](#)
Напряжение с датчика температуры, необработанные данные с АЦП.
- unsigned int [Joy_ADC](#)
Джойстик, необработанные данные с АЦП.
- unsigned int [Pot_ADC](#)
Напряжение на аналоговом входе, необработанные данные с АЦП
- unsigned int [L5_ADC](#)
Напряжение питания USB после current sense резистора, необработанные данные с АЦП.
- unsigned int [H5_ADC](#)
Напряжение питания USB, необработанные данные с АЦП
- int [A1Voltage](#)
"Выходное напряжение на 1 выводе обмотки A" откалиброванные данные.
- int [A2Voltage](#)
"Выходное напряжение на 2 выводе обмотки A" откалиброванные данные.
- int [B1Voltage](#)
"Выходное напряжение на 1 выводе обмотки B" откалиброванные данные.
- int [B2Voltage](#)
"Выходное напряжение на 2 выводе обмотки B" откалиброванные данные.
- int [SupVoltage](#)
"Напряжение питания ключей H-моста" откалиброванные данные.
- int [ACurrent](#)
"Ток через обмотку A" откалиброванные данные.
- int [BCurrent](#)
"Ток через обмотку B" откалиброванные данные.
- int [FullCurrent](#)
"Полный ток" откалиброванные данные.
- int [Temp](#)
Температура, откалиброванные данные.
- int [Joy](#)

- Джойстик во внутренних единицах.
- int [Pot](#)
Аналоговый вход во внутренних единицах.
- int [L5](#)
Напряжение питания USB после current sense резистора
- int [H5](#)
Напряжение питания USB.
- unsigned int deprecated
- int [R](#)
Сопротивление обмоток двигателя(для шагового двигателя), в мОм
- int [L](#)
Псевдоиндуктивность обмоток двигателя(для шагового двигателя), в мкГн

4.2.1 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](#)

4.2.2 Поля

4.2.2.1 int A1Voltage

"Выходное напряжение на 1 выводе обмотки A" откалиброванные данные.

4.2.2.2 unsigned int A1Voltage_ADC

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

4.2.2.3 int A2Voltage

"Выходное напряжение на 2 выводе обмотки A" откалиброванные данные.

4.2.2.4 unsigned int A2Voltage_ADC

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

4.2.2.5 int ACurrent

"Ток через обмотку A" откалиброванные данные.

4.2.2.6 unsigned int ACurrent_ADC

"Ток через обмотку A" необработанные данные с АЦП.

4.2.2.7 int B1Voltage

"Выходное напряжение на 1 выводе обмотки В" откалиброванные данные.

4.2.2.8 unsigned int B1Voltage_ADC

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

4.2.2.9 int B2Voltage

"Выходное напряжение на 2 выводе обмотки В" откалиброванные данные.

4.2.2.10 unsigned int B2Voltage_ADC

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

4.2.2.11 int BCurrent

"Ток через обмотку В" откалиброванные данные.

4.2.2.12 unsigned int BCurrent_ADC

"Ток через обмотку В" необработанные данные с АЦП.

4.2.2.13 int FullCurrent

"Полный ток" откалиброванные данные.

4.2.2.14 unsigned int FullCurrent_ADC

"Полный ток" необработанные данные с АЦП.

4.2.2.15 int Joy

Джойстик во внутренних единицах.

Диапазон: 0..10000

4.2.2.16 unsigned int Joy_ADC

Джойстик, необработанные данные с АЦП.

4.2.2.17 unsigned int L5_ADC

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

4.2.2.18 int Pot

Аналоговый вход во внутренних единицах.

Диапазон: 0..10000

4.2.2.19 `int SupVoltage`

"Напряжение питания ключей H-моста" откалиброванные данные.

4.2.2.20 `unsigned int SupVoltage_ADC`

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

4.2.2.21 `int Temp`

Температура, откалиброванные данные.

4.2.2.22 `unsigned int Temp_ADC`

Напряжение с датчика температуры, необработанные данные с АЦП.

4.3 Структура `brake_settings_t`

Настройки тормоза.

Поля данных

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

4.3.1 Подробное описание

Настройки тормоза.

Эта структура содержит параметры управления тормозом.

См. также

```
set_brake_settings  
get_brake_settings  
get_brake_settings, set_brake_settings
```

4.3.2 Поля

4.3.2.1 `unsigned int BrakeFlags`

Флаги настроек тормоза.

4.3.2.2 `unsigned int t1`

Время в мс между включением питания мотора и отключением тормоза.

4.3.2.3 `unsigned int t2`

Время в мс между отключением тормоза и готовностью к движению.

Все команды движения начинают выполняться только по истечении этого времени.

4.3.2.4 `unsigned int t3`

Время в мс между остановкой мотора и включением тормоза.

4.3.2.5 `unsigned int t4`

Время в мс между включением тормоза и отключением питания мотора.

4.4 Структура `calibration_settings_t`

Калибровочные коэффициенты.

Поля данных

- `float CSS1_A`
Коэффициент масштабирования для аналоговых измерений тока в обмотке А.
- `float CSS1_B`
Коэффициент сдвига для аналоговых измерений тока в обмотке А.
- `float CSS2_A`
Коэффициент масштабирования для аналоговых измерений тока в обмотке В.
- `float CSS2_B`
Коэффициент сдвига для аналоговых измерений тока в обмотке В.
- `float FullCurrent_A`
Коэффициент масштабирования для аналоговых измерений полного тока.
- `float FullCurrent_B`
Коэффициент сдвига для аналоговых измерений полного тока.

4.4.1 Подробное описание

Калибровочные коэффициенты.

Эта структура содержит калибровочные коэффициенты.

См. также

```
get_calibration_settings  
set_calibration_settings  
get_calibration_settings, set_calibration_settings
```

4.4.2 Поля

4.4.2.1 `float CSS1_A`

Коэффициент масштабирования для аналоговых измерений тока в обмотке А.

4.4.2.2 float CSS1_B

Коэффициент сдвига для аналоговых измерений тока в обмотке A.

4.4.2.3 float CSS2_A

Коэффициент масштабирования для аналоговых измерений тока в обмотке B.

4.4.2.4 float CSS2_B

Коэффициент сдвига для аналоговых измерений тока в обмотке B.

4.4.2.5 float FullCurrent_A

Коэффициент масштабирования для аналоговых измерений полного тока.

4.4.2.6 float FullCurrent_B

Коэффициент сдвига для аналоговых измерений полного тока.

4.5 Структура calibration_t

Структура калибровок

Поля данных

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

4.5.1 Подробное описание

Структура калибровок

4.6 Структура chart_data_t

Дополнительное состояние устройства.

Поля данных

- int [WindingVoltageA](#)
В случае ШД, напряжение на обмотке A; в случае бесщеточного, напряжение на первой обмотке; в случае DC на единственной.
- int [WindingVoltageB](#)
В случае ШД, напряжение на обмотке B; в случае бесщеточного, напряжение на второй обмотке; в случае DC не используется.
- int [WindingVoltageC](#)
В случае бесщеточного, напряжение на третьей обмотке; в случае ШД и DC не используется.

- `int WindingCurrentA`
В случае ШД, ток в обмотке А; в случае бесщеточного, ток в первой обмотке; в случае DC в единственной.
- `int WindingCurrentB`
В случае ШД, ток в обмотке В; в случае бесщеточного, ток в второй обмотке; в случае DC не используется.
- `int WindingCurrentC`
В случае бесщеточного, ток в третьей обмотке; в случае ШД и DC не используется.
- `unsigned int Pot`
Значение на аналоговом входе.
- `unsigned int Joy`
Положение джойстика в десяти тысячных долях.
- `int DutyCycle`
Коэффициент заполнения ШИМ.

4.6.1 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](#)

4.6.2 Поля

4.6.2.1 `int DutyCycle`

Коэффициент заполнения ШИМ.

4.6.2.2 `unsigned int Joy`

Положение джойстика в десяти тысячных долях.

Диапазон: 0..10000

4.6.2.3 `unsigned int Pot`

Значение на аналоговом входе.

Диапазон: 0..10000

4.6.2.4 `int WindingCurrentA`

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

4.6.2.5 `int WindingCurrentB`

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

4.6.2.6 `int WindingCurrentC`

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

4.6.2.7 `int WindingVoltageA`

В случае ШД, напряжение на обмотке A; в случае бесщеточного, напряжение на первой обмотке; в случае DC на единственной.

4.6.2.8 `int WindingVoltageB`

В случае ШД, напряжение на обмотке B; в случае бесщеточного, напряжение на второй обмотке; в случае DC не используется.

4.6.2.9 `int WindingVoltageC`

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

4.7 Структура `command_add_sync_in_action_calb_t`

Поля данных

- float [Position](#)
Желаемая позиция или смещение.
- unsigned int [Time](#)
Время, за которое требуется достичь требуемой позиции, в микросекундах.

4.7.1 Поля

4.7.1.1 `float Position`

Желаемая позиция или смещение.

4.7.1.2 `unsigned int Time`

Время, за которое требуется достичь требуемой позиции, в микросекундах.

4.8 Структура `command_add_sync_in_action_t`

Это команда добавляет один элемент в буфер FIFO команд.

Поля данных

- int [Position](#)
Желаемая позиция или смещение (целая часть)
- int [uPosition](#)
Дробная часть позиции или смещения в микрошагах.
- unsigned int [Time](#)
Время, за которое требуется достичь требуемой позиции, в микросекундах.

4.8.1 Подробное описание

Это команда добавляет один элемент в буфер FIFO команд.

См. также

[command_add_sync_in_action](#)

4.8.2 Поля

4.8.2.1 `unsigned int Time`

Время, за которое требуется достичь требуемой позиции, в микросекундах.

4.8.2.2 `int uPosition`

Дробная часть позиции или смещения в микрошагах.

Используется только с шаговым двигателем. Диапазон: -255..255.

4.9 Структура `command_change_motor_t`

Поля данных

- `unsigned int` [Motor](#)
Номер мотора, на который следует переключить реле [0..1].

4.10 Структура `control_settings_calb_t`

Поля данных

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

4.10.1 Поля

4.10.1.1 `unsigned int Flags`

[Флаги управления.](#)

4.10.1.2 `unsigned int MaxClickTime`

Максимальное время клика.

До истечения этого времени первая скорость не включается.

4.10.1.3 float MaxSpeed[10]

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

4.10.1.4 unsigned int Timeout[9]

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

4.11 Структура control_settings_t

Настройки управления.

Поля данных

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

4.11.1 Подробное описание

Настройки управления.

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

См. также

```
set_control_settings  
get_control_settings  
get_control_settings, set_control_settings
```

4.11.2 Поля

4.11.2.1 `unsigned int Flags`

[Флаги управления.](#)

4.11.2.2 `unsigned int MaxClickTime`

Максимальное время клика.

До истечения этого времени первая скорость не включается.

4.11.2.3 `unsigned int MaxSpeed[10]`

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

Диапазон: 0..100000.

4.11.2.4 `unsigned int Timeout[9]`

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

4.11.2.5 `int uDeltaPosition`

Дробная часть смещения в микрошагах.

Используется только с шаговым двигателем. Диапазон: -255..255.

4.11.2.6 `unsigned int uMaxSpeed[10]`

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

4.12 Структура `controller_name_t`

Пользовательское имя контроллера и флаги настройки.

Поля данных

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

4.12.1 Подробное описание

Пользовательское имя контроллера и флаги настройки.

См. также

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

4.12.2 Поля

4.12.2.1 `char ControllerName[17]`

Пользовательское имя контроллера.

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

4.12.2.2 `unsigned int CtrlFlags`

[Флаги настроек контроллера.](#)

4.13 Структура `ctp_settings_t`

Настройки контроля позиции(для шагового двигателя).

Поля данных

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

4.13.1 Подробное описание

Настройки контроля позиции(для шагового двигателя).

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

См. также

```
set_ctp_settings  
get_ctp_settings  
get_ctp_settings, set_ctp_settings
```

4.13.2 Поля

4.13.2.1 `unsigned int CTPFlags`

[Флаги контроля позиции.](#)

4.13.2.2 unsigned int CTPMinError

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

Измеряется в шагах ШД.

4.14 Структура debug_read_t

Отладочные данные.

Поля данных

- uint8_t [DebugData](#) [128]

Отладочные данные.

4.14.1 Подробное описание

Отладочные данные.

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

См. также

[get_debug_read](#)

4.14.2 Поля

4.14.2.1 uint8_t DebugData[128]

Отладочные данные.

4.15 Структура debug_write_t

Отладочные данные.

Поля данных

- uint8_t [DebugData](#) [128]

Отладочные данные.

4.15.1 Подробное описание

Отладочные данные.

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

См. также

[set_debug_write](#)

4.15.2 Поля

4.15.2.1 uint8_t DebugData[128]

Отладочные данные.

4.16 Структура device_information_t

Команда чтения информации о контроллере.

Поля данных

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

4.16.1 Подробное описание

Команда чтения информации о контроллере.

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

См. также

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

4.16.2 Поля

4.16.2.1 unsigned int Major

Основной номер версии железа.

4.16.2.2 unsigned int Minor

Второстепенный номер версии железа.

4.16.2.3 unsigned int Release

Номер правок этой версии железа.

4.17 Структура device_network_information_t

Структура данных с информацией о сетевом устройстве.

Поля данных

- uint32_t [ipv4](#)
IPv4 address, passed in network byte order (big-endian byte order)
- char [nodename](#) [16]
Name of the Bindy node which hosts the device.
- uint32_t [axis_state](#)
Flags representing device state.
- char [locker_username](#) [16]
Name of the user who locked the device (if any)
- char [locker_nodename](#) [16]
Bindy node name, which was used to lock the device (if any)
- time_t [locked_time](#)
Time the lock was acquired at (UTC, microseconds since the epoch)

4.17.1 Подробное описание

Структура данных с информацией о сетевом устройстве.

4.18 Структура edges_settings_calb_t

Поля данных

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

4.18.1 Поля

4.18.1.1 unsigned int BorderFlags

[Флаги границ.](#)

4.18.1.2 unsigned int EnderFlags

[Флаги концевых выключателей.](#)

4.18.1.3 float LeftBorder

Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.

4.18.1.4 float RightBorder

Позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER.

4.19 Структура edges_settings_t

Настройки границ.

Поля данных

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

4.19.1 Подробное описание

Настройки границ.

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

См. также

```
set_edges_settings  
get_edges_settings  
get_edges_settings, set_edges_settings
```

4.19.2 Поля

4.19.2.1 unsigned int BorderFlags

[Флаги границ.](#)

4.19.2.2 unsigned int EnderFlags

[Флаги концевых выключателей.](#)

4.19.2.3 int LeftBorder

Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.

4.19.2.4 `int RightBorder`

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

4.19.2.5 `int uLeftBorder`

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

Диапазон: -255..255.

4.19.2.6 `int uRightBorder`

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

Диапазон: -255..255.

4.20 Структура `encoder_information_t`

Информация об энкодере.

Поля данных

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

4.20.1 Подробное описание

Информация об энкодере.

См. также

```
set_encoder_information  
get_encoder_information  
get_encoder_information, set_encoder_information
```

4.20.2 Поля

4.20.2.1 `char Manufacturer`[17]

Производитель.

Максимальная длина строки: 16 символов.

4.20.2.2 `char PartNumber`[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

4.21 Структура `encoder_settings_t`

Настройки энкодера.

Поля данных

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

4.21.1 Подробное описание

Настройки энкодера.

См. также

```
set_encoder_settings  
get_encoder_settings  
get_encoder_settings, set_encoder_settings
```

4.21.2 Поля

4.21.2.1 unsigned int EncoderSettings

[Флаги настроек энкодера.](#)

4.21.2.2 float MaxCurrentConsumption

Максимальное потребление тока (мА).

Тип данных: float.

4.21.2.3 float MaxOperatingFrequency

Максимальная частота (кГц).

Тип данных: float.

4.21.2.4 float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

4.21.2.5 float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

4.22 Структура engine_settings_calb_t

Поля данных

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

4.22.1 Поля

4.22.1.1 float Antipplay

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

Используется, если установлен флаг ENGINE_ANTIPLAY.

4.22.1.2 unsigned int EngineFlags

[Флаги параметров мотора.](#)

4.22.1.3 unsigned int MicrostepMode

[Флаги параметров микрошагового режима.](#)

4.22.1.4 unsigned int NomCurrent

Номинальный ток через мотор.

Ток стабилизируется для шаговых и может быть ограничен для DC(если установлен флаг ENGINE_LIMIT_CURR). Диапазон: 15..8000

4.22.1.5 float NomSpeed

Номинальная скорость.

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

4.22.1.6 unsigned int NomVoltage

Номинальное напряжение мотора.

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

4.22.1.7 unsigned int StepsPerRev

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

Диапазон: 1..65535.

4.23 Структура engine_settings_t

Настройки мотора.

Поля данных

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

4.23.1 Подробное описание

Настройки мотора.

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

См. также

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

4.23.2 Поля

4.23.2.1 int Antiplay

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

Используется, если установлен флаг ENGINE_ANTIPLAY.

4.23.2.2 unsigned int EngineFlags

Флаги параметров мотора.

4.23.2.3 unsigned int MicrostepMode

Флаги параметров микрошагового режима.

4.23.2.4 unsigned int NomCurrent

Номинальный ток через мотор.

Ток стабилизируется для шаговых и может быть ограничен для DC(если установлен флаг ENGINE_LIMIT_CURR). Диапазон: 15..8000

4.23.2.5 unsigned int NomSpeed

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

Контроллер будет сохранять скорость мотора не выше номинальной, если установлен флаг ENGINE_LIMIT_RPM. Диапазон: 1..100000.

4.23.2.6 unsigned int NomVoltage

Номинальное напряжение мотора.

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

4.23.2.7 unsigned int StepsPerRev

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

Диапазон: 1..65535.

4.23.2.8 unsigned int uNomSpeed

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

4.24 Структура entype_settings_t

Настройки типа мотора и типа силового драйвера.

Поля данных

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

4.24.1 Подробное описание

Настройки типа мотора и типа силового драйвера.

Эта структура содержит настройки типа мотора и типа силового драйвера.

Аргументы

id	идентификатор устройства
EngineType	тип мотора
DriverType	тип силового драйвера

См. также

`get_entype_settings`, `set_entype_settings`

4.24.2 Поля

4.24.2.1 unsigned int `DriverType`

Флаги, определяющие тип силового драйвера.

4.24.2.2 unsigned int `EngineType`

Флаги, определяющие тип мотора.

4.25 Структура `extio_settings_t`

Настройки EXTIO.

Поля данных

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

4.25.1 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](#)

4.25.2 Поля

4.25.2.1 unsigned int EXTIOModeFlags

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

4.25.2.2 unsigned int EXTIOSetupFlags

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

4.26 Структура feedback_settings_t

Настройки обратной связи.

Поля данных

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

4.26.1 Подробное описание

Настройки обратной связи.

Эта структура содержит настройки обратной связи.

См. также

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

4.26.2 Поля

4.26.2.1 unsigned int FeedbackFlags

[Флаги обратной связи.](#)

4.26.2.2 unsigned int FeedbackType

[Тип обратной связи.](#)

4.26.2.3 int HallShift

Фазовый сдвиг между выходным сигналом на обмотках BLDC двигателя и входным сигналом на датчиках Холла(0 - при активном только датчике холла А подается положительный потенциал на обмотку А и отрицательный потенциал на обмотку В).

4.26.2.4 unsigned int HallSPR

Количество отсчётов датчиков Холла на оборот.

4.27 Структура gear_information_t

Информация о редукторе.

Поля данных

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

4.27.1 Подробное описание

Информация о редукторе.

См. также

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

4.27.2 Поля

4.27.2.1 char Manufacturer[17]

Производитель.

Максимальная длина строки: 16 символов.

4.27.2.2 char PartNumber[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

4.28 Структура gear_settings_t

Настройки редуктора.

Поля данных

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

4.28.1 Подробное описание

Настройки редуктора.

См. также

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

4.28.2 Поля

4.28.2.1 float Efficiency

КПД редуктора (%).

Тип данных: float.

4.28.2.2 float InputInertia

Эквивалентная входная инерция редуктора(г см²).

Тип данных: float.

4.28.2.3 float MaxOutputBacklash

Выходной люфт редуктора (градус).

Тип данных: float.

4.28.2.4 float RatedInputSpeed

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

Тип данных: float.

4.28.2.5 float `RatedInputTorque`

Максимальный крутящий момент (Н м).

Тип данных: float.

4.28.2.6 float `ReductionIn`

Входной коэффициент редуктора.

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

4.28.2.7 float `ReductionOut`

Выходной коэффициент редуктора.

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

4.29 Структура `get_position_calb_t`

Поля данных

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

4.29.1 Поля

4.29.1.1 long_t `EncPosition`

Позиция энкодера.

4.29.1.2 float `Position`

Позиция двигателя.

4.30 Структура `get_position_t`

Данные о позиции.

Поля данных

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

4.30.1 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](#)

4.30.2 Поля

4.30.2.1 `long_t EncPosition`

Позиция энкодера.

4.30.2.2 `int uPosition`

Позиция в микрошагах(используется только с шаговыми двигателями).

4.31 Структура `hallsensor_information_t`

Информация об датчиках Холла.

Поля данных

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

4.31.1 Подробное описание

Информация об датчиках Холла.

См. также

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

4.31.2 Поля

4.31.2.1 `char Manufacturer[17]`

Производитель.

Максимальная длина строки: 16 символов.

4.31.2.2 `char PartNumber[25]`

Серия и номер модели.

Максимальная длина строки: 24 символа.

4.32 Структура `hallsensor_settings_t`

Настройки датчиков Холла.

Поля данных

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

4.32.1 Подробное описание

Настройки датчиков Холла.

См. также

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

4.32.2 Поля

4.32.2.1 float `MaxCurrentConsumption`

Максимальное потребление тока (мА).

Тип данных: float.

4.32.2.2 float `MaxOperatingFrequency`

Максимальная частота (кГц).

Тип данных: float.

4.32.2.3 float `SupplyVoltageMax`

Максимальное напряжение питания (В).

Тип данных: float.

4.32.2.4 float `SupplyVoltageMin`

Минимальное напряжение питания (В).

Тип данных: float.

4.33 Структура home_settings_calb_t

Поля данных

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

4.33.1 Поля

4.33.1.1 float FastHome

Скорость первого движения.

4.33.1.2 float HomeDelta

Расстояние отхода от точки останова.

4.33.1.3 unsigned int HomeFlags

[Флаги настроек команды home.](#)

4.33.1.4 float SlowHome

Скорость второго движения.

4.34 Структура home_settings_t

Настройки калибровки позиции.

Поля данных

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

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

- unsigned int [HomeFlags](#)
[Флаги настроек команды home.](#)

4.34.1 Подробное описание

Настройки калибровки позиции.

Эта структура содержит настройки, используемые при калибровке позиции.

См. также

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

4.34.2 Поля

4.34.2.1 unsigned int FastHome

Скорость первого движения.

Диапазон: 0..100000

4.34.2.2 int HomeDelta

Расстояние отхода от точки останова.

4.34.2.3 unsigned int HomeFlags

[Флаги настроек команды home.](#)

4.34.2.4 unsigned int SlowHome

Скорость второго движения.

Диапазон: 0..100000.

4.34.2.5 unsigned int uFastHome

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

4.34.2.6 int uHomeDelta

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

Диапазон: -255..255.

4.34.2.7 unsigned int uSlowHome

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

4.35 Структура `init_random_t`

Случайный ключ.

Поля данных

- `uint8_t key` [16]
Случайный ключ.

4.35.1 Подробное описание

Случайный ключ.

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

См. также

[get_init_random](#)

4.35.2 Поля

4.35.2.1 `uint8_t key`[16]

Случайный ключ.

4.36 Структура `joystick_settings_t`

Настройки джойстика.

Поля данных

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

4.36.1 Подробное описание

Настройки джойстика.

Команда чтения настроек и калибровки джойстика. При отклонении джойстика более чем на Dead-Zone от центрального положения начинается движение со скоростью, определяемой отклонением

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

См. также

```
set_joystick_settings  
get_joystick_settings  
get_joystick_settings, set_joystick_settings
```

4.36.2 Поля

4.36.2.1 unsigned int `DeadZone`

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

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

4.36.2.2 unsigned int `ExpFactor`

Фактор экспоненциальной нелинейности отклика джойстика.

4.36.2.3 unsigned int `JoyCenter`

Значение в шагах джойстика, соответствующее неотклонённому устройству.

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

4.36.2.4 unsigned int `JoyFlags`

Флаги джойстика.

4.36.2.5 unsigned int `JoyHighEnd`

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

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

4.36.2.6 unsigned int `JoyLowEnd`

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

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

4.37 Структура `motor_information_t`

Информация о двигателе.

Поля данных

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

4.37.1 Подробное описание

Информация о двигателе.

См. также

[set_motor_information](#)
[get_motor_information](#)
[get_motor_information](#), [set_motor_information](#)

4.37.2 Поля

4.37.2.1 char Manufacturer[17]

Производитель.

Максимальная длина строки: 16 символов.

4.37.2.2 char PartNumber[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

4.38 Структура motor_settings_t

Настройки двигателя.

Поля данных

- unsigned int [MotorType](#)
Флаг типа двигателя.
- unsigned int [ReservedField](#)
Зарезервировано
- unsigned int [Poles](#)
Кол-во пар полюсов у DC или BLDC двигателя или кол-во шагов на оборот для шагового двигателя.
- unsigned int [Phases](#)
Кол-во фаз у BLDC двигателя.
- float [NominalVoltage](#)
Номинальное напряжение на обмотке (В).
- float [NominalCurrent](#)
Максимальный постоянный ток в обмотке для DC и BLDC двигателей, номинальный ток в обмотке для шаговых двигателей (А).
- float [NominalSpeed](#)
Номинальная скорость (об/мин).

- float [NominalTorque](#)
Номинальный крутящий момент (мН м).
- float [NominalPower](#)
Номинальная мощность (Вт).
- float [WindingResistance](#)
Сопротивление обмотки DC двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC двигателя (Ом).
- float [WindingInductance](#)
Индуктивность обмотки DC двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC двигателя (мГн).
- float [RotorInertia](#)
Инерция ротора (г см²).
- float [StallTorque](#)
Крутящий момент удержания позиции для шагового двигателя или крутящий момент при неподвижном роторе для других типов двигателей (мН м).
- float [DetentTorque](#)
Момент удержания позиции с незапитанными обмотками (мН м).
- float [TorqueConstant](#)
Константа крутящего момента, определяющая коэффициент пропорциональности максимального момента силы ротора от протекающего в обмотке тока (мН м/А).
- float [SpeedConstant](#)
Константа скорости, определяющая значение или амплитуду напряжения наведённой индукции при вращении ротора DC или BLDC двигателя (об/мин / В) или шагового двигателя (шаг/с / В).
- float [SpeedTorqueGradient](#)
Градиент крутящего момента (об/мин / мН м).
- float [MechanicalTimeConstant](#)
Механическая постоянная времени (мс).
- float [MaxSpeed](#)
Максимальная разрешённая скорость для шаговых двигателей (шаг/с) или для DC и BLDC двигателей (об/мин).
- float [MaxCurrent](#)
Максимальный ток в обмотке (А).
- float [MaxCurrentTime](#)
Безопасная длительность максимального тока в обмотке (мс).
- float [NoLoadCurrent](#)
Ток потребления в холостом режиме (А).
- float [NoLoadSpeed](#)
Скорость в холостом режиме (об/мин).

4.38.1 Подробное описание

Настройки двигателя.

См. также

```
set_motor_settings  
get_motor_settings  
get_motor_settings, set_motor_settings
```

4.38.2 Поля

4.38.2.1 float DetentTorque

Момент удержания позиции с незапитанными обмотками (мН м).

Тип данных: float.

4.38.2.2 float MaxCurrent

Максимальный ток в обмотке (А).

Тип данных: float.

4.38.2.3 float MaxCurrentTime

Безопасная длительность максимального тока в обмотке (мс).

Тип данных: float.

4.38.2.4 float MaxSpeed

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

Тип данных: float.

4.38.2.5 float MechanicalTimeConstant

Механическая постоянная времени (мс).

Тип данных: float.

4.38.2.6 unsigned int MotorType

Флаг типа двигателя.

4.38.2.7 float NoLoadCurrent

Ток потребления в холостом режиме (А).

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

4.38.2.8 float NoLoadSpeed

Скорость в холостом режиме (об/мин).

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

4.38.2.9 float NominalCurrent

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

Тип данных: float.

4.38.2.10 float NominalPower

Номинальная мощность(Вт).

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

4.38.2.11 float NominalSpeed

Номинальная скорость (об/мин).

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

4.38.2.12 float NominalTorque

Номинальный крутящий момент (мН м).

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

4.38.2.13 float NominalVoltage

Номинальное напряжение на обмотке (В).

Тип данных: float.

4.38.2.14 unsigned int Phases

Кол-во фаз у BLDC двигателя.

4.38.2.15 unsigned int Poles

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

4.38.2.16 float RotorInertia

Инерция ротора (г см²).

Тип данных: float.

4.38.2.17 float SpeedConstant

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

Тип данных: float.

4.38.2.18 float SpeedTorqueGradient

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

Тип данных: float.

4.38.2.19 float StallTorque

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

Тип данных: float.

4.38.2.20 float TorqueConstant

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

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

4.38.2.21 float WindingInductance

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

Тип данных: float.

4.38.2.22 float WindingResistance

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

Тип данных: float.

4.39 Структура move_settings_calb_t

Поля данных

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

4.39.1 Поля

4.39.1.1 float Accel

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

4.39.1.2 float AntipplaySpeed

Скорость в режиме антилюфта.

4.39.1.3 float Decel

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

4.39.1.4 float Speed

Заданная скорость.

4.40 Структура move_settings_t

Настройки движения.

Поля данных

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

4.40.1 Подробное описание

Настройки движения.

См. также

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

4.40.2 Поля

4.40.2.1 unsigned int Accel

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

Диапазон: 1..65535.

4.40.2.2 unsigned int AntiplaySpeed

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

Диапазон: 0..100000.

4.40.2.3 unsigned int Decel

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

Диапазон: 1..65535.

4.40.2.4 unsigned int Speed

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

Диапазон: 0..100000.

4.40.2.5 unsigned int uAntiplaySpeed

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

Используется только с шаговым мотором.

4.40.2.6 `unsigned int uSpeed`

Заданная скорость в единицах деления микрошага в секунду.

Используется только с шаговым мотором.

4.41 Структура `nonvolatile_memory_t`

Пользовательские данные для сохранения во FRAM.

Поля данных

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

4.41.1 Подробное описание

Пользовательские данные для сохранения во FRAM.

См. также

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

4.41.2 Поля

4.41.2.1 `unsigned int UserData[7]`

Пользовательские данные.

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

4.42 Структура `pid_settings_t`

Настройки ПИД.

Поля данных

- `unsigned int KpU`
Пропорциональный коэффициент ПИД контура по напряжению
- `unsigned int KiU`
Интегральный коэффициент ПИД контура по напряжению
- `unsigned int KdU`
Дифференциальный коэффициент ПИД контура по напряжению

4.42.1 Подробное описание

Настройки ПИД.

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

См. также

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

4.43 Структура power_settings_t

Настройки питания шагового мотора.

Поля данных

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

4.43.1 Подробное описание

Настройки питания шагового мотора.

См. также

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

4.43.2 Поля

4.43.2.1 unsigned int CurrentSetTime

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

4.43.2.2 unsigned int CurrReductDelay

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

4.43.2.3 unsigned int HoldCurrent

Ток мотора в режиме удержания, в процентах от номинального.

Диапазон: 0..100.

4.43.2.4 unsigned int PowerFlags

[Флаги параметров питания шагового мотора.](#)

4.43.2.5 `unsigned int PowerOffDelay`

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

4.44 Структура `secure_settings_t`

Эта структура содержит необработанные данные с АЦП и нормированные значения.

Поля данных

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

4.44.1 Подробное описание

Эта структура содержит необработанные данные с АЦП и нормированные значения.

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

См. также

`get_secure_settings`
`set_secure_settings`
`get_secure_settings, set_secure_settings`

4.44.2 Поля

4.44.2.1 `unsigned int CriticalIpwr`

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

4.44.2.2 `unsigned int CriticalIusb`

Максимальный ток USB, вызывающий состояние ALARM, в мА.

4.44.2.3 `unsigned int CriticalUpwr`

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

4.44.2.4 `unsigned int CriticalUusb`

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

4.44.2.5 `unsigned int Flags`

[Флаги критических параметров.](#)

4.44.2.6 `unsigned int LowUpwrOff`

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

4.44.2.7 `unsigned int MinimumUusb`

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

4.45 Структура `serial_number_t`

Структура с серийным номером и версией железа.

Поля данных

- `unsigned int SN`
Новый серийный номер платы.
- `uint8_t Key [32]`
Ключ защиты для установки серийного номера (256 бит).
- `unsigned int Major`
Основной номер версии железа.
- `unsigned int Minor`
Второстепенный номер версии железа.
- `unsigned int Release`
Номер правок этой версии железа.

4.45.1 Подробное описание

Структура с серийным номером и версией железа.

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

См. также

[set_serial_number](#)

4.45.2 Поля

4.45.2.1 `uint8_t Key[32]`

Ключ защиты для установки серийного номера (256 бит).

4.45.2.2 `unsigned int Major`

Основной номер версии железа.

4.45.2.3 `unsigned int Minor`

Второстепенный номер версии железа.

4.45.2.4 `unsigned int Release`

Номер правок этой версии железа.

4.45.2.5 `unsigned int SN`

Новый серийный номер платы.

4.46 Структура `set_position_calb_t`

Поля данных

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

4.46.1 Поля

4.46.1.1 `long_t EncPosition`

Позиция энкодера.

4.46.1.2 `unsigned int PosFlags`

Флаги установки положения.

4.46.1.3 `float Position`

Позиция двигателя.

4.47 Структура `set_position_t`

Данные о позиции.

Поля данных

- `int Position`
Позиция в основных шагах двигателя

- `int` [uPosition](#)
Позиция в микрошагах(используется только с шаговыми двигателями).
- `long_t` [EncPosition](#)
Позиция энкодера.
- `unsigned int` [PosFlags](#)
Флаги установки положения.

4.47.1 Подробное описание

Данные о позиции.

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

См. также

[set_position](#)

4.47.2 Поля

4.47.2.1 `long_t` `EncPosition`

Позиция энкодера.

4.47.2.2 `unsigned int` `PosFlags`

[Флаги установки положения.](#)

4.47.2.3 `int` `uPosition`

Позиция в микрошагах(используется только с шаговыми двигателями).

4.48 Структура `stage_information_t`

Информация о позиционере.

Поля данных

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

4.48.1 Подробное описание

Информация о позиционере.

См. также

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

4.48.2 Поля

4.48.2.1 char Manufacturer[17]

Производитель.

Максимальная длина строки: 16 символов.

4.48.2.2 char PartNumber[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

4.49 Структура stage_name_t

Пользовательское имя подвижки.

Поля данных

- char [PositionerName](#) [17]

Пользовательское имя подвижки.

4.49.1 Подробное описание

Пользовательское имя подвижки.

См. также

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

4.49.2 Поля

4.49.2.1 char PositionerName[17]

Пользовательское имя подвижки.

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

4.50 Структура stage_settings_t

Настройки позиционера.

Поля данных

- float [LeadScrewPitch](#)

Шаг ходового винта в мм.

- char [Units](#) [9]

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

- float [MaxSpeed](#)

Максимальная скорость (Units/c).

- float [TravelRange](#)
Диапазон перемещения (Units).
- float [SupplyVoltageMin](#)
Минимальное напряжение питания (В).
- float [SupplyVoltageMax](#)
Максимальное напряжение питания (В).
- float [MaxCurrentConsumption](#)
Максимальный ток потребления (А).
- float [HorizontalLoadCapacity](#)
Горизонтальная грузоподъемность (кг).
- float [VerticalLoadCapacity](#)
Вертикальная грузоподъемность (кг).

4.50.1 Подробное описание

Настройки позиционера.

См. также

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

4.50.2 Поля

4.50.2.1 float HorizontalLoadCapacity

Горизонтальная грузоподъемность (кг).

Тип данных: float.

4.50.2.2 float LeadScrewPitch

Шаг ходового винта в мм.

Тип данных: float.

4.50.2.3 float MaxCurrentConsumption

Максимальный ток потребления (А).

Тип данных: float.

4.50.2.4 float MaxSpeed

Максимальная скорость (Units/c).

Тип данных: float.

4.50.2.5 float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

4.50.2.6 float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

4.50.2.7 float TravelRange

Диапазон перемещения (Units).

Тип данных: float.

4.50.2.8 char Units[9]

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

4.50.2.9 float VerticalLoadCapacity

Вертикальная грузоподъемность (кг).

Тип данных: float.

4.51 Структура status_calb_t

Поля данных

- unsigned int [MoveSts](#)
Флаги состояния движения.
- unsigned int [MvCmdSts](#)
Состояние команды движения.
- unsigned int [PWRSts](#)
Флаги состояния питания шагового мотора.
- unsigned int [EncSts](#)
Состояние энкодера.
- unsigned int [WindSts](#)
Состояние обмоток.
- float [CurPosition](#)
Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь.
- long_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.
- int [Upwr](#)
Напряжение на силовой части.
- int [Iusb](#)
Ток потребления по USB.
- int [Uusb](#)
Напряжение на USB.

- int [CurT](#)
Температура процессора в десятых долях градусов цельсия.
- unsigned int [Flags](#)
[Флаги состояния.](#)
- unsigned int [GPIOFlags](#)
[Флаги состояния GPIO входов.](#)
- unsigned int [CmdBufFreeSpace](#)
Это поле показывает количество свободных ячеек буфера цепочки синхронизации.

4.51.1 Поля

4.51.1.1 unsigned int CmdBufFreeSpace

Это поле показывает количество свободных ячеек буфера цепочки синхронизации.

4.51.1.2 float CurPosition

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

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

4.51.1.3 float CurSpeed

Текущая скорость.

4.51.1.4 int CurT

Температура процессора в десятых долях градусов цельсия.

4.51.1.5 long_t EncPosition

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

4.51.1.6 unsigned int EncSts

[Состояние энкодера.](#)

4.51.1.7 unsigned int Flags

[Флаги состояния.](#)

4.51.1.8 unsigned int GPIOFlags

[Флаги состояния GPIO входов.](#)

4.51.1.9 int Ipwr

Ток потребления силовой части.

4.51.1.10 int lusb

Ток потребления по USB.

4.51.1.11 unsigned int MoveSts

Флаги состояния движения.

4.51.1.12 unsigned int MvCmdSts

Состояние команды движения.

4.51.1.13 unsigned int PWRSts

Флаги состояния питания шагового мотора.

4.51.1.14 int Upwr

Напряжение на силовой части.

4.51.1.15 int Uusb

Напряжение на USB.

4.51.1.16 unsigned int WindSts

Состояние обмоток.

4.52 Структура status_t

Состояние устройства.

Поля данных

- unsigned int [MoveSts](#)
Флаги состояния движения.
- unsigned int [MvCmdSts](#)
Состояние команды движения.
- unsigned int [PWRSts](#)
Флаги состояния питания шагового мотора.
- unsigned int [EncSts](#)
Состояние энкодера.
- unsigned int [WindSts](#)
Состояние обмоток.
- int [CurPosition](#)
Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь.
- int [uCurPosition](#)
Дробная часть текущей позиции в микрошагах (-255..255).
- long_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 (-255..255).
- int [Ipwr](#)
Ток потребления силовой части.
- int [Upwr](#)
Напряжение на силовой части.
- int [Iusb](#)
Ток потребления по USB.
- int [Uusb](#)
Напряжение на USB.
- int [Curt](#)
Температура процессора в десятых долях градусов цельсия.
- unsigned int [Flags](#)
Флаги состояния.
- unsigned int [GPIOFlags](#)
Флаги состояния GPIO входов.
- unsigned int [CmdBufFreeSpace](#)
Это поле показывает количество свободных ячеек буфера цепочки синхронизации.

4.52.1 Подробное описание

Состояние устройства.

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

См. также

`get_status_impl`

4.52.2 Поля

4.52.2.1 unsigned int CmdBufFreeSpace

Это поле показывает количество свободных ячеек буфера цепочки синхронизации.

4.52.2.2 int CurPosition

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

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

4.52.2.3 int CurSpeed

Текущая скорость.

4.52.2.4 int CurT

Температура процессора в десятых долях градусов цельсия.

4.52.2.5 long_t EncPosition

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

4.52.2.6 unsigned int EncSts

Состояние энкодера.

4.52.2.7 unsigned int Flags

Флаги состояния.

4.52.2.8 unsigned int GPIOFlags

Флаги состояния GPIO входов.

4.52.2.9 int Ipwr

Ток потребления силовой части.

4.52.2.10 int Iusb

Ток потребления по USB.

4.52.2.11 unsigned int MoveSts

Флаги состояния движения.

4.52.2.12 unsigned int MvCmdSts

Состояние команды движения.

4.52.2.13 unsigned int PWRSts

Флаги состояния питания шагового мотора.

4.52.2.14 int uCurPosition

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

Используется только с шаговым двигателем.

4.52.2.15 int uCurSpeed

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

Используется только с шаговым двигателем.

4.52.2.16 `int Upwr`

Напряжение на силовой части.

4.52.2.17 `int Uusb`

Напряжение на USB.

4.52.2.18 `unsigned int WindSts`

Состояние обмоток.

4.53 Структура `sync_in_settings_calb_t`

Поля данных

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

4.53.1 Поля

4.53.1.1 `unsigned int ClutterTime`

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

4.53.1.2 `float Position`

Желаемая позиция или смещение.

4.53.1.3 `float Speed`

Заданная скорость.

4.53.1.4 `unsigned int SyncInFlags`

Флаги настроек синхронизации входа.

4.54 Структура `sync_in_settings_t`

Настройки входной синхронизации.

Поля данных

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

4.54.1 Подробное описание

Настройки входной синхронизации.

Эта структура содержит все настройки, определяющие поведение входа синхронизации.

См. также

```
get_sync_in_settings  
set_sync_in_settings  
get_sync_in_settings, set_sync_in_settings
```

4.54.2 Поля

4.54.2.1 unsigned int ClutterTime

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

4.54.2.2 unsigned int Speed

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

Диапазон: 0..100000.

4.54.2.3 unsigned int SyncInFlags

Флаги настроек синхронизации входа.

4.54.2.4 int uPosition

Дробная часть позиции или смещения в микрошагах.

Используется только с шаговым двигателем. Диапазон: -255..255.

4.54.2.5 unsigned int uSpeed

Заданная скорость в микрошагах в секунду.

Используется только с шаговым мотором.

4.55 Структура `sync_out_settings_calb_t`

Поля данных

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

4.55.1 Поля

4.55.1.1 float Accuracy

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

4.55.1.2 unsigned int SyncOutFlags

Флаги настроек синхронизации выхода.

4.55.1.3 unsigned int SyncOutPeriod

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

4.55.1.4 unsigned int SyncOutPulseSteps

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

4.56 Структура `sync_out_settings_t`

Настройки выходной синхронизации.

Поля данных

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

- unsigned int [uAccuracy](#)

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

4.56.1 Подробное описание

Настройки выходной синхронизации.

Эта структура содержит все настройки, определяющие поведение выхода синхронизации.

См. также

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

4.56.2 Поля

4.56.2.1 unsigned int Accuracy

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

4.56.2.2 unsigned int SyncOutFlags

[Флаги настроек синхронизации выхода.](#)

4.56.2.3 unsigned int SyncOutPeriod

Период генерации импульсов, используется при установленном флаге SYNCOUT_ONPERIOD.

4.56.2.4 unsigned int SyncOutPulseSteps

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

4.56.2.5 unsigned int uAccuracy

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

4.57 Структура `uart_settings_t`

Настройки UART.

Поля данных

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

4.57.1 Подробное описание

Настройки UART.

Эта структура содержит настройки UART.

См. также

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

4.57.2 Поля

4.57.2.1 `unsigned int UARTSetupFlags`

[Флаги настроек четности команды uart.](#)

Глава 5

Файлы

5.1 Файл `ximc.h`

Заголовочный файл для библиотеки `libximc`.

Структуры данных

- struct `calibration_t`
Структура калибровок
- struct `device_network_information_t`
Структура данных с информацией о сетевом устройстве.
- struct `feedback_settings_t`
Настройки обратной связи.
- struct `home_settings_t`
Настройки калибровки позиции.
- struct `home_settings_calb_t`
- struct `move_settings_t`
Настройки движения.
- struct `move_settings_calb_t`
- struct `engine_settings_t`
Настройки мотора.
- struct `engine_settings_calb_t`
- struct `entype_settings_t`
Настройки типа мотора и типа силового драйвера.
- struct `power_settings_t`
Настройки питания шагового мотора.
- struct `secure_settings_t`
Эта структура содержит необработанные данные с АЦП и нормированные значения.
- struct `edges_settings_t`
Настройки границ.
- struct `edges_settings_calb_t`
- struct `pid_settings_t`
Настройки ПИД.
- struct `sync_in_settings_t`
Настройки входной синхронизации.
- struct `sync_in_settings_calb_t`
- struct `sync_out_settings_t`

- Настройки выходной синхронизации.
- struct [sync_out_settings_calb_t](#)
- struct [extio_settings_t](#)
- Настройки EXTIO.
- struct [brake_settings_t](#)
- Настройки тормоза.
- struct [control_settings_t](#)
- Настройки управления.
- struct [control_settings_calb_t](#)
- struct [joystick_settings_t](#)
- Настройки джойстика.
- struct [ctp_settings_t](#)
- Настройки контроля позиции(для шагового двигателя).
- struct [uart_settings_t](#)
- Настройки UART.
- struct [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 [command_add_sync_in_action_t](#)
- Это команда добавляет один элемент в буфер FIFO команд.
- struct [command_add_sync_in_action_calb_t](#)
- struct [get_position_t](#)
- Данные о позиции.
- struct [get_position_calb_t](#)
- struct [set_position_t](#)
- Данные о позиции.
- struct [set_position_calb_t](#)
- struct [status_t](#)
- Состояние устройства.
- struct [status_calb_t](#)
- struct [chart_data_t](#)
- Дополнительное состояние устройства.
- struct [device_information_t](#)
- Команда чтения информации о контроллере.
- struct [serial_number_t](#)
- Структура с серийным номером 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`
Настройки двигателя.
- struct `encoder_information_t`
Информация об энкодере.
- struct `encoder_settings_t`
Настройки энкодера.
- struct `hallsensor_information_t`
Информация об датчиках Холла.
- struct `hallsensor_settings_t`
Настройки датчиков Холла.
- struct `gear_information_t`
Информация о редукторе.
- struct `gear_settings_t`
Настройки редуктора.
- struct `accessories_settings_t`
Информация о дополнительных аксессуарах.
- struct `init_random_t`
Случайный ключ.
- struct `command_change_motor_t`

Макросы

- #define `XIMC_API`
Library import macro Macros allows to automatically import function from shared library.
- #define `XIMC_CALLCONV`
Library calling convention macros.
- #define `XIMC_RETTYPE` void*
Thread return type.
- #define `device_undefined` -1
Макрос, означающий неопределенное устройство

Результаты выполнения команд

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

Уровень логирования

- #define `LOGLEVEL_ERROR` 0x01
Уровень логирования - ошибка
- #define `LOGLEVEL_WARNING` 0x02
Уровень логирования - предупреждение
- #define `LOGLEVEL_INFO` 0x03

- Уровень логирования - информация
- #define LOGLEVEL_DEBUG 0x04
- Уровень логирования - отладка

Флаги поиска устройств

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

Флаги состояния движения

Возвращаются командой get_status.

См. также

get_status
status_t::move_state
status_t::MoveSts, get_status_impl

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

Флаги настроек контроллера

См. также

set_controller_name
get_controller_name
controller_name_t::CtrlFlags, get_controller_name, set_controller_name

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

Флаги состояния питания шагового мотора

Возвращаются командой get_status.

См. также

status_t::power_state
get_status
status_t::PWRSts, get_status_impl

- #define PWR_STATE_UNKNOWN 0x00
- Неизвестное состояние, которое не должно никогда реализовываться.
- #define PWR_STATE_OFF 0x01
- Обмотки мотора разомкнуты и не управляются драйвером.
- #define PWR_STATE_NORM 0x03
- Обмотки запитаны номинальным током.
- #define PWR_STATE_REDUCT 0x04
- Обмотки намеренно запитаны уменьшенным током от рабочего для снижения потребляемой мощности.
- #define PWR_STATE_MAX 0x05

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

Флаги состояния

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

См. также

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

- #define STATE_CONTR 0x0003F
Флаги состояния контроллера.
- #define STATE_ERRC 0x00001
Недопустимая команда.
- #define STATE_ERRD 0x00002
Нарушение целостности данных.
- #define STATE_ERRV 0x00004
Недопустимое значение данных.
- #define STATE_EEPROM_CONNECTED 0x00010
Подключена память EEPROM с настройками.
- #define STATE_IS_HOMED 0x00020
Калибровка выполнена
- #define STATE_SECUR 0x3FFC0
Флаги опасности.
- #define STATE_ALARM 0x00040
Контроллер находится в состоянии ALARM, показывая, что случилась какая-то опасная ситуация.
- #define STATE_CTP_ERROR 0x00080
Контроль позиции нарушен(используется только с шаговым двигателем).
- #define STATE_POWER_OVERHEAT 0x00100
Перегрелась силовая часть платы.
- #define STATE_CONTROLLER_OVERHEAT 0x00200
Перегрелась микросхема контроллера.
- #define STATE_OVERLOAD_POWER_VOLTAGE 0x00400
Превышено напряжение на силовой части.
- #define STATE_OVERLOAD_POWER_CURRENT 0x00800
Превышен максимальный ток потребления силовой части.
- #define STATE_OVERLOAD_USB_VOLTAGE 0x01000
Превышено напряжение на USB.
- #define STATE_LOW_USB_VOLTAGE 0x02000
Слишком низкое напряжение на USB.
- #define STATE_OVERLOAD_USB_CURRENT 0x04000
Превышен максимальный ток потребления USB.
- #define STATE_BORDERS_SWAP_MISSET 0x08000
Достижение неверной границы.
- #define STATE_LOW_POWER_VOLTAGE 0x10000
Напряжение на силовой части ниже чем напряжение Low Voltage Protection.
- #define STATE_H_BRIDGE_FAULT 0x20000
Получен сигнал от драйвера о неисправности
- #define STATE_CURRENT_MOTOR_BITS 0xC0000
Флаги выбора мотора.
- #define STATE_CURRENT_MOTOR0 0x00000
Мотор 0.
- #define STATE_CURRENT_MOTOR1 0x40000
Мотор 1.
- #define STATE_CURRENT_MOTOR2 0x80000

- Мотор 2.
#define STATE_CURRENT_MOTOR3 0xC0000
- Мотор 3.

Флаги состояния GPIO входов

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

См. также

```
status_t::flags
get_status
status_t::GPIOFlags, get_status_impl
```

- #define STATE_DIG_SIGNAL 0xFFFF
Флаги цифровых сигналов.
- #define STATE_RIGHT_EDGE 0x0001
Достижение правой границы.
- #define STATE_LEFT_EDGE 0x0002
Достижение левой границы.
- #define STATE_BUTTON_RIGHT 0x0004
Состояние кнопки "вправо" (1, если нажата).
- #define STATE_BUTTON_LEFT 0x0008
Состояние кнопки "влево" (1, если нажата).
- #define STATE_GPIO_PINOUT 0x0010
Если флаг установлен, ввод/вывод общего назначения работает как выход; если флаг сброшен, ввод/вывод работает как вход.
- #define STATE_GPIO_LEVEL 0x0020
Состояние ввода/вывода общего назначения.
- #define STATE_HALL_A 0x0040
Состояние вывода датчика холла(а)(флаг "1", если датчик активен).
- #define STATE_HALL_B 0x0080
Состояние вывода датчика холла(б)(флаг "1", если датчик активен).
- #define STATE_HALL_C 0x0100
Состояние вывода датчика холла(с)(флаг "1", если датчик активен).
- #define STATE_BRAKE 0x0200
Состояние вывода управления тормозом(флаг "1" - если на тормоз подаётся питание, "0" - если тормоз не запитан).
- #define STATE_REV_SENSOR 0x0400
Состояние вывода датчика оборотов(флаг "1", если датчик активен).
- #define STATE_SYNC_INPUT 0x0800
Состояние входа синхронизации(1, если вход синхронизации активен).
- #define STATE_SYNC_OUTPUT 0x1000
Состояние выхода синхронизации(1, если выход синхронизации активен).
- #define STATE_ENC_A 0x2000
Состояние ножки А энкодера(флаг "1", если энкодер активен).
- #define STATE_ENC_B 0x4000
Состояние ножки В энкодера(флаг "1", если энкодер активен).

Состояние энкодера

Состояние энкодера, подключенного к контроллеру.

См. также

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

- #define ENC_STATE_ABSENT 0x00
Энкодер не подключен.

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

Состояние обмоток

Состояние обмоток двигателя, подключенного к контроллеру.

См. также

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

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

Состояние команды движения

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

См. также

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

- `#define MVCMD_NAME_BITS 0x3F`
Битовая маска активной команды.
- `#define MVCMD_UKNWN 0x00`
Неизвестная команда.
- `#define MVCMD_MOVE 0x01`
Команда move.
- `#define MVCMD_MOVR 0x02`
Команда movr.
- `#define MVCMD_LEFT 0x03`
Команда left.
- `#define MVCMD_RIGHT 0x04`
Команда right.
- `#define MVCMD_STOP 0x05`
Команда stop.

- `#define MVCMD_HOME 0x06`
Команда home.
- `#define MVCMD_LOFT 0x07`
Команда loft.
- `#define MVCMD_SSTP 0x08`
Команда плавной остановки(SSTP).
- `#define MVCMD_ERROR 0x40`
Состояние завершения движения (1 - команда движения выполнена с ошибкой, 0 - команда движения выполнена корректно).
- `#define MVCMD_RUNNING 0x80`
Состояние команды движения (0 - команда движения выполнена, 1 - команда движения сейчас выполняется).

Флаги параметров мотора

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

См. также

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

- `#define ENGINE_REVERSE 0x01`
Флаг реверса.
- `#define ENGINE_CURRENT_AS_RMS 0x02`
Флаг интерпретации значения тока.
- `#define ENGINE_MAX_SPEED 0x04`
Флаг максимальной скорости.
- `#define ENGINE_ANTIPLAY 0x08`
Компенсация люфта.
- `#define ENGINE_ACCEL_ON 0x10`
Ускорение.
- `#define ENGINE_LIMIT_VOLT 0x20`
Номинальное напряжение мотора.
- `#define ENGINE_LIMIT_CURR 0x40`
Номинальный ток мотора.
- `#define ENGINE_LIMIT_RPM 0x80`
Номинальная частота вращения мотора.

Флаги параметров микрошагового режима

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

См. также

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

- `#define MICROSTEP_MODE_FULL 0x01`
Полношаговый режим.
- `#define MICROSTEP_MODE_FRAC_2 0x02`
Деление шага 1/2.
- `#define MICROSTEP_MODE_FRAC_4 0x03`
Деление шага 1/4.
- `#define MICROSTEP_MODE_FRAC_8 0x04`
Деление шага 1/8.

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

Флаги, определяющие тип мотора

Определяют тип мотора. Возвращаются командой `get_entype_settings`.

См. также

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

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

Флаги, определяющие тип силового драйвера

Определяют тип силового драйвера. Возвращаются командой `get_entype_settings`.

См. также

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

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

Флаги параметров питания шагового мотора

Возвращаются командой `get_power_settings`.

См. также

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

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

Флаги критических параметров.

Возвращаются командой `get_secure_settings`.

См. также

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

- `#define ALARM_ON_DRIVER_OVERHEATING 0x01`
Если флаг установлен, то войти в состояние Alarm при получении сигнала подступающего перегрева с драйвера.
- `#define LOW_UPWR_PROTECTION 0x02`
Если установлен, то выключать силовую часть при напряжении меньшем LowUpwrOff.
- `#define H_BRIDGE_ALERT 0x04`
Если установлен, то выключать силовую часть при сигнале неполадки в одном из транзисторных мостов.
- `#define ALARM_ON_BORDERS_SWAP_MISSET 0x08`
Если флаг установлен, то войти в состояние Alarm при получении сигнала с противоположного концевика.
- `#define ALARM_FLAGS_STICKING 0x10`
Если флаг установлен, то только по команде STOP возможен сброс всех флагов ALARM.
- `#define USB_BREAK_RECONNECT 0x20`
Если флаг установлен, то будет включен блок перезагрузки USB при поломке связи.

Флаги установки положения

Возвращаются командой `get_position`.

См. также

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

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

Тип обратной связи.

См. также

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

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

Флаги обратной связи.

См. также

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

- `#define FEEDBACK_ENC_REVERSE 0x01`
Обратный счет у энкодера.
- `#define FEEDBACK_HALL_REVERSE 0x02`
Обратный счёт позиции по датчикам Холла.
- `#define FEEDBACK_ENC_TYPE_BITS 0xC0`
Биты, отвечающие за тип энкодера.
- `#define FEEDBACK_ENC_TYPE_AUTO 0x00`
Определять тип энкодера автоматически.
- `#define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40`
Недифференциальный энкодер.
- `#define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80`
Дифференциальный энкодер.

Флаги настроек синхронизации входа

См. также

```
sync_settings_t::syncin_flags
get_sync_settings
set_sync_settings
sync_in_settings_t::SyncInFlags, get_sync_in_settings, set_sync_in_settings
```

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

Флаги настроек синхронизации выхода

См. также

```
sync_settings_t::syncout_flags
get_sync_settings
set_sync_settings
sync_out_settings_t::SyncOutFlags, get_sync_out_settings, set_sync_out_settings
```

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

Флаги настройки работы внешнего ввода/вывода

См. также

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

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

Флаги настройки режимов внешнего ввода/вывода

См. также

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

- `#define EXTIO_SETUP_MODE_IN_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.
- `#define EXTIO_SETUP_MODE_OUT_MOTOR_FOUND 0x50`
Ножка находится в активном состоянии при обнаружении подключенного двигателя (первой обмотки).

Флаги границ

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

См. также

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

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

Флаги концевых выключателей

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

См. также

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

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

Флаги настроек тормоза

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

См. также

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

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

Флаги управления

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

См. также

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

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

Флаги джойстика

Управляют состояниями джойстика.

См. также

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

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

Флаги контроля позиции

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

См. также

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

- #define **CTP_ENABLED** 0x01
Контроль позиции включен, если флаг установлен.
- #define **CTP_BASE** 0x02
Опорой является датчик оборотов, если флаг установлен; иначе - энкодер.
- #define **CTP_ALARM_ON_ERROR** 0x04
Войти в состояние ALARM при расхождении позиции, если флаг установлен.

- `#define REV_SENS_INV 0x08`
Сенсор считается активным, когда на нём 0, а инвертирование делает активным уровнем 1.
- `#define CTP_ERROR_CORRECTION 0x10`
Корректировать ошибки, возникающие при проскальзывании, если флаг установлен.

Флаги настроек команды home

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

См. также

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

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

Флаги настроек четности команды uart

См. также

```
uart_settings_t::UARTSetupFlags, get_uart_settings, set_uart_settings
```

- `#define UART_PARITY_BITS 0x03`
Биты, отвечающие за выбор четности.
- `#define UART_PARITY_BIT_EVEN 0x00`
Бит 1, если чет
- `#define UART_PARITY_BIT_ODD 0x01`
Бит 1, если нечет
- `#define UART_PARITY_BIT_SPACE 0x02`
Бит четности всегда 0.
- `#define UART_PARITY_BIT_MARK 0x03`

- Бит четности всегда 1.
- `#define UART_PARITY_BIT_USE 0x04`
Бит чётности не используется, если "0"; бит четности используется, если "1".
- `#define UART_STOP_BIT 0x08`
Если установлен, один стоповый бит; иначе - 2 стоповых бита

Флаг типа двигателя

См. также

`motor_settings_t::MotorType, get_motor_settings, set_motor_settings`

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

Флаги настроек энкодера

См. также

`accessories_settings_t::MBSettings, get_accessories_settings, set_accessories_settings`

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

Флаги настроек температурного датчика

См. также

`accessories_settings_t::LimitSwitchesSettings, get_accessories_settings, set_accessories_settings`

- `#define TS_TYPE_BITS 0x07`
Биты, отвечающие за тип температурного датчика.
- `#define TS_TYPE_UNKNOWN 0x00`
Неизвестный сенсор
- `#define TS_TYPE_THERMOCOUPLE 0x01`
Термопара

- `#define TS_TYPE_SEMICONDUCTOR 0x02`
Полупроводниковый температурный датчик
- `#define TS_AVAILABLE 0x08`
Если флаг установлен, то датчик температуры доступен
- `#define LS_ON_SW1_AVAILABLE 0x01`
Если флаг установлен, то концевик, подключенный к ножке SW1, доступен
- `#define LS_ON_SW2_AVAILABLE 0x02`
Если флаг установлен, то концевик, подключенный к ножке SW2, доступен
- `#define LS_SW1_ACTIVE_LOW 0x04`
Если флаг установлен, то концевик, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте
- `#define LS_SW2_ACTIVE_LOW 0x08`
Если флаг установлен, то концевик, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте
- `#define LS_SHORTED 0x10`
Если флаг установлен, то концевики закорочены.

Определения типов

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

Функции

Группа команд настройки контроллера

Функции для чтения/записи большинства настроек контроллера.

- `result_t XIMC_API set_feedback_settings (device_t id, const feedback_settings_t *feedback_settings)`
Запись настроек обратной связи.
- `result_t XIMC_API get_feedback_settings (device_t id, feedback_settings_t *feedback_settings)`
Чтение настроек обратной связи
- `result_t XIMC_API set_home_settings (device_t id, const home_settings_t *home_settings)`
Команда записи настроек для подхода в home position.
- `result_t XIMC_API set_home_settings_calb (device_t id, const home_settings_calb_t *home_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API get_home_settings (device_t id, home_settings_t *home_settings)`
Команда чтения настроек для подхода в home position.
- `result_t XIMC_API get_home_settings_calb (device_t id, home_settings_calb_t *home_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_move_settings (device_t id, const move_settings_t *move_settings)`

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

- `result_t XIMC_API set_move_settings_calb (device_t id, const move_settings_calb_t *move_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API get_move_settings (device_t id, move_settings_t *move_settings)`

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

- `result_t XIMC_API get_move_settings_calb (device_t id, move_settings_calb_t *move_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_engine_settings (device_t id, const engine_settings_t *engine_settings)`

Запись настроек мотора.

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

Чтение настроек мотора.

- `result_t XIMC_API get_engine_settings_calb (device_t id, engine_settings_calb_t *engine_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_entype_settings (device_t id, const entype_settings_t *entype_settings)`

Запись информации о типе мотора и типе силового драйвера.

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

Возвращает информацию о типе мотора и силового драйвера.

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

Команда записи параметров питания мотора.

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

Команда чтения параметров питания мотора.

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

Команда записи установок защит.

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

Команда записи установок защит.

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

Запись настроек границ и концевых выключателей.

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

Чтение настроек границ и концевых выключателей.

- `result_t XIMC_API get_edges_settings_calb (device_t id, edges_settings_calb_t *edges_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_pid_settings (device_t id, const pid_settings_t *pid_settings)`

Запись ПИД коэффициентов.

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

Чтение ПИД коэффициентов.

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

Запись настроек для входного импульса синхронизации.

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

Чтение настроек для входного импульса синхронизации.

- `result_t XIMC_API get_sync_in_settings_calb (device_t id, sync_in_settings_calb_t *sync_in_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_sync_out_settings (device_t id, const sync_out_settings_t *sync_out_settings)`

Запись настроек для выходного импульса синхронизации.

- `result_t XIMC_API set_sync_out_settings_calb (device_t id, const sync_out_settings_calb_t *sync_out_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API get_sync_out_settings (device_t id, sync_out_settings_t *sync_out_settings)`
Чтение настроек для выходного импульса синхронизации.
- `result_t XIMC_API get_sync_out_settings_calb (device_t id, sync_out_settings_calb_t *sync_out_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_extio_settings (device_t id, const extio_settings_t *extio_settings)`
Команда записи параметров настройки режимов внешнего ввода/вывода.
- `result_t XIMC_API get_extio_settings (device_t id, extio_settings_t *extio_settings)`
Команда чтения параметров настройки режимов внешнего ввода/вывода.
- `result_t XIMC_API set_brake_settings (device_t id, const brake_settings_t *brake_settings)`
Запись настроек управления тормозом.
- `result_t XIMC_API get_brake_settings (device_t id, brake_settings_t *brake_settings)`
Чтение настроек управления тормозом.
- `result_t XIMC_API set_control_settings (device_t id, const control_settings_t *control_settings)`
Запись настроек управления мотором.
- `result_t XIMC_API set_control_settings_calb (device_t id, const control_settings_calb_t *control_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API get_control_settings (device_t id, control_settings_t *control_settings)`
Чтение настроек управления мотором.
- `result_t XIMC_API get_control_settings_calb (device_t id, control_settings_calb_t *control_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_joystick_settings (device_t id, const joystick_settings_t *joystick_settings)`
Запись настроек джойстика.
- `result_t XIMC_API get_joystick_settings (device_t id, joystick_settings_t *joystick_settings)`
Чтение настроек джойстика.
- `result_t XIMC_API set_ctp_settings (device_t id, const ctp_settings_t *ctp_settings)`
Запись настроек контроля позиции(для шагового двигателя).
- `result_t XIMC_API get_ctp_settings (device_t id, ctp_settings_t *ctp_settings)`
Чтение настроек контроля позиции(для шагового двигателя).
- `result_t XIMC_API set_uart_settings (device_t id, const uart_settings_t *uart_settings)`
Команда записи настроек UART.
- `result_t XIMC_API get_uart_settings (device_t id, uart_settings_t *uart_settings)`
Команда чтения настроек UART.
- `result_t XIMC_API set_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.

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_add_sync_in_action \(device_t id, const command_add_sync_in_action_t *the_command_add_sync_in_action\)](#)
Это команда добавляет один элемент в буфер FIFO команд, выполняемых при получении входного импульса синхронизации.
- [result_t XIMC_API command_add_sync_in_action_calb \(device_t id, const command_add_sync_in_action_calb_t *the_command_add_sync_in_action_calb, const calibration_t *calibration\)](#)
- [result_t XIMC_API command_power_off \(device_t id\)](#)
Немедленное отключение питания двигателя вне зависимости от его состояния.
- [result_t XIMC_API command_move \(device_t id, int Position, int uPosition\)](#)
При получении команды "move" двигатель начинает перемещаться (если не используется режим "ТТЛСинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), к точке указанной в полях Position, uPosition.
- [result_t XIMC_API command_move_calb \(device_t id, float Position, const calibration_t *calibration\)](#)
- [result_t XIMC_API command_movr \(device_t id, int DeltaPosition, int uDeltaPosition\)](#)
При получении команды "movr" двигатель начинает смещаться (если не используется режим "ТТЛСинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), влево или вправо (зависит от знака DeltaPosition) на количество импульсов указанное в полях DeltaPosition, uDeltaPosition.
- [result_t XIMC_API command_movr_calb \(device_t id, float DeltaPosition, const calibration_t *calibration\)](#)
- [result_t XIMC_API command_home \(device_t id\)](#)
Поля скоростей знаковые.
- [result_t XIMC_API command_left \(device_t id\)](#)
При получении команды "left" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), влево.
- [result_t XIMC_API command_right \(device_t id\)](#)
При получении команды "right" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), вправо.
- [result_t XIMC_API command_loft \(device_t id\)](#)
При получении команды "loft" двигатель смещается из текущей точки на расстояние GENG:::-Antiplay, затем двигается в ту же точку.
- [result_t XIMC_API command_sstp \(device_t id\)](#)
Плавная остановка.
- [result_t XIMC_API get_position \(device_t id, get_position_t *the_get_position\)](#)
Считывает значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.
- [result_t XIMC_API get_position_calb \(device_t id, get_position_calb_t *the_get_position_calb, const calibration_t *calibration\)](#)
- [result_t XIMC_API set_position \(device_t id, const set_position_t *the_set_position\)](#)
Устанавливает произвольное значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.
- [result_t XIMC_API set_position_calb \(device_t id, const set_position_calb_t *the_set_position_calb, const calibration_t *calibration\)](#)
- [result_t XIMC_API command_zero \(device_t id\)](#)
Устанавливает текущую позицию и позицию в которую осуществляется движение по командам move и movr равными нулю для всех случаев, кроме движения к позиции назначения.

Группа команд сохранения и загрузки настроек

- [result_t XIMC_API command_save_settings \(device_t id\)](#)
При получении команды контроллер выполняет операцию сохранения текущих настроек во встроенную энергонезависимую память контроллера.

- `result_t XIMC_API command_read_settings (device_t id)`
Чтение всех настроек контроллера из flash памяти в оперативную, заменяя текущие настройки.
- `result_t XIMC_API command_save_robust_settings (device_t id)`
При получении команды контроллер выполняет операцию сохранения важных настроек (калибровочные коэффициенты и т.
- `result_t XIMC_API command_read_robust_settings (device_t id)`
Чтение важных настроек (калибровочные коэффициенты и т.
- `result_t XIMC_API command_eesave_settings (device_t id)`
Запись настроек контроллера в EEPROM память позиционера, которые непосредственно связаны с позиционером и не меняются без его механической переделки.
- `result_t XIMC_API command_eeread_settings (device_t id)`
Чтение настроек контроллера из EEPROM памяти позиционера, которые непосредственно связаны с позиционером и не меняются без его механической переделки.
- `result_t XIMC_API 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 об датчиках Холла в EEPROM.
- `result_t XIMC_API get_hallsensor_information (device_t id, hallsensor_information_t *hallsensor_information)`
Чтение информации об датчиках Холла из EEPROM.
- `result_t XIMC_API set_hallsensor_settings (device_t id, const hallsensor_settings_t *hallsensor_settings)`
Запись настроек датчиков Холла в EEPROM.
- `result_t XIMC_API get_hallsensor_settings (device_t id, hallsensor_settings_t *hallsensor_settings)`
Чтение настроек датчиков Холла из EEPROM.
- `result_t XIMC_API set_gear_information (device_t id, const gear_information_t *gear_information)`
Запись информации о редукторе в EEPROM.
- `result_t XIMC_API get_gear_information (device_t id, gear_information_t *gear_information)`
Чтение информации о редукторе из EEPROM.
- `result_t XIMC_API set_gear_settings (device_t id, const gear_settings_t *gear_settings)`
Запись настроек редуктора в EEPROM.
- `result_t XIMC_API get_gear_settings (device_t id, gear_settings_t *gear_settings)`
Чтение настроек редуктора из EEPROM.
- `result_t XIMC_API set_accessories_settings (device_t id, const accessories_settings_t *accessories_settings)`
Запись информации о дополнительных аксессуарах в EEPROM.
- `result_t XIMC_API get_accessories_settings (device_t id, accessories_settings_t *accessories_settings)`
Чтение информации о дополнительных аксессуарах из EEPROM.
- `result_t XIMC_API get_bootloader_version (device_t id, unsigned int *Major, unsigned int *Minor, unsigned int *Release)`
Чтение номера версии прошивки контроллера.
- `result_t XIMC_API get_init_random (device_t id, init_random_t *init_random)`
Чтение случайного числа из контроллера.
- `result_t XIMC_API command_change_motor (device_t id, const command_change_motor_t *the_command_change_motor)`
- `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 *name, uint8_t *ret)`
Проверка наличия прошивки в контроллере
- `result_t XIMC_API command_update_firmware (const char *name, const uint8_t *data, uint32_t data_size)`
Обновление прошивки

- `result_t XIMC_API write_key` (const char *name, uint8_t *key)
Запись ключа защиты. Функция используется только производителем.
- `result_t XIMC_API command_reset` (device_t id)
Перезагрузка контроллера.
- `result_t XIMC_API command_clear_fram` (device_t id)
Очистка FRAM памяти контроллера.

Управление устройством

Функции поиска и открытия/закрытия устройств

- `typedef char * pchar`
Не обращайтесь на меня внимание
- `typedef void(XIMC_CALLCONV * logging_callback_t)(int loglevel, const wchar_t *message, void *user_data)`
Прототип функции обратного вызова для логирования
- `device_t XIMC_API open_device` (const char *name)
Открывает устройство по имени name и возвращает идентификатор, который будет использоваться для обращения к устройству.
- `result_t XIMC_API close_device` (device_t *id)
Закрывает устройство
- `result_t XIMC_API probe_device` (const char *name)
Проверяет, является ли устройство с именем name XIMC-совместимым.
- `result_t XIMC_API set_bindy_key` (const char *keyfilepath)
Устанавливает ключ шифрования сетевой подсистемы (bindy).
- `device_enumeration_t XIMC_API enumerate_devices` (int enumerate_flags, const char *hints)
Перечисляет все XIMC-совместимые устройства.
- `result_t XIMC_API free_enumerate_devices` (device_enumeration_t device_enumeration)
Освобождает память, выделенную enumerate_devices.
- `int XIMC_API get_device_count` (device_enumeration_t device_enumeration)
Возвращает количество подключенных устройств.
- `pchar XIMC_API get_device_name` (device_enumeration_t device_enumeration, int device_index)
Возвращает имя подключенного устройства из перечисления устройств.
- `result_t XIMC_API get_enumerate_device_serial` (device_enumeration_t device_enumeration, int device_index, uint32_t *serial)
Возвращает серийный номер подключенного устройства из перечисления устройств.
- `result_t XIMC_API get_enumerate_device_information` (device_enumeration_t device_enumeration, int device_index, device_information_t *device_information)
Возвращает информацию о подключенном устройстве из перечисления устройств.
- `result_t XIMC_API get_enumerate_device_controller_name` (device_enumeration_t device_enumeration, int device_index, controller_name_t *controller_name)
Возвращает имя подключенного устройства из перечисления устройств.
- `result_t XIMC_API get_enumerate_device_stage_name` (device_enumeration_t device_enumeration, int device_index, stage_name_t *stage_name)
Возвращает имя подвижки для подключенного устройства из перечисления устройств.
- `result_t XIMC_API get_enumerate_device_network_information` (device_enumeration_t device_enumeration, int device_index, device_network_information_t *device_network_information)
Возвращает сетевую информацию о подключенном устройстве из перечисления устройств.
- `result_t XIMC_API reset_locks` ()
Снимает блокировку библиотеки в экстренном случае.
- `result_t XIMC_API ximc_fix_usbser_sys` (const char *device_name)

- Исправление ошибки драйвера USB в Windows.
- void `XIMC_API msec_sleep` (unsigned int msec)
 - Приостанавливает работу на указанное время
- void `XIMC_API ximc_version` (char *version)
 - Возвращает версию библиотеки
- void `XIMC_API logging_callback_stderr_wide` (int loglevel, const wchar_t *message, 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)
 - Состояние устройства в калиброванных единицах.
- `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.

5.1.1 Подробное описание

Заголовочный файл для библиотеки libximc.

5.1.2 Макросы

5.1.2.1 #define ALARM_ON_DRIVER_OVERHEATING 0x01

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

Иначе - игнорировать подступающий перегрев с драйвера.

5.1.2.2 #define BORDER_IS_ENCODER 0x01

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

Если флаг сброшен, границы определяются концевыми выключателями.

5.1.2.3 #define BORDER_STOP_LEFT 0x02

Если флаг установлен, мотор останавливается при достижении левой границы.

5.1.2.4 #define BORDER_STOP_RIGHT 0x04

Если флаг установлен, мотор останавливается при достижении правой границы.

5.1.2.5 `#define BORDERS_SWAP_MISSET_DETECTION 0x08`

Если флаг установлен, мотор останавливается при достижении обеих границ.
Нужен для предотвращения поломки двигателя при неправильных настройках концевиков

5.1.2.6 `#define BRAKE_ENABLED 0x01`

Управление тормозом включено, если флаг установлен.

5.1.2.7 `#define BRAKE_ENG_PWROFF 0x02`

Тормоз отключает питание шагового мотора, если флаг установлен.

5.1.2.8 `#define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04`

Левая кнопка нормально разомкнутая, если флаг установлен.

5.1.2.9 `#define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08`

Правая кнопка нормально разомкнутая, если флаг установлен.

5.1.2.10 `#define CONTROL_MODE_BITS 0x03`

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

5.1.2.11 `#define CONTROL_MODE_JOY 0x01`

Управление с помощью джойстика.

5.1.2.12 `#define CONTROL_MODE_LR 0x02`

Управление с помощью кнопок left/right.

5.1.2.13 `#define CONTROL_MODE_OFF 0x00`

Управление отключено.

5.1.2.14 `#define CTP_ALARM_ON_ERROR 0x04`

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

5.1.2.15 `#define CTP_BASE 0x02`

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

5.1.2.16 `#define CTP_ENABLED 0x01`

Контроль позиции включен, если флаг установлен.

5.1.2.17 `#define CTP_ERROR_CORRECTION 0x10`

Корректировать ошибки, возникающие при проскальзывании, если флаг установлен.

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

5.1.2.18 `#define DRIVER_TYPE_DISCRETE_FET 0x01`

Силовой драйвер на дискретных мосфет-ключках.

Используется по умолчанию.

5.1.2.19 `#define DRIVER_TYPE_EXTERNAL 0x03`

Внешний силовой драйвер.

5.1.2.20 `#define DRIVER_TYPE_INTEGRATE 0x02`

Силовой драйвер с использованием ключей, интегрированных в микросхему.

5.1.2.21 `#define EEPROM_PRECEDENCE 0x01`

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

5.1.2.22 `#define ENC_STATE_ABSENT 0x00`

Энкодер не подключен.

5.1.2.23 `#define ENC_STATE_MALFUNC 0x02`

Энкодер подключен и неисправен.

5.1.2.24 `#define ENC_STATE_OK 0x04`

Энкодер подключен и работает адекватно.

5.1.2.25 `#define ENC_STATE_REVERS 0x03`

Энкодер подключен и исправен, но считает в другую сторону.

5.1.2.26 `#define ENC_STATE_UNKNOWN 0x01`

Состояние энкодера неизвестно.

5.1.2.27 `#define ENDER_SW1_ACTIVE_LOW 0x02`

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

5.1.2.28 `#define ENDER_SW2_ACTIVE_LOW 0x04`

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

5.1.2.29 `#define ENDER_SWAP 0x01`

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

5.1.2.30 `#define ENGINE_ACCEL_ON 0x10`

Ускорение.

Если флаг установлен, движение происходит с ускорением.

5.1.2.31 `#define ENGINE_ANTIPLAY 0x08`

Компенсация люфта.

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

5.1.2.32 `#define ENGINE_CURRENT_AS_RMS 0x02`

Флаг интерпретации значения тока.

Если флаг установлен, то задаваемое значение тока интерпретируется как среднеквадратичное значение тока, если флаг снят, то задаваемое значение тока интерпретируется как максимальная амплитуда тока.

5.1.2.33 `#define ENGINE_LIMIT_CURR 0x40`

Номинальный ток мотора.

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

5.1.2.34 `#define ENGINE_LIMIT_RPM 0x80`

Номинальная частота вращения мотора.

Если флаг установлен, частота вращения ограничивается заданным номинальным значением.

5.1.2.35 `#define ENGINE_LIMIT_VOLT 0x20`

Номинальное напряжение мотора.

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

5.1.2.36 `#define ENGINE_MAX_SPEED 0x04`

Флаг максимальной скорости.

Если флаг установлен, движение происходит на максимальной скорости.

5.1.2.37 `#define ENGINE_REVERSE 0x01`

Флаг реверса.

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

5.1.2.38 `#define ENGINE_TYPE_2DC 0x02`

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

5.1.2.39 `#define ENGINE_TYPE_BRUSHLESS 0x05`

Безщеточный мотор.

5.1.2.40 `#define ENGINE_TYPE_DC 0x01`

Мотор постоянного тока.

5.1.2.41 `#define ENGINE_TYPE_NONE 0x00`

Это значение не нужно использовать.

5.1.2.42 `#define ENGINE_TYPE_STEP 0x03`

Шаговый мотор.

5.1.2.43 `#define ENGINE_TYPE_TEST 0x04`

Скважность в обмотках фиксирована.

Используется только производителем.

5.1.2.44 `#define ENUMERATE_PROBE 0x01`

Проверить, является ли устройство XIMC-совместимым.

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

5.1.2.45 `#define EXTIO_SETUP_INVERT 0x02`

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

5.1.2.46 `#define EXTIO_SETUP_MODE_IN_ALARM 0x05`

Войти в состояние ALARM при переходе сигнала в активное состояние.

5.1.2.47 `#define EXTIO_SETUP_MODE_IN_BITS 0x0F`

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

5.1.2.48 `#define EXTIO_SETUP_MODE_IN_HOME 0x04`

Выполняется команда HOME.

5.1.2.49 `#define EXTIO_SETUP_MODE_IN_MOVR 0x03`

Выполняется команда MOVR с последними настройками.

5.1.2.50 `#define EXTIO_SETUP_MODE_IN_NOP 0x00`

Ничего не делать.

5.1.2.51 `#define EXTIO_SETUP_MODE_IN_PWOF 0x02`

Выполняет команду PWOF, обесточивая обмотки двигателя.

5.1.2.52 `#define EXTIO_SETUP_MODE_IN_STOP 0x01`

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

5.1.2.53 `#define EXTIO_SETUP_MODE_OUT_ALARM 0x30`

Ножка находится в активном состоянии при нахождении в состоянии ALARM.

5.1.2.54 `#define EXTIO_SETUP_MODE_OUT_BITS 0xF0`

Биты выбора поведения на выходе.

5.1.2.55 `#define EXTIO_SETUP_MODE_OUT_MOTOR_FOUND 0x50`

Ножка находится в активном состоянии при обнаружении подключенного двигателя (первой обмотки).

5.1.2.56 `#define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40`

Ножка находится в активном состоянии при подаче питания на обмотки.

5.1.2.57 `#define EXTIO_SETUP_MODE_OUT_MOVING 0x20`

Ножка находится в активном состоянии при движении.

5.1.2.58 `#define EXTIO_SETUP_MODE_OUT_OFF 0x00`

Ножка всегда в неактивном состоянии.

5.1.2.59 `#define EXTIO_SETUP_MODE_OUT_ON 0x10`

Ножка всегда в активном состоянии.

5.1.2.60 `#define EXTIO_SETUP_OUTPUT 0x01`

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

5.1.2.61 `#define FEEDBACK_EMF 0x04`

Обратная связь по ЭДС.

5.1.2.62 `#define FEEDBACK_ENC_REVERSE 0x01`

Обратный счет у энкодера.

5.1.2.63 `#define FEEDBACK_ENC_TYPE_AUTO 0x00`

Определять тип энкодера автоматически.

5.1.2.64 `#define FEEDBACK_ENC_TYPE_BITS 0xC0`

Биты, отвечающие за тип энкодера.

5.1.2.65 `#define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80`

Дифференциальный энкодер.

5.1.2.66 `#define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40`

Недифференциальный энкодер.

5.1.2.67 `#define FEEDBACK_ENCODER 0x01`

Обратная связь с помощью энкодера.

5.1.2.68 `#define FEEDBACK_ENCODERHALL 0x03`

Обратная связь с помощью датчика Холла.

5.1.2.69 `#define FEEDBACK_HALL_REVERSE 0x02`

Обратный счёт позиции по датчикам Холла.

5.1.2.70 `#define FEEDBACK_NONE 0x05`

Обратная связь отсутствует.

5.1.2.71 `#define HOME_DIR_FIRST 0x001`

Определяет направление первоначального движения мотора после поступления команды HOME.

Если флаг установлен - вправо; иначе - влево.

5.1.2.72 `#define HOME_DIR_SECOND 0x002`

Определяет направление второго движения мотора.

Если флаг установлен - вправо; иначе - влево.

5.1.2.73 `#define HOME_HALF_MV 0x008`

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

5.1.2.74 `#define HOME_MV_SEC_EN 0x004`

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

5.1.2.75 `#define HOME_STOP_FIRST_BITS 0x030`

Биты, отвечающие за выбор сигнала завершения первого движения.

5.1.2.76 `#define HOME_STOP_FIRST_LIM 0x030`

Первое движение завершается по сигналу с концевика.

5.1.2.77 `#define HOME_STOP_FIRST_REV 0x010`

Первое движение завершается по сигналу с Revolution sensor.

5.1.2.78 `#define HOME_STOP_FIRST_SYN 0x020`

Первое движение завершается по сигналу со входа синхронизации.

5.1.2.79 `#define HOME_STOP_SECOND_BITS 0x0C0`

Биты, отвечающие за выбор сигнала завершения второго движения.

5.1.2.80 `#define HOME_STOP_SECOND_LIM 0x0C0`

Второе движение завершается по сигналу с концевика.

5.1.2.81 `#define HOME_STOP_SECOND_REV 0x040`

Второе движение завершается по сигналу с Revolution sensor.

5.1.2.82 `#define HOME_STOP_SECOND_SYN 0x080`

Второе движение завершается по сигналу со входа синхронизации.

5.1.2.83 `#define HOME_USE_FAST 0x100`

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

5.1.2.84 `#define JOY_REVERSE 0x01`

Реверс воздействия джойстика.

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

5.1.2.85 `#define LOW_UPWR_PROTECTION 0x02`

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

5.1.2.86 `#define LS_SHORTED 0x10`

Если флаг установлен, то концевики закорочены.

5.1.2.87 `#define MICROSTEP_MODE_FRAC_128 0x08`

Деление шага 1/128.

5.1.2.88 `#define MICROSTEP_MODE_FRAC_16 0x05`

Деление шага 1/16.

5.1.2.89 `#define MICROSTEP_MODE_FRAC_2 0x02`

Деление шага 1/2.

5.1.2.90 `#define MICROSTEP_MODE_FRAC_256 0x09`

Деление шага 1/256.

5.1.2.91 `#define MICROSTEP_MODE_FRAC_32 0x06`

Деление шага 1/32.

5.1.2.92 `#define MICROSTEP_MODE_FRAC_4 0x03`

Деление шага 1/4.

5.1.2.93 `#define MICROSTEP_MODE_FRAC_64 0x07`

Деление шага 1/64.

5.1.2.94 `#define MICROSTEP_MODE_FRAC_8 0x04`

Деление шага 1/8.

5.1.2.95 `#define MICROSTEP_MODE_FULL 0x01`

Полношаговый режим.

5.1.2.96 `#define MOVE_STATE_ANTIPLAY 0x04`

Выполняется компенсация люфта, если флаг установлен.

5.1.2.97 `#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`.

5.1.2.98 `#define MOVE_STATE_TARGET_SPEED 0x02`

Флаг устанавливается при достижении заданной скорости.

5.1.2.99 `#define MVCMD_ERROR 0x40`

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

Имеет смысл если `MVCMD_RUNNING` указывает на завершение движения.

5.1.2.100 `#define MVCMD_HOME 0x06`

Команда home.

5.1.2.101 `#define MVCMD_LEFT 0x03`

Команда left.

5.1.2.102 `#define MVCMD_LOFT 0x07`

Команда loft.

5.1.2.103 `#define MVCMD_MOVE 0x01`

Команда move.

5.1.2.104 `#define MVCMD_MOVR 0x02`

Команда movr.

5.1.2.105 `#define MVCMD_NAME_BITS 0x3F`

Битовая маска активной команды.

5.1.2.106 `#define MVCMD_RIGHT 0x04`

Команда rigt.

5.1.2.107 `#define MVCMD_RUNNING 0x80`

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

5.1.2.108 `#define MVCMD_SSTP 0x08`

Команда плавной остановки(SSTP).

5.1.2.109 `#define MVCMD_STOP 0x05`

Команда stop.

5.1.2.110 `#define MVCMD_UNKNWN 0x00`

Неизвестная команда.

5.1.2.111 `#define POWER_OFF_ENABLED 0x02`

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

Иначе - не снимать.

5.1.2.112 `#define POWER_REDUCT_ENABLED 0x01`

Если флаг установлен, уменьшить ток по прошествии CurrReductDelay.

Иначе - не уменьшать.

5.1.2.113 `#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.

5.1.2.114 `#define PWR_STATE_MAX 0x05`

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

5.1.2.115 `#define PWR_STATE_NORM 0x03`

Обмотки запитаны номинальным током.

5.1.2.116 `#define PWR_STATE_OFF 0x01`

Обмотки мотора разомкнуты и не управляются драйвером.

5.1.2.117 `#define PWR_STATE_REDUCT 0x04`

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

5.1.2.118 `#define PWR_STATE_UNKNOWN 0x00`

Неизвестное состояние, которое не должно никогда реализовываться.

5.1.2.119 `#define REV_SENS_INV 0x08`

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

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

5.1.2.120 `#define SETPOS_IGNORE_ENCODER 0x02`

Если установлен, то счётчик энкодера не обновляется.

5.1.2.121 `#define SETPOS_IGNORE_POSITION 0x01`

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

5.1.2.122 `#define STATE_ALARM 0x00040`

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

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

5.1.2.123 `#define STATE_BORDERS_SWAP_MISSET 0x08000`

Достижение неверной границы.

5.1.2.124 `#define STATE_BRAKE 0x0200`

Состояние вывода управления тормозом(флаг "1" - если на тормоз подаётся питание, "0" - если тормоз не запитан).

5.1.2.125 `#define STATE_BUTTON_LEFT 0x0008`

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

5.1.2.126 `#define STATE_BUTTON_RIGHT 0x0004`

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

5.1.2.127 `#define STATE_CONTR 0x0003F`

Флаги состояния контроллера.

5.1.2.128 `#define STATE_CONTROLLER_OVERHEAT 0x00200`

Перегрелась микросхема контроллера.

5.1.2.129 `#define STATE_CTP_ERROR 0x00080`

Контроль позиции нарушен(используется только с шаговым двигателем).

5.1.2.130 `#define STATE_CURRENT_MOTOR0 0x00000`

Мотор 0.

5.1.2.131 `#define STATE_CURRENT_MOTOR1 0x40000`

Мотор 1.

5.1.2.132 `#define STATE_CURRENT_MOTOR2 0x80000`

Мотор 2.

5.1.2.133 `#define STATE_CURRENT_MOTOR3 0xC0000`

Мотор 3.

5.1.2.134 `#define STATE_CURRENT_MOTOR_BITS 0xC0000`

Флаги выбора мотора.

5.1.2.135 `#define STATE_DIG_SIGNAL 0xFFFF`

Флаги цифровых сигналов.

5.1.2.136 `#define STATE_EEPROM_CONNECTED 0x00010`

Подключена память EEPROM с настройками.

5.1.2.137 `#define STATE_ENC_A 0x2000`

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

5.1.2.138 `#define STATE_ENC_B 0x4000`

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

5.1.2.139 `#define STATE_ERRC 0x00001`

Недопустимая команда.

5.1.2.140 `#define STATE_ERRD 0x00002`

Нарушение целостности данных.

5.1.2.141 `#define STATE_ERRV 0x00004`

Недопустимое значение данных.

5.1.2.142 `#define STATE_GPIO_LEVEL 0x0020`

Состояние ввода/вывода общего назначения.

5.1.2.143 `#define STATE_GPIO_PINOUT 0x0010`

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

5.1.2.144 `#define STATE_HALL_A 0x0040`

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

5.1.2.145 `#define STATE_HALL_B 0x0080`

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

5.1.2.146 `#define STATE_HALL_C 0x0100`

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

5.1.2.147 `#define STATE_LEFT_EDGE 0x0002`

Достижение левой границы.

5.1.2.148 `#define STATE_LOW_USB_VOLTAGE 0x02000`

Слишком низкое напряжение на USB.

5.1.2.149 `#define STATE_OVERLOAD_POWER_CURRENT 0x00800`

Превышен максимальный ток потребления силовой части.

5.1.2.150 `#define STATE_OVERLOAD_POWER_VOLTAGE 0x00400`

Превышено напряжение на силовой части.

5.1.2.151 `#define STATE_OVERLOAD_USB_CURRENT 0x04000`

Превышен максимальный ток потребления USB.

5.1.2.152 `#define STATE_OVERLOAD_USB_VOLTAGE 0x01000`

Превышено напряжение на USB.

5.1.2.153 `#define STATE_POWER_OVERHEAT 0x00100`

Перегрелась силовая часть платы.

5.1.2.154 `#define STATE_REV_SENSOR 0x0400`

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

5.1.2.155 `#define STATE_RIGHT_EDGE 0x0001`

Достижение правой границы.

5.1.2.156 `#define STATE_SECUR 0x3FFC0`

Флаги опасности.

5.1.2.157 `#define STATE_SYNC_INPUT 0x0800`

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

5.1.2.158 `#define STATE_SYNC_OUTPUT 0x1000`

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

5.1.2.159 `#define SYNCIN_ENABLED 0x01`

Включение необходимости импульса синхронизации для начала движения.

5.1.2.160 `#define SYNCIN_INVERT 0x02`

Если установлен - срабатывает по переходу из 1 в 0.

Иначе - из 0 в 1.

5.1.2.161 `#define SYNCOUT_ENABLED 0x01`

Синхронизация выхода работает согласно настройкам, если флаг установлен.

В ином случае значение выхода фиксировано и подчиняется SYNCOUT_STATE.

5.1.2.162 `#define SYNCOUT_IN_STEPS 0x08`

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

5.1.2.163 `#define SYNCOUT_INVERT 0x04`

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

5.1.2.164 `#define SYNCOUT_ONPERIOD 0x40`

Выдать импульс синхронизации после прохождения SyncOutPeriod отсчётов.

5.1.2.165 `#define SYNCOUT_ONSTART 0x10`

Генерация синхронизирующего импульса при начале движения.

5.1.2.166 `#define SYNCOUT_ONSTOP 0x20`

Генерация синхронизирующего импульса при остановке.

5.1.2.167 `#define SYNCOUT_STATE 0x02`

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

5.1.2.168 `#define TS_TYPE_BITS 0x07`

Биты, отвечающие за тип температурного датчика.

5.1.2.169 `#define UART_PARITY_BITS 0x03`

Биты, отвечающие за выбор четности.

5.1.2.170 `#define WIND_A_STATE_ABSENT 0x00`

Обмотка A не подключена.

5.1.2.171 `#define WIND_A_STATE_MALFUNC 0x02`

Короткое замыкание на обмотке A.

5.1.2.172 `#define WIND_A_STATE_OK 0x03`

Обмотка A работает адекватно.

5.1.2.173 `#define WIND_A_STATE_UNKNOWN 0x01`

Состояние обмотки A неизвестно.

5.1.2.174 `#define WIND_B_STATE_ABSENT 0x00`

Обмотка B не подключена.

5.1.2.175 `#define WIND_B_STATE_MALFUNC 0x20`

Короткое замыкание на обмотке B.

5.1.2.176 `#define WIND_B_STATE_OK 0x30`

Обмотка B работает адекватно.

5.1.2.177 `#define WIND_B_STATE_UNKNOWN 0x10`

Состояние обмотки B неизвестно.

5.1.2.178 #define XIMC_API

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

It automatically expands to dllimport on msvc when including header file

5.1.3 Типы

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

Прототип функции обратного вызова для логирования

Аргументы

loglevel	уровень логирования
message	сообщение

5.1.4 Функции

5.1.4.1 result_t XIMC_API close_device (device_t * id)

Закрывает устройство

Аргументы

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

5.1.4.2 result_t XIMC_API command_add_sync_in_action (device_t id, const command_add_sync_in_action_t * the_command_add_sync_in_action)

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

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

Аргументы

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

5.1.4.3 result_t XIMC_API command_clear_fram (device_t id)

Очистка FRAM памяти контроллера.

Функция используется только производителем.

Аргументы

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

5.1.4.4 `result_t XIMC_API command_eeread_settings (device_t id)`

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

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

Аргументы

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

5.1.4.5 `result_t XIMC_API command_eesave_settings (device_t id)`

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

Функция должна использоваться только производителем.

Аргументы

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

5.1.4.6 `result_t XIMC_API command_home (device_t id)`

Поля скоростей знаковые.

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

Аргументы

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

См. также

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

5.1.4.7 `result_t XIMC_API command_homezero (device_t id)`

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

Это удобный путь для калибровки нулевой позиции.

Аргументы

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

5.1.4.8 result_t XIMC_API command_left (device_t id)

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

Аргументы

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

5.1.4.9 result_t XIMC_API command_loft (device_t id)

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

Аргументы

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

5.1.4.10 result_t XIMC_API command_move (device_t id, int Position, int uPosition)

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

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

Аргументы

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

5.1.4.11 result_t XIMC_API command_movr (device_t id, int DeltaPosition, int uDeltaPosition)

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

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

Аргументы

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

5.1.4.12 `result_t XIMC_API command_power_off (device_t id)`

Немедленное отключение питания двигателя вне зависимости от его состояния.

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

Аргументы

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

См. также

[get_power_settings](#)
[set_power_settings](#)

5.1.4.13 `result_t XIMC_API command_read_robust_settings (device_t id)`

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

Аргументы

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

5.1.4.14 `result_t XIMC_API command_read_settings (device_t id)`

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

Аргументы

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

5.1.4.15 `result_t XIMC_API command_reset (device_t id)`

Перезагрузка контроллера.

Функция используется только производителем.

Аргументы

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

5.1.4.16 `result_t XIMC_API command_right (device_t id)`

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

Аргументы

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

5.1.4.17 `result_t XIMC_API command_save_robust_settings (device_t id)`

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

п.) во встроенную энергонезависимую память контроллера.

Аргументы

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

5.1.4.18 `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

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

5.1.4.19 `result_t XIMC_API command_sstp (device_t id)`

Плавная остановка.

Двигатель останавливается с ускорением замедления.

Аргументы

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

5.1.4.20 `result_t XIMC_API command_stop (device_t id)`

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

Аргументы

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

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

Обновление прошивки

Аргументы

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

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

Ожидание остановки контроллера

Аргументы

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

5.1.4.23 `result_t XIMC_API command_zero (device_t id)`

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

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

Аргументы

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

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

Перечисляет все XIMC-совместимые устройства.

Аргументы

in	enumerate_flags	флаги поиска устройств
in	hints	дополнительная информация для поиска <code>hints</code> это строка вида "ключ1=значение1\ключ2=значение2". Неизвестные пары ключ-значение игнорируются. Список ключей: <code>addr</code> - используется вместе с флагом <code>ENUMERATE_NETWORK</code> . Ненулевое значение это адрес или список адресов с перечислением через запятую удаленных хостов на которых происходит поиск устройств, отсутствующее значение это подключение посредством широковещательного запроса.

5.1.4.25 `result_t XIMC_API free_enumerate_devices (device_enumeration_t device_enumeration)`

Освобождает память, выделенную `enumerate_devices`.

Аргументы

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

5.1.4.26 result_t XIMC_API get_accessories_settings (device_t id, accessories_settings_t * accessories_settings)

Чтение информации о дополнительных аксессуарах из EEPROM.

Аргументы

	id	идентификатор устройства
out	accessories_ - settings	структура, содержащая информацию о дополнительных аксессуарах

5.1.4.27 result_t XIMC_API get_analog_data (device_t id, analog_data_t * analog_data)

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

Эта функция используется для тестирования и калибровки устройства.

Аргументы

	id	идентификатор устройства
out	analog_data	аналоговые данные

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

Чтение номера версии прошивки контроллера.

Аргументы

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

5.1.4.29 result_t XIMC_API get_brake_settings (device_t id, brake_settings_t * brake_settings)

Чтение настроек управления тормозом.

Аргументы

	id	идентификатор устройства
out	brake_settings	структура, содержащая настройки управления тормозом

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

Команда чтения калибровочных коэффициентов.

Эта функция заполняет структуру калибровочных коэффициентов.

См. также

[calibration_settings_t](#)

Аргументы

	id	идентификатор устройства
out	calibration_settings	калибровочные коэффициенты

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

Команда чтения состояния обмоток и других не часто используемых данных.

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

См. также

[chart_data_t](#)

Аргументы

	id	идентификатор устройства
out	chart_data	структура chart_data.

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

Чтение настроек управления мотором.

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

Аргументы

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

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

Чтение пользовательского имени контроллера и настроек из FRAM.

Аргументы

	id	идентификатор устройства
out	controller_name	структура, содержащая установленное пользовательское имя контроллера и флаги настроек

5.1.4.34 `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 (СТП_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 СТП_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 СТПMinError, устанавливается флаг STATE_CTP_ERROR. При управлении ШД с датчиком оборотов (СТП_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 СТПMinError устанавливается флаг STATE_CTP_ERROR.

Аргументы

	id	идентификатор устройства
out	ctp_settings	структура, содержащая настройки контроля позиции

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

Чтение данных из прошивки для отладки и поиска неисправностей.

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

Аргументы

	id	идентификатор устройства
out	debug_read	Данные для отладки.

5.1.4.36 `int XIMC_API get_device_count (device_enumeration_t device_enumeration)`

Возвращает количество подключенных устройств.

Аргументы

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

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

Возвращает информацию об устройстве.

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

Аргументы

	id	идентификатор устройства.
out	device_information	информация об устройстве Информация об устройстве.

См. также

[get_device_information](#)

5.1.4.38 pchar XIMC_API get_device_name (device_enumeration_t device_enumeration, int device_index)

Возвращает имя подключенного устройства из перечисления устройств.

Возвращает имя устройства с номером device_index.

Аргументы

in	device_enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	номер устройства

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

Чтение настроек границ и концевых выключателей.

См. также

[set_edges_settings](#)

Аргументы

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

5.1.4.40 result_t XIMC_API get_encoder_information (device_t id, encoder_information_t * encoder_information)

Чтение информации об энкодере из EEPROM.

Аргументы

	id	идентификатор устройства
out	encoder_information	структура, содержащая информацию об энкодере

5.1.4.41 result_t XIMC_API get_encoder_settings (device_t id, encoder_settings_t * encoder_settings)

Чтение настроек энкодера из EEPROM.

Аргументы

	id	идентификатор устройства
out	encoder_settings	структура, содержащая настройки энкодера

5.1.4.42 result_t XIMC_API get_engine_settings (device_t id, engine_settings_t * engine_settings)

Чтение настроек мотора.

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

См. также

[set_engine_settings](#)

Аргументы

	id	идентификатор устройства
out	engine_settings	структура с настройками мотора

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

Возвращает информацию о типе мотора и силового драйвера.

Аргументы

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

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

Возвращает имя подключенного устройства из перечисления устройств.

Возвращает имя устройства с номером device_index.

Аргументы

in	device_enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	номер устройства
out	controller	name имя устройства

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

Возвращает информацию о подключенном устройстве из перечисления устройств.

Возвращает информацию о устройстве с номером device_index.

Аргументы

in	device_enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	номер устройства
out	device_information	информация об устройстве

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

Возвращает сетевую информацию о подключенном устройстве из перечисления устройств.

Возвращает сетевую информацию о устройстве с номером `device_index`.

Аргументы

in	device_enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	номер устройства
out	device_network_information	сетевая информация об устройстве

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

Возвращает серийный номер подключенного устройства из перечисления устройств.

Возвращает серийный номер устройства с номером `device_index`.

Аргументы

in	device_enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	номер устройства
in	serial	серийный номер устройства

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

Возвращает имя подвижки для подключенного устройства из перечисления устройств.

Возвращает имя подвижки устройства с номером `device_index`.

Аргументы

in	device_enumeration	закрытый указатель на данные о перечисленных устройствах
in	device_index	номер устройства
out	stage	name имя подвижки

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

Команда чтения параметров настройки режимов внешнего ввода/вывода.

См. также

[set_extio_settings](#)

Аргументы

	id	идентификатор устройства
out	extio_settings	настройки EXTIO

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

Чтение настроек обратной связи

Аргументы

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

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

Чтение номера версии прошивки контроллера.

Аргументы

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

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

Чтение информации о редукторе из EEPROM.

Аргументы

	id	идентификатор устройства
out	gear_information	структура, содержащая информацию о редукторе

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

Чтение настроек редуктора из EEPROM.

Аргументы

	id	идентификатор устройства
out	gear_settings	структура, содержащая настройки редуктора

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

Чтение информации об датчиках Холла из EEPROM.

Аргументы

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

out	hallsensor_ information	структура, содержащая информацию об датчиках Холла
-----	----------------------------	--

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

Чтение настроек датчиков Холла из EEPROM.

Аргументы

	id	идентификатор устройства
out	hallsensor_ settings	структура, содержащая настройки датчиков Холла

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

Команда чтения настроек для подхода в home position.

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

См. также

[home_settings_t](#)

Аргументы

	id	идентификатор устройства
out	home_settings	настройки калибровки позиции

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

Чтение случайного числа из контроллера.

Аргументы

	id	идентификатор устройства
out	случайная	последовательность, сгенерированная контроллером

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

Чтение настроек джойстика.

При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость, а 100% отклонения соответствует MaxSpeed i , где $i=0$, если предыдущим использованием этого режима не было выбрано другое i . Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Расчёт DeadZone проиллюстрирован на графике: <!/attachments/download/5563/range25p.png>! Зависимость между отклонением и скоростью экспоненциальная, что позволяет без переключения режимов скорости сочетать высокую подвижность

и точность. На графике ниже показан пример экспоненциальной зависимости скорости и работы мертвой зоны. [!/attachments/download/3092/ExpJoystick.png!](http://attachments/download/3092/ExpJoystick.png) Параметр нелинейности можно менять. Нулевой параметр нелинейности соответствует линейной зависимости.

Аргументы

	id	идентификатор устройства
out	joystick_ settings	структура, содержащая настройки джойстика

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

Чтение информации о двигателе из EEPROM.

Аргументы

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

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

Чтение настроек двигателя из EEPROM.

Аргументы

	id	идентификатор устройства
out	motor_settings	структура, содержащая настройки двигателя

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

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

Аргументы

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

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

Чтение пользовательских данных из FRAM.

Аргументы

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

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

Чтение ПИД коэффициентов.

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

См. также

[set_pid_settings](#)

Аргументы

	id	идентификатор устройства
out	pid_settings	настройки ПИД

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

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

Аргументы

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

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

Команда чтения параметров питания мотора.

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

Аргументы

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

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

Команда записи установок защит.

Аргументы

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

См. также

`status_t::flags`

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

Чтение серийного номера контроллера.

Аргументы

	id	идентификатор устройства
out	SerialNumber	серийный номер контроллера

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

Чтение информации о позиционере из EEPROM.

Аргументы

	id	идентификатор устройства
out	stage_information	структура, содержащая информацию о позиционере

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

Чтение пользовательского имени подвижки из EEPROM.

Аргументы

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

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

Чтение настроек позиционера из EEPROM.

Аргументы

	id	идентификатор устройства
out	stage_settings	структура, содержащая настройки позиционера

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

Возвращает информацию о текущем состоянии устройства.

Аргументы

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

См. также

[get_status](#)

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

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](#)

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

Чтение настроек для входного импульса синхронизации.

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

См. также

[set_sync_in_settings](#)

Аргументы

	id	идентификатор устройства
out	sync_in_settings	настройки синхронизации

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

Чтение настроек для выходного импульса синхронизации.

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

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

Команда чтения настроек UART.

Эта функция заполняет структуру настроек UART.

См. также

[uart_settings_t](#)

Аргументы

	Speed	Скорость UART
out	uart_settings	настройки UART

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

Перезагрузка в прошивку в контроллере

Аргументы

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

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

Проверка наличия прошивки в контроллере

Аргументы

	name	имя устройства
out	ret	ноль, если прошивка присутствует

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

Простая функция логирования на stderr в узких (однобайтных) символах

Аргументы

loglevel	уровень логирования
message	сообщение

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

Простая функция логирования на stderr в широких символах

Аргументы

loglevel	уровень логирования
message	сообщение

5.1.4.80 `void XIMC_API msec_sleep (unsigned int msec)`

Приостанавливает работу на указанное время

Аргументы

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

5.1.4.81 device_t XIMC_API open_device (const char * name)

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

Аргументы

in	name	- имя устройства Имя устройства имеет вид "xi-com:port" или "xi-net://host/serial" или "xi-emu://file". Для USB-COM устройства "port" это имя устройства в ОС. Например "xi-com:\\.\\COM3" в Windows или "xi-com:/dev/tty.s123" в Linux/Mac. Для сетевого устройства "host" это IPv4 адрес или полностью определённое имя домена, "serial" это серийный номер устройства в шестнадцатеричной системе. Например "xi-net://192.168.0.1/00001234" или "xi-net://hostname.com/89ABCDEF". Для виртуального устройства "file" это путь к файлу с сохранённым состоянием устройства. Если файл не существует, он будет создан и инициализирован значениями по умолчанию. Например "xi-emu://C:\\dir\\file.bin" в Windows или "xi-emu:///home/user/file.bin" в Linux/Mac.
----	------	--

5.1.4.82 result_t XIMC_API probe_device (const char * name)

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

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

Аргументы

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

5.1.4.83 result_t XIMC_API service_command_updf (device_t id)

Команда переводит контроллер в режим обновления прошивки.

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

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

Запись информации о дополнительных аксессуарах в EEPROM.

Функция должна использоваться только производителем.

Аргументы

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

5.1.4.85 result_t XIMC_API set_bindy_key (const char * keyfilepath)

Устанавливает ключ шифрования сетевой подсистемы (bindy).

Аргументы

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

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

Запись настроек управления тормозом.

Аргументы

	id	идентификатор устройства
in	brake_settings	структура, содержащая настройки управления тормозом

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

Команда записи калибровочных коэффициентов.

Эта функция записывает структуру калибровочных коэффициентов в память контроллера.

См. также

[calibration_settings_t](#)

Аргументы

	id	идентификатор устройства
in	calibration_settings	калибровочные коэффициенты

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

Запись настроек управления мотором.

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

Аргументы

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

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

Запись пользовательского имени контроллера и настроек в FRAM.

Аргументы

	id	идентификатор устройства
in	controller_information	структура, содержащая информацию о контроллере

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

Запись настроек контроля позиции(для шагового двигателя).

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

Аргументы

	id	идентификатор устройства
in	ctp_settings	структура, содержащая настройки контроля позиции

5.1.4.91 `result_t XIMC_API set_debug_write (device_t id, const debug_write_t * debug_write)`

Запись данных в прошивку для отладки и поиска неисправностей.

Аргументы

	id	идентификатор устройства
in	debug_write	Данные для отладки.

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

Запись настроек границ и концевых выключателей.

См. также

[get_edges_settings](#)

Аргументы

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

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

Запись информации об энкодере в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	encoder_information	структура, содержащая информацию об энкодере

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

Запись настроек энкодера в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	encoder_settings	структура, содержащая настройки энкодера

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

Запись настроек мотора.

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

См. также

[get_engine_settings](#)

Аргументы

	id	идентификатор устройства
in	engine_settings	структура с настройками мотора

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

Запись информации о типе мотора и типе силового драйвера.

Аргументы

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

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

Команда записи параметров настройки режимов внешнего ввода/вывода.

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

См. также

[get_extio_settings](#)

Аргументы

	id	идентификатор устройства
in	extio_settings	настройки ЕХТЮ

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

Запись настроек обратной связи.

Аргументы

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

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

Запись информации о редукторе в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	gear_information	структура, содержащая информацию о редукторе

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

Запись настроек редуктора в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	gear_settings	структура, содержащая настройки редуктора

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

Запись информации об датчиках Холла в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	hallsensor_ - information	структура, содержащая информацию об датчиках Холла

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

Запись настроек датчиков Холла в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	hallsensor_ - settings	структура, содержащая настройки датчиков Холла

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

Команда записи настроек для подхода в home position.

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

См. также

[home_settings_t](#)


Аргументы

	id	идентификатор устройства
out	home_settings	настройки калибровки позиции

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

Запись настроек джойстика.

При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость, а 100% отклонения соответствует MaxSpeed i , где $i=0$, если предыдущим использованием этого режима не было выбрано другое i . Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Расчёт DeadZone проиллюстрирован на графике: <!/attachments/download/5563/range25p.png>! Зависимость между отклонением и скоростью экспо-

ненциальная, что позволяет без переключения режимов скорости сочетать высокую подвижность и точность. На графике ниже показан пример экспоненциальной зависимости скорости и работы мертвой зоны. ! Параметр нелинейности можно менять. Нулевой параметр нелинейности соответствует линейной зависимости.

Аргументы

	id	идентификатор устройства
in	joystick_settings	структура, содержащая настройки джойстика

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

Устанавливает функцию обратного вызова для логирования.

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

Аргументы

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

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

Запись информации о двигателе в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	motor_information	структура, содержащая информацию о двигателе

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

Запись настроек двигателя в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	motor_settings	структура, содержащая настройки двигателя

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

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

Аргументы

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

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

Запись пользовательских данных во FRAM.

Аргументы

	id	идентификатор устройства
in	nonvolatile_memory	структура, содержащая установленные пользовательские данные

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

Запись ПИД коэффициентов.

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

См. также

[get_pid_settings](#)

Аргументы

	id	идентификатор устройства
in	pid_settings	настройки ПИД

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

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

То есть меняется основной показатель положения.

Аргументы

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

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

Команда записи параметров питания мотора.

Используется только с шаговым двигателем.

Аргументы

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

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

Команда записи установок зашит.

Аргументы

	id	идентификатор устройства
	secure_settings	структура с настройками критических значений

См. также

`status_t::flags`

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

Запись серийного номера и версии железа во flash память контроллера.

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

Аргументы

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

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

Запись информации о позиционере в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	stage_information	структура, содержащая информацию о позиционере

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

Запись пользовательского имени подвижки в EEPROM.

Аргументы

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

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

Запись настроек позиционера в EEPROM.

Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	stage_settings	структура, содержащая настройки позиционера

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

Запись настроек для входного импульса синхронизации.

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

См. также

[get_sync_in_settings](#)

Аргументы

	id	идентификатор устройства
in	sync_in_settings	настройки синхронизации

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

Запись настроек для выходного импульса синхронизации.

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

См. также

[get_sync_in_settings](#)

Аргументы

	id	идентификатор устройства
in	sync_in_settings	настройки синхронизации

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

Команда записи настроек UART.

Эта функция записывает структуру настроек UART в память контроллера.

См. также

[uart_settings_t](#)

Аргументы

	Speed	Скорость UART
in	uart_settings	настройки UART

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

Запись ключа защиты Функция используется только производителем.

Аргументы

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

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

Исправление ошибки драйвера USB в Windows.

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

5.1.4.123 `void XIMC_API ximc_version (char * version)`

Возвращает версию библиотеки

Аргументы

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

Предметный указатель

- A1Voltage
 - analog_data_t, [12](#)
- A1Voltage_ADC
 - analog_data_t, [12](#)
- A2Voltage
 - analog_data_t, [12](#)
- A2Voltage_ADC
 - analog_data_t, [12](#)
- ACurrent
 - analog_data_t, [12](#)
- ACurrent_ADC
 - analog_data_t, [12](#)
- Accel
 - move_settings_calb_t, [47](#)
 - move_settings_t, [48](#)
- accessories_settings_t, [9](#)
 - LimitSwitchesSettings, [10](#)
 - MBRatedCurrent, [10](#)
 - MBRatedVoltage, [10](#)
 - MBSettings, [10](#)
 - MBTorque, [10](#)
 - MagneticBrakeInfo, [10](#)
 - TSGrad, [10](#)
 - TSMMax, [10](#)
 - TSMMin, [10](#)
 - TSSettings, [10](#)
 - TemperatureSensorInfo, [10](#)
- Accuracy
 - sync_out_settings_calb_t, [64](#)
 - sync_out_settings_t, [65](#)
- analog_data_t, [11](#)
 - A1Voltage, [12](#)
 - A1Voltage_ADC, [12](#)
 - A2Voltage, [12](#)
 - A2Voltage_ADC, [12](#)
 - ACurrent, [12](#)
 - ACurrent_ADC, [12](#)
 - B1Voltage, [12](#)
 - B1Voltage_ADC, [13](#)
 - B2Voltage, [13](#)
 - B2Voltage_ADC, [13](#)
 - BCurrent, [13](#)
 - BCurrent_ADC, [13](#)
 - FullCurrent, [13](#)
 - FullCurrent_ADC, [13](#)
 - Joy, [13](#)
 - Joy_ADC, [13](#)
 - L5_ADC, [13](#)
 - Pot, [13](#)
 - SupVoltage, [13](#)
 - SupVoltage_ADC, [14](#)
 - Temp, [14](#)
 - Temp_ADC, [14](#)
- Antiplay
 - engine_settings_calb_t, [29](#)
 - engine_settings_t, [31](#)
- AntiplaySpeed
 - move_settings_calb_t, [47](#)
 - move_settings_t, [48](#)
- B1Voltage
 - analog_data_t, [12](#)
- B1Voltage_ADC
 - analog_data_t, [13](#)
- B2Voltage
 - analog_data_t, [13](#)
- B2Voltage_ADC
 - analog_data_t, [13](#)
- BCurrent
 - analog_data_t, [13](#)
- BCurrent_ADC
 - analog_data_t, [13](#)
- BORDER_IS_ENCODER
 - ximc.h, [90](#)
- BORDER_STOP_LEFT
 - ximc.h, [90](#)
- BORDER_STOP_RIGHT
 - ximc.h, [90](#)
- BRAKE_ENABLED
 - ximc.h, [91](#)
- BRAKE_ENG_PWROFF
 - ximc.h, [91](#)
- BorderFlags
 - edges_settings_calb_t, [25](#)
 - edges_settings_t, [26](#)
- brake_settings_t, [14](#)
 - BrakeFlags, [14](#)
 - t1, [14](#)
 - t2, [15](#)
 - t3, [15](#)
 - t4, [15](#)
- BrakeFlags
 - brake_settings_t, [14](#)
- CONTROL_MODE_BITS
 - ximc.h, [91](#)
- CONTROL_MODE_JOY
 - ximc.h, [91](#)

- CONTROL_MODE_LR
 - ximc.h, [91](#)
- CONTROL_MODE_OFF
 - ximc.h, [91](#)
- CSS1_A
 - calibration_settings_t, [15](#)
- CSS1_B
 - calibration_settings_t, [15](#)
- CSS2_A
 - calibration_settings_t, [16](#)
- CSS2_B
 - calibration_settings_t, [16](#)
- CTP_ALARM_ON_ERROR
 - ximc.h, [91](#)
- CTP_BASE
 - ximc.h, [91](#)
- CTP_ENABLED
 - ximc.h, [91](#)
- CTP_ERROR_CORRECTION
 - ximc.h, [91](#)
- CTPFlags
 - ctp_settings_t, [22](#)
- CTPMinError
 - ctp_settings_t, [22](#)
- calibration_settings_t, [15](#)
 - CSS1_A, [15](#)
 - CSS1_B, [15](#)
 - CSS2_A, [16](#)
 - CSS2_B, [16](#)
 - FullCurrent_A, [16](#)
 - FullCurrent_B, [16](#)
- calibration_t, [16](#)
- chart_data_t, [16](#)
 - DutyCycle, [17](#)
 - Joy, [17](#)
 - Pot, [17](#)
 - WindingCurrentA, [17](#)
 - WindingCurrentB, [17](#)
 - WindingCurrentC, [17](#)
 - WindingVoltageA, [18](#)
 - WindingVoltageB, [18](#)
 - WindingVoltageC, [18](#)
- close_device
 - ximc.h, [106](#)
- ClutterTime
 - sync_in_settings_calb_t, [62](#)
 - sync_in_settings_t, [63](#)
- CmdBufFreeSpace
 - status_calb_t, [58](#)
 - status_t, [60](#)
- command_add_sync_in_action
 - ximc.h, [106](#)
- command_add_sync_in_action_calb_t, [18](#)
 - Position, [18](#)
 - Time, [18](#)
- command_add_sync_in_action_t, [18](#)
 - Time, [19](#)
 - uPosition, [19](#)
- command_change_motor_t, [19](#)
- command_clear_fram
 - ximc.h, [106](#)
- command_eeread_settings
 - ximc.h, [106](#)
- command_eesave_settings
 - ximc.h, [107](#)
- command_home
 - ximc.h, [107](#)
- command_homezero
 - ximc.h, [107](#)
- command_left
 - ximc.h, [108](#)
- command_loft
 - ximc.h, [108](#)
- command_move
 - ximc.h, [108](#)
- command_movr
 - ximc.h, [108](#)
- command_power_off
 - ximc.h, [109](#)
- command_read_robust_settings
 - ximc.h, [109](#)
- command_read_settings
 - ximc.h, [109](#)
- command_reset
 - ximc.h, [109](#)
- command_right
 - ximc.h, [109](#)
- command_save_robust_settings
 - ximc.h, [109](#)
- command_save_settings
 - ximc.h, [110](#)
- command_sstp
 - ximc.h, [110](#)
- command_stop
 - ximc.h, [110](#)
- command_update_firmware
 - ximc.h, [110](#)
- command_wait_for_stop
 - ximc.h, [110](#)
- command_zero
 - ximc.h, [111](#)
- control_settings_calb_t, [19](#)
 - Flags, [19](#)
 - MaxClickTime, [19](#)
 - MaxSpeed, [19](#)
 - Timeout, [20](#)
- control_settings_t, [20](#)
 - Flags, [21](#)
 - MaxClickTime, [21](#)
 - MaxSpeed, [21](#)
 - Timeout, [21](#)
 - uDeltaPosition, [21](#)
 - uMaxSpeed, [21](#)
- controller_name_t, [21](#)
 - ControllerName, [22](#)
 - CtrlFlags, [22](#)

- ControllerName
 - controller_name_t, 22
- CriticalIpwr
 - secure_settings_t, 51
- CriticalIusb
 - secure_settings_t, 51
- CriticalUpwr
 - secure_settings_t, 51
- CriticalUusb
 - secure_settings_t, 51
- ctp_settings_t, 22
 - CTPFlags, 22
 - CTPMinError, 22
- CtrlFlags
 - controller_name_t, 22
- CurPosition
 - status_calb_t, 58
 - status_t, 60
- CurSpeed
 - status_calb_t, 58
 - status_t, 60
- CurT
 - status_calb_t, 58
 - status_t, 60
- CurrReductDelay
 - power_settings_t, 50
- CurrentSetTime
 - power_settings_t, 50
- DRIVER_TYPE_EXTERNAL
 - ximc.h, 92
- DeadZone
 - joystick_settings_t, 42
- debug_read_t, 23
 - DebugData, 23
- debug_write_t, 23
 - DebugData, 24
- DebugData
 - debug_read_t, 23
 - debug_write_t, 24
- Decel
 - move_settings_calb_t, 47
 - move_settings_t, 48
- DetentTorque
 - motor_settings_t, 44
- device_information_t, 24
 - Major, 24
 - Minor, 24
 - Release, 24
- device_network_information_t, 25
- DriverType
 - entype_settings_t, 32
- DutyCycle
 - chart_data_t, 17
- EEPROM_PRECEDENCE
 - ximc.h, 92
- ENC_STATE_ABSENT
 - ximc.h, 92
- ENC_STATE_MALFUNC
 - ximc.h, 92
- ENC_STATE_OK
 - ximc.h, 92
- ENC_STATE_REVERS
 - ximc.h, 92
- ENC_STATE_UNKNOWN
 - ximc.h, 92
- ENDER_SW1_ACTIVE_LOW
 - ximc.h, 92
- ENDER_SW2_ACTIVE_LOW
 - ximc.h, 92
- ENDER_SWAP
 - ximc.h, 92
- ENGINE_ACCEL_ON
 - ximc.h, 93
- ENGINE_ANTIPLAY
 - ximc.h, 93
- ENGINE_LIMIT_CURR
 - ximc.h, 93
- ENGINE_LIMIT_RPM
 - ximc.h, 93
- ENGINE_LIMIT_VOLT
 - ximc.h, 93
- ENGINE_MAX_SPEED
 - ximc.h, 93
- ENGINE_REVERSE
 - ximc.h, 93
- ENGINE_TYPE_2DC
 - ximc.h, 94
- ENGINE_TYPE_DC
 - ximc.h, 94
- ENGINE_TYPE_NONE
 - ximc.h, 94
- ENGINE_TYPE_STEP
 - ximc.h, 94
- ENGINE_TYPE_TEST
 - ximc.h, 94
- ENUMERATE_PROBE
 - ximc.h, 94
- EXTIO_SETUP_INVERT
 - ximc.h, 94
- EXTIO_SETUP_OUTPUT
 - ximc.h, 95
- EXTIOModeFlags
 - extio_settings_t, 33
- EXTIOSetupFlags
 - extio_settings_t, 33
- edges_settings_calb_t, 25
 - BorderFlags, 25
 - EnderFlags, 25
 - LeftBorder, 25
 - RightBorder, 25
- edges_settings_t, 26
 - BorderFlags, 26
 - EnderFlags, 26
 - LeftBorder, 26
 - RightBorder, 26

- uLeftBorder, 27
- uRightBorder, 27
- Efficiency
 - gear_settings_t, 35
- EncPosition
 - get_position_calb_t, 36
 - get_position_t, 37
 - set_position_calb_t, 53
 - set_position_t, 54
 - status_calb_t, 58
 - status_t, 61
- EncSts
 - status_calb_t, 58
 - status_t, 61
- encoder_information_t, 27
 - Manufacturer, 27
 - PartNumber, 27
- encoder_settings_t, 27
 - EncoderSettings, 28
 - MaxCurrentConsumption, 28
 - MaxOperatingFrequency, 28
 - SupplyVoltageMax, 28
 - SupplyVoltageMin, 28
- EncoderSettings
 - encoder_settings_t, 28
- EnderFlags
 - edges_settings_calb_t, 25
 - edges_settings_t, 26
- engine_settings_calb_t, 29
 - Antiplay, 29
 - EngineFlags, 29
 - MicrostepMode, 29
 - NomCurrent, 29
 - NomSpeed, 29
 - NomVoltage, 29
 - StepsPerRev, 30
- engine_settings_t, 30
 - Antiplay, 31
 - EngineFlags, 31
 - MicrostepMode, 31
 - NomCurrent, 31
 - NomSpeed, 31
 - NomVoltage, 31
 - StepsPerRev, 31
 - uNomSpeed, 31
- EngineFlags
 - engine_settings_calb_t, 29
 - engine_settings_t, 31
- EngineType
 - entype_settings_t, 32
- entype_settings_t, 31
 - DriverType, 32
 - EngineType, 32
- enumerate_devices
 - ximc.h, 111
- ExpFactor
 - joystick_settings_t, 42
- extio_settings_t, 32
 - EXTIOModeFlags, 33
 - EXTIOSetupFlags, 33
- FEEDBACK_EMF
 - ximc.h, 96
- FEEDBACK_ENC_REVERSE
 - ximc.h, 96
- FEEDBACK_ENCODER
 - ximc.h, 96
- FEEDBACK_ENCODERHALL
 - ximc.h, 96
- FEEDBACK_NONE
 - ximc.h, 96
- FastHome
 - home_settings_calb_t, 39
 - home_settings_t, 40
- feedback_settings_t, 33
 - FeedbackFlags, 33
 - FeedbackType, 33
 - HallSPR, 34
 - HallShift, 33
- FeedbackFlags
 - feedback_settings_t, 33
- FeedbackType
 - feedback_settings_t, 33
- Flags
 - control_settings_calb_t, 19
 - control_settings_t, 21
 - secure_settings_t, 52
 - status_calb_t, 58
 - status_t, 61
- free_enumerate_devices
 - ximc.h, 111
- FullCurrent
 - analog_data_t, 13
- FullCurrent_A
 - calibration_settings_t, 16
- FullCurrent_ADC
 - analog_data_t, 13
- FullCurrent_B
 - calibration_settings_t, 16
- GPIOFlags
 - status_calb_t, 58
 - status_t, 61
- gear_information_t, 34
 - Manufacturer, 34
 - PartNumber, 34
- gear_settings_t, 34
 - Efficiency, 35
 - InputInertia, 35
 - MaxOutputBacklash, 35
 - RatedInputSpeed, 35
 - RatedInputTorque, 35
 - ReductionIn, 36
 - ReductionOut, 36
- get_accessories_settings
 - ximc.h, 112
- get_analog_data

- ximc.h, [112](#)
- get_bootloader_version
 - ximc.h, [112](#)
- get_brake_settings
 - ximc.h, [112](#)
- get_calibration_settings
 - ximc.h, [112](#)
- get_chart_data
 - ximc.h, [113](#)
- get_control_settings
 - ximc.h, [113](#)
- get_controller_name
 - ximc.h, [113](#)
- get_ctp_settings
 - ximc.h, [113](#)
- get_debug_read
 - ximc.h, [114](#)
- get_device_count
 - ximc.h, [114](#)
- get_device_information
 - ximc.h, [114](#)
- get_device_name
 - ximc.h, [114](#)
- get_edges_settings
 - ximc.h, [115](#)
- get_encoder_information
 - ximc.h, [115](#)
- get_encoder_settings
 - ximc.h, [115](#)
- get_engine_settings
 - ximc.h, [115](#)
- get_entype_settings
 - ximc.h, [116](#)
- get_enumerate_device_controller_name
 - ximc.h, [116](#)
- get_enumerate_device_information
 - ximc.h, [116](#)
- get_enumerate_device_network_information
 - ximc.h, [116](#)
- get_enumerate_device_serial
 - ximc.h, [117](#)
- get_enumerate_device_stage_name
 - ximc.h, [117](#)
- get_extio_settings
 - ximc.h, [117](#)
- get_feedback_settings
 - ximc.h, [118](#)
- get_firmware_version
 - ximc.h, [118](#)
- get_gear_information
 - ximc.h, [118](#)
- get_gear_settings
 - ximc.h, [118](#)
- get_hallsensor_information
 - ximc.h, [118](#)
- get_hallsensor_settings
 - ximc.h, [119](#)
- get_home_settings
 - ximc.h, [119](#)
- get_init_random
 - ximc.h, [119](#)
- get_joystick_settings
 - ximc.h, [119](#)
- get_motor_information
 - ximc.h, [120](#)
- get_motor_settings
 - ximc.h, [120](#)
- get_move_settings
 - ximc.h, [120](#)
- get_nonvolatile_memory
 - ximc.h, [120](#)
- get_pid_settings
 - ximc.h, [120](#)
- get_position
 - ximc.h, [121](#)
- get_position_calb_t, [36](#)
 - EncPosition, [36](#)
 - Position, [36](#)
- get_position_t, [36](#)
 - EncPosition, [37](#)
 - uPosition, [37](#)
- get_power_settings
 - ximc.h, [121](#)
- get_secure_settings
 - ximc.h, [121](#)
- get_serial_number
 - ximc.h, [121](#)
- get_stage_information
 - ximc.h, [122](#)
- get_stage_name
 - ximc.h, [122](#)
- get_stage_settings
 - ximc.h, [122](#)
- get_status
 - ximc.h, [122](#)
- get_status_calb
 - ximc.h, [123](#)
- get_sync_in_settings
 - ximc.h, [123](#)
- get_sync_out_settings
 - ximc.h, [123](#)
- get_uart_settings
 - ximc.h, [123](#)
- goto_firmware
 - ximc.h, [124](#)
- HOME_DIR_FIRST
 - ximc.h, [96](#)
- HOME_DIR_SECOND
 - ximc.h, [96](#)
- HOME_HALF_MV
 - ximc.h, [97](#)
- HOME_MV_SEC_EN
 - ximc.h, [97](#)
- HOME_STOP_FIRST_LIM
 - ximc.h, [97](#)
- HOME_STOP_FIRST_REV

- ximc.h, 97
- HOME_STOP_FIRST_SYN
 - ximc.h, 97
- HOME_USE_FAST
 - ximc.h, 97
- HallSPR
 - feedback_settings_t, 34
- HallShift
 - feedback_settings_t, 33
- hallsensor_information_t, 37
 - Manufacturer, 37
 - PartNumber, 37
- hallsensor_settings_t, 38
 - MaxCurrentConsumption, 38
 - MaxOperatingFrequency, 38
 - SupplyVoltageMax, 38
 - SupplyVoltageMin, 38
- has_firmware
 - ximc.h, 124
- HoldCurrent
 - power_settings_t, 50
- home_settings_calb_t, 39
 - FastHome, 39
 - HomeDelta, 39
 - HomeFlags, 39
 - SlowHome, 39
- home_settings_t, 39
 - FastHome, 40
 - HomeDelta, 40
 - HomeFlags, 40
 - SlowHome, 40
 - uFastHome, 40
 - uHomeDelta, 40
 - uSlowHome, 40
- HomeDelta
 - home_settings_calb_t, 39
 - home_settings_t, 40
- HomeFlags
 - home_settings_calb_t, 39
 - home_settings_t, 40
- HorizontalLoadCapacity
 - stage_settings_t, 56
- init_random_t, 41
 - key, 41
- InputInertia
 - gear_settings_t, 35
- Ipwr
 - status_calb_t, 58
 - status_t, 61
- Iusb
 - status_calb_t, 58
 - status_t, 61
- JOY_REVERSE
 - ximc.h, 97
- Joy
 - analog_data_t, 13
 - chart_data_t, 17
- Joy_ADC
 - analog_data_t, 13
- JoyCenter
 - joystick_settings_t, 42
- JoyFlags
 - joystick_settings_t, 42
- JoyHighEnd
 - joystick_settings_t, 42
- JoyLowEnd
 - joystick_settings_t, 42
- joystick_settings_t, 41
 - DeadZone, 42
 - ExpFactor, 42
 - JoyCenter, 42
 - JoyFlags, 42
 - JoyHighEnd, 42
 - JoyLowEnd, 42
- Key
 - serial_number_t, 52
- key
 - init_random_t, 41
- L5_ADC
 - analog_data_t, 13
- LOW_UPWR_PROTECTION
 - ximc.h, 98
- LS_SHORTED
 - ximc.h, 98
- LeadScrewPitch
 - stage_settings_t, 56
- LeftBorder
 - edges_settings_calb_t, 25
 - edges_settings_t, 26
- LimitSwitchesSettings
 - accessories_settings_t, 10
- logging_callback_stderr_narrow
 - ximc.h, 124
- logging_callback_stderr_wide
 - ximc.h, 124
- logging_callback_t
 - ximc.h, 106
- LowUpwrOff
 - secure_settings_t, 52
- MBRatedCurrent
 - accessories_settings_t, 10
- MBRatedVoltage
 - accessories_settings_t, 10
- MBSettings
 - accessories_settings_t, 10
- MBTorque
 - accessories_settings_t, 10
- MICROSTEP_MODE_FULL
 - ximc.h, 98
- MOVE_STATE_ANTIPLAY
 - ximc.h, 98
- MOVE_STATE_MOVING
 - ximc.h, 99

- MVCMD_ERROR
 - ximc.h, [99](#)
- MVCMD_HOME
 - ximc.h, [99](#)
- MVCMD_LEFT
 - ximc.h, [99](#)
- MVCMD_LOFT
 - ximc.h, [99](#)
- MVCMD_MOVE
 - ximc.h, [99](#)
- MVCMD_MOVR
 - ximc.h, [99](#)
- MVCMD_NAME_BITS
 - ximc.h, [99](#)
- MVCMD_RIGHT
 - ximc.h, [99](#)
- MVCMD_RUNNING
 - ximc.h, [99](#)
- MVCMD_SSTP
 - ximc.h, [100](#)
- MVCMD_STOP
 - ximc.h, [100](#)
- MVCMD_UKNWN
 - ximc.h, [100](#)
- MagneticBrakeInfo
 - accessories_settings_t, [10](#)
- Major
 - device_information_t, [24](#)
 - serial_number_t, [52](#)
- Manufacturer
 - encoder_information_t, [27](#)
 - gear_information_t, [34](#)
 - hallsensor_information_t, [37](#)
 - motor_information_t, [43](#)
 - stage_information_t, [55](#)
- MaxClickTime
 - control_settings_calb_t, [19](#)
 - control_settings_t, [21](#)
- MaxCurrent
 - motor_settings_t, [44](#)
- MaxCurrentConsumption
 - encoder_settings_t, [28](#)
 - hallsensor_settings_t, [38](#)
 - stage_settings_t, [56](#)
- MaxCurrentTime
 - motor_settings_t, [45](#)
- MaxOperatingFrequency
 - encoder_settings_t, [28](#)
 - hallsensor_settings_t, [38](#)
- MaxOutputBacklash
 - gear_settings_t, [35](#)
- MaxSpeed
 - control_settings_calb_t, [19](#)
 - control_settings_t, [21](#)
 - motor_settings_t, [45](#)
 - stage_settings_t, [56](#)
- MechanicalTimeConstant
 - motor_settings_t, [45](#)
- MicrostepMode
 - engine_settings_calb_t, [29](#)
 - engine_settings_t, [31](#)
- MinimumUusb
 - secure_settings_t, [52](#)
- Minor
 - device_information_t, [24](#)
 - serial_number_t, [53](#)
- motor_information_t, [42](#)
 - Manufacturer, [43](#)
 - PartNumber, [43](#)
- motor_settings_t, [43](#)
 - DetentTorque, [44](#)
 - MaxCurrent, [44](#)
 - MaxCurrentTime, [45](#)
 - MaxSpeed, [45](#)
 - MechanicalTimeConstant, [45](#)
 - MotorType, [45](#)
 - NoLoadCurrent, [45](#)
 - NoLoadSpeed, [45](#)
 - NominalCurrent, [45](#)
 - NominalPower, [45](#)
 - NominalSpeed, [45](#)
 - NominalTorque, [46](#)
 - NominalVoltage, [46](#)
 - Phases, [46](#)
 - Poles, [46](#)
 - RotorInertia, [46](#)
 - SpeedConstant, [46](#)
 - SpeedTorqueGradient, [46](#)
 - StallTorque, [46](#)
 - TorqueConstant, [46](#)
 - WindingInductance, [47](#)
 - WindingResistance, [47](#)
- MotorType
 - motor_settings_t, [45](#)
- move_settings_calb_t, [47](#)
 - Accel, [47](#)
 - AntipplaySpeed, [47](#)
 - Decel, [47](#)
 - Speed, [47](#)
- move_settings_t, [47](#)
 - Accel, [48](#)
 - AntipplaySpeed, [48](#)
 - Decel, [48](#)
 - Speed, [48](#)
 - uAntipplaySpeed, [48](#)
 - uSpeed, [48](#)
- MoveSts
 - status_calb_t, [59](#)
 - status_t, [61](#)
- msec_sleep
 - ximc.h, [124](#)
- MvCmdSts
 - status_calb_t, [59](#)
 - status_t, [61](#)
- NoLoadCurrent
 - motor_settings_t, [45](#)

- NoLoadSpeed
 - motor_settings_t, [45](#)
- NomCurrent
 - engine_settings_calb_t, [29](#)
 - engine_settings_t, [31](#)
- NomSpeed
 - engine_settings_calb_t, [29](#)
 - engine_settings_t, [31](#)
- NomVoltage
 - engine_settings_calb_t, [29](#)
 - engine_settings_t, [31](#)
- NominalCurrent
 - motor_settings_t, [45](#)
- NominalPower
 - motor_settings_t, [45](#)
- NominalSpeed
 - motor_settings_t, [45](#)
- NominalTorque
 - motor_settings_t, [46](#)
- NominalVoltage
 - motor_settings_t, [46](#)
- nonvolatile_memory_t, [49](#)
 - UserData, [49](#)
- open_device
 - ximc.h, [124](#)
- POWER_OFF_ENABLED
 - ximc.h, [100](#)
- POWER_REDUCE_ENABLED
 - ximc.h, [100](#)
- POWER_SMOOTH_CURRENT
 - ximc.h, [100](#)
- PWR_STATE_MAX
 - ximc.h, [100](#)
- PWR_STATE_NORM
 - ximc.h, [100](#)
- PWR_STATE_OFF
 - ximc.h, [100](#)
- PWR_STATE_REDUCE
 - ximc.h, [100](#)
- PWR_STATE_UNKNOWN
 - ximc.h, [100](#)
- PWRSts
 - status_calb_t, [59](#)
 - status_t, [61](#)
- PartNumber
 - encoder_information_t, [27](#)
 - gear_information_t, [34](#)
 - hallsensor_information_t, [37](#)
 - motor_information_t, [43](#)
 - stage_information_t, [55](#)
- Phases
 - motor_settings_t, [46](#)
- pid_settings_t, [49](#)
- Poles
 - motor_settings_t, [46](#)
- PosFlags
 - set_position_calb_t, [53](#)
 - set_position_t, [54](#)
- Position
 - command_add_sync_in_action_calb_t, [18](#)
 - get_position_calb_t, [36](#)
 - set_position_calb_t, [53](#)
 - sync_in_settings_calb_t, [62](#)
- PositionerName
 - stage_name_t, [55](#)
- Pot
 - analog_data_t, [13](#)
 - chart_data_t, [17](#)
- power_settings_t, [50](#)
 - CurrReductDelay, [50](#)
 - CurrentSetTime, [50](#)
 - HoldCurrent, [50](#)
 - PowerFlags, [50](#)
 - PowerOffDelay, [50](#)
- PowerFlags
 - power_settings_t, [50](#)
- PowerOffDelay
 - power_settings_t, [50](#)
- probe_device
 - ximc.h, [125](#)
- REV_SENS_INV
 - ximc.h, [101](#)
- RatedInputSpeed
 - gear_settings_t, [35](#)
- RatedInputTorque
 - gear_settings_t, [35](#)
- ReductionIn
 - gear_settings_t, [36](#)
- ReductionOut
 - gear_settings_t, [36](#)
- Release
 - device_information_t, [24](#)
 - serial_number_t, [53](#)
- RightBorder
 - edges_settings_calb_t, [25](#)
 - edges_settings_t, [26](#)
- RotorInertia
 - motor_settings_t, [46](#)
- SN
 - serial_number_t, [53](#)
- STATE_ALARM
 - ximc.h, [101](#)
- STATE_BRAKE
 - ximc.h, [101](#)
- STATE_BUTTON_LEFT
 - ximc.h, [101](#)
- STATE_BUTTON_RIGHT
 - ximc.h, [101](#)
- STATE_CONTR
 - ximc.h, [101](#)
- STATE_CTP_ERROR
 - ximc.h, [101](#)
- STATE_CURRENT_MOTOR0
 - ximc.h, [102](#)

- STATE_CURRENT_MOTOR1
 - ximc.h, [102](#)
- STATE_CURRENT_MOTOR2
 - ximc.h, [102](#)
- STATE_CURRENT_MOTOR3
 - ximc.h, [102](#)
- STATE_DIG_SIGNAL
 - ximc.h, [102](#)
- STATE_ENC_A
 - ximc.h, [102](#)
- STATE_ENC_B
 - ximc.h, [102](#)
- STATE_ERRC
 - ximc.h, [102](#)
- STATE_ERRD
 - ximc.h, [102](#)
- STATE_ERRV
 - ximc.h, [102](#)
- STATE_GPIO_LEVEL
 - ximc.h, [102](#)
- STATE_GPIO_PINOUT
 - ximc.h, [103](#)
- STATE_HALL_A
 - ximc.h, [103](#)
- STATE_HALL_B
 - ximc.h, [103](#)
- STATE_HALL_C
 - ximc.h, [103](#)
- STATE_LEFT_EDGE
 - ximc.h, [103](#)
- STATE_POWER_OVERHEAT
 - ximc.h, [103](#)
- STATE_REV_SENSOR
 - ximc.h, [103](#)
- STATE_RIGHT_EDGE
 - ximc.h, [104](#)
- STATE_SECUR
 - ximc.h, [104](#)
- STATE_SYNC_INPUT
 - ximc.h, [104](#)
- STATE_SYNC_OUTPUT
 - ximc.h, [104](#)
- SYNCIN_ENABLED
 - ximc.h, [104](#)
- SYNCIN_INVERT
 - ximc.h, [104](#)
- SYNCOUT_ENABLED
 - ximc.h, [104](#)
- SYNCOUT_IN_STEPS
 - ximc.h, [104](#)
- SYNCOUT_INVERT
 - ximc.h, [104](#)
- SYNCOUT_ONPERIOD
 - ximc.h, [104](#)
- SYNCOUT_ONSTART
 - ximc.h, [104](#)
- SYNCOUT_ONSTOP
 - ximc.h, [104](#)
- SYNCOUT_STATE
 - ximc.h, [105](#)
- secure_settings_t, [51](#)
 - CriticalIpwr, [51](#)
 - CriticalUusb, [51](#)
 - CriticalUpwr, [51](#)
 - CriticalUusb, [51](#)
 - Flags, [52](#)
 - LowUpwrOff, [52](#)
 - MinimumUusb, [52](#)
- serial_number_t, [52](#)
 - Key, [52](#)
 - Major, [52](#)
 - Minor, [53](#)
 - Release, [53](#)
 - SN, [53](#)
- service_command_updf
 - ximc.h, [125](#)
- set_accessories_settings
 - ximc.h, [125](#)
- set_bindy_key
 - ximc.h, [125](#)
- set_brake_settings
 - ximc.h, [126](#)
- set_calibration_settings
 - ximc.h, [126](#)
- set_control_settings
 - ximc.h, [126](#)
- set_controller_name
 - ximc.h, [126](#)
- set_ctp_settings
 - ximc.h, [127](#)
- set_debug_write
 - ximc.h, [127](#)
- set_edges_settings
 - ximc.h, [127](#)
- set_encoder_information
 - ximc.h, [127](#)
- set_encoder_settings
 - ximc.h, [128](#)
- set_engine_settings
 - ximc.h, [128](#)
- set_entype_settings
 - ximc.h, [128](#)
- set_extio_settings
 - ximc.h, [128](#)
- set_feedback_settings
 - ximc.h, [129](#)
- set_gear_information
 - ximc.h, [129](#)
- set_gear_settings
 - ximc.h, [129](#)
- set_hallsensor_information
 - ximc.h, [129](#)
- set_hallsensor_settings
 - ximc.h, [130](#)
- set_home_settings
 - ximc.h, [130](#)

- set_joystick_settings
 - ximc.h, [130](#)
- set_logging_callback
 - ximc.h, [131](#)
- set_motor_information
 - ximc.h, [131](#)
- set_motor_settings
 - ximc.h, [131](#)
- set_move_settings
 - ximc.h, [131](#)
- set_nonvolatile_memory
 - ximc.h, [132](#)
- set_pid_settings
 - ximc.h, [132](#)
- set_position
 - ximc.h, [132](#)
- set_position_calb_t, [53](#)
 - EncPosition, [53](#)
 - PosFlags, [53](#)
 - Position, [53](#)
- set_position_t, [53](#)
 - EncPosition, [54](#)
 - PosFlags, [54](#)
 - uPosition, [54](#)
- set_power_settings
 - ximc.h, [132](#)
- set_secure_settings
 - ximc.h, [133](#)
- set_serial_number
 - ximc.h, [133](#)
- set_stage_information
 - ximc.h, [133](#)
- set_stage_name
 - ximc.h, [133](#)
- set_stage_settings
 - ximc.h, [134](#)
- set_sync_in_settings
 - ximc.h, [134](#)
- set_sync_out_settings
 - ximc.h, [134](#)
- set_uart_settings
 - ximc.h, [134](#)
- SlowHome
 - home_settings_calb_t, [39](#)
 - home_settings_t, [40](#)
- Speed
 - move_settings_calb_t, [47](#)
 - move_settings_t, [48](#)
 - sync_in_settings_calb_t, [62](#)
 - sync_in_settings_t, [63](#)
- SpeedConstant
 - motor_settings_t, [46](#)
- SpeedTorqueGradient
 - motor_settings_t, [46](#)
- stage_information_t, [54](#)
 - Manufacturer, [55](#)
 - PartNumber, [55](#)
- stage_name_t, [55](#)
 - PositionerName, [55](#)
- stage_settings_t, [55](#)
 - HorizontalLoadCapacity, [56](#)
 - LeadScrewPitch, [56](#)
 - MaxCurrentConsumption, [56](#)
 - MaxSpeed, [56](#)
 - SupplyVoltageMax, [56](#)
 - SupplyVoltageMin, [56](#)
 - TravelRange, [57](#)
 - Units, [57](#)
 - VerticalLoadCapacity, [57](#)
- StallTorque
 - motor_settings_t, [46](#)
- status_calb_t, [57](#)
 - CmdBufFreeSpace, [58](#)
 - CurPosition, [58](#)
 - CurSpeed, [58](#)
 - CurT, [58](#)
 - EncPosition, [58](#)
 - EncSts, [58](#)
 - Flags, [58](#)
 - GPIOFlags, [58](#)
 - Ipwr, [58](#)
 - Iusb, [58](#)
 - MoveSts, [59](#)
 - MvCmdSts, [59](#)
 - PWRSts, [59](#)
 - Upwr, [59](#)
 - Uusb, [59](#)
 - WindSts, [59](#)
- status_t, [59](#)
 - CmdBufFreeSpace, [60](#)
 - CurPosition, [60](#)
 - CurSpeed, [60](#)
 - CurT, [60](#)
 - EncPosition, [61](#)
 - EncSts, [61](#)
 - Flags, [61](#)
 - GPIOFlags, [61](#)
 - Ipwr, [61](#)
 - Iusb, [61](#)
 - MoveSts, [61](#)
 - MvCmdSts, [61](#)
 - PWRSts, [61](#)
 - uCurPosition, [61](#)
 - uCurSpeed, [61](#)
 - Upwr, [61](#)
 - Uusb, [62](#)
 - WindSts, [62](#)
- StepsPerRev
 - engine_settings_calb_t, [30](#)
 - engine_settings_t, [31](#)
- SupVoltage
 - analog_data_t, [13](#)
- SupVoltage_ADC
 - analog_data_t, [14](#)
- SupplyVoltageMax
 - encoder_settings_t, [28](#)

- hallsensor_settings_t, 38
- stage_settings_t, 56
- SupplyVoltageMin
 - encoder_settings_t, 28
 - hallsensor_settings_t, 38
 - stage_settings_t, 56
- sync_in_settings_calb_t, 62
 - ClutterTime, 62
 - Position, 62
 - Speed, 62
 - SyncInFlags, 62
- sync_in_settings_t, 62
 - ClutterTime, 63
 - Speed, 63
 - SyncInFlags, 63
 - uPosition, 63
 - uSpeed, 63
- sync_out_settings_calb_t, 64
 - Accuracy, 64
 - SyncOutFlags, 64
 - SyncOutPeriod, 64
 - SyncOutPulseSteps, 64
- sync_out_settings_t, 64
 - Accuracy, 65
 - SyncOutFlags, 65
 - SyncOutPeriod, 65
 - SyncOutPulseSteps, 65
 - uAccuracy, 65
- SyncInFlags
 - sync_in_settings_calb_t, 62
 - sync_in_settings_t, 63
- SyncOutFlags
 - sync_out_settings_calb_t, 64
 - sync_out_settings_t, 65
- SyncOutPeriod
 - sync_out_settings_calb_t, 64
 - sync_out_settings_t, 65
- SyncOutPulseSteps
 - sync_out_settings_calb_t, 64
 - sync_out_settings_t, 65
- t1
 - brake_settings_t, 14
- t2
 - brake_settings_t, 15
- t3
 - brake_settings_t, 15
- t4
 - brake_settings_t, 15
- TS_TYPE_BITS
 - ximc.h, 105
- TSGrad
 - accessories_settings_t, 10
- TSMMax
 - accessories_settings_t, 10
- TSMIn
 - accessories_settings_t, 10
- TSSettings
 - accessories_settings_t, 10
- Temp
 - analog_data_t, 14
- Temp_ADC
 - analog_data_t, 14
- TemperatureSensorInfo
 - accessories_settings_t, 10
- Time
 - command_add_sync_in_action_calb_t, 18
 - command_add_sync_in_action_t, 19
- Timeout
 - control_settings_calb_t, 20
 - control_settings_t, 21
- TorqueConstant
 - motor_settings_t, 46
- TravelRange
 - stage_settings_t, 57
- UART_PARITY_BITS
 - ximc.h, 105
- UARTSetupFlags
 - uart_settings_t, 66
- uAccuracy
 - sync_out_settings_t, 65
- uAntiplaySpeed
 - move_settings_t, 48
- uCurPosition
 - status_t, 61
- uCurSpeed
 - status_t, 61
- uDeltaPosition
 - control_settings_t, 21
- uFastHome
 - home_settings_t, 40
- uHomeDelta
 - home_settings_t, 40
- uLeftBorder
 - edges_settings_t, 27
- uMaxSpeed
 - control_settings_t, 21
- uNomSpeed
 - engine_settings_t, 31
- uPosition
 - command_add_sync_in_action_t, 19
 - get_position_t, 37
 - set_position_t, 54
 - sync_in_settings_t, 63
- uRightBorder
 - edges_settings_t, 27
- uSlowHome
 - home_settings_t, 40
- uSpeed
 - move_settings_t, 48
 - sync_in_settings_t, 63
- uart_settings_t, 65
 - UARTSetupFlags, 66
- Units
 - stage_settings_t, 57
- Upwr
 - status_calb_t, 59

- status_t, [61](#)
- UserData
 - nonvolatile_memory_t, [49](#)
- Uusb
 - status_calb_t, [59](#)
 - status_t, [62](#)
- VerticalLoadCapacity
 - stage_settings_t, [57](#)
- WIND_A_STATE_ABSENT
 - ximc.h, [105](#)
- WIND_A_STATE_OK
 - ximc.h, [105](#)
- WIND_B_STATE_ABSENT
 - ximc.h, [105](#)
- WIND_B_STATE_OK
 - ximc.h, [105](#)
- WindSts
 - status_calb_t, [59](#)
 - status_t, [62](#)
- WindingCurrentA
 - chart_data_t, [17](#)
- WindingCurrentB
 - chart_data_t, [17](#)
- WindingCurrentC
 - chart_data_t, [17](#)
- WindingInductance
 - motor_settings_t, [47](#)
- WindingResistance
 - motor_settings_t, [47](#)
- WindingVoltageA
 - chart_data_t, [18](#)
- WindingVoltageB
 - chart_data_t, [18](#)
- WindingVoltageC
 - chart_data_t, [18](#)
- write_key
 - ximc.h, [135](#)
- XIMC_API
 - ximc.h, [105](#)
- ximc.h, [67](#)
 - BORDER_IS_ENCODER, [90](#)
 - BORDER_STOP_LEFT, [90](#)
 - BORDER_STOP_RIGHT, [90](#)
 - BRAKE_ENABLED, [91](#)
 - BRAKE_ENG_PWROFF, [91](#)
 - CONTROL_MODE_BITS, [91](#)
 - CONTROL_MODE_JOY, [91](#)
 - CONTROL_MODE_LR, [91](#)
 - CONTROL_MODE_OFF, [91](#)
 - CTP_ALARM_ON_ERROR, [91](#)
 - CTP_BASE, [91](#)
 - CTP_ENABLED, [91](#)
 - close_device, [106](#)
 - command_add_sync_in_action, [106](#)
 - command_clear_fram, [106](#)
 - command_eeread_settings, [106](#)
 - command_eesave_settings, [107](#)
 - command_home, [107](#)
 - command_homezero, [107](#)
 - command_left, [108](#)
 - command_loft, [108](#)
 - command_move, [108](#)
 - command_movr, [108](#)
 - command_power_off, [109](#)
 - command_read_robust_settings, [109](#)
 - command_read_settings, [109](#)
 - command_reset, [109](#)
 - command_right, [109](#)
 - command_save_robust_settings, [109](#)
 - command_save_settings, [110](#)
 - command_sstp, [110](#)
 - command_stop, [110](#)
 - command_update_firmware, [110](#)
 - command_wait_for_stop, [110](#)
 - command_zero, [111](#)
 - EEPROM_PRECEDENCE, [92](#)
 - ENC_STATE_ABSENT, [92](#)
 - ENC_STATE_MALFUNC, [92](#)
 - ENC_STATE_OK, [92](#)
 - ENC_STATE_REVERS, [92](#)
 - ENC_STATE_UNKNOWN, [92](#)
 - ENDER_SWAP, [92](#)
 - ENGINE_ACCEL_ON, [93](#)
 - ENGINE_ANTIPLAY, [93](#)
 - ENGINE_LIMIT_CURR, [93](#)
 - ENGINE_LIMIT_RPM, [93](#)
 - ENGINE_LIMIT_VOLT, [93](#)
 - ENGINE_MAX_SPEED, [93](#)
 - ENGINE_REVERSE, [93](#)
 - ENGINE_TYPE_2DC, [94](#)
 - ENGINE_TYPE_DC, [94](#)
 - ENGINE_TYPE_NONE, [94](#)
 - ENGINE_TYPE_STEP, [94](#)
 - ENGINE_TYPE_TEST, [94](#)
 - ENUMERATE_PROBE, [94](#)
 - EXTIO_SETUP_INVERT, [94](#)
 - EXTIO_SETUP_OUTPUT, [95](#)
 - enumerate_devices, [111](#)
 - FEEDBACK_EMF, [96](#)
 - FEEDBACK_ENCODER, [96](#)
 - FEEDBACK_ENCODERHALL, [96](#)
 - FEEDBACK_NONE, [96](#)
 - free_enumerate_devices, [111](#)
 - get_accessories_settings, [112](#)
 - get_analog_data, [112](#)
 - get_bootloader_version, [112](#)
 - get_brake_settings, [112](#)
 - get_calibration_settings, [112](#)
 - get_chart_data, [113](#)
 - get_control_settings, [113](#)
 - get_controller_name, [113](#)
 - get_ctp_settings, [113](#)
 - get_debug_read, [114](#)
 - get_device_count, [114](#)

- get_device_information, 114
- get_device_name, 114
- get_edges_settings, 115
- get_encoder_information, 115
- get_encoder_settings, 115
- get_engine_settings, 115
- get_entype_settings, 116
- get_enumerate_device_controller_name, 116
- get_enumerate_device_information, 116
- get_enumerate_device_network_information, 116
- get_enumerate_device_serial, 117
- get_enumerate_device_stage_name, 117
- get_extio_settings, 117
- get_feedback_settings, 118
- get_firmware_version, 118
- get_gear_information, 118
- get_gear_settings, 118
- get_hallsensor_information, 118
- get_hallsensor_settings, 119
- get_home_settings, 119
- get_init_random, 119
- get_joystick_settings, 119
- get_motor_information, 120
- get_motor_settings, 120
- get_move_settings, 120
- get_nonvolatile_memory, 120
- get_pid_settings, 120
- get_position, 121
- get_power_settings, 121
- get_secure_settings, 121
- get_serial_number, 121
- get_stage_information, 122
- get_stage_name, 122
- get_stage_settings, 122
- get_status, 122
- get_status_calb, 123
- get_sync_in_settings, 123
- get_sync_out_settings, 123
- get_uart_settings, 123
- goto_firmware, 124
- HOME_DIR_FIRST, 96
- HOME_DIR_SECOND, 96
- HOME_HALF_MV, 97
- HOME_MV_SEC_EN, 97
- HOME_USE_FAST, 97
- has_firmware, 124
- JOY_REVERSE, 97
- LOW_UPWR_PROTECTION, 98
- LS_SHORTED, 98
- logging_callback_stderr_narrow, 124
- logging_callback_stderr_wide, 124
- logging_callback_t, 106
- MICROSTEP_MODE_FULL, 98
- MOVE_STATE_ANTIPLAY, 98
- MOVE_STATE_MOVING, 99
- MVCMD_ERROR, 99
- MVCMD_HOME, 99
- MVCMD_LEFT, 99
- MVCMD_LOFT, 99
- MVCMD_MOVE, 99
- MVCMD_MOVR, 99
- MVCMD_NAME_BITS, 99
- MVCMD_RIGHT, 99
- MVCMD_RUNNING, 99
- MVCMD_SSTP, 100
- MVCMD_STOP, 100
- MVCMD_UKNWN, 100
- msec_sleep, 124
- open_device, 124
- POWER_OFF_ENABLED, 100
- PWR_STATE_MAX, 100
- PWR_STATE_NORM, 100
- PWR_STATE_OFF, 100
- PWR_STATE_REDUCT, 100
- PWR_STATE_UNKNOWN, 100
- probe_device, 125
- REV_SENS_INV, 101
- STATE_ALARM, 101
- STATE_BRAKE, 101
- STATE_BUTTON_LEFT, 101
- STATE_BUTTON_RIGHT, 101
- STATE_CONTR, 101
- STATE_CTP_ERROR, 101
- STATE_CURRENT_MOTOR0, 102
- STATE_CURRENT_MOTOR1, 102
- STATE_CURRENT_MOTOR2, 102
- STATE_CURRENT_MOTOR3, 102
- STATE_DIG_SIGNAL, 102
- STATE_ENC_A, 102
- STATE_ENC_B, 102
- STATE_ERRC, 102
- STATE_ERRD, 102
- STATE_ERRV, 102
- STATE_GPIO_LEVEL, 102
- STATE_GPIO_PINOUT, 103
- STATE_HALL_A, 103
- STATE_HALL_B, 103
- STATE_HALL_C, 103
- STATE_LEFT_EDGE, 103
- STATE_REV_SENSOR, 103
- STATE_RIGHT_EDGE, 104
- STATE_SECUR, 104
- STATE_SYNC_INPUT, 104
- STATE_SYNC_OUTPUT, 104
- SYNCIN_ENABLED, 104
- SYNCIN_INVERT, 104
- SYNCOUT_ENABLED, 104
- SYNCOUT_IN_STEPS, 104
- SYNCOUT_INVERT, 104
- SYNCOUT_ONPERIOD, 104
- SYNCOUT_ONSTART, 104
- SYNCOUT_ONSTOP, 104
- SYNCOUT_STATE, 105
- service_command_updf, 125
- set_accessories_settings, 125

- set_bindy_key, [125](#)
- set_brake_settings, [126](#)
- set_calibration_settings, [126](#)
- set_control_settings, [126](#)
- set_controller_name, [126](#)
- set_ctp_settings, [127](#)
- set_debug_write, [127](#)
- set_edges_settings, [127](#)
- set_encoder_information, [127](#)
- set_encoder_settings, [128](#)
- set_engine_settings, [128](#)
- set_entype_settings, [128](#)
- set_extio_settings, [128](#)
- set_feedback_settings, [129](#)
- set_gear_information, [129](#)
- set_gear_settings, [129](#)
- set_hallsensor_information, [129](#)
- set_hallsensor_settings, [130](#)
- set_home_settings, [130](#)
- set_joystick_settings, [130](#)
- set_logging_callback, [131](#)
- set_motor_information, [131](#)
- set_motor_settings, [131](#)
- set_move_settings, [131](#)
- set_nonvolatile_memory, [132](#)
- set_pid_settings, [132](#)
- set_position, [132](#)
- set_power_settings, [132](#)
- set_secure_settings, [133](#)
- set_serial_number, [133](#)
- set_stage_information, [133](#)
- set_stage_name, [133](#)
- set_stage_settings, [134](#)
- set_sync_in_settings, [134](#)
- set_sync_out_settings, [134](#)
- set_uart_settings, [134](#)
- TS_TYPE_BITS, [105](#)
- UART_PARITY_BITS, [105](#)
- WIND_A_STATE_OK, [105](#)
- WIND_B_STATE_OK, [105](#)
- write_key, [135](#)
- XIMC_API, [105](#)
- ximc_fix_usbser_sys, [135](#)
- ximc_version, [135](#)
- ximc_fix_usbser_sys
- ximc.h, [135](#)
- ximc_version
- ximc.h, [135](#)