

libximc  
2.8.0

Generated by Doxygen 1.8.2

Thu Jun 4 2015 16:24:23

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	About	1
1.2	System requirements	1
1.2.1	For rebuilding library	1
1.2.2	For using library	2
<b>2</b>	<b>How to rebuild library</b>	<b>3</b>
2.1	Building on generic UNIX	3
2.2	Building on debian-based linux systems	3
2.3	Building on redhat-based linux systems	3
2.4	Building on FreeBSD	4
2.5	Buliding on Mac OS X	4
2.6	Buliding on Windows	4
2.7	Source code access	4
<b>3</b>	<b>How to use with...</b>	<b>5</b>
3.1	Usage with 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	MATLAB	6
3.5	Generic logging facility	7
<b>4</b>	<b>Data Structure Documentation</b>	<b>8</b>
4.1	accessories_settings.t Struct Reference	8
4.1.1	Detailed Description	8
4.1.2	Field Documentation	9
4.1.2.1	LimitSwitchesSettings	9

4.1.2.2	MagneticBrakeInfo . . . . .	9
4.1.2.3	MBRatedCurrent . . . . .	9
4.1.2.4	MBRatedVoltage . . . . .	9
4.1.2.5	MBSettings . . . . .	9
4.1.2.6	MBTorque . . . . .	9
4.1.2.7	TemperatureSensorInfo . . . . .	9
4.1.2.8	TSGrad . . . . .	9
4.1.2.9	TSMax . . . . .	9
4.1.2.10	TSTMin . . . . .	9
4.1.2.11	TSSettings . . . . .	9
4.2	analog_data.t Struct Reference . . . . .	10
4.2.1	Detailed Description . . . . .	11
4.2.2	Field Documentation . . . . .	11
4.2.2.1	A1Voltage . . . . .	11
4.2.2.2	A1Voltage_ADC . . . . .	11
4.2.2.3	A2Voltage . . . . .	11
4.2.2.4	A2Voltage_ADC . . . . .	11
4.2.2.5	ACurrent . . . . .	11
4.2.2.6	ACurrent_ADC . . . . .	11
4.2.2.7	B1Voltage . . . . .	12
4.2.2.8	B1Voltage_ADC . . . . .	12
4.2.2.9	B2Voltage . . . . .	12
4.2.2.10	B2Voltage_ADC . . . . .	12
4.2.2.11	BCurrent . . . . .	12
4.2.2.12	BCurrent_ADC . . . . .	12
4.2.2.13	FullCurrent . . . . .	12
4.2.2.14	FullCurrent_ADC . . . . .	12
4.2.2.15	Joy . . . . .	12
4.2.2.16	Joy_ADC . . . . .	12
4.2.2.17	L . . . . .	12
4.2.2.18	L5 . . . . .	12
4.2.2.19	L5_ADC . . . . .	13
4.2.2.20	Pot . . . . .	13
4.2.2.21	R . . . . .	13
4.2.2.22	SupVoltage . . . . .	13
4.2.2.23	SupVoltage_ADC . . . . .	13
4.2.2.24	Temp . . . . .	13
4.2.2.25	Temp_ADC . . . . .	13
4.3	brake_settings.t Struct Reference . . . . .	13
4.3.1	Detailed Description . . . . .	14

4.3.2	Field Documentation	14
4.3.2.1	BrakeFlags	14
4.3.2.2	t1	14
4.3.2.3	t2	14
4.3.2.4	t3	14
4.3.2.5	t4	14
4.4	calibration.t Struct Reference	14
4.4.1	Detailed Description	14
4.5	chart_data.t Struct Reference	15
4.5.1	Detailed Description	15
4.5.2	Field Documentation	15
4.5.2.1	DutyCycle	15
4.5.2.2	Joy	15
4.5.2.3	Pot	16
4.5.2.4	WindingCurrentA	16
4.5.2.5	WindingCurrentB	16
4.5.2.6	WindingCurrentC	16
4.5.2.7	WindingVoltageA	16
4.5.2.8	WindingVoltageB	16
4.5.2.9	WindingVoltageC	16
4.6	command_add_sync_in_action_calb.t Struct Reference	16
4.6.1	Field Documentation	16
4.6.1.1	Position	16
4.6.1.2	Time	17
4.7	command_add_sync_in_action.t Struct Reference	17
4.7.1	Detailed Description	17
4.7.2	Field Documentation	17
4.7.2.1	Time	17
4.7.2.2	uPosition	17
4.8	control_settings_calb.t Struct Reference	17
4.8.1	Field Documentation	18
4.8.1.1	Flags	18
4.8.1.2	MaxClickTime	18
4.8.1.3	MaxSpeed	18
4.8.1.4	Timeout	18
4.9	control_settings.t Struct Reference	18
4.9.1	Detailed Description	18
4.9.2	Field Documentation	19
4.9.2.1	Flags	19
4.9.2.2	MaxClickTime	19

4.9.2.3	MaxSpeed	19
4.9.2.4	Timeout	19
4.9.2.5	uDeltaPosition	19
4.9.2.6	uMaxSpeed	19
4.10	controller_name.t Struct Reference	19
4.10.1	Detailed Description	20
4.10.2	Field Documentation	20
4.10.2.1	ControllerName	20
4.10.2.2	CtrlFlags	20
4.11	ctp_settings.t Struct Reference	20
4.11.1	Detailed Description	20
4.11.2	Field Documentation	20
4.11.2.1	CTPFlags	20
4.11.2.2	CTPMinError	21
4.12	debug_read.t Struct Reference	21
4.12.1	Detailed Description	21
4.12.2	Field Documentation	21
4.12.2.1	DebugData	21
4.13	debug_write.t Struct Reference	21
4.13.1	Detailed Description	21
4.13.2	Field Documentation	22
4.13.2.1	DebugData	22
4.14	device_information.t Struct Reference	22
4.14.1	Detailed Description	22
4.14.2	Field Documentation	22
4.14.2.1	Major	22
4.14.2.2	Minor	22
4.14.2.3	Release	22
4.15	device_network_information.t Struct Reference	23
4.15.1	Detailed Description	23
4.16	edges_settings_calb.t Struct Reference	23
4.16.1	Field Documentation	23
4.16.1.1	BorderFlags	23
4.16.1.2	EnderFlags	23
4.16.1.3	LeftBorder	23
4.16.1.4	RightBorder	24
4.17	edges_settings.t Struct Reference	24
4.17.1	Detailed Description	24
4.17.2	Field Documentation	24
4.17.2.1	BorderFlags	24

4.17.2.2	EnderFlags	24
4.17.2.3	LeftBorder	24
4.17.2.4	RightBorder	25
4.17.2.5	uLeftBorder	25
4.17.2.6	uRightBorder	25
4.18	encoder_information_t Struct Reference	25
4.18.1	Detailed Description	25
4.18.2	Field Documentation	25
4.18.2.1	Manufacturer	25
4.18.2.2	PartNumber	25
4.19	encoder_settings_t Struct Reference	26
4.19.1	Detailed Description	26
4.19.2	Field Documentation	26
4.19.2.1	EncoderSettings	26
4.19.2.2	MaxCurrentConsumption	26
4.19.2.3	MaxOperatingFrequency	26
4.19.2.4	SupplyVoltageMax	26
4.19.2.5	SupplyVoltageMin	27
4.20	engine_settings_calb_t Struct Reference	27
4.20.1	Field Documentation	27
4.20.1.1	Antiplay	27
4.20.1.2	EngineFlags	27
4.20.1.3	MicrostepMode	27
4.20.1.4	NomCurrent	27
4.20.1.5	NomSpeed	27
4.20.1.6	NomVoltage	28
4.20.1.7	StepsPerRev	28
4.21	engine_settings_t Struct Reference	28
4.21.1	Detailed Description	28
4.21.2	Field Documentation	29
4.21.2.1	Antiplay	29
4.21.2.2	EngineFlags	29
4.21.2.3	MicrostepMode	29
4.21.2.4	NomCurrent	29
4.21.2.5	NomSpeed	29
4.21.2.6	NomVoltage	29
4.21.2.7	StepsPerRev	29
4.21.2.8	uNomSpeed	29
4.22	entype_settings_t Struct Reference	29
4.22.1	Detailed Description	30

4.22.2	Field Documentation	30
4.22.2.1	DriverType	30
4.22.2.2	EngineType	30
4.23	extio_settings.t Struct Reference	30
4.23.1	Detailed Description	30
4.23.2	Field Documentation	31
4.23.2.1	EXTIOModeFlags	31
4.23.2.2	EXTIOSetupFlags	31
4.24	feedback_settings.t Struct Reference	31
4.24.1	Detailed Description	31
4.24.2	Field Documentation	31
4.24.2.1	FeedbackFlags	31
4.24.2.2	FeedbackType	32
4.24.2.3	HallShift	32
4.24.2.4	HallSPR	32
4.25	gear_information.t Struct Reference	32
4.25.1	Detailed Description	32
4.25.2	Field Documentation	32
4.25.2.1	Manufacturer	32
4.25.2.2	PartNumber	32
4.26	gear_settings.t Struct Reference	32
4.26.1	Detailed Description	33
4.26.2	Field Documentation	33
4.26.2.1	Efficiency	33
4.26.2.2	InputInertia	33
4.26.2.3	MaxOutputBacklash	33
4.26.2.4	RatedInputSpeed	33
4.26.2.5	RatedInputTorque	34
4.26.2.6	ReductionIn	34
4.26.2.7	ReductionOut	34
4.27	get_position_calb.t Struct Reference	34
4.27.1	Field Documentation	34
4.27.1.1	EncPosition	34
4.27.1.2	Position	34
4.28	get_position.t Struct Reference	34
4.28.1	Detailed Description	35
4.28.2	Field Documentation	35
4.28.2.1	EncPosition	35
4.29	hallsensor_information.t Struct Reference	35
4.29.1	Detailed Description	35

4.29.2	Field Documentation	35
4.29.2.1	Manufacturer	35
4.29.2.2	PartNumber	35
4.30	hallsensor_settings_t Struct Reference	35
4.30.1	Detailed Description	36
4.30.2	Field Documentation	36
4.30.2.1	MaxCurrentConsumption	36
4.30.2.2	MaxOperatingFrequency	36
4.30.2.3	SupplyVoltageMax	36
4.30.2.4	SupplyVoltageMin	36
4.31	home_settings_calb_t Struct Reference	36
4.31.1	Field Documentation	37
4.31.1.1	FastHome	37
4.31.1.2	HomeDelta	37
4.31.1.3	HomeFlags	37
4.31.1.4	SlowHome	37
4.32	home_settings_t Struct Reference	37
4.32.1	Detailed Description	38
4.32.2	Field Documentation	38
4.32.2.1	FastHome	38
4.32.2.2	HomeDelta	38
4.32.2.3	HomeFlags	38
4.32.2.4	SlowHome	38
4.32.2.5	uFastHome	38
4.32.2.6	uHomeDelta	38
4.32.2.7	uSlowHome	38
4.33	init_random_t Struct Reference	38
4.33.1	Detailed Description	39
4.33.2	Field Documentation	39
4.33.2.1	key	39
4.34	joystick_settings_t Struct Reference	39
4.34.1	Detailed Description	39
4.34.2	Field Documentation	40
4.34.2.1	DeadZone	40
4.34.2.2	ExpFactor	40
4.34.2.3	JoyCenter	40
4.34.2.4	JoyFlags	40
4.34.2.5	JoyHighEnd	40
4.34.2.6	JoyLowEnd	40
4.35	motor_information_t Struct Reference	40



4.35.1 Detailed Description . . . . .	41
4.35.2 Field Documentation . . . . .	41
4.35.2.1 Manufacturer . . . . .	41
4.35.2.2 PartNumber . . . . .	41
4.36 motor_settings.t Struct Reference . . . . .	41
4.36.1 Detailed Description . . . . .	42
4.36.2 Field Documentation . . . . .	42
4.36.2.1 DetentTorque . . . . .	42
4.36.2.2 MaxCurrent . . . . .	42
4.36.2.3 MaxCurrentTime . . . . .	43
4.36.2.4 MaxSpeed . . . . .	43
4.36.2.5 MechanicalTimeConstant . . . . .	43
4.36.2.6 MotorType . . . . .	43
4.36.2.7 NoLoadCurrent . . . . .	43
4.36.2.8 NoLoadSpeed . . . . .	43
4.36.2.9 NominalCurrent . . . . .	43
4.36.2.10 NominalPower . . . . .	43
4.36.2.11 NominalSpeed . . . . .	43
4.36.2.12 NominalTorque . . . . .	44
4.36.2.13 NominalVoltage . . . . .	44
4.36.2.14 Phases . . . . .	44
4.36.2.15 Poles . . . . .	44
4.36.2.16 RotorInertia . . . . .	44
4.36.2.17 SpeedConstant . . . . .	44
4.36.2.18 SpeedTorqueGradient . . . . .	44
4.36.2.19 StallTorque . . . . .	44
4.36.2.20 TorqueConstant . . . . .	44
4.36.2.21 WindingInductance . . . . .	45
4.36.2.22 WindingResistance . . . . .	45
4.37 move_settings_calb.t Struct Reference . . . . .	45
4.37.1 Field Documentation . . . . .	45
4.37.1.1 Accel . . . . .	45
4.37.1.2 AntiplaySpeed . . . . .	45
4.37.1.3 Decel . . . . .	45
4.37.1.4 Speed . . . . .	45
4.38 move_settings.t Struct Reference . . . . .	45
4.38.1 Detailed Description . . . . .	46
4.38.2 Field Documentation . . . . .	46
4.38.2.1 Accel . . . . .	46
4.38.2.2 AntiplaySpeed . . . . .	46

4.38.2.3	Decel	46
4.38.2.4	Speed	46
4.38.2.5	uAntiplaySpeed	46
4.38.2.6	uSpeed	47
4.39	pid_settings.t Struct Reference	47
4.39.1	Detailed Description	47
4.40	power_settings.t Struct Reference	47
4.40.1	Detailed Description	48
4.40.2	Field Documentation	48
4.40.2.1	CurrentSetTime	48
4.40.2.2	CurrReductDelay	48
4.40.2.3	HoldCurrent	48
4.40.2.4	PowerFlags	48
4.40.2.5	PowerOffDelay	48
4.41	secure_settings.t Struct Reference	48
4.41.1	Detailed Description	49
4.41.2	Field Documentation	49
4.41.2.1	Criticalpwr	49
4.41.2.2	Criticalusb	49
4.41.2.3	CriticalT	49
4.41.2.4	CriticalUpwr	49
4.41.2.5	CriticalUusb	49
4.41.2.6	Flags	49
4.41.2.7	LowUpwrOff	49
4.41.2.8	MinimumUusb	49
4.42	serial_number.t Struct Reference	50
4.42.1	Detailed Description	50
4.42.2	Field Documentation	50
4.42.2.1	Key	50
4.42.2.2	Major	50
4.42.2.3	Minor	50
4.42.2.4	Release	50
4.42.2.5	SN	50
4.43	set_position_calb.t Struct Reference	51
4.43.1	Field Documentation	51
4.43.1.1	EncPosition	51
4.43.1.2	PosFlags	51
4.43.1.3	Position	51
4.44	set_position.t Struct Reference	51
4.44.1	Detailed Description	51

4.44.2	Field Documentation	52
4.44.2.1	EncPosition	52
4.44.2.2	PosFlags	52
4.45	stage_information_t Struct Reference	52
4.45.1	Detailed Description	52
4.45.2	Field Documentation	52
4.45.2.1	Manufacturer	52
4.45.2.2	PartNumber	52
4.46	stage_name_t Struct Reference	52
4.46.1	Detailed Description	53
4.46.2	Field Documentation	53
4.46.2.1	PositionerName	53
4.47	stage_settings_t Struct Reference	53
4.47.1	Detailed Description	54
4.47.2	Field Documentation	54
4.47.2.1	HorizontalLoadCapacity	54
4.47.2.2	LeadScrewPitch	54
4.47.2.3	MaxCurrentConsumption	54
4.47.2.4	MaxSpeed	54
4.47.2.5	SupplyVoltageMax	54
4.47.2.6	SupplyVoltageMin	54
4.47.2.7	TravelRange	54
4.47.2.8	Units	55
4.47.2.9	VerticalLoadCapacity	55
4.48	status_calb_t Struct Reference	55
4.48.1	Field Documentation	56
4.48.1.1	CmdBufFreeSpace	56
4.48.1.2	CurPosition	56
4.48.1.3	CurSpeed	56
4.48.1.4	CurT	56
4.48.1.5	EncPosition	56
4.48.1.6	EncSts	56
4.48.1.7	Flags	56
4.48.1.8	GPIOFlags	56
4.48.1.9	Ipwr	56
4.48.1.10	Iusb	56
4.48.1.11	MoveSts	56
4.48.1.12	MvCmdSts	56
4.48.1.13	PWRSts	57
4.48.1.14	Upwr	57

4.48.1.15 Uusb . . . . .	57
4.48.1.16 WindSts . . . . .	57
4.49 status_t Struct Reference . . . . .	57
4.49.1 Detailed Description . . . . .	58
4.49.2 Field Documentation . . . . .	58
4.49.2.1 CmdBufFreeSpace . . . . .	58
4.49.2.2 CurPosition . . . . .	58
4.49.2.3 CurSpeed . . . . .	58
4.49.2.4 CurT . . . . .	58
4.49.2.5 EncPosition . . . . .	58
4.49.2.6 EncSts . . . . .	58
4.49.2.7 Flags . . . . .	58
4.49.2.8 GPIOFlags . . . . .	59
4.49.2.9 lpwr . . . . .	59
4.49.2.10 lusb . . . . .	59
4.49.2.11 MoveSts . . . . .	59
4.49.2.12 MvCmdSts . . . . .	59
4.49.2.13 PWRSts . . . . .	59
4.49.2.14 uCurPosition . . . . .	59
4.49.2.15 uCurSpeed . . . . .	59
4.49.2.16 Upwr . . . . .	59
4.49.2.17 Uusb . . . . .	59
4.49.2.18 WindSts . . . . .	59
4.50 sync_in_settings_calb_t Struct Reference . . . . .	60
4.50.1 Field Documentation . . . . .	60
4.50.1.1 ClutterTime . . . . .	60
4.50.1.2 Position . . . . .	60
4.50.1.3 Speed . . . . .	60
4.50.1.4 SyncInFlags . . . . .	60
4.51 sync_in_settings_t Struct Reference . . . . .	60
4.51.1 Detailed Description . . . . .	61
4.51.2 Field Documentation . . . . .	61
4.51.2.1 ClutterTime . . . . .	61
4.51.2.2 Speed . . . . .	61
4.51.2.3 SyncInFlags . . . . .	61
4.51.2.4 uPosition . . . . .	61
4.51.2.5 uSpeed . . . . .	61
4.52 sync_out_settings_calb_t Struct Reference . . . . .	61
4.52.1 Field Documentation . . . . .	62
4.52.1.1 Accuracy . . . . .	62

4.52.1.2	SyncOutFlags	62
4.52.1.3	SyncOutPeriod	62
4.52.1.4	SyncOutPulseSteps	62
4.53	sync_out_settings_t Struct Reference	62
4.53.1	Detailed Description	62
4.53.2	Field Documentation	63
4.53.2.1	Accuracy	63
4.53.2.2	SyncOutFlags	63
4.53.2.3	SyncOutPeriod	63
4.53.2.4	SyncOutPulseSteps	63
4.53.2.5	uAccuracy	63
4.54	uart_settings_t Struct Reference	63
4.54.1	Detailed Description	63
4.54.2	Field Documentation	64
4.54.2.1	UARTSetupFlags	64
<b>5</b>	<b>File Documentation</b>	<b>65</b>
5.1	ximc.h File Reference	65
5.1.1	Detailed Description	85
5.1.2	Macro Definition Documentation	85
5.1.2.1	ALARM_ON_DRIVER_OVERHEATING	85
5.1.2.2	BORDER_IS_ENCODER	86
5.1.2.3	BORDER_STOP_LEFT	86
5.1.2.4	BORDER_STOP_RIGHT	86
5.1.2.5	BORDERS_SWAP_MISSET_DETECTION	86
5.1.2.6	BRAKE_ENABLED	86
5.1.2.7	BRAKE_ENG_PWROFF	86
5.1.2.8	CONTROL_BTN_LEFT_PUSHED_OPEN	86
5.1.2.9	CONTROL_BTN_RIGHT_PUSHED_OPEN	86
5.1.2.10	CONTROL_MODE_BITS	86
5.1.2.11	CONTROL_MODE_JOY	86
5.1.2.12	CONTROL_MODE_LR	86
5.1.2.13	CONTROL_MODE_OFF	86
5.1.2.14	CTP_ALARM_ON_ERROR	87
5.1.2.15	CTP_BASE	87
5.1.2.16	CTP_ENABLED	87
5.1.2.17	DRIVER_TYPE_DISCRETE_FET	87
5.1.2.18	DRIVER_TYPE_EXTERNAL	87
5.1.2.19	DRIVER_TYPE_INTEGRATE	87
5.1.2.20	EEPROM_PRECEDENCE	87

5.1.2.21	ENC.STATE_ABSENT	87
5.1.2.22	ENC.STATE_MALFUNC	87
5.1.2.23	ENC.STATE_OK	87
5.1.2.24	ENC.STATE_REVERS	87
5.1.2.25	ENC.STATE_UNKNOWN	87
5.1.2.26	ENDER.SW1_ACTIVE_LOW	88
5.1.2.27	ENDER.SW2_ACTIVE_LOW	88
5.1.2.28	ENDER.SWAP	88
5.1.2.29	ENGINE.ACCEL_ON	88
5.1.2.30	ENGINE.ANTIPLAY	88
5.1.2.31	ENGINE.LIMIT_CURR	88
5.1.2.32	ENGINE.LIMIT_RPM	88
5.1.2.33	ENGINE.LIMIT_VOLT	88
5.1.2.34	ENGINE.MAX_SPEED	88
5.1.2.35	ENGINE.REVERSE	88
5.1.2.36	ENGINE.TYPE_2DC	89
5.1.2.37	ENGINE.TYPE_BRUSHLESS	89
5.1.2.38	ENGINE.TYPE_DC	89
5.1.2.39	ENGINE.TYPE_NONE	89
5.1.2.40	ENGINE.TYPE_STEP	89
5.1.2.41	ENGINE.TYPE_TEST	89
5.1.2.42	ENUMERATE_PROBE	89
5.1.2.43	EXTIO.SETUP_INVERT	89
5.1.2.44	EXTIO.SETUP_MODE_IN_HOME	89
5.1.2.45	EXTIO.SETUP_MODE_IN_MOVR	89
5.1.2.46	EXTIO.SETUP_MODE_IN_NOP	89
5.1.2.47	EXTIO.SETUP_MODE_IN_PWOF	90
5.1.2.48	EXTIO.SETUP_MODE_IN_STOP	90
5.1.2.49	EXTIO.SETUP_MODE_OUT_ALARM	90
5.1.2.50	EXTIO.SETUP_MODE_OUT_MOTOR_FOUND	90
5.1.2.51	EXTIO.SETUP_MODE_OUT_MOTOR_ON	90
5.1.2.52	EXTIO.SETUP_MODE_OUT_MOVING	90
5.1.2.53	EXTIO.SETUP_MODE_OUT_OFF	90
5.1.2.54	EXTIO.SETUP_MODE_OUT_ON	90
5.1.2.55	EXTIO.SETUP_OUTPUT	90
5.1.2.56	FEEDBACK_EMF	90
5.1.2.57	FEEDBACK_ENC_REVERSE	90
5.1.2.58	FEEDBACK_ENCODER	90
5.1.2.59	FEEDBACK_ENCODERHALL	91
5.1.2.60	FEEDBACK_HALL_REVERSE	91

5.1.2.61	FEEDBACK_NONE	91
5.1.2.62	H_BRIDGE_ALERT	91
5.1.2.63	HOME_DIR_FIRST	91
5.1.2.64	HOME_DIR_SECOND	91
5.1.2.65	HOME_HALF_MV	91
5.1.2.66	HOME_MV_SEC_EN	91
5.1.2.67	HOME_STOP_FIRST_BITS	91
5.1.2.68	HOME_STOP_FIRST_LIM	91
5.1.2.69	HOME_STOP_FIRST_REV	91
5.1.2.70	HOME_STOP_FIRST_SYN	91
5.1.2.71	HOME_STOP_SECOND_BITS	92
5.1.2.72	HOME_STOP_SECOND_LIM	92
5.1.2.73	HOME_STOP_SECOND_REV	92
5.1.2.74	HOME_STOP_SECOND_SYN	92
5.1.2.75	JOY_REVERSE	92
5.1.2.76	LOW_UPWR_PROTECTION	92
5.1.2.77	MICROSTEP_MODE_FRAC_128	92
5.1.2.78	MICROSTEP_MODE_FRAC_16	92
5.1.2.79	MICROSTEP_MODE_FRAC_2	92
5.1.2.80	MICROSTEP_MODE_FRAC_256	92
5.1.2.81	MICROSTEP_MODE_FRAC_32	92
5.1.2.82	MICROSTEP_MODE_FRAC_4	92
5.1.2.83	MICROSTEP_MODE_FRAC_64	93
5.1.2.84	MICROSTEP_MODE_FRAC_8	93
5.1.2.85	MICROSTEP_MODE_FULL	93
5.1.2.86	MOVE_STATE_ANTIPLAY	93
5.1.2.87	MOVE_STATE_MOVING	93
5.1.2.88	MOVE_STATE_TARGET_SPEED	93
5.1.2.89	MVCMD_ERROR	93
5.1.2.90	MVCMD_HOME	93
5.1.2.91	MVCMD_LEFT	93
5.1.2.92	MVCMD_LOFT	93
5.1.2.93	MVCMD_MOVE	93
5.1.2.94	MVCMD_MOVR	93
5.1.2.95	MVCMD_NAME_BITS	94
5.1.2.96	MVCMD_RIGHT	94
5.1.2.97	MVCMD_RUNNING	94
5.1.2.98	MVCMD_SSTP	94
5.1.2.99	MVCMD_STOP	94
5.1.2.100	MVCMD_UKNWN	94

5.1.2.101 POWER_OFF_ENABLED . . . . .	94
5.1.2.102 POWER_REDUCT_ENABLED . . . . .	94
5.1.2.103 POWER_SMOOTH_CURRENT . . . . .	94
5.1.2.104 PWR_STATE_MAX . . . . .	94
5.1.2.105 PWR_STATE_NORM . . . . .	94
5.1.2.106 PWR_STATE_OFF . . . . .	94
5.1.2.107 PWR_STATE_REDUCT . . . . .	95
5.1.2.108 PWR_STATE_UNKNOWN . . . . .	95
5.1.2.109 REV_SENS_INV . . . . .	95
5.1.2.110 SETPOS_IGNORE_ENCODER . . . . .	95
5.1.2.111 SETPOS_IGNORE_POSITION . . . . .	95
5.1.2.112 STATE_ALARM . . . . .	95
5.1.2.113 STATE_BORDERS_SWAP_MISSET . . . . .	95
5.1.2.114 STATE_BRAKE . . . . .	95
5.1.2.115 STATE_BUTTON_LEFT . . . . .	95
5.1.2.116 STATE_BUTTON_RIGHT . . . . .	95
5.1.2.117 STATE_CONTR . . . . .	95
5.1.2.118 STATE_CONTROLLER_OVERHEAT . . . . .	95
5.1.2.119 STATE_CTP_ERROR . . . . .	96
5.1.2.120 STATE_DIG_SIGNAL . . . . .	96
5.1.2.121 STATE_EEPROM_CONNECTED . . . . .	96
5.1.2.122 STATE_ENC_A . . . . .	96
5.1.2.123 STATE_ENC_B . . . . .	96
5.1.2.124 STATE_ERRC . . . . .	96
5.1.2.125 STATE_ERRD . . . . .	96
5.1.2.126 STATE_ERRV . . . . .	96
5.1.2.127 STATE_GPIO_LEVEL . . . . .	96
5.1.2.128 STATE_GPIO_PINOUT . . . . .	96
5.1.2.129 STATE_HALL_A . . . . .	96
5.1.2.130 STATE_HALL_B . . . . .	96
5.1.2.131 STATE_HALL_C . . . . .	97
5.1.2.132 STATE_LEFT_EDGE . . . . .	97
5.1.2.133 STATE_LOW_USB_VOLTAGE . . . . .	97
5.1.2.134 STATE_OVERLOAD_POWER_CURRENT . . . . .	97
5.1.2.135 STATE_OVERLOAD_POWER_VOLTAGE . . . . .	97
5.1.2.136 STATE_OVERLOAD_USB_CURRENT . . . . .	97
5.1.2.137 STATE_OVERLOAD_USB_VOLTAGE . . . . .	97
5.1.2.138 STATE_POWER_OVERHEAT . . . . .	97
5.1.2.139 STATE_REV_SENSOR . . . . .	97
5.1.2.140 STATE_RIGHT_EDGE . . . . .	97



5.1.2.141	STATE_SECUR	97
5.1.2.142	STATE_SYNC_INPUT	97
5.1.2.143	STATE_SYNC_OUTPUT	98
5.1.2.144	SYNCIN_ENABLED	98
5.1.2.145	SYNCIN_GOTOPOSITION	98
5.1.2.146	SYNCIN_INVERT	98
5.1.2.147	SYNCOUT_ENABLED	98
5.1.2.148	SYNCOUT_IN_STEPS	98
5.1.2.149	SYNCOUT_INVERT	98
5.1.2.150	SYNCOUT_ONPERIOD	98
5.1.2.151	SYNCOUT_ONSTART	98
5.1.2.152	SYNCOUT_ONSTOP	98
5.1.2.153	SYNCOUT_STATE	98
5.1.2.154	UART_PARITY_BITS	98
5.1.2.155	WIND_A_STATE_ABSENT	99
5.1.2.156	WIND_A_STATE_MALFUNC	99
5.1.2.157	WIND_A_STATE_OK	99
5.1.2.158	WIND_A_STATE_UNKNOWN	99
5.1.2.159	WIND_B_STATE_ABSENT	99
5.1.2.160	WIND_B_STATE_MALFUNC	99
5.1.2.161	WIND_B_STATE_OK	99
5.1.2.162	WIND_B_STATE_UNKNOWN	99
5.1.2.163	XIMC_API	99
5.1.3	Typedef Documentation	99
5.1.3.1	logging_callback_t	99
5.1.4	Function Documentation	100
5.1.4.1	close_device	100
5.1.4.2	command_add_sync_in_action	100
5.1.4.3	command_clear_fram	100
5.1.4.4	command_eeread_settings	100
5.1.4.5	command_eesave_settings	100
5.1.4.6	command_home	101
5.1.4.7	command_left	101
5.1.4.8	command_loft	101
5.1.4.9	command_move	101
5.1.4.10	command_movr	102
5.1.4.11	command_power_off	102
5.1.4.12	command_read_settings	102
5.1.4.13	command_reset	102
5.1.4.14	command_right	102

5.1.4.15	<a href="#">command_save_settings</a>	103
5.1.4.16	<a href="#">command_sstp</a>	103
5.1.4.17	<a href="#">command_stop</a>	103
5.1.4.18	<a href="#">command_update_firmware</a>	103
5.1.4.19	<a href="#">command_zero</a>	103
5.1.4.20	<a href="#">enumerate_devices</a>	104
5.1.4.21	<a href="#">free_enumerate_devices</a>	104
5.1.4.22	<a href="#">get_accessories_settings</a>	104
5.1.4.23	<a href="#">get_analog_data</a>	104
5.1.4.24	<a href="#">get_bootloader_version</a>	104
5.1.4.25	<a href="#">get_brake_settings</a>	105
5.1.4.26	<a href="#">get_chart_data</a>	105
5.1.4.27	<a href="#">get_control_settings</a>	105
5.1.4.28	<a href="#">get_controller_name</a>	105
5.1.4.29	<a href="#">get_ctp_settings</a>	106
5.1.4.30	<a href="#">get_debug_read</a>	106
5.1.4.31	<a href="#">get_device_count</a>	106
5.1.4.32	<a href="#">get_device_information</a>	106
5.1.4.33	<a href="#">get_device_name</a>	107
5.1.4.34	<a href="#">get_edges_settings</a>	107
5.1.4.35	<a href="#">get_encoder_information</a>	107
5.1.4.36	<a href="#">get_encoder_settings</a>	107
5.1.4.37	<a href="#">get_engine_settings</a>	107
5.1.4.38	<a href="#">get_entype_settings</a>	108
5.1.4.39	<a href="#">get_enumerate_device_information</a>	108
5.1.4.40	<a href="#">get_enumerate_device_serial</a>	108
5.1.4.41	<a href="#">get_extio_settings</a>	108
5.1.4.42	<a href="#">get_feedback_settings</a>	109
5.1.4.43	<a href="#">get_firmware_version</a>	109
5.1.4.44	<a href="#">get_gear_information</a>	109
5.1.4.45	<a href="#">get_gear_settings</a>	109
5.1.4.46	<a href="#">get_hallsensor_information</a>	110
5.1.4.47	<a href="#">get_hallsensor_settings</a>	110
5.1.4.48	<a href="#">get_home_settings</a>	110
5.1.4.49	<a href="#">get_init_random</a>	110
5.1.4.50	<a href="#">get_joystick_settings</a>	110
5.1.4.51	<a href="#">get_motor_information</a>	111
5.1.4.52	<a href="#">get_motor_settings</a>	111
5.1.4.53	<a href="#">get_move_settings</a>	111
5.1.4.54	<a href="#">get_pid_settings</a>	111

5.1.4.55	<a href="#">get_position</a>	112
5.1.4.56	<a href="#">get_power_settings</a>	112
5.1.4.57	<a href="#">get_secure_settings</a>	112
5.1.4.58	<a href="#">get_serial_number</a>	112
5.1.4.59	<a href="#">get_stage_information</a>	112
5.1.4.60	<a href="#">get_stage_name</a>	113
5.1.4.61	<a href="#">get_stage_settings</a>	113
5.1.4.62	<a href="#">get_status</a>	113
5.1.4.63	<a href="#">get_status_calb</a>	113
5.1.4.64	<a href="#">get_sync_in_settings</a>	113
5.1.4.65	<a href="#">get_sync_out_settings</a>	114
5.1.4.66	<a href="#">get_uart_settings</a>	114
5.1.4.67	<a href="#">goto_firmware</a>	114
5.1.4.68	<a href="#">has_firmware</a>	115
5.1.4.69	<a href="#">logging_callback_stderr_narrow</a>	115
5.1.4.70	<a href="#">logging_callback_stderr_wide</a>	115
5.1.4.71	<a href="#">msec_sleep</a>	115
5.1.4.72	<a href="#">open_device</a>	115
5.1.4.73	<a href="#">probe_device</a>	115
5.1.4.74	<a href="#">service_command_updf</a>	116
5.1.4.75	<a href="#">set_accessories_settings</a>	116
5.1.4.76	<a href="#">set_brake_settings</a>	116
5.1.4.77	<a href="#">set_control_settings</a>	116
5.1.4.78	<a href="#">set_controller_name</a>	116
5.1.4.79	<a href="#">set_ctp_settings</a>	117
5.1.4.80	<a href="#">set_debug_write</a>	117
5.1.4.81	<a href="#">set_edges_settings</a>	117
5.1.4.82	<a href="#">set_encoder_information</a>	117
5.1.4.83	<a href="#">set_encoder_settings</a>	118
5.1.4.84	<a href="#">set_engine_settings</a>	118
5.1.4.85	<a href="#">set_entype_settings</a>	118
5.1.4.86	<a href="#">set_extio_settings</a>	118
5.1.4.87	<a href="#">set_feedback_settings</a>	119
5.1.4.88	<a href="#">set_gear_information</a>	119
5.1.4.89	<a href="#">set_gear_settings</a>	119
5.1.4.90	<a href="#">set_hallsensor_information</a>	119
5.1.4.91	<a href="#">set_hallsensor_settings</a>	120
5.1.4.92	<a href="#">set_home_settings</a>	120
5.1.4.93	<a href="#">set_joystick_settings</a>	120
5.1.4.94	<a href="#">set_logging_callback</a>	120

5.1.4.95	set_motor_information	121
5.1.4.96	set_motor_settings	121
5.1.4.97	set_move_settings	121
5.1.4.98	set_pid_settings	121
5.1.4.99	set_position	122
5.1.4.100	set_power_settings	122
5.1.4.101	set_secure_settings	122
5.1.4.102	set_serial_number	122
5.1.4.103	set_stage_information	122
5.1.4.104	set_stage_name	123
5.1.4.105	set_stage_settings	123
5.1.4.106	set_sync_in_settings	123
5.1.4.107	set_sync_out_settings	123
5.1.4.108	set_uart_settings	124
5.1.4.109	write_key	124
5.1.4.110	ximc_fix_usbser_sys	124
5.1.4.111	ximc_version	124

# Chapter 1

## Introduction

### 1.1 About

Congratulations on choosing XIMC multi-platform programming library! This document contains all information about XIMC library. It utilizes well known virtual COM-port interface, so you can use it on Windows 7, Windows Vista, Windows XP, Windows Server 2003, Windows 2000, Linux, Mac OS X. XIMC multi-platform programming library supports plug/unplug on the fly. One program can control one device. Multiple processes (programs) that control one device simultaneously are not allowed.

### 1.2 System requirements

#### 1.2.1 For rebuilding library

On Windows:

- Windows 2000 or later, 64-bit system (if compiling both architectures) or 32-bit system.
- Microsoft Visual C++ 2008 or later
- cygwin with tar, bison, flex installed

On Linux or FreeBSD:

- 64-bit or/and 32-bit system
- gcc 4 or later
- common autotools: autoconf, autoheader, aclocal, automake, autoreconf, libtool
- gmake
- doxygen - for building docs
- LaTeX distribution (TeX or TeXlive) - for building docs
- flex 2.5.30+
- bison
- mercurial (for building developer version from hg)

On Mac OS X:

- XCode 4

- doxygen
- mactex
- autotools
- mercurial (for building developer version from hg)

If mercurial is used, please enable 'purge' extension by adding to `~/.hgrc` following lines:

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

### 1.2.2 For using library

Supported operating systems (32 or 64 bit):

- Mac OS X 10.6
- Windows 2000 or later
- Autotools-compatible unix. Package is installed from sources.
- Linux debian-based 32 and 64 bit. DEB package is built against Debian Squeeze 7
- Linux debian-based ARM. DEB package is built on Ubuntu 14.04
- Linux rpm-based. RPM is built against OpenSUSE 12
- FreeBSD 9. Package is provided.

Build requirements:

- Windows: Microsoft Visual C++ 2008 or mingw (currently not supported)
- UNIX: gcc 4, gmake
- Mac OS X: XCode 4

## Chapter 2

# How to rebuild library

### 2.1 Building on generic UNIX

Generic version could be built with standard autotools.

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

Built files (library, headers, documentation) are installed to `./dist/local` directory. It is a generic developer build. Sometimes you need to specify additional parameters to command line for your machine. Please look to following OS sections.

### 2.2 Building on debian-based linux systems

Requirement: 64-bit and 32-bit debian system, ubuntu Typical set of packages: gcc, autotools, autoconf, libtool, dpkg-dev, flex, bison, doxygen, texlive, mercurial Full set of packages: debhelper g++ mercurial make 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 For ARM cross-compiling install gcc-arm-linux-gnueabi from your ARM toolchain.

It's required to match library and host architecture: 64-bit library can be built only at 64-bit host, 32-bit library - only at 32-bit host. ARM library is built with armhf cross-compiler gcc-arm-linux-gnueabi.

To build library and package invoke a script:

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

For ARM library replace 'libdeb' with 'libdebarm'.

Grab packages from `./ximc/deb` and locally installed binaries from `./dist/local`.

### 2.3 Building on redhat-based linux systems

Requirement: 64-bit redhat-based system (Fedora, Red Hat, SUSE) Typical set of packages: gcc, autotools, autoconf, libtool, flex, bison, doxygen, texlive, mercurial Full set of packages: 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

It's possible to build both 32- and 64-bit libraries on 64-bit host system. 64-bit library can't be built on 32-bit system.

To build library and package invoke a script:

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

Grab packages from `./ximc/rpm` and locally installed binaries from `./dist/local`.

## 2.4 Building on FreeBSD

Requirement: 64-bit or 32-bit FreeBSD Typical set of packages: gcc, autotools, autoconf, libtool, flex, bison, doxygen, teTeX, mercurial

It's required to match library and host architecture. Also you need to fix a `configure.ac` to exclude SOVER from the package name (freebsd does not use linux conventions on library versioning).

To build a library and package invoke following command. It requires sudo privileges for port installing and specially crafted `/usr/ports/local` tree. Consult a script for details.

```
$ ./build.sh libfreebsd
```

Grab packages from `./ximc/freebsd`.

## 2.5 Building on Mac OS X

To build and package a script invoke a script:

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

Built library (classical and framework), examples (classical and `.app`), documentation are located at `./ximc/macosx`, locally installed binaries from `./dist/local`.

## 2.6 Building on Windows

Requirements: 64-bit windows (build script builds both architectures), cygwin (must be installed to a default path), mercurial.

Invoke a script:

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

Grab packages from `./deb/win32` and `./deb/win64`

To build debug version of the library set environment variable "DEBUG" to "true" before running the build script.

## 2.7 Source code access

XIMC source codes are given under special request.



## Chapter 3

# How to use with...

Library usage can be examined from test application testapp. Non-C languages are supported because library supports stdcall calling convention and so can be used with a variety of languages.

C test project is located at 'examples/testapp' directory, C# test project - at 'examples/testcs', VB.NET - 'examples/testvbnet', Delphi 6 - 'examples/testdelphi', sample bindings for MATLAB - 'examples/testmatlab'

NOTE: SDK requires Microsoft Visual C++ Redistributable Package 9.0.307291 (provided with SDK - vc8redist.x86 or vc8redist.x64)

### 3.1 Usage with C

#### 3.1.1 Visual C++

Testapp can be built using testapp.sln. Library must be compiled with MS Visual C++ too, mingw-library isn't supported. Make sure that Microsoft Visual C++ Redistributable Package is installed.

Open solution examples/testapp/testapp.sln, build and run from the IDE.

#### 3.1.2 MinGW

MinGW is a port of GCC to win32 platform. It's required to install MinGW package. Currently not supported

MinGW-compiled testapp can be built with MS Visual C++ or mingw library.

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

Then copy library libximc.dll to current directory and launch testapp.exe.

#### 3.1.3 C++ Builder

First of all you should create C++ Builder-style import library. Visual C++ library is not compatible with BCB. Invoke:

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

Then compile test application:

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

### 3.1.4 XCode

Test app should be built with XCode project testapp.xcodeproj. Library is a Mac OS X framework, and at example application it's bundled inside testapp.app

Then launch application testapp.app and check activity output in Console.app.

### 3.1.5 GCC

Make sure that libximc (rpm, deb, freebsd package or tarball) is installed at your system. Installation of package should be performed with a package manager of operating system. On OS X plain dylib library and a framework is provided.

Note that user should belong to system group which allows access to a serial port (dip or serial, for example).

Test application can be built with the installed library with the following script:

```
$ make
```

In case of cross-compilation (target architecture differs from the current system architecture) feed -m64 or -m32 flag to compiler. On OS X it's needed to use -arch flag instead to build an universal binary. Please consult a compiler documentation.

Then launch the application as:

```
$ make run
```

Note: make run on OS X copies a library to the current directory. If you want to use library from the custom directory please be sure to specify LD\_LIBRARY\_PATH or DYLD\_LIBRARY\_PATH to the directory with the library.

## 3.2 .NET

Wrapper assembly for libximc.dll is wrappers/csharp/ximcnet.dll. It is provided with two different architectures and depends on .NET 2.0.

Test .NET applications for Visual Studio 2008 is located at testcs (for C#) and testvbnet (for VB.NET) respectively. Open solutions, build and run.

## 3.3 Delphi

Wrapper for libximc.dll is a unit wrappers/delphi/ximc.pas

Console test application for is located at testdelphi. Tested with Delphi 6 and only 32-bit version.

Just compile, place DLL near the executable and run program.

## 3.4 MATLAB

Sample MATLAB program testximc.m is provided at the directory examples/testmatlab. Change current directory to the examples/matlab and copy [ximc.h](#), libximc.dll, bindy.dll, xiwrapper.dll to the current directory. If you have 64-bit matlab, you must additionally copy libximc.thunk.pwin64.dll and ximc.m to the current directory. With 32-bit matlab launch

```
testximc32.m
```

With 64-bit matlab launch

testximc64.m

They differs only with library loading instructions.

## 3.5 Generic logging facility

If you want to turn on file logging, you should run the program that uses libximc library with the "XILOG" environment variable set to desired file name. This file will be opened for writing on the first log event and will be closed when the program which uses libximc terminates. Data which is sent to/received from the controller is logged along with port open and close events.

## Chapter 4

# Data Structure Documentation

### 4.1 accessories\_settings\_t Struct Reference

Additional accessories information.

#### Data Fields

- char [MagneticBrakeInfo](#) [25]  
*The manufacturer and the part number of magnetic brake, the maximum string length is 24 characters.*
- float [MBRatedVoltage](#)  
*Rated voltage for controlling the magnetic brake (B).*
- float [MBRatedCurrent](#)  
*Rated current for controlling the magnetic brake (A).*
- float [MBTorque](#)  
*Retention moment (mN m).*
- unsigned int [MBSettings](#)  
*Magnetic brake settings flags.*
- char [TemperatureSensorInfo](#) [25]  
*The manufacturer and the part number of the temperature sensor, the maximum string length: 24 characters.*
- float [TSMIn](#)  
*The minimum measured temperature (degrees Celsius) Data type: float.*
- float [TSMaX](#)  
*The maximum measured temperature (degrees Celsius) Data type: float.*
- float [TSGrad](#)  
*The temperature gradient (V/degrees Celsius).*
- unsigned int [TSSettings](#)  
*Temperature sensor settings flags.*
- unsigned int [LimitSwitchesSettings](#)  
*Temperature sensor settings flags.*

#### 4.1.1 Detailed Description

Additional accessories information.

See Also

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

### 4.1.2 Field Documentation

#### 4.1.2.1 unsigned int LimitSwitchesSettings

[Temperature sensor settings flags.](#)

#### 4.1.2.2 char MagneticBrakeInfo[25]

The manufacturer and the part number of magnetic brake, the maximum string length is 24 characters.

#### 4.1.2.3 float MBRatedCurrent

Rated current for controlling the magnetic brake (A).

Data type: float.

#### 4.1.2.4 float MBRatedVoltage

Rated voltage for controlling the magnetic brake (B).

Data type: float.

#### 4.1.2.5 unsigned int MBSettings

[Magnetic brake settings flags.](#)

#### 4.1.2.6 float MBTorque

Retention moment (mN m).

Data type: float.

#### 4.1.2.7 char TemperatureSensorInfo[25]

The manufacturer and the part number of the temperature sensor, the maximum string length: 24 characters.

#### 4.1.2.8 float TSGrad

The temperature gradient (V/degrees Celsius).

Data type: float.

#### 4.1.2.9 float TSMax

The maximum measured temperature (degrees Celsius) Data type: float.

#### 4.1.2.10 float TSMin

The minimum measured temperature (degrees Celsius) Data type: float.

#### 4.1.2.11 unsigned int TSSettings

[Temperature sensor settings flags.](#)

## 4.2 analog\_data\_t Struct Reference

Analog data.

### Data Fields

- unsigned int [A1Voltage\\_ADC](#)  
*"Voltage on pin 1 winding A" raw data from ADC.*
- unsigned int [A2Voltage\\_ADC](#)  
*"Voltage on pin 2 winding A" raw data from ADC.*
- unsigned int [B1Voltage\\_ADC](#)  
*"Voltage on pin 1 winding B" raw data from ADC.*
- unsigned int [B2Voltage\\_ADC](#)  
*"Voltage on pin 2 winding B" raw data from ADC.*
- unsigned int [SupVoltage\\_ADC](#)  
*"Voltage on the top of MOSFET full bridge" raw data from ADC.*
- unsigned int [ACurrent\\_ADC](#)  
*"Winding A current" raw data from ADC.*
- unsigned int [BCurrent\\_ADC](#)  
*"Winding B current" raw data from ADC.*
- unsigned int [FullCurrent\\_ADC](#)  
*"Full current" raw data from ADC.*
- unsigned int [Temp\\_ADC](#)  
*Voltage from temperature sensor, raw data from ADC.*
- unsigned int [Joy\\_ADC](#)  
*Joystick raw data from ADC.*
- unsigned int [Pot\\_ADC](#)  
*Voltage on analog input, raw data from ADC.*
- unsigned int [L5\\_ADC](#)  
*USB supply voltage after the current sense resistor, from ADC.*
- unsigned int [H5\\_ADC](#)  
*Power supply USB from ADC.*
- int [A1Voltage](#)  
*"Voltage on pin 1 winding A" calibrated data.*
- int [A2Voltage](#)  
*"Voltage on pin 2 winding A" calibrated data.*
- int [B1Voltage](#)  
*"Voltage on pin 1 winding B" calibrated data.*
- int [B2Voltage](#)  
*"Voltage on pin 2 winding B" calibrated data.*
- int [SupVoltage](#)  
*"Voltage on the top of MOSFET full bridge" calibrated data.*
- int [ACurrent](#)  
*"Winding A current" calibrated data.*
- int [BCurrent](#)  
*"Winding B current" calibrated data.*
- int [FullCurrent](#)  
*"Full current" calibrated data.*
- int [Temp](#)  
*Temperature, calibrated data.*

- int [Joy](#)  
*Joystick, calibrated data.*
- int [Pot](#)  
*Analog input, calibrated data.*
- int [L5](#)  
*USB supply voltage after the current sense resistor.*
- int [H5](#)  
*Power supply USB.*
- unsigned int **deprecated**
- int [R](#)  
*Motor winding resistance in mOhms(is only used with stepper motor).*
- int [L](#)  
*Motor winding pseudo inductance in uHn(is only used with stepper motor).*

### 4.2.1 Detailed Description

Analog data.

This structure contains raw analog data from ADC embedded on board. These data used for device testing and deep recalibration by manufacturer only.

See Also

[get\\_analog\\_data](#)  
[get\\_analog\\_data](#)

### 4.2.2 Field Documentation

#### 4.2.2.1 int A1Voltage

"Voltage on pin 1 winding A" calibrated data.

#### 4.2.2.2 unsigned int A1Voltage\_ADC

"Voltage on pin 1 winding A" raw data from ADC.

#### 4.2.2.3 int A2Voltage

"Voltage on pin 2 winding A" calibrated data.

#### 4.2.2.4 unsigned int A2Voltage\_ADC

"Voltage on pin 2 winding A" raw data from ADC.

#### 4.2.2.5 int ACurrent

"Winding A current" calibrated data.

#### 4.2.2.6 unsigned int ACurrent\_ADC

"Winding A current" raw data from ADC.

4.2.2.7 int B1Voltage

"Voltage on pin 1 winding B" calibrated data.

4.2.2.8 unsigned int B1Voltage\_ADC

"Voltage on pin 1 winding B" raw data from ADC.

4.2.2.9 int B2Voltage

"Voltage on pin 2 winding B" calibrated data.

4.2.2.10 unsigned int B2Voltage\_ADC

"Voltage on pin 2 winding B" raw data from ADC.

4.2.2.11 int BCurrent

"Winding B current" calibrated data.

4.2.2.12 unsigned int BCurrent\_ADC

"Winding B current" raw data from ADC.

4.2.2.13 int FullCurrent

"Full current" calibrated data.

4.2.2.14 unsigned int FullCurrent\_ADC

"Full current" raw data from ADC.

4.2.2.15 int Joy

Joystick, calibrated data.

Range: 0..10000

4.2.2.16 unsigned int Joy\_ADC

Joystick raw data from ADC.

4.2.2.17 int L

Motor winding pseudo inductance in uHn(is only used with stepper motor).

4.2.2.18 int L5

USB supply voltage after the current sense resistor.



4.2.2.19 unsigned int L5\_ADC

USB supply voltage after the current sense resistor, from ADC.

4.2.2.20 int Pot

Analog input, calibrated data.

Range: 0..10000

4.2.2.21 int R

Motor winding resistance in mOhms(is only used with stepper motor).

4.2.2.22 int SupVoltage

"Voltage on the top of MOSFET full bridge" calibrated data.

4.2.2.23 unsigned int SupVoltage\_ADC

"Voltage on the top of MOSFET full bridge" raw data from ADC.

4.2.2.24 int Temp

Temperature, calibrated data.

4.2.2.25 unsigned int Temp\_ADC

Voltage from temperature sensor, raw data from ADC.

## 4.3 brake\_settings\_t Struct Reference

Brake settings.

### Data Fields

- unsigned int [t1](#)  
*Time in ms between turn on motor power and turn off brake.*
- unsigned int [t2](#)  
*Time in ms between turn off brake and moving readiness.*
- unsigned int [t3](#)  
*Time in ms between motor stop and turn on brake.*
- unsigned int [t4](#)  
*Time in ms between turn on brake and turn off motor power.*
- unsigned int [BrakeFlags](#)  
*Brake settings flags.*

### 4.3.1 Detailed Description

Brake settings.

This structure contains parameters of brake control.

See Also

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

### 4.3.2 Field Documentation

#### 4.3.2.1 unsigned int BrakeFlags

[Brake settings flags](#).

#### 4.3.2.2 unsigned int t1

Time in ms between turn on motor power and turn off brake.

#### 4.3.2.3 unsigned int t2

Time in ms between turn off brake and moving readiness.

All moving commands will execute after this interval.

#### 4.3.2.4 unsigned int t3

Time in ms between motor stop and turn on brake.

#### 4.3.2.5 unsigned int t4

Time in ms between turn on brake and turn off motor power.

## 4.4 calibration\_t Struct Reference

Calibration companion structure.

### Data Fields

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

### 4.4.1 Detailed Description

Calibration companion structure.

## 4.5 chart\_data\_t Struct Reference

Additional device state.

### Data Fields

- int [WindingVoltageA](#)  
*In the case step motor, the voltage across the winding A; in the case of a brushless, the voltage on the first coil, in the case of the only DC.*
- int [WindingVoltageB](#)  
*In the case step motor, the voltage across the winding B; in case of a brushless, the voltage on the second winding, and in the case of DC is not used.*
- int [WindingVoltageC](#)  
*In the case of a brushless, the voltage on the third winding, in the case step motor and DC is not used.*
- int [WindingCurrentA](#)  
*In the case step motor, the current in the coil A; brushless if the current in the first coil, and in the case of a single DC.*
- int [WindingCurrentB](#)  
*In the case step motor, the current in the coil B; brushless if the current in the second coil, and in the case of DC is not used.*
- int [WindingCurrentC](#)  
*In the case of a brushless, the current in the third winding, in the case step motor and DC is not used.*
- unsigned int [Pot](#)  
*Analog input value in ten-thousandths.*
- unsigned int [Joy](#)  
*The joystick position in the ten-thousandths.*
- int [DutyCycle](#)  
*Duty cycle of PWM.*

### 4.5.1 Detailed Description

Additional device state.

This structure contains additional values such as winding's voltages, currents and temperature.

See Also

[get\\_chart\\_data](#)  
[get\\_chart\\_data](#)

### 4.5.2 Field Documentation

#### 4.5.2.1 int DutyCycle

Duty cycle of PWM.

#### 4.5.2.2 unsigned int Joy

The joystick position in the ten-thousandths.

Range: 0..10000

4.5.2.3 `unsigned int Pot`

Analog input value in ten-thousandths.

Range: 0..10000

4.5.2.4 `int WindingCurrentA`

In the case step motor, the current in the coil A; brushless if the current in the first coil, and in the case of a single DC.

4.5.2.5 `int WindingCurrentB`

In the case step motor, the current in the coil B; brushless if the current in the second coil, and in the case of DC is not used.

4.5.2.6 `int WindingCurrentC`

In the case of a brushless, the current in the third winding, in the case step motor and DC is not used.

4.5.2.7 `int WindingVoltageA`

In the case step motor, the voltage across the winding A; in the case of a brushless, the voltage on the first coil, in the case of the only DC.

4.5.2.8 `int WindingVoltageB`

In the case step motor, the voltage across the winding B; in case of a brushless, the voltage on the second winding, and in the case of DC is not used.

4.5.2.9 `int WindingVoltageC`

In the case of a brushless, the voltage on the third winding, in the case step motor and DC is not used.

## 4.6 `command_add_sync_in_action_calb_t` Struct Reference

### Data Fields

- float [Position](#)  
*Desired position or shift.*
- unsigned int [Time](#)  
*Time for which you want to achieve the desired position in microseconds.*

### 4.6.1 Field Documentation

4.6.1.1 `float Position`

Desired position or shift.

## 4.6.1.2 unsigned int Time

Time for which you want to achieve the desired position in microseconds.

4.7 `command_add_sync_in_action_t` Struct Reference

This command adds one element of the FIFO commands.

## Data Fields

- int [Position](#)  
*Desired position or shift (whole steps)*
- int [uPosition](#)  
*The fractional part of a position or shift in microsteps.*
- unsigned int [Time](#)  
*Time for which you want to achieve the desired position in microseconds.*

## 4.7.1 Detailed Description

This command adds one element of the FIFO commands.

See Also

[command\\_add\\_sync\\_in\\_action](#)

## 4.7.2 Field Documentation

## 4.7.2.1 unsigned int Time

Time for which you want to achieve the desired position in microseconds.

## 4.7.2.2 int uPosition

The fractional part of a position or shift in microsteps.

Is only used with stepper motor. Range: -255..255.

4.8 `control_settings_calb_t` Struct Reference

## Data Fields

- float [MaxSpeed](#) [10]  
*Array of speeds using with joystick and button control.*
- unsigned int [Timeout](#) [9]  
*timeout[i] is time in ms, after that max\_speed[i+1] is applying.*
- unsigned int [MaxClickTime](#)  
*Maximum click time.*
- unsigned int [Flags](#)  
*Control flags.*
- float [DeltaPosition](#)  
*Shift (delta) of position.*

### 4.8.1 Field Documentation

#### 4.8.1.1 unsigned int Flags

Control flags.

#### 4.8.1.2 unsigned int MaxClickTime

Maximum click time.

Prior to the expiration of this time the first speed isn't enabled.

#### 4.8.1.3 float MaxSpeed[10]

Array of speeds using with joystick and button control.

#### 4.8.1.4 unsigned int Timeout[9]

timeout[i] is time in ms, after that max\_speed[i+1] is applying.

It is using with buttons control only.

## 4.9 control\_settings\_t Struct Reference

Control settings.

### Data Fields

- unsigned int [MaxSpeed](#) [10]  
*Array of speeds (full step) using with joystick and button control.*
- unsigned int [uMaxSpeed](#) [10]  
*Array of speeds (1/256 microstep) using with joystick and button control.*
- unsigned int [Timeout](#) [9]  
*timeout[i] is time in ms, after that max\_speed[i+1] is applying.*
- unsigned int [MaxClickTime](#)  
*Maximum click time.*
- unsigned int [Flags](#)  
*Control flags.*
- int [DeltaPosition](#)  
*Shift (delta) of position.*
- int [uDeltaPosition](#)  
*Fractional part of the shift in micro steps.*

### 4.9.1 Detailed Description

Control settings.

This structure contains control parameters. When choosing CTL\_MODE=1 switches motor control with the joystick. In this mode, the joystick to the maximum engine tends Move at MaxSpeed [i], where i=0 if the previous use This mode is not selected another i. Buttons switch the room rate i. When CTL\_MODE=2 is switched on motor control using the Left / right. When you click on the button motor starts to move in the appropriate direction at a speed MaxSpeed [0], at the end of time Timeout[i] motor move at a speed MaxSpeed [i+1]. at Transition from MaxSpeed

[i] on MaxSpeed [i+1] to acceleration, as usual. The figure above shows the sensitivity of the joystick feature on its position.

See Also

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

## 4.9.2 Field Documentation

### 4.9.2.1 unsigned int Flags

[Control flags](#).

### 4.9.2.2 unsigned int MaxClickTime

Maximum click time.

Prior to the expiration of this time the first speed isn't enabled.

### 4.9.2.3 unsigned int MaxSpeed[10]

Array of speeds (full step) using with joystick and button control.

Range: 0..100000.

### 4.9.2.4 unsigned int Timeout[9]

timeout[i] is time in ms, after that max.speed[i+1] is applying.

It is using with buttons control only.

### 4.9.2.5 int uDeltaPosition

Fractional part of the shift in micro steps.

Is only used with stepper motor. Range: -255..255.

### 4.9.2.6 unsigned int uMaxSpeed[10]

Array of speeds (1/256 microstep) using with joystick and button control.

## 4.10 controller\_name\_t Struct Reference

Controller user name and flags of setting.

### Data Fields

- char [ControllerName](#) [17]  
*User controller name.*
- unsigned int [CtrlFlags](#)  
*Flags of internal controller settings.*

#### 4.10.1 Detailed Description

Controller user name and flags of setting.

See Also

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

#### 4.10.2 Field Documentation

##### 4.10.2.1 char ControllerName[17]

User conroller name.

Can be set by user for his/her convinience. Max string length: 16 chars.

##### 4.10.2.2 unsigned int CtrlFlags

[Flags of internal controller settings.](#)

### 4.11 ctp\_settings\_t Struct Reference

Control position settings(is only used with stepper motor).

#### Data Fields

- unsigned int [CTPMinError](#)  
*Minimum contrast steps from step motor encoder position, wich set STATE\_CTP\_ERROR flag.*
- unsigned int [CTPFlags](#)  
*Position control flags.*

#### 4.11.1 Detailed Description

Control position settings(is only used with stepper motor).

When controlling the step motor with encoder (CTP\_BASE 0) it is possible to detect the loss of steps. The controller knows the number of steps per revolution (GENG :: StepsPerRev) and the encoder resolution (GFBS :: IPT). When the control (flag CTP\_ENABLED), the controller stores the current position in the footsteps of SM and the current position of the encoder. Further, at each step of the position encoder is converted into steps and if the difference is greater CTPMinError, a flag STATE\_CTP\_ERROR and set ALARM state. When controlling the step motor with speed sensor (CTP\_BASE 1), the position is controlled by him. The active edge of input clock controller stores the current value of steps. Further, at each turn checks how many steps shifted. When a mismatch CTPMinError a flag STATE\_CTP\_ERROR and set ALARM state.

See Also

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

#### 4.11.2 Field Documentation

##### 4.11.2.1 unsigned int CTPFlags

[Position control flags.](#)



## 4.11.2.2 unsigned int CTPMinError

Minimum contrast steps from step motor encoder position, wich set STATE\_CTP\_ERROR flag.

Measured in steps step motor.

## 4.12 debug\_read\_t Struct Reference

Debug data.

### Data Fields

- unsigned int [DebugData](#) [128]  
*Arbitrary debug data.*

### 4.12.1 Detailed Description

Debug data.

These data are used for device debugging by manufacturer only.

See Also

[get.debug.read](#)

### 4.12.2 Field Documentation

## 4.12.2.1 unsigned int DebugData[128]

Arbitrary debug data.

## 4.13 debug\_write\_t Struct Reference

Debug data.

### Data Fields

- unsigned int [DebugData](#) [128]  
*Arbitrary debug data.*

### 4.13.1 Detailed Description

Debug data.

These data are used for device debugging by manufacturer only.

See Also

[set.debug.write](#)

### 4.13.2 Field Documentation

#### 4.13.2.1 unsigned int DebugData[128]

Arbitrary debug data.

## 4.14 device\_information\_t Struct Reference

Read command controller information.

### Data Fields

- char [Manufacturer](#) [5]  
*Manufacturer.*
- char [ManufacturerId](#) [3]  
*Manufacturer id.*
- char [ProductDescription](#) [9]  
*Product description.*
- unsigned int [Major](#)  
*The major number of the hardware version.*
- unsigned int [Minor](#)  
*Minor number of the hardware version.*
- unsigned int [Release](#)  
*Number of edits this release of hardware.*

### 4.14.1 Detailed Description

Read command controller information.

The controller responds to this command in any state. Manufacturer field for all XI \*\* devices should contain the string "XIMC" (validation is performed on it) The remaining fields contain information about the device.

See Also

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

### 4.14.2 Field Documentation

#### 4.14.2.1 unsigned int Major

The major number of the hardware version.

#### 4.14.2.2 unsigned int Minor

Minor number of the hardware version.

#### 4.14.2.3 unsigned int Release

Number of edits this release of hardware.

## 4.15 device\_network\_information\_t Struct Reference

TODO.

### Data Fields

- uint32\_t [ipv4](#)  
*IPv4 address, passed in host byte order.*
- char [nodename](#) [16]  
*TODO.*
- uint32\_t [axis\\_state](#)  
*TODO.*
- char [locker\\_username](#) [16]  
*TODO.*
- char [locker\\_nodename](#) [16]  
*TODO.*
- time\_t [locked\\_time](#)  
*TODO.*

### 4.15.1 Detailed Description

TODO.

## 4.16 edges\_settings\_calb\_t Struct Reference

### Data Fields

- unsigned int [BorderFlags](#)  
*Border flags.*
- unsigned int [EnderFlags](#)  
*Limit switches flags.*
- float [LeftBorder](#)  
*Left border position, used if BORDER.IS.ENCODER flag is set.*
- float [RightBorder](#)  
*Right border position, used if BORDER.IS.ENCODER flag is set.*

### 4.16.1 Field Documentation

#### 4.16.1.1 unsigned int BorderFlags

[Border flags.](#)

#### 4.16.1.2 unsigned int EnderFlags

[Limit switches flags.](#)

#### 4.16.1.3 float LeftBorder

Left border position, used if BORDER.IS.ENCODER flag is set.

## 4.16.1.4 float RightBorder

Right border position, used if BORDER.IS.ENCODER flag is set.

## 4.17 edges\_settings\_t Struct Reference

Edges settings.

## Data Fields

- unsigned int [BorderFlags](#)  
*Border flags.*
- unsigned int [EnderFlags](#)  
*Limit switches flags.*
- int [LeftBorder](#)  
*Left border position, used if BORDER.IS.ENCODER flag is set.*
- int [uLeftBorder](#)  
*Left border position in 1/256 microsteps(used with stepper motor only).*
- int [RightBorder](#)  
*Right border position, used if BORDER.IS.ENCODER flag is set.*
- int [uRightBorder](#)  
*Right border position in 1/256 microsteps.*

## 4.17.1 Detailed Description

Edges settings.

This structure contains border and limit switches settings. Please load new engine settings when you change positioner etc. Please note that wrong engine settings lead to device malfunction, can lead to irreversible damage of board.

See Also

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

## 4.17.2 Field Documentation

## 4.17.2.1 unsigned int BorderFlags

[Border flags.](#)

## 4.17.2.2 unsigned int EnderFlags

[Limit switches flags.](#)

## 4.17.2.3 int LeftBorder

Left border position, used if BORDER.IS.ENCODER flag is set.

## 4.17.2.4 int RightBorder

Right border position, used if BORDER\_IS\_ENCODER flag is set.

## 4.17.2.5 int uLeftBorder

Left border position in 1/256 microsteps(used with stepper motor only).

Range: -255..255.

## 4.17.2.6 int uRightBorder

Right border position in 1/256 microsteps.

Used with stepper motor only. Range: -255..255.

## 4.18 encoder\_information\_t Struct Reference

Encoder information.

### Data Fields

- char [Manufacturer](#) [17]  
*Manufacturer.*
- char [PartNumber](#) [25]  
*Series and PartNumber.*

### 4.18.1 Detailed Description

Encoder information.

See Also

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

### 4.18.2 Field Documentation

## 4.18.2.1 char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

## 4.18.2.2 char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

## 4.19 encoder\_settings\_t Struct Reference

Encoder settings.

### Data Fields

- float [MaxOperatingFrequency](#)  
*Max operation frequency (kHz).*
- float [SupplyVoltageMin](#)  
*Minimum supply voltage (V).*
- float [SupplyVoltageMax](#)  
*Maximum supply voltage (V).*
- float [MaxCurrentConsumption](#)  
*Max current consumption (mA).*
- unsigned int [PPR](#)  
*The number of counts per revolution.*
- unsigned int [EncoderSettings](#)  
*Encoder settings flags.*

### 4.19.1 Detailed Description

Encoder settings.

See Also

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

### 4.19.2 Field Documentation

#### 4.19.2.1 unsigned int EncoderSettings

[Encoder settings flags.](#)

#### 4.19.2.2 float MaxCurrentConsumption

Max current consumption (mA).

Data type: float.

#### 4.19.2.3 float MaxOperatingFrequency

Max operation frequency (kHz).

Data type: float.

#### 4.19.2.4 float SupplyVoltageMax

Maximum supply voltage (V).

Data type: float.

4.19.2.5 float SupplyVoltageMin

Minimum supply voltage (V).

Data type: float.

## 4.20 engine\_settings\_calb\_t Struct Reference

### Data Fields

- unsigned int [NomVoltage](#)  
*Rated voltage.*
- unsigned int [NomCurrent](#)  
*Rated current.*
- float [NomSpeed](#)  
*Nominal speed.*
- unsigned int [EngineFlags](#)  
*Flags of engine settings.*
- float [Antiplay](#)  
*Number of pulses or steps for backlash (play) compensation procedure.*
- unsigned int [MicrostepMode](#)  
*Flags of microstep mode.*
- unsigned int [StepsPerRev](#)  
*Number of full steps per revolution(Used with stepper motor only).*

### 4.20.1 Field Documentation

4.20.1.1 float Antiplay

Number of pulses or steps for backlash (play) compensation procedure.

Used if ENGINE\_ANTIPLAY flag is set.

4.20.1.2 unsigned int EngineFlags

[Flags of engine settings.](#)

4.20.1.3 unsigned int MicrostepMode

[Flags of microstep mode.](#)

4.20.1.4 unsigned int NomCurrent

Rated current.

Controller will keep current consumed by motor below this value if ENGINE\_LIMIT\_CURR flag is set. Range: 15..8000

4.20.1.5 float NomSpeed

Nominal speed.

Controller will keep motor speed below this value if ENGINE\_LIMIT\_RPM flag is set.

## 4.20.1.6 unsigned int NomVoltage

Rated voltage.

Controller will keep the voltage drop on motor below this value if ENGINE\_LIMIT\_VOLT flag is set(Used with DC only). Range: 1..65535

## 4.20.1.7 unsigned int StepsPerRev

Number of full steps per revolution(Used with stepper motor only).

Range: 1..65535.

## 4.21 engine\_settings\_t Struct Reference

Engine settings.

## Data Fields

- unsigned int [NomVoltage](#)  
*Rated voltage.*
- unsigned int [NomCurrent](#)  
*Rated current.*
- unsigned int [NomSpeed](#)  
*Nominal speed (in whole steps/s or rpm for DC and stepper motor as a master encoder).*
- unsigned int [uNomSpeed](#)  
*The fractional part of a nominal speed in microsteps (is only used with stepper motor).*
- unsigned int [EngineFlags](#)  
*Flags of engine settings.*
- int [Antiplay](#)  
*Number of pulses or steps for backlash (play) compensation procedure.*
- unsigned int [MicrostepMode](#)  
*Flags of microstep mode.*
- unsigned int [StepsPerRev](#)  
*Number of full steps per revolution(Used with stepper motor only).*

## 4.21.1 Detailed Description

Engine settings.

This structure contains useful motor settings. These settings specify motor shaft movement algorithm, list of limitations and rated characteristics. All boards are supplied with standart set of engine setting on controller's flash memory. Please load new engine settings when you change motor, encoder, positioner etc. Please note that wrong engine settings lead to device malfunction, can lead to irreversible damage of board.

See Also

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



### 4.21.2 Field Documentation

#### 4.21.2.1 int Antiplay

Number of pulses or steps for backlash (play) compensation procedure.

Used if ENGINE\_ANTIPLAY flag is set.

#### 4.21.2.2 unsigned int EngineFlags

[Flags of engine settings.](#)

#### 4.21.2.3 unsigned int MicrostepMode

[Flags of microstep mode.](#)

#### 4.21.2.4 unsigned int NomCurrent

Rated current.

Controller will keep current consumed by motor below this value if ENGINE\_LIMIT\_CURR flag is set. Range: 15..8000

#### 4.21.2.5 unsigned int NomSpeed

Nominal speed (in whole steps/s or rpm for DC and stepper motor as a master encoder).

Controller will keep motor shaft RPM below this value if ENGINE\_LIMIT\_RPM flag is set. Range: 1..100000.

#### 4.21.2.6 unsigned int NomVoltage

Rated voltage.

Controller will keep the voltage drop on motor below this value if ENGINE\_LIMIT\_VOLT flag is set(Used with DC only). Range: 1..65535

#### 4.21.2.7 unsigned int StepsPerRev

Number of full steps per revolution(Used with stepper motor only).

Range: 1..65535.

#### 4.21.2.8 unsigned int uNomSpeed

The fractional part of a nominal speed in microsteps (is only used with stepper motor).

## 4.22 entype\_settings\_t Struct Reference

Engine type and driver type settings.

## Data Fields

- unsigned int [EngineType](#)  
*Flags of engine type.*
- unsigned int [DriverType](#)  
*Flags of driver type.*

## 4.22.1 Detailed Description

Engine type and driver type settings.

## Parameters

<i>id</i>	an identifier of device
<i>EngineType</i>	engine type
<i>DriverType</i>	driver type

## See Also

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

## 4.22.2 Field Documentation

## 4.22.2.1 unsigned int DriverType

[Flags of driver type.](#)

## 4.22.2.2 unsigned int EngineType

[Flags of engine type.](#)

## 4.23 extio\_settings\_t Struct Reference

EXTIO settings.

## Data Fields

- unsigned int [EXTIOSetupFlags](#)  
*External IO setup flags.*
- unsigned int [EXTIOModeFlags](#)  
*External IO mode flags.*

## 4.23.1 Detailed Description

EXTIO settings.

This structure contains all EXTIO settings. By default input event are signalled through rising front and output states are signalled by high logic state.

See Also

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

### 4.23.2 Field Documentation

#### 4.23.2.1 unsigned int EXTIOModeFlags

[External IO mode flags.](#)

#### 4.23.2.2 unsigned int EXTIOSetupFlags

[External IO setup flags.](#)

## 4.24 feedback\_settings\_t Struct Reference

Feedback settings.

### Data Fields

- unsigned int [IPS](#)  
*The number of measured counts per revolution encoder.*
- unsigned int [FeedbackType](#)  
*Feedback type.*
- unsigned int [FeedbackFlags](#)  
*Describes feedback flags.*
- unsigned int [HallSPR](#)  
*The number of hall steps per revolution.*
- int [HallShift](#)  
*Phase shift between output signal on BLDC engine and hall sensor input(0 - when only active the Hall sensor, the output state is a positive voltage on the winding A and a negative voltage on the winding B).*

### 4.24.1 Detailed Description

Feedback settings.

This structure contains feedback settings.

See Also

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

### 4.24.2 Field Documentation

#### 4.24.2.1 unsigned int FeedbackFlags

[Describes feedback flags.](#)

4.24.2.2 unsigned int FeedbackType

[Feedback type.](#)

4.24.2.3 int HallShift

Phase shift between output signal on BLDC engine and hall sensor input(0 - when only active the Hall sensor, the output state is a positive voltage on the winding A and a negative voltage on the winding B).

4.24.2.4 unsigned int HallSPR

The number of hall steps per revolution.

## 4.25 gear\_information\_t Struct Reference

Gear information.

### Data Fields

- char [Manufacturer](#) [17]  
*Manufacturer.*
- char [PartNumber](#) [25]  
*Series and PartNumber.*

### 4.25.1 Detailed Description

Gear information.

See Also

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

### 4.25.2 Field Documentation

4.25.2.1 char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

4.25.2.2 char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

## 4.26 gear\_settings\_t Struct Reference

Gear settings.

## Data Fields

- float [ReductionIn](#)  
*Input reduction coefficient.*
- float [ReductionOut](#)  
*Output reduction coefficient.*
- float [RatedInputTorque](#)  
*Max continuous torque (N m).*
- float [RatedInputSpeed](#)  
*Max speed on the input shaft (rpm).*
- float [MaxOutputBacklash](#)  
*Output backlash of the reduction gear(degree).*
- float [InputInertia](#)  
*Equivalent input gear inertia (g cm2).*
- float [Efficiency](#)  
*Reduction gear efficiency (%).*

### 4.26.1 Detailed Description

Gear settings.

See Also

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

### 4.26.2 Field Documentation

#### 4.26.2.1 float Efficiency

Reduction gear efficiency (%).

Data type: float.

#### 4.26.2.2 float InputInertia

Equivalent input gear inertia (g cm2).

Data type: float.

#### 4.26.2.3 float MaxOutputBacklash

Output backlash of the reduction gear(degree).

Data type: float.

#### 4.26.2.4 float RatedInputSpeed

Max speed on the input shaft (rpm).

Data type: float.

## 4.26.2.5 float RatedInputTorque

Max continuous torque (N m).

Data type: float.

## 4.26.2.6 float ReductionIn

Input reduction coefficient.

(Output = (ReductionOut / ReductionIn) \* Input) Data type: float.

## 4.26.2.7 float ReductionOut

Output reduction coefficient.

(Output = (ReductionOut / ReductionIn) \* Input) Data type: float.

## 4.27 get\_position\_calb\_t Struct Reference

## Data Fields

- float [Position](#)  
*The position in the engine.*
- long\_t [EncPosition](#)  
*Encoder position.*

## 4.27.1 Field Documentation

## 4.27.1.1 long\_t EncPosition

Encoder position.

## 4.27.1.2 float Position

The position in the engine.

## 4.28 get\_position\_t Struct Reference

Position information.

## Data Fields

- int [Position](#)  
*The position of the whole steps in the engine.*
- int [uPosition](#)  
*Microstep position is only used with stepper motors.*
- long\_t [EncPosition](#)  
*Encoder position.*

#### 4.28.1 Detailed Description

Position information.

Useful structure that contains position value in steps and micro for stepper motor and encoder steps of all engines.

See Also

[get\\_position](#)

#### 4.28.2 Field Documentation

##### 4.28.2.1 long\_t EncPosition

Encoder position.

### 4.29 hallsensor\_information\_t Struct Reference

Hall sensor information.

#### Data Fields

- char [Manufacturer](#) [17]  
*Manufacturer.*
- char [PartNumber](#) [25]  
*Series and PartNumber.*

#### 4.29.1 Detailed Description

Hall sensor information.

See Also

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

#### 4.29.2 Field Documentation

##### 4.29.2.1 char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

##### 4.29.2.2 char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

### 4.30 hallsensor\_settings\_t Struct Reference

Hall sensor settings.

## Data Fields

- float [MaxOperatingFrequency](#)  
*Max operation frequency (kHz).*
- float [SupplyVoltageMin](#)  
*Minimum supply voltage (V).*
- float [SupplyVoltageMax](#)  
*Maximum supply voltage (V).*
- float [MaxCurrentConsumption](#)  
*Max current consumption (mA).*
- unsigned int [PPR](#)  
*The number of counts per revolution.*

## 4.30.1 Detailed Description

Hall sensor settings.

See Also

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

## 4.30.2 Field Documentation

## 4.30.2.1 float MaxCurrentConsumption

Max current consumption (mA).

Data type: float.

## 4.30.2.2 float MaxOperatingFrequency

Max operation frequency (kHz).

Data type: float.

## 4.30.2.3 float SupplyVoltageMax

Maximum supply voltage (V).

Data type: float.

## 4.30.2.4 float SupplyVoltageMin

Minimum supply voltage (V).

Data type: float.

## 4.31 home\_settings\_calb\_t Struct Reference

## Data Fields

- float [FastHome](#)



- *Speed used for first motion.*
- float [SlowHome](#)
  - *Speed used for second motion.*
- float [HomeDelta](#)
  - *Distance from break point.*
- unsigned int [HomeFlags](#)
  - *Home settings flags.*

#### 4.31.1 Field Documentation

##### 4.31.1.1 float FastHome

Speed used for first motion.

##### 4.31.1.2 float HomeDelta

Distance from break point.

##### 4.31.1.3 unsigned int HomeFlags

[Home settings flags.](#)

##### 4.31.1.4 float SlowHome

Speed used for second motion.

## 4.32 home\_settings\_t Struct Reference

Position calibration settings.

### Data Fields

- unsigned int [FastHome](#)
  - *Speed used for first motion.*
- unsigned int [uFastHome](#)
  - *Part of the speed for first motion, microsteps.*
- unsigned int [SlowHome](#)
  - *Speed used for second motion.*
- unsigned int [uSlowHome](#)
  - *Part of the speed for second motion, microsteps.*
- int [HomeDelta](#)
  - *Distance from break point.*
- int [uHomeDelta](#)
  - *Part of the delta distance, microsteps.*
- unsigned int [HomeFlags](#)
  - *Home settings flags.*

#### 4.32.1 Detailed Description

Position calibration settings.

This structure contains settings used in position calibrating. It specify behaviour of calibrating position.

See Also

[get\\_home\\_settings](#)  
[set\\_home\\_settings](#)  
[command\\_home](#)  
[get\\_home\\_settings](#), [set\\_home\\_settings](#)

#### 4.32.2 Field Documentation

##### 4.32.2.1 unsigned int FastHome

Speed used for first motion.

Range: 0..100000.

##### 4.32.2.2 int HomeDelta

Distance from break point.

##### 4.32.2.3 unsigned int HomeFlags

[Home settings flags](#).

##### 4.32.2.4 unsigned int SlowHome

Speed used for second motion.

Range: 0..100000.

##### 4.32.2.5 unsigned int uFastHome

Part of the speed for first motion, microsteps.

##### 4.32.2.6 int uHomeDelta

Part of the delta distance, microsteps.

Range: -255..255.

##### 4.32.2.7 unsigned int uSlowHome

Part of the speed for second motion, microsteps.

### 4.33 init\_random\_t Struct Reference

Random key.

## Data Fields

- unsigned int [key](#) [16]  
*Random key.*

### 4.33.1 Detailed Description

Random key.

Structure that contains random key used in encryption of WKEY and SSER command contents.

See Also

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

### 4.33.2 Field Documentation

#### 4.33.2.1 unsigned int key[16]

Random key.

## 4.34 joystick\_settings\_t Struct Reference

Joystick settings.

## Data Fields

- unsigned int [JoyLowEnd](#)  
*Joystick lower end position.*
- unsigned int [JoyCenter](#)  
*Joystick center position.*
- unsigned int [JoyHighEnd](#)  
*Joystick higher end position.*
- unsigned int [ExpFactor](#)  
*Exponential nonlinearity factor.*
- unsigned int [DeadZone](#)  
*Joystick dead zone.*
- unsigned int [JoyFlags](#)  
*Joystick flags.*

### 4.34.1 Detailed Description

Joystick settings.

This structure contains joystick parameters. If joystick position is outside DeadZone limits from the central position a movement with speed, defined by the joystick DeadZone edge to 100% deviation, begins. Joystick positions inside DeadZone limits correspond to zero speed (soft stop of motion) and positions beyond Low and High limits correspond MaxSpeed [i] or -MaxSpeed [i] (see command SCTL), where i = 0 by default and can be changed with left/right buttons (see command SCTL). If next speed in list is zero (both integer and microstep parts), the button press is ignored. First speed in list shouldn't be zero. The relationship between the deviation and the rate is exponential, allowing no switching speed combine high mobility and accuracy.

See Also

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

#### 4.34.2 Field Documentation

##### 4.34.2.1 unsigned int DeadZone

Joystick dead zone.

##### 4.34.2.2 unsigned int ExpFactor

Exponential nonlinearity factor.

##### 4.34.2.3 unsigned int JoyCenter

Joystick center position.

Range: 0..10000.

##### 4.34.2.4 unsigned int JoyFlags

[Joystick flags](#).

##### 4.34.2.5 unsigned int JoyHighEnd

Joystick higher end position.

Range: 0..10000.

##### 4.34.2.6 unsigned int JoyLowEnd

Joystick lower end position.

Range: 0..10000.

## 4.35 motor\_information\_t Struct Reference

motor information.

### Data Fields

- char [Manufacturer](#) [17]  
*Manufacturer.*
- char [PartNumber](#) [25]  
*Series and PartNumber.*

### 4.35.1 Detailed Description

motor information.

See Also

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

### 4.35.2 Field Documentation

#### 4.35.2.1 char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

#### 4.35.2.2 char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

## 4.36 motor\_settings\_t Struct Reference

motor settings.

### Data Fields

- unsigned int [MotorType](#)  
*Motor Type flags.*
- unsigned int [ReservedField](#)  
*Reserved.*
- unsigned int [Poles](#)  
*Number of pole pairs for DC or BLDC motors or number of steps per rotation for stepper motor.*
- unsigned int [Phases](#)  
*Number of phases for BLDC motors.*
- float [NominalVoltage](#)  
*Nominal voltage on winding (V).*
- float [NominalCurrent](#)  
*Maximum direct current in winding for DC and BLDC engines, nominal current in windings for stepper motor (A).*
- float [NominalSpeed](#)  
*Nominal speed(rpm).*
- float [NominalTorque](#)  
*Nominal torque(mNm).*
- float [NominalPower](#)  
*Nominal power(W).*
- float [WindingResistance](#)  
*Resistance of windings for DC engine, each of two windings for stepper motor or each of there windings for BLDC engine(Ohm).*
- float [WindingInductance](#)

*Inductance of windings for DC engine, each of two windings for stepper motor or each of there windings for BLDC engine(mH).*

- float [RotorInertia](#)

*Rotor inertia(g cm2).*

- float [StallTorque](#)

*Torque hold position for a stepper motor or torque at a motionless rotor for other types of engines (mN m).*

- float [DetentTorque](#)

*Holding torque position with un-powered coils (mN m).*

- float [TorqueConstant](#)

*Torque constant, which determines the aspect ratio of maximum moment of force from the rotor current flowing in the coil (mN m / A).*

- float [SpeedConstant](#)

*Velocity constant, which determines the value or amplitude of the induced voltage on the motion of DC or BLDC motor (rpm / V) or stepper motor (steps/s / V).*

- float [SpeedTorqueGradient](#)

*Speed torque gradient (rpm / mN m).*

- float [MechanicalTimeConstant](#)

*Mechanical time constant (ms).*

- float [MaxSpeed](#)

*The maximum speed for stepper motors (steps/s) or DC and BLDC motors (rmp).*

- float [MaxCurrent](#)

*The maximum current in the winding (A).*

- float [MaxCurrentTime](#)

*Safe duration of overcurrent in the winding (ms).*

- float [NoLoadCurrent](#)

*The current consumption in idle mode (A).*

- float [NoLoadSpeed](#)

*Idle speed (rpm).*

#### 4.36.1 Detailed Description

motor settings.

See Also

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

#### 4.36.2 Field Documentation

##### 4.36.2.1 float DetentTorque

Holding torque position with un-powered coils (mN m).

Data type: float.

##### 4.36.2.2 float MaxCurrent

The maximum current in the winding (A).

Data type: float.

## 4.36.2.3 float MaxCurrentTime

Safe duration of overcurrent in the winding (ms).

Data type: float.

## 4.36.2.4 float MaxSpeed

The maximum speed for stepper motors (steps/s) or DC and BLDC motors (rpm).

Data type: float.

## 4.36.2.5 float MechanicalTimeConstant

Mechanical time constant (ms).

Data type: float.

## 4.36.2.6 unsigned int MotorType

[Motor Type flags.](#)

## 4.36.2.7 float NoLoadCurrent

The current consumption in idle mode (A).

Used for DC and BLDC motors. Data type: float.

## 4.36.2.8 float NoLoadSpeed

Idle speed (rpm).

Used for DC and BLDC motors. Data type: float.

## 4.36.2.9 float NominalCurrent

Maximum direct current in winding for DC and BLDC engines, nominal current in windings for stepper motor (A).

Data type: float.

## 4.36.2.10 float NominalPower

Nominal power(W).

Used for DC and BLDC engine. Data type: float.

## 4.36.2.11 float NominalSpeed

Nominal speed(rpm).

Used for DC and BLDC engine. Data type: float.

## 4.36.2.12 float NominalTorque

Nominal torque(mN m).

Used for DC and BLDC engine. Data type: float.

## 4.36.2.13 float NominalVoltage

Nominal voltage on winding (B).

Data type: float

## 4.36.2.14 unsigned int Phases

Number of phases for BLDC motors.

## 4.36.2.15 unsigned int Poles

Number of pole pairs for DC or BLDC motors or number of steps per rotation for stepper motor.

## 4.36.2.16 float RotorInertia

Rotor inertia(g cm<sup>2</sup>).

Data type: float.

## 4.36.2.17 float SpeedConstant

Velocity constant, which determines the value or amplitude of the induced voltage on the motion of DC or BLDC motor (rpm / V) or stepper motor (steps/s / V).

Data type: float.

## 4.36.2.18 float SpeedTorqueGradient

Speed torque gradient (rpm / mN m).

Data type: float.

## 4.36.2.19 float StallTorque

Torque hold position for a stepper motor or torque at a motionless rotor for other types of engines (mN m).

Data type: float.

## 4.36.2.20 float TorqueConstant

Torque constant, which determines the aspect ratio of maximum moment of force from the rotor current flowing in the coil (mN m / A).

Used mainly for DC motors. Data type: float.



## 4.36.2.21 float WindingInductance

Inductance of windings for DC engine, each of two windings for stepper motor or each of there windings for BLDC engine(mH).

Data type: float.

## 4.36.2.22 float WindingResistance

Resistance of windings for DC engine, each of two windings for stepper motor or each of there windings for BLDC engine(Ohm).

Data type: float.

## 4.37 move\_settings\_calb\_t Struct Reference

## Data Fields

- float [Speed](#)  
*Target speed.*
- float [Accel](#)  
*Motor shaft acceleration, steps/s<sup>2</sup>(stepper motor) or RPM/s(DC).*
- float [Decel](#)  
*Motor shaft deceleration, steps/s<sup>2</sup>(stepper motor) or RPM/s(DC).*
- float [AntiplaySpeed](#)  
*Speed in antiplay mode.*

## 4.37.1 Field Documentation

## 4.37.1.1 float Accel

Motor shaft acceleration, steps/s<sup>2</sup>(stepper motor) or RPM/s(DC).

## 4.37.1.2 float AntiplaySpeed

Speed in antiplay mode.

## 4.37.1.3 float Decel

Motor shaft deceleration, steps/s<sup>2</sup>(stepper motor) or RPM/s(DC).

## 4.37.1.4 float Speed

Target speed.

## 4.38 move\_settings\_t Struct Reference

Move settings.

## Data Fields

- unsigned int [Speed](#)  
*Target speed (for stepper motor: steps/s, for DC: rpm).*
- unsigned int [uSpeed](#)  
*Target speed in 1/256 microsteps/s.*
- unsigned int [Accel](#)  
*Motor shaft acceleration, steps/s<sup>2</sup>(stepper motor) or RPM/s(DC).*
- unsigned int [Decel](#)  
*Motor shaft deceleration, steps/s<sup>2</sup>(stepper motor) or RPM/s(DC).*
- unsigned int [AntiplaySpeed](#)  
*Speed in antiplay mode, full steps/s(stepper motor) or RPM(DC).*
- unsigned int [uAntiplaySpeed](#)  
*Speed in antiplay mode, 1/256 microsteps/s.*

## 4.38.1 Detailed Description

Move settings.

See Also

[set.move.settings](#)  
[get.move.settings](#)  
[get.move.settings](#), [set.move.settings](#)

## 4.38.2 Field Documentation

## 4.38.2.1 unsigned int Accel

Motor shaft acceleration, steps/s<sup>2</sup>(stepper motor) or RPM/s(DC).

Range: 1..65535.

## 4.38.2.2 unsigned int AntiplaySpeed

Speed in antiplay mode, full steps/s(stepper motor) or RPM(DC).

Range: 0..100000.

## 4.38.2.3 unsigned int Decel

Motor shaft deceleration, steps/s<sup>2</sup>(stepper motor) or RPM/s(DC).

Range: 1..65535.

## 4.38.2.4 unsigned int Speed

Target speed (for stepper motor: steps/s, for DC: rpm).

Range: 0..100000.

## 4.38.2.5 unsigned int uAntiplaySpeed

Speed in antiplay mode, 1/256 microsteps/s.

Used with stepper motor only.

4.38.2.6 unsigned int uSpeed

Target speed in 1/256 microsteps/s.

Using with stepper motor only.

## 4.39 pid\_settings\_t Struct Reference

PID settings.

### Data Fields

- unsigned int [KpU](#)  
*Proportional gain for voltage PID routine.*
- unsigned int [KiU](#)  
*Integral gain for voltage PID routine.*
- unsigned int [KdU](#)  
*Differential gain for voltage PID routine.*

### 4.39.1 Detailed Description

PID settings.

This structure contains factors for PID routine. It specify behaviour of PID routine for voltage. These factors are slightly different for different positioners. All boards are supplied with standart set of PID setting on controller's flash memory. Please load new PID settings when you change positioner. Please note that wrong PID settings lead to device malfunction.

See Also

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

## 4.40 power\_settings\_t Struct Reference

Step motor power settings.

### Data Fields

- unsigned int [HoldCurrent](#)  
*Current in holding regime, percent of nominal.*
- unsigned int [CurrReductDelay](#)  
*Time in ms from going to STOP state to reducing current.*
- unsigned int [PowerOffDelay](#)  
*Time in s from going to STOP state to turning power off.*
- unsigned int [CurrentSetTime](#)  
*Time in ms to reach nominal current.*
- unsigned int [PowerFlags](#)  
*Flags of power settings of stepper motor.*

#### 4.40.1 Detailed Description

Step motor power settings.

See Also

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

#### 4.40.2 Field Documentation

##### 4.40.2.1 `unsigned int CurrentSetTime`

Time in ms to reach nominal current.

##### 4.40.2.2 `unsigned int CurrReductDelay`

Time in ms from going to STOP state to reducing current.

##### 4.40.2.3 `unsigned int HoldCurrent`

Current in holding regime, percent of nominal.

Range: 0..100.

##### 4.40.2.4 `unsigned int PowerFlags`

[Flags of power settings of stepper motor.](#)

##### 4.40.2.5 `unsigned int PowerOffDelay`

Time in s from going to STOP state to turning power off.

### 4.41 `secure_settings_t` Struct Reference

This structure contains raw analog data from ADC embedded on board.

#### Data Fields

- `unsigned int` [LowUpwrOff](#)  
*Lower voltage limit to turn off the motor, in mV.*
- `unsigned int` [CriticalIpwr](#)  
*Maximum motor current which triggers ALARM state, in mA.*
- `unsigned int` [CriticalUpwr](#)  
*Maximum motor voltage which triggers ALARM state, in mV.*
- `unsigned int` [CriticalT](#)  
*Maximum temperature, which triggers ALARM state, in tenths of degrees Celcius.*
- `unsigned int` [CriticalIusb](#)  
*Maximum USB current which triggers ALARM state, in mA.*
- `unsigned int` [CriticalUusb](#)

*Maximum USB voltage which triggers ALARM state, in mV.*

- unsigned int [MinimumUusb](#)

*Minimum USB voltage which triggers ALARM state, in mV.*

- unsigned int [Flags](#)

[Flags of secure settings.](#)

#### 4.41.1 Detailed Description

This structure contains raw analog data from ADC embedded on board.

These data used for device testing and deep recalibration by manufacturer only.

See Also

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

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

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

#### 4.41.2 Field Documentation

##### 4.41.2.1 unsigned int CriticalIpwr

Maximum motor current which triggers ALARM state, in mA.

##### 4.41.2.2 unsigned int CriticalIusb

Maximum USB current which triggers ALARM state, in mA.

##### 4.41.2.3 unsigned int CriticalT

Maximum temperature, which triggers ALARM state, in tenths of degrees Celcius.

##### 4.41.2.4 unsigned int CriticalUpwr

Maximum motor voltage which triggers ALARM state, in mV.

##### 4.41.2.5 unsigned int CriticalUusb

Maximum USB voltage which triggers ALARM state, in mV.

##### 4.41.2.6 unsigned int Flags

[Flags of secure settings.](#)

##### 4.41.2.7 unsigned int LowUpwrOff

Lower voltage limit to turn off the motor, in mV.

##### 4.41.2.8 unsigned int MinimumUusb

Minimum USB voltage which triggers ALARM state, in mV.

## 4.42 serial\_number\_t Struct Reference

Serial number structure and hardware version.

### Data Fields

- unsigned int [SN](#)  
*New board serial number.*
- unsigned int [Key](#) [32]  
*Protection key (256 bit).*
- unsigned int [Major](#)  
*The major number of the hardware version.*
- unsigned int [Minor](#)  
*Minor number of the hardware version.*
- unsigned int [Release](#)  
*Number of edits this release of hardware.*

### 4.42.1 Detailed Description

Serial number structure and hardware version.

The structure keep new serial number, hardware version and valid key. The SN and hardware version are changed and saved when transmitted key matches stored key. Can be used by manufacturer only.

See Also

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

### 4.42.2 Field Documentation

#### 4.42.2.1 unsigned int Key[32]

Protection key (256 bit).

#### 4.42.2.2 unsigned int Major

The major number of the hardware version.

#### 4.42.2.3 unsigned int Minor

Minor number of the hardware version.

#### 4.42.2.4 unsigned int Release

Number of edits this release of hardware.

#### 4.42.2.5 unsigned int SN

New board serial number.

## 4.43 set\_position\_calb\_t Struct Reference

### Data Fields

- float [Position](#)  
*The position in the engine.*
- long\_t [EncPosition](#)  
*Encoder position.*
- unsigned int [PosFlags](#)  
*Position setting flags.*

### 4.43.1 Field Documentation

#### 4.43.1.1 long\_t EncPosition

Encoder position.

#### 4.43.1.2 unsigned int PosFlags

[Position setting flags.](#)

#### 4.43.1.3 float Position

The position in the engine.

## 4.44 set\_position\_t Struct Reference

Position information.

### Data Fields

- int [Position](#)  
*The position of the whole steps in the engine.*
- int [uPosition](#)  
*Microstep position is only used with stepper motors.*
- long\_t [EncPosition](#)  
*Encoder position.*
- unsigned int [PosFlags](#)  
*Position setting flags.*

### 4.44.1 Detailed Description

Position information.

Useful structure that contains position value in steps and micro for stepper motor and encoder steps of all engines.

See Also

[set\\_position](#)

#### 4.44.2 Field Documentation

##### 4.44.2.1 long\_t EncPosition

Encoder position.

##### 4.44.2.2 unsigned int PosFlags

[Position setting flags.](#)

### 4.45 stage\_information\_t Struct Reference

Stage information.

#### Data Fields

- char [Manufacturer](#) [17]  
*Manufacturer.*
- char [PartNumber](#) [25]  
*Series and PartNumber.*

#### 4.45.1 Detailed Description

Stage information.

See Also

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

#### 4.45.2 Field Documentation

##### 4.45.2.1 char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

##### 4.45.2.2 char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

### 4.46 stage\_name\_t Struct Reference

Stage user name.



## Data Fields

- char [PositionerName](#) [17]  
*User positioner name.*

## 4.46.1 Detailed Description

Stage user name.

See Also

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

## 4.46.2 Field Documentation

## 4.46.2.1 char PositionerName[17]

User positioner name.

Can be set by user for his/her convinience. Max string length: 16 chars.

## 4.47 stage\_settings\_t Struct Reference

Stage settings.

## Data Fields

- float [LeadScrewPitch](#)  
*Lead screw pitch (mm).*
- char [Units](#) [9]  
*Units for MaxSpeed and TravelRange fields of the structure (steps, degrees, mm, ...).*
- float [MaxSpeed](#)  
*Max speed (Units/c).*
- float [TravelRange](#)  
*Travel range (Units).*
- float [SupplyVoltageMin](#)  
*Supply voltage minimum (V).*
- float [SupplyVoltageMax](#)  
*Supply voltage maximum (V).*
- float [MaxCurrentConsumption](#)  
*Max current consumption (A).*
- float [HorizontalLoadCapacity](#)  
*Horizontal load capacity (kg).*
- float [VerticalLoadCapacity](#)  
*Vertical load capacity (kg).*

#### 4.47.1 Detailed Description

Stage settings.

See Also

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

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

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

#### 4.47.2 Field Documentation

##### 4.47.2.1 float HorizontalLoadCapacity

Horizontal load capacity (kg).

Data type: float.

##### 4.47.2.2 float LeadScrewPitch

Lead screw pitch (mm).

Data type: float.

##### 4.47.2.3 float MaxCurrentConsumption

Max current consumption (A).

Data type: float.

##### 4.47.2.4 float MaxSpeed

Max speed (Units/c).

Data type: float.

##### 4.47.2.5 float SupplyVoltageMax

Supply voltage maximum (V).

Data type: float.

##### 4.47.2.6 float SupplyVoltageMin

Supply voltage minimum (V).

Data type: float.

##### 4.47.2.7 float TravelRange

Travel range (Units).

Data type: float.

## 4.47.2.8 char Units[9]

Units for MaxSpeed and TravelRange fields of the structure (steps, degrees, mm, ...).

Max string length: 8 chars.

## 4.47.2.9 float VerticalLoadCapacity

Vertical load capacity (kg).

Data type: float.

## 4.48 status\_calb\_t Struct Reference

## Data Fields

- unsigned int [MoveSts](#)  
*Flags of move state.*
- unsigned int [MvCmdSts](#)  
*Move command state.*
- unsigned int [PWRSts](#)  
*Flags of power state of stepper motor.*
- unsigned int [EncSts](#)  
*Encoder state.*
- unsigned int [WindSts](#)  
*Winding state.*
- float [CurPosition](#)  
*Current position.*
- long\_t [EncPosition](#)  
*Current encoder position.*
- float [CurSpeed](#)  
*Motor shaft speed.*
- int [lpwr](#)  
*Engine current.*
- int [Upwr](#)  
*Power supply voltage.*
- int [lusb](#)  
*USB current consumption.*
- int [Uusb](#)  
*USB voltage.*
- int [CurT](#)  
*Temperature in tenths of degrees C.*
- unsigned int [Flags](#)  
*Status flags.*
- unsigned int [GPIOFlags](#)  
*Status flags.*
- unsigned int [CmdBufFreeSpace](#)  
*This field shows the amount of free cells buffer synchronization chain.*

#### 4.48.1 Field Documentation

##### 4.48.1.1 unsigned int CmdBufFreeSpace

This field shows the amount of free cells buffer synchronization chain.

##### 4.48.1.2 float CurPosition

Current position.

##### 4.48.1.3 float CurSpeed

Motor shaft speed.

##### 4.48.1.4 int CurT

Temperature in tenths of degrees C.

##### 4.48.1.5 long\_t EncPosition

Current encoder position.

##### 4.48.1.6 unsigned int EncSts

[Encoder state.](#)

##### 4.48.1.7 unsigned int Flags

[Status flags.](#)

##### 4.48.1.8 unsigned int GPIOFlags

[Status flags.](#)

##### 4.48.1.9 int Ipwr

Engine current.

##### 4.48.1.10 int Iusb

USB current consumption.

##### 4.48.1.11 unsigned int MoveSts

[Flags of move state.](#)

##### 4.48.1.12 unsigned int MvCmdSts

[Move command state.](#)

4.48.1.13 unsigned int PWRSts

Flags of power state of stepper motor.

4.48.1.14 int Upwr

Power supply voltage.

4.48.1.15 int Uusb

USB voltage.

4.48.1.16 unsigned int WindSts

Winding state.

## 4.49 status\_t Struct Reference

Device state.

### Data Fields

- unsigned int [MoveSts](#)  
*Flags of move state.*
- unsigned int [MvCmdSts](#)  
*Move command state.*
- unsigned int [PWRSts](#)  
*Flags of power state of stepper motor.*
- unsigned int [EncSts](#)  
*Encoder state.*
- unsigned int [WindSts](#)  
*Winding state.*
- int [CurPosition](#)  
*Current position.*
- int [uCurPosition](#)  
*Step motor shaft position in 1/256 microsteps.*
- long\_t [EncPosition](#)  
*Current encoder position.*
- int [CurSpeed](#)  
*Motor shaft speed.*
- int [uCurSpeed](#)  
*Part of motor shaft speed in 1/256 microsteps.*
- int [lpwr](#)  
*Engine current.*
- int [Upwr](#)  
*Power supply voltage.*
- int [lusb](#)  
*USB current consumption.*
- int [Uusb](#)

- USB voltage.*
- int [CurT](#)  
*Temperature in tenths of degrees C.*
- unsigned int [Flags](#)  
*Status flags.*
- unsigned int [GPIOFlags](#)  
*Status flags.*
- unsigned int [CmdBufFreeSpace](#)  
*This field shows the amount of free cells buffer synchronization chain.*

#### 4.49.1 Detailed Description

Device state.

Useful structure that contains current controller state, including speed, position and boolean flags.

See Also

`get_status_impl`

#### 4.49.2 Field Documentation

##### 4.49.2.1 unsigned int CmdBufFreeSpace

This field shows the amount of free cells buffer synchronization chain.

##### 4.49.2.2 int CurPosition

Current position.

##### 4.49.2.3 int CurSpeed

Motor shaft speed.

##### 4.49.2.4 int CurT

Temperature in tenths of degrees C.

##### 4.49.2.5 long\_t EncPosition

Current encoder position.

##### 4.49.2.6 unsigned int EncSts

[Encoder state.](#)

##### 4.49.2.7 unsigned int Flags

[Status flags.](#)

4.49.2.8 unsigned int GPIOFlags

[Status flags.](#)

4.49.2.9 int Ipwr

Engine current.

4.49.2.10 int Iusb

USB current consumption.

4.49.2.11 unsigned int MoveSts

[Flags of move state.](#)

4.49.2.12 unsigned int MvCmdSts

[Move command state.](#)

4.49.2.13 unsigned int PWRSts

[Flags of power state of stepper motor.](#)

4.49.2.14 int uCurPosition

Step motor shaft position in 1/256 microsteps.

Used only with stepper motor.

4.49.2.15 int uCurSpeed

Part of motor shaft speed in 1/256 microsteps.

Used only with stepper motor.

4.49.2.16 int Upwr

Power supply voltage.

4.49.2.17 int Uusb

USB voltage.

4.49.2.18 unsigned int WindSts

[Winding state.](#)

## 4.50 sync\_in\_settings\_calb\_t Struct Reference

### Data Fields

- unsigned int [SyncInFlags](#)  
*Flags for synchronization input setup.*
- unsigned int [ClutterTime](#)  
*Input synchronization pulse dead time (mks).*
- float [Position](#)  
*Desired position or shift.*
- float [Speed](#)  
*Target speed.*

### 4.50.1 Field Documentation

#### 4.50.1.1 unsigned int ClutterTime

Input synchronization pulse dead time (mks).

#### 4.50.1.2 float Position

Desired position or shift.

#### 4.50.1.3 float Speed

Target speed.

#### 4.50.1.4 unsigned int SyncInFlags

[Flags for synchronization input setup.](#)

## 4.51 sync\_in\_settings\_t Struct Reference

Synchronization settings.

### Data Fields

- unsigned int [SyncInFlags](#)  
*Flags for synchronization input setup.*
- unsigned int [ClutterTime](#)  
*Input synchronization pulse dead time (mks).*
- int [Position](#)  
*Desired position or shift (whole steps)*
- int [uPosition](#)  
*The fractional part of a position or shift in microsteps.*
- unsigned int [Speed](#)  
*Target speed (for stepper motor: steps/s, for DC: rpm).*
- unsigned int [uSpeed](#)  
*Target speed in microsteps/s.*



### 4.51.1 Detailed Description

Synchronization settings.

This structure contains all synchronization settings, modes, periods and flags. It specifies behaviour of input synchronization. All boards are supplied with standart set of these settings.

See Also

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

### 4.51.2 Field Documentation

#### 4.51.2.1 unsigned int ClutterTime

Input synchronization pulse dead time (mks).

#### 4.51.2.2 unsigned int Speed

Target speed (for stepper motor: steps/s, for DC: rpm).

Range: 0..100000.

#### 4.51.2.3 unsigned int SyncInFlags

[Flags for synchronization input setup.](#)

#### 4.51.2.4 int uPosition

The fractional part of a position or shift in microsteps.

Is used with stepper motor. Range: -255..255.

#### 4.51.2.5 unsigned int uSpeed

Target speed in microsteps/s.

Using with stepper motor only.

## 4.52 sync\_out\_settings\_calb\_t Struct Reference

### Data Fields

- unsigned int [SyncOutFlags](#)  
[Flags of synchronization output.](#)
- unsigned int [SyncOutPulseSteps](#)  
*This value specifies duration of output pulse.*
- unsigned int [SyncOutPeriod](#)  
*This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT\_ONPERIOD is set.*
- float [Accuracy](#)  
*This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop.*

### 4.52.1 Field Documentation

#### 4.52.1.1 float Accuracy

This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop.

#### 4.52.1.2 unsigned int SyncOutFlags

Flags of synchronization output.

#### 4.52.1.3 unsigned int SyncOutPeriod

This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT\_ONPERIOD is set.

#### 4.52.1.4 unsigned int SyncOutPulseSteps

This value specifies duration of output pulse.

It is measured microseconds when SYNCOUT.IN.STEPS flag is cleared or in encoder pulses or motor steps when SYNCOUT.IN.STEPS is set.

## 4.53 sync\_out\_settings\_t Struct Reference

Synchronization settings.

### Data Fields

- unsigned int [SyncOutFlags](#)  
*Flags of synchronization output.*
- unsigned int [SyncOutPulseSteps](#)  
*This value specifies duration of output pulse.*
- unsigned int [SyncOutPeriod](#)  
*This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT\_ONPERIOD is set.*
- unsigned int [Accuracy](#)  
*This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop.*
- unsigned int [uAccuracy](#)  
*This is the neighborhood around the target coordinates in micro steps (only used with stepper motor).*

### 4.53.1 Detailed Description

Synchronization settings.

This structure contains all synchronization settings, modes, periods and flags. It specifies behaviour of output synchronization. All boards are supplied with standart set of these settings.

See Also

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

### 4.53.2 Field Documentation

#### 4.53.2.1 unsigned int Accuracy

This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop.

#### 4.53.2.2 unsigned int SyncOutFlags

Flags of synchronization output.

#### 4.53.2.3 unsigned int SyncOutPeriod

This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT\_ONPERIOD is set.

#### 4.53.2.4 unsigned int SyncOutPulseSteps

This value specifies duration of output pulse.

It is measured microseconds when SYNCOUT\_IN\_STEPS flag is cleared or in encoder pulses or motor steps when SYNCOUT\_IN\_STEPS is set.

#### 4.53.2.5 unsigned int uAccuracy

This is the neighborhood around the target coordinates in micro steps (only used with stepper motor).

## 4.54 `uart_settings_t` Struct Reference

UART settings.

### Data Fields

- unsigned int [Speed](#)  
*UART speed.*
- unsigned int [UARTSetupFlags](#)  
*UART parity flags.*

### 4.54.1 Detailed Description

UART settings.

This structure contains UART settings.

See Also

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

#### 4.54.2 Field Documentation

##### 4.54.2.1 unsigned int UARTSetupFlags

[UART parity flags.](#)

# Chapter 5

## File Documentation

### 5.1 ximc.h File Reference

Header file for libximc library.

#### Data Structures

- struct [calibration\\_t](#)  
*Calibration companion structure.*
- struct [device\\_network\\_information\\_t](#)  
*TODO.*
- struct [feedback\\_settings\\_t](#)  
*Feedback settings.*
- struct [home\\_settings\\_t](#)  
*Position calibration settings.*
- struct [home\\_settings\\_calb\\_t](#)
- struct [move\\_settings\\_t](#)  
*Move settings.*
- struct [move\\_settings\\_calb\\_t](#)
- struct [engine\\_settings\\_t](#)  
*Engine settings.*
- struct [engine\\_settings\\_calb\\_t](#)
- struct [entype\\_settings\\_t](#)  
*Engine type and driver type settings.*
- struct [power\\_settings\\_t](#)  
*Step motor power settings.*
- struct [secure\\_settings\\_t](#)  
*This structure contains raw analog data from ADC embedded on board.*
- struct [edges\\_settings\\_t](#)  
*Edges settings.*
- struct [edges\\_settings\\_calb\\_t](#)
- struct [pid\\_settings\\_t](#)  
*PID settings.*
- struct [sync\\_in\\_settings\\_t](#)  
*Synchronization settings.*
- struct [sync\\_in\\_settings\\_calb\\_t](#)
- struct [sync\\_out\\_settings\\_t](#)

*Synchronization settings.*

- struct [sync\\_out\\_settings\\_calb\\_t](#)
- struct [extio\\_settings\\_t](#)

*EXTIO settings.*

- struct [brake\\_settings\\_t](#)

*Brake settings.*

- struct [control\\_settings\\_t](#)

*Control settings.*

- struct [control\\_settings\\_calb\\_t](#)
- struct [joystick\\_settings\\_t](#)

*Joystick settings.*

- struct [ctp\\_settings\\_t](#)

*Control position settings(is only used with stepper motor).*

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

*UART settings.*

- struct [controller\\_name\\_t](#)

*Controller user name and flags of setting.*

- struct [command\\_add\\_sync\\_in\\_action\\_t](#)

*This command adds one element of the FIFO commands.*

- struct [command\\_add\\_sync\\_in\\_action\\_calb\\_t](#)
- struct [get\\_position\\_t](#)

*Position information.*

- struct [get\\_position\\_calb\\_t](#)
- struct [set\\_position\\_t](#)

*Position information.*

- struct [set\\_position\\_calb\\_t](#)
- struct [status\\_t](#)

*Device state.*

- struct [status\\_calb\\_t](#)
- struct [chart\\_data\\_t](#)

*Additional device state.*

- struct [device\\_information\\_t](#)

*Read command controller information.*

- struct [serial\\_number\\_t](#)

*Serial number structure and hardware version.*

- struct [analog\\_data\\_t](#)

*Analog data.*

- struct [debug\\_read\\_t](#)

*Debug data.*

- struct [debug\\_write\\_t](#)

*Debug data.*

- struct [stage\\_name\\_t](#)

*Stage user name.*

- struct [stage\\_information\\_t](#)

*Stage information.*

- struct [stage\\_settings\\_t](#)

*Stage settings.*

- struct [motor\\_information\\_t](#)

*motor information.*

- struct [motor\\_settings\\_t](#)

*motor settings.*

- struct [encoder\\_information\\_t](#)  
*Encoder information.*
- struct [encoder\\_settings\\_t](#)  
*Encoder settings.*
- struct [hallsensor\\_information\\_t](#)  
*Hall sensor information.*
- struct [hallsensor\\_settings\\_t](#)  
*Hall sensor settings.*
- struct [gear\\_information\\_t](#)  
*Gear information.*
- struct [gear\\_settings\\_t](#)  
*Gear settings.*
- struct [accessories\\_settings\\_t](#)  
*Additional accessories information.*
- struct [init\\_random\\_t](#)  
*Random key.*

## Macros

- #define [XIMC\\_API](#)  
*Library import macro Macros allows to automatically import function from shared library.*
- #define [XIMC\\_CALLCONV](#)  
*Library calling convention macros.*
- #define [device\\_undefined](#) -1  
*Handle specified undefined device.*

## Result statuses

- #define [result\\_ok](#) 0  
*success*
- #define [result\\_error](#) -1  
*generic error*
- #define [result\\_not\\_implemented](#) -2  
*function is not implemented*
- #define [result\\_value\\_error](#) -3  
*value error*
- #define [result\\_noddevice](#) -4  
*device is lost*

## Logging level

- #define [LOGLEVEL\\_ERROR](#) 0x01  
*Logging level - error.*
- #define [LOGLEVEL\\_WARNING](#) 0x02  
*Logging level - warning.*
- #define [LOGLEVEL\\_INFO](#) 0x03  
*Logging level - info.*
- #define [LOGLEVEL\\_DEBUG](#) 0x04  
*Logging level - debug.*

## Enumerate devices flags

- #define [ENUMERATE\\_PROBE](#) 0x01

- *Check if a device with OS name name is XIMC device.*  
• #define [ENUMERATE\\_ALL\\_COM](#) 0x02  
*Check all COM devices.*
- #define [ENUMERATE\\_NETWORK](#) 0x04  
*Check network devices.*

### Flags of move state

Specify move states.

See Also

[get\\_status](#)  
[status\\_t::move\\_state](#)  
[status\\_t::MoveSts](#), [get\\_status\\_impl](#)

- #define [MOVE\\_STATE\\_MOVING](#) 0x01  
*This flag indicates that controller is trying to move the motor.*
- #define [MOVE\\_STATE\\_TARGET\\_SPEED](#) 0x02  
*Target speed is reached, if flag set.*
- #define [MOVE\\_STATE\\_ANTIPLAY](#) 0x04  
*Motor is playing compensation, if flag set.*

### Flags of internal controller settings

See Also

[set\\_controller\\_name](#)  
[get\\_controller\\_name](#)  
[controller\\_name\\_t::CtrlFlags](#), [get\\_controller\\_name](#), [set\\_controller\\_name](#)

- #define [EEPROM\\_PRECEDENCE](#) 0x01  
*If the flag is set settings from external EEPROM override controller settings.*

### Flags of power state of stepper motor

Specify power states.

See Also

[status\\_t::power\\_state](#)  
[get\\_status](#)  
[status\\_t::PWRSts](#), [get\\_status\\_impl](#)

- #define [PWR\\_STATE\\_UNKNOWN](#) 0x00  
*Unknown state, should never happen.*
- #define [PWR\\_STATE\\_OFF](#) 0x01  
*Motor windings are disconnected from the driver.*
- #define [PWR\\_STATE\\_NORM](#) 0x03  
*Motor windings are powered by nominal current.*
- #define [PWR\\_STATE\\_REDUCT](#) 0x04  
*Motor windings are powered by reduced current to lower power consumption.*
- #define [PWR\\_STATE\\_MAX](#) 0x05  
*Motor windings are powered by maximum current driver can provide at this voltage.*

### Status flags

GPIO state flags returned by device query. Contains boolean part of controller state. May be combined with bitwise OR.



See Also

`status_t::flags`

`get_status`

`status_t::GPIOFlags`, `get_status_impl`

- #define `STATE_CONTR` 0x0003F  
*Flags of controller states.*
- #define `STATE_ERRC` 0x00001  
*Command error encountered.*
- #define `STATE_ERRD` 0x00002  
*Data integrity error encountered.*
- #define `STATE_ERRV` 0x00004  
*Value error encountered.*
- #define `STATE_EEPROM_CONNECTED` 0x00010  
*EEPROM with settings is connected.*
- #define `STATE_SECUR` 0x3FFC0  
*Flags of security.*
- #define `STATE_ALARM` 0x00040  
*Controller is in alarm state indicating that something dangerous had happened.*
- #define `STATE_CTP_ERROR` 0x00080  
*Control position error(is only used with stepper motor).*
- #define `STATE_POWER_OVERHEAT` 0x00100  
*Power driver overheat.*
- #define `STATE_CONTROLLER_OVERHEAT` 0x00200  
*Controller overheat.*
- #define `STATE_OVERLOAD_POWER_VOLTAGE` 0x00400  
*Power voltage exceeds safe limit.*
- #define `STATE_OVERLOAD_POWER_CURRENT` 0x00800  
*Power current exceeds safe limit.*
- #define `STATE_OVERLOAD_USB_VOLTAGE` 0x01000  
*USB voltage exceeds safe limit.*
- #define `STATE_LOW_USB_VOLTAGE` 0x02000  
*USB voltage is insufficient for normal operation.*
- #define `STATE_OVERLOAD_USB_CURRENT` 0x04000  
*USB current exceeds safe limit.*
- #define `STATE_BORDERS_SWAP_MISSET` 0x08000  
*Engine stuck at the wrong edge.*
- #define `STATE_LOW_POWER_VOLTAGE` 0x10000  
*Power voltage is lower than Low Voltage Protection limit.*
- #define `STATE_H_BRIDGE_FAULT` 0x20000  
*Signal from the driver that fault happened.*
- #define `STATE_DIG_SIGNAL` 0xFFFF  
*Flags of digital signals.*
- #define `STATE_RIGHT_EDGE` 0x0001  
*Engine stuck at the right edge.*
- #define `STATE_LEFT_EDGE` 0x0002  
*Engine stuck at the left edge.*
- #define `STATE_BUTTON_RIGHT` 0x0004  
*Button "right" state (1 if pressed).*
- #define `STATE_BUTTON_LEFT` 0x0008  
*Button "left" state (1 if pressed).*
- #define `STATE_GPIO_PINOUT` 0x0010  
*External GPIO works as Out, if flag set; otherwise works as In.*
- #define `STATE_GPIO_LEVEL` 0x0020  
*State of external GPIO pin.*
- #define `STATE_HALL_A` 0x0040  
*State of Hall.a pin.*
- #define `STATE_HALL_B` 0x0080  
*State of Hall.b pin.*

- #define [STATE\\_HALL\\_C](#) 0x0100  
*State of Hall.c pin.*
- #define [STATE\\_BRAKE](#) 0x0200  
*State of Brake pin.*
- #define [STATE\\_REV\\_SENSOR](#) 0x0400  
*State of Revolution sensor pin.*
- #define [STATE\\_SYNC\\_INPUT](#) 0x0800  
*State of Sync input pin.*
- #define [STATE\\_SYNC\\_OUTPUT](#) 0x1000  
*State of Sync output pin.*
- #define [STATE\\_ENC\\_A](#) 0x2000  
*State of encoder A pin.*
- #define [STATE\\_ENC\\_B](#) 0x4000  
*State of encoder B pin.*

### Encoder state

Encoder state returned by device query.

See Also

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

- #define [ENC\\_STATE\\_ABSENT](#) 0x00  
*Encoder is absent.*
- #define [ENC\\_STATE\\_UNKNOWN](#) 0x01  
*Encoder state is unknown.*
- #define [ENC\\_STATE\\_MALFUNC](#) 0x02  
*Encoder is connected and malfunctioning.*
- #define [ENC\\_STATE\\_REVERS](#) 0x03  
*Encoder is connected and operational but counts in otyher direction.*
- #define [ENC\\_STATE\\_OK](#) 0x04  
*Encoder is connected and working properly.*

### Winding state

Motor winding state returned by device query.

See Also

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

- #define [WIND\\_A.STATE\\_ABSENT](#) 0x00  
*Winding A is disconnected.*
- #define [WIND\\_A.STATE\\_UNKNOWN](#) 0x01  
*Winding A state is unknown.*
- #define [WIND\\_A.STATE\\_MALFUNC](#) 0x02  
*Winding A is short-circuited.*
- #define [WIND\\_A.STATE\\_OK](#) 0x03  
*Winding A is connected and working properly.*
- #define [WIND\\_B.STATE\\_ABSENT](#) 0x00  
*Winding B is disconnected.*
- #define [WIND\\_B.STATE\\_UNKNOWN](#) 0x10  
*Winding B state is unknown.*
- #define [WIND\\_B.STATE\\_MALFUNC](#) 0x20  
*Winding B is short-circuited.*
- #define [WIND\\_B.STATE\\_OK](#) 0x30  
*Winding B is connected and working properly.*

### Move command state

Move command (`command_move`, `command_movr`, `command_left`, `command_right`, `command_stop`, `command_home`, `command_loft`, `command_sstp`) and its state (`run`, `finished`, `error`).

See Also

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

- #define [MVCMD\\_NAME\\_BITS](#) 0x3F  
Move command bit mask.
- #define [MVCMD\\_UKNWN](#) 0x00  
Unknown command.
- #define [MVCMD\\_MOVE](#) 0x01  
Command move.
- #define [MVCMD\\_MOVR](#) 0x02  
Command movr.
- #define [MVCMD\\_LEFT](#) 0x03  
Command left.
- #define [MVCMD\\_RIGHT](#) 0x04  
Command right.
- #define [MVCMD\\_STOP](#) 0x05  
Command stop.
- #define [MVCMD\\_HOME](#) 0x06  
Command home.
- #define [MVCMD\\_LOFT](#) 0x07  
Command loft.
- #define [MVCMD\\_SSTP](#) 0x08  
Command soft stop.
- #define [MVCMD\\_ERROR](#) 0x40  
Finish state (1 - move command have finished with an error, 0 - move command have finished correctly).
- #define [MVCMD\\_RUNNING](#) 0x80  
Move command state (0 - move command have finished, 1 - move command is being executed).

### Flags of engine settings

Specify motor shaft movement algorithm and list of limitations. Flags returned by query of engine settings. May be combined with bitwise OR.

See Also

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

- #define [ENGINE\\_REVERSE](#) 0x01  
Reverse flag.
- #define [ENGINE\\_MAX\\_SPEED](#) 0x04  
Max speed flag.
- #define [ENGINE\\_ANTIPLAY](#) 0x08  
Play compensation flag.
- #define [ENGINE\\_ACCEL\\_ON](#) 0x10  
Acceleration enable flag.
- #define [ENGINE\\_LIMIT\\_VOLT](#) 0x20  
Maximum motor voltage limit enable flag(is only used with DC motor).
- #define [ENGINE\\_LIMIT\\_CURR](#) 0x40  
Maximum motor current limit enable flag(is only used with DC motor).
- #define [ENGINE\\_LIMIT\\_RPM](#) 0x80  
Maximum motor speed limit enable flag.

### Flags of microstep mode

Specify settings of microstep mode. Using with step motors. Flags returned by query of engine settings. May be combined with bitwise OR

See Also

`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  
Full step mode.
- #define [MICROSTEP\\_MODE\\_FRAC\\_2](#) 0x02  
1/2 step mode.
- #define [MICROSTEP\\_MODE\\_FRAC\\_4](#) 0x03  
1/4 step mode.
- #define [MICROSTEP\\_MODE\\_FRAC\\_8](#) 0x04  
1/8 step mode.
- #define [MICROSTEP\\_MODE\\_FRAC\\_16](#) 0x05  
1/16 step mode.
- #define [MICROSTEP\\_MODE\\_FRAC\\_32](#) 0x06  
1/32 step mode.
- #define [MICROSTEP\\_MODE\\_FRAC\\_64](#) 0x07  
1/64 step mode.
- #define [MICROSTEP\\_MODE\\_FRAC\\_128](#) 0x08  
1/128 step mode.
- #define [MICROSTEP\\_MODE\\_FRAC\\_256](#) 0x09  
1/256 step mode.

### Flags of engine type

Specify motor type. Flags returned by query of engine settings.

See Also

`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  
A value that shouldn't be used.
- #define [ENGINE\\_TYPE\\_DC](#) 0x01  
DC motor.
- #define [ENGINE\\_TYPE\\_2DC](#) 0x02  
2 DC motors.
- #define [ENGINE\\_TYPE\\_STEP](#) 0x03  
Step motor.
- #define [ENGINE\\_TYPE\\_TEST](#) 0x04  
Duty cycle are fixed.
- #define [ENGINE\\_TYPE\\_BRUSHLESS](#) 0x05  
Brushless motor.

### Flags of driver type

Specify driver type. Flags returned by query of engine settings.

See Also

[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  
*Driver with discrete FET keys.*
- #define [DRIVER\\_TYPE\\_INTEGRATE](#) 0x02  
*Driver with integrated IC.*
- #define [DRIVER\\_TYPE\\_EXTERNAL](#) 0x03  
*External driver.*

### Flags of power settings of stepper motor

Specify power settings. Flags returned by query of power settings.

See Also

[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  
*Current reduction enabled after CurrReductDelay, if this flag is set.*
- #define [POWER\\_OFF\\_ENABLED](#) 0x02  
*Power off enabled after PowerOffDelay, if this flag is set.*
- #define [POWER\\_SMOOTH\\_CURRENT](#) 0x04  
*Current ramp-up/down is performed smoothly during current.set.time, if this flag is set.*

### Flags of secure settings

Specify secure settings. Flags returned by query of secure settings.

See Also

[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  
*If this flag is set enter Alarm state on driver overheat signal.*
- #define [LOW\\_UPWR\\_PROTECTION](#) 0x02  
*If this flag is set turn off motor when voltage is lower than LowUpwrOff.*
- #define [H\\_BRIDGE\\_ALERT](#) 0x04  
*If this flag is set then turn off the power unit with a signal problem in one of the transistor bridge.*
- #define [ALARM\\_ON\\_BORDERS\\_SWAP\\_MISSET](#) 0x08  
*If this flag is set enter Alarm state on borders swap misset.*
- #define [ALARM\\_FLAGS\\_STICKING](#) 0x10  
*If this flag is set only a STOP command can turn all alarms to 0.*
- #define [USB\\_BREAK\\_RECONNECT](#) 0x20  
*If this flag is set USB brake reconnect module will be enable.*

### Position setting flags

Flags used in setting of position.

See Also

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

- #define [SETPOS\\_IGNORE\\_POSITION](#) 0x01  
*Will not reload position in steps/microsteps if this flag is set.*
- #define [SETPOS\\_IGNORE\\_ENCODER](#) 0x02  
*Will not reload encoder state if this flag is set.*

### Feedback type.

See Also

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

- #define [FEEDBACK\\_ENCODER](#) 0x01  
*Feedback by encoder.*
- #define [FEEDBACK\\_ENCODERHALL](#) 0x03  
*Feedback by Hall detector.*
- #define [FEEDBACK\\_EMF](#) 0x04  
*Feedback by EMF.*
- #define [FEEDBACK\\_NONE](#) 0x05  
*Feedback is absent.*

### Describes feedback flags.

See Also

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

- #define [FEEDBACK\\_ENC\\_REVERSE](#) 0x01  
*Reverse count of encoder.*
- #define [FEEDBACK\\_HALL\\_REVERSE](#) 0x02  
*Reverse count position on the Hall sensor.*

### Flags for synchronization input setup

See Also

[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  
*Synchronization in mode is enabled, if this flag is set.*
- #define [SYNCIN\\_INVERT](#) 0x02  
*Trigger on falling edge if flag is set, on rising edge otherwise.*
- #define [SYNCIN\\_GOTOPOSITION](#) 0x04  
*The engine is go to position specified in Position and uPosition, if this flag is set.*

### Flags of synchronization output

See Also

[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  
*Synchronization out pin follows the synchronization logic, if set.*
- #define [SYNCOUT\\_STATE](#) 0x02  
*When output state is fixed by negative SYNCOUT\_ENABLED flag, the pin state is in accordance with this flag state.*
- #define [SYNCOUT\\_INVERT](#) 0x04  
*Low level is active, if set, and high level is active otherwise.*
- #define [SYNCOUT\\_IN\\_STEPS](#) 0x08  
*Use motor steps/encoder pulses instead of milliseconds for output pulse generation if the flag is set.*
- #define [SYNCOUT\\_ONSTART](#) 0x10  
*Generate synchronization pulse when movement starts.*
- #define [SYNCOUT\\_ONSTOP](#) 0x20  
*Generate synchronization pulse when movement stops.*
- #define [SYNCOUT\\_ONPERIOD](#) 0x40  
*Generate synchronization pulse every SyncOutPeriod encoder pulses.*

### External IO setup flags

See Also

[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  
*EXTIO works as output if flag is set, works as input otherwise.*
- #define [EXTIO\\_SETUP\\_INVERT](#) 0x02  
*Interpret EXTIO states and fronts inverted if flag is set.*

### External IO mode flags

See Also

[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\\_NOP](#) 0x00  
*Do nothing.*
- #define [EXTIO\\_SETUP\\_MODE\\_IN\\_STOP](#) 0x01  
*Issue STOP command, ceasing the engine movement.*
- #define [EXTIO\\_SETUP\\_MODE\\_IN\\_PWOF](#) 0x02  
*Issue PWOF command, powering off all engine windings.*
- #define [EXTIO\\_SETUP\\_MODE\\_IN\\_MOVR](#) 0x03  
*Issue MOVR command with last used settings.*
- #define [EXTIO\\_SETUP\\_MODE\\_IN\\_HOME](#) 0x04  
*Issue HOME command.*
- #define [EXTIO\\_SETUP\\_MODE\\_OUT\\_OFF](#) 0x00  
*EXTIO pin always set in inactive state.*
- #define [EXTIO\\_SETUP\\_MODE\\_OUT\\_ON](#) 0x10  
*EXTIO pin always set in active state.*
- #define [EXTIO\\_SETUP\\_MODE\\_OUT\\_MOVING](#) 0x20

- *EXTIO pin stays active during moving state.*  
• #define [EXTIO.SETUP\\_MODE\\_OUT\\_ALARM](#) 0x30
- *EXTIO pin stays active during Alarm state.*  
• #define [EXTIO.SETUP\\_MODE\\_OUT\\_MOTOR\\_ON](#) 0x40
- *EXTIO pin stays active when windings are powered.*  
• #define [EXTIO.SETUP\\_MODE\\_OUT\\_MOTOR\\_FOUND](#) 0x50
- *EXTIO pin stays active when motor is connected (first winding).*

### Border flags

Specify types of borders and motor behaviour on borders. May be combined with bitwise OR.

See Also

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

- #define [BORDER.IS\\_ENCODER](#) 0x01  
*Borders are fixed by predetermined encoder values, if set; borders position on limit switches, if not set.*
- #define [BORDER.STOP\\_LEFT](#) 0x02  
*Motor should stop on left border.*
- #define [BORDER.STOP\\_RIGHT](#) 0x04  
*Motor should stop on right border.*
- #define [BORDERS.SWAP\\_MISSET\\_DETECTION](#) 0x08  
*Motor should stop on both borders.*

### Limit switches flags

Specify electrical behaviour of limit switches like order and pulled positions. May be combined with bitwise OR.

See Also

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

- #define [ENDER.SWAP](#) 0x01  
*First limit switch on the right side, if set; otherwise on the left side.*
- #define [ENDER.SW1\\_ACTIVE\\_LOW](#) 0x02  
*1 - Limit switch connected to pin SW1 is triggered by a low level on pin.*
- #define [ENDER.SW2\\_ACTIVE\\_LOW](#) 0x04  
*1 - Limit switch connected to pin SW2 is triggered by a low level on pin.*

### Brake settings flags

Specify behaviour of brake. May be combined with bitwise OR.

See Also

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

- #define [BRAKE.ENABLED](#) 0x01  
*Brake control is enabled, if this flag is set.*
- #define [BRAKE.ENG\\_PWROFF](#) 0x02  
*Brake turns off power of step motor, if this flag is set.*

### Control flags

Specify motor control settings by joystick or buttons. May be combined with bitwise OR.



See Also

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

- #define [CONTROL\\_MODE\\_BITS](#) 0x03  
*Bits to control engine by joystick or buttons.*
- #define [CONTROL\\_MODE\\_OFF](#) 0x00  
*Control is disabled.*
- #define [CONTROL\\_MODE\\_JOY](#) 0x01  
*Control by joystick.*
- #define [CONTROL\\_MODE\\_LR](#) 0x02  
*Control by left/right buttons.*
- #define [CONTROL\\_BTN\\_LEFT\\_PUSHED\\_OPEN](#) 0x04  
*Pushed left button corresponds to open contact, if this flag is set.*
- #define [CONTROL\\_BTN\\_RIGHT\\_PUSHED\\_OPEN](#) 0x08  
*Pushed right button corresponds to open contact, if this flag is set.*

### Joystick flags

Control joystick states.

See Also

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

- #define [JOY\\_REVERSE](#) 0x01  
*Joystick action is reversed.*

### Position control flags

Specify settings of position control. May be combined with bitwise OR.

See Also

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

- #define [CTP\\_ENABLED](#) 0x01  
*Position control is enabled, if flag set.*
- #define [CTP\\_BASE](#) 0x02  
*Position control is based on revolution sensor, if this flag is set; otherwise it is based on encoder.*
- #define [CTP\\_ALARM\\_ON\\_ERROR](#) 0x04  
*Set ALARM on mismatch, if flag set.*
- #define [REV\\_SENS\\_INV](#) 0x08  
*Sensor is active when it 0 and invert makes active level 1.*

### Home settings flags

Specify behaviour for home command. May be combined with bitwise OR.

See Also

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

- #define [HOME\\_DIR\\_FIRST](#) 0x01  
*Flag defines direction of 1st motion after execution of home command.*

- #define [HOME\\_DIR\\_SECOND](#) 0x02  
*Flag defines direction of 2nd motion.*
- #define [HOME\\_MV\\_SEC\\_EN](#) 0x04  
*Use the second phase of calibration to the home position, if set; otherwise the second phase is skipped.*
- #define [HOME\\_HALF\\_MV](#) 0x08  
*If the flag is set, the stop signals are ignored in start of second movement the first half-turn.*
- #define [HOME\\_STOP\\_FIRST\\_BITS](#) 0x30  
*Bits of the first stop selector.*
- #define [HOME\\_STOP\\_FIRST\\_REV](#) 0x10  
*First motion stops by revolution sensor.*
- #define [HOME\\_STOP\\_FIRST\\_SYN](#) 0x20  
*First motion stops by synchronization input.*
- #define [HOME\\_STOP\\_FIRST\\_LIM](#) 0x30  
*First motion stops by limit switch.*
- #define [HOME\\_STOP\\_SECOND\\_BITS](#) 0xC0  
*Bits of the second stop selector.*
- #define [HOME\\_STOP\\_SECOND\\_REV](#) 0x40  
*Second motion stops by revolution sensor.*
- #define [HOME\\_STOP\\_SECOND\\_SYN](#) 0x80  
*Second motion stops by synchronization input.*
- #define [HOME\\_STOP\\_SECOND\\_LIM](#) 0xC0  
*Second motion stops by limit switch.*

### UART parity flags

See Also

[uart.settings.t::UARTSetupFlags](#), [get\\_uart\\_settings](#), [set\\_uart\\_settings](#)

- #define [UART\\_PARITY\\_BITS](#) 0x03  
*Bits of the parity.*
- #define [UART\\_PARITY\\_BIT\\_EVEN](#) 0x00  
*Parity bit 1, if even.*
- #define [UART\\_PARITY\\_BIT\\_ODD](#) 0x01  
*Parity bit 1, if odd.*
- #define [UART\\_PARITY\\_BIT\\_SPACE](#) 0x02  
*Parity bit always 0.*
- #define [UART\\_PARITY\\_BIT\\_MARK](#) 0x03  
*Parity bit always 1.*
- #define [UART\\_PARITY\\_BIT\\_USE](#) 0x04  
*None parity.*
- #define [UART\\_STOP\\_BIT](#) 0x08  
*If set - one stop bit, else two stop bit.*

### Motor Type flags

See Also

[motor.settings.t::MotorType](#), [get\\_motor\\_settings](#), [set\\_motor\\_settings](#)

- #define [MOTOR.TYPE\\_UNKNOWN](#) 0x00  
*Unknown type of engine.*
- #define [MOTOR.TYPE\\_STEP](#) 0x01  
*Step engine.*
- #define [MOTOR.TYPE\\_DC](#) 0x02  
*DC engine.*
- #define [MOTOR.TYPE\\_BLDC](#) 0x03  
*BLDC engine.*

### Encoder settings flags

See Also

[encoder\\_settings\\_t::EncoderSettings](#), [get\\_encoder\\_settings](#), [set\\_encoder\\_settings](#)

- #define [ENCSET\\_DIFFERENTIAL\\_OUTPUT](#) 0x001  
*If flag is set the encoder has differential output, else single ended output.*
- #define [ENCSET\\_PUSHPULL\\_OUTPUT](#) 0x004  
*If flag is set the encoder has push-pull output, else open drain output.*
- #define [ENCSET\\_INDEXCHANNEL\\_PRESENT](#) 0x010  
*If flag is set the encoder has index channel, else encoder hasn't it.*
- #define [ENCSET\\_REVOLUTIONSENSOR\\_PRESENT](#) 0x040  
*If flag is set the encoder has revolution sensor, else encoder hasn't it.*
- #define [ENCSET\\_REVOLUTIONSENSOR\\_ACTIVE\\_HIGH](#) 0x100  
*If flag is set the revolution sensor active state is high logic state, else active state is low logic state.*

### Magnetic brake settings flags

See Also

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

- #define [MB\\_AVAILABLE](#) 0x01  
*If flag is set the magnetic brake is available.*
- #define [MB\\_POWERED\\_HOLD](#) 0x02  
*If this flag is set the magnetic brake is on when powered.*

### Temperature sensor settings flags

See Also

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

- #define [TS\\_TYPE\\_BITS](#) 0x07  
*Bits of the temperature sensor type.*
- #define [TS\\_TYPE\\_UNKNOWN](#) 0x00  
*Unknow type of sensor.*
- #define [TS\\_TYPE\\_THERMOCOUPLE](#) 0x01  
*Thermocouple.*
- #define [TS\\_TYPE\\_SEMICONDUCTOR](#) 0x02  
*The semiconductor temperature sensor.*
- #define [TS\\_AVAILABLE](#) 0x08  
*If flag is set the temperature sensor is available.*
- #define [LS\\_ON\\_SW1\\_AVAILABLE](#) 0x01  
*If flag is set the limit switch connected to pin SW1 is available.*
- #define [LS\\_ON\\_SW2\\_AVAILABLE](#) 0x02  
*If flag is set the limit switch connected to pin SW2 is available.*
- #define [LS\\_SW1\\_ACTIVE\\_LOW](#) 0x04  
*If flag is set the limit switch connected to pin SW1 is triggered by a low level on pin.*
- #define [LS\\_SW2\\_ACTIVE\\_LOW](#) 0x08  
*If flag is set the limit switch connected to pin SW2 is triggered by a low level on pin.*
- #define [LS\\_SHORTED](#) 0x10  
*If flag is set the Limit switches is shorted.*

## Typedefs

- typedef unsigned long long **ulong\_t**
- typedef long long **long\_t**
- typedef int **device\_t**  
*Type describes device identifier.*
- typedef int **result\_t**  
*Type specifies result of any operation.*
- typedef uint32\_t **device\_enumeration\_t**  
*Type describes device enumeration structure.*
- typedef struct **calibration\_t** **calibration\_t**  
*Calibration companion structure.*
- typedef struct **device\_network\_information\_t** **device\_network\_information\_t**  
*TODO.*

## Functions

### Controller settings setup

Functions for adjusting engine read/write almost all controller settings.

- **result\_t** XIMC\_API **set\_feedback\_settings** (**device\_t** id, const **feedback\_settings\_t** \*feedback\_settings)  
*Set feedback settings.*
- **result\_t** XIMC\_API **get\_feedback\_settings** (**device\_t** id, **feedback\_settings\_t** \*feedback\_settings)  
*Read feedback settings.*
- **result\_t** XIMC\_API **set\_home\_settings** (**device\_t** id, const **home\_settings\_t** \*home\_settings)  
*Set home settings.*
- **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)  
*Read home settings.*
- **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)  
*Set command setup movement (speed, acceleration, threshold and etc).*
- **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)  
*Read command setup movement (speed, acceleration, threshold and etc).*
- **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)  
*Set engine settings.*
- **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)  
*Read engine settings.*
- **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)  
*Set engine type and driver type.*
- **result\_t** XIMC\_API **get\_entype\_settings** (**device\_t** id, **entype\_settings\_t** \*entype\_settings)  
*Return engine type and driver type.*
- **result\_t** XIMC\_API **set\_power\_settings** (**device\_t** id, const **power\_settings\_t** \*power\_settings)  
*Set settings of step motor power control.*
- **result\_t** XIMC\_API **get\_power\_settings** (**device\_t** id, **power\_settings\_t** \*power\_settings)  
*Read settings of step motor power control.*

- [result\\_t XIMC\\_API set\\_secure\\_settings](#) ([device\\_t](#) id, const [secure\\_settings\\_t](#) \*secure\_settings)  
*Set protection settings.*
- [result\\_t XIMC\\_API get\\_secure\\_settings](#) ([device\\_t](#) id, [secure\\_settings\\_t](#) \*secure\_settings)  
*Read protection settings.*
- [result\\_t XIMC\\_API set\\_edges\\_settings](#) ([device\\_t](#) id, const [edges\\_settings\\_t](#) \*edges\_settings)  
*Set border and limit switches settings.*
- [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)  
*Read border and limit switches settings.*
- [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)  
*Set PID settings.*
- [result\\_t XIMC\\_API get\\_pid\\_settings](#) ([device\\_t](#) id, [pid\\_settings\\_t](#) \*pid\_settings)  
*Read PID settings.*
- [result\\_t XIMC\\_API set\\_sync\\_in\\_settings](#) ([device\\_t](#) id, const [sync\\_in\\_settings\\_t](#) \*sync\_in\_settings)  
*Set input synchronization settings.*
- [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)  
*Read input synchronization settings.*
- [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)  
*Set output synchronization settings.*
- [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)  
*Read output synchronization settings.*
- [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)  
*Set EXTIO settings.*
- [result\\_t XIMC\\_API get\\_extio\\_settings](#) ([device\\_t](#) id, [extio\\_settings\\_t](#) \*extio\_settings)  
*Read EXTIO settings.*
- [result\\_t XIMC\\_API set\\_brake\\_settings](#) ([device\\_t](#) id, const [brake\\_settings\\_t](#) \*brake\_settings)  
*Set settings of brake control.*
- [result\\_t XIMC\\_API get\\_brake\\_settings](#) ([device\\_t](#) id, [brake\\_settings\\_t](#) \*brake\_settings)  
*Read settings of brake control.*
- [result\\_t XIMC\\_API set\\_control\\_settings](#) ([device\\_t](#) id, const [control\\_settings\\_t](#) \*control\_settings)  
*Set settings of motor control.*
- [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)  
*Read settings of motor control.*
- [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)  
*Set settings of joystick.*
- [result\\_t XIMC\\_API get\\_joystick\\_settings](#) ([device\\_t](#) id, [joystick\\_settings\\_t](#) \*joystick\_settings)  
*Read settings of joystick.*
- [result\\_t XIMC\\_API set\\_ctp\\_settings](#) ([device\\_t](#) id, const [ctp\\_settings\\_t](#) \*ctp\_settings)  
*Set settings of control position(is only used with stepper motor).*
- [result\\_t XIMC\\_API get\\_ctp\\_settings](#) ([device\\_t](#) id, [ctp\\_settings\\_t](#) \*ctp\_settings)  
*Read settings of control position(is only used with stepper motor).*
- [result\\_t XIMC\\_API set\\_uart\\_settings](#) ([device\\_t](#) id, const [uart\\_settings\\_t](#) \*uart\_settings)  
*Set UART settings.*
- [result\\_t XIMC\\_API get\\_uart\\_settings](#) ([device\\_t](#) id, [uart\\_settings\\_t](#) \*uart\_settings)  
*Read UART settings.*

- [result\\_t XIMC\\_API set\\_controller\\_name](#) ([device\\_t](#) id, const [controller\\_name\\_t](#) \*controller\_name)  
*Write user controller name and flags of setting from FRAM.*
- [result\\_t XIMC\\_API get\\_controller\\_name](#) ([device\\_t](#) id, [controller\\_name\\_t](#) \*controller\_name)  
*Read user controller name and flags of setting from FRAM.*

### Group of commands movement control

- [result\\_t XIMC\\_API command\\_stop](#) ([device\\_t](#) id)  
*Immediately stop the engine, the transition to the STOP, mode key BREAK (winding short-circuited), the regime "retention" is deactivated for DC motors, keeping current in the windings for stepper motors (with Power management settings).*
- [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)  
*This command adds one element of the FIFO commands that are executed when input clock pulse.*
- [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)  
*Immediately power off motor regardless its state.*
- [result\\_t XIMC\\_API command\\_move](#) ([device\\_t](#) id, int Position, int uPosition)  
*Upon receiving the command "move" the engine starts to move with pre-set parameters (speed, acceleration, retention), to the point specified to the 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)  
*Upon receiving the command "movr" engine starts to move with pre-set parameters (speed, acceleration, hold), left or right (depending on the sign of DeltaPosition) by the number of pulses specified in the fields 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)  
*The positive direction is to the right.*
- [result\\_t XIMC\\_API command\\_left](#) ([device\\_t](#) id)  
*Start continuous moving to the left.*
- [result\\_t XIMC\\_API command\\_right](#) ([device\\_t](#) id)  
*Start continuous moving to the right.*
- [result\\_t XIMC\\_API command\\_loft](#) ([device\\_t](#) id)  
*Upon receiving the command "loft" the engine is shifted from the current point to a distance GENG :: Antipaly, then move to the same point.*
- [result\\_t XIMC\\_API command\\_sstp](#) ([device\\_t](#) id)  
*soft stop engine.*
- [result\\_t XIMC\\_API get\\_position](#) ([device\\_t](#) id, [get\\_position\\_t](#) \*the\_get\_position)  
*Reads the value position in steps and micro for stepper motor and encoder steps all engines.*
- [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)  
*Sets any position value in steps and micro for stepper motor and encoder steps of all engines.*
- [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)  
*Sets the current position and the position in which the traffic moves by the move command and movr zero for all cases, except for movement to the target position.*

### Group of commands to save and load settings

- [result\\_t XIMC\\_API command\\_save\\_settings](#) ([device\\_t](#) id)  
*Save all settings from controller's RAM to controller's flash memory, replacing previous data in controller's flash memory.*
- [result\\_t XIMC\\_API command\\_read\\_settings](#) ([device\\_t](#) id)  
*Read all settings from controller's flash memory to controller's RAM, replacing previous data in controller's RAM.*
- [result\\_t XIMC\\_API command\\_eesave\\_settings](#) ([device\\_t](#) id)  
*Save settings from controller's RAM to stage's EEPROM memory, which spontaneity connected to stage and it isn't change without it mechanical reconstruction.*

- [result\\_t XIMC\\_API command\\_eeread\\_settings](#) ([device\\_t](#) id)  
*Read settings from controller's RAM to stage's EEPROM memory, which spontaneity connected to stage and it isn't change without it mechanical reconstruction.*
- [result\\_t XIMC\\_API get\\_chart\\_data](#) ([device\\_t](#) id, [chart\\_data\\_t](#) \*chart\_data)  
*Return device electrical parameters, useful for charts.*
- [result\\_t XIMC\\_API get\\_serial\\_number](#) ([device\\_t](#) id, unsigned int \*SerialNumber)  
*Read device serial number.*
- [result\\_t XIMC\\_API get\\_firmware\\_version](#) ([device\\_t](#) id, unsigned int \*Major, unsigned int \*Minor, unsigned int \*Release)  
*Read controller's firmware version.*
- [result\\_t XIMC\\_API service\\_command\\_updf](#) ([device\\_t](#) id)  
*Command puts the controller to update the firmware.*

### Service commands

- [result\\_t XIMC\\_API set\\_serial\\_number](#) ([device\\_t](#) id, const [serial\\_number\\_t](#) \*serial\_number)  
*Write device serial number and hardware version to controller's flash memory.*
- [result\\_t XIMC\\_API get\\_analog\\_data](#) ([device\\_t](#) id, [analog\\_data\\_t](#) \*analog\_data)  
*Read analog data structure that contains raw analog data from ADC embedded on board.*
- [result\\_t XIMC\\_API get\\_debug\\_read](#) ([device\\_t](#) id, [debug\\_read\\_t](#) \*debug\_read)  
*Read data from firmware for debug purpose.*
- [result\\_t XIMC\\_API set\\_debug\\_write](#) ([device\\_t](#) id, const [debug\\_write\\_t](#) \*debug\_write)  
*Write data to firmware for debug purpose.*

### Group of commands to work with EEPROM

- [result\\_t XIMC\\_API set\\_stage\\_name](#) ([device\\_t](#) id, const [stage\\_name\\_t](#) \*stage\_name)  
*Write user stage name from EEPROM.*
- [result\\_t XIMC\\_API get\\_stage\\_name](#) ([device\\_t](#) id, [stage\\_name\\_t](#) \*stage\_name)  
*Read user stage name from EEPROM.*
- [result\\_t XIMC\\_API set\\_stage\\_information](#) ([device\\_t](#) id, const [stage\\_information\\_t](#) \*stage\_information)  
*Set stage information to EEPROM.*
- [result\\_t XIMC\\_API get\\_stage\\_information](#) ([device\\_t](#) id, [stage\\_information\\_t](#) \*stage\_information)  
*Read stage information from EEPROM.*
- [result\\_t XIMC\\_API set\\_stage\\_settings](#) ([device\\_t](#) id, const [stage\\_settings\\_t](#) \*stage\_settings)  
*Set stage settings to EEPROM.*
- [result\\_t XIMC\\_API get\\_stage\\_settings](#) ([device\\_t](#) id, [stage\\_settings\\_t](#) \*stage\_settings)  
*Read stage settings from EEPROM.*
- [result\\_t XIMC\\_API set\\_motor\\_information](#) ([device\\_t](#) id, const [motor\\_information\\_t](#) \*motor\_information)  
*Set motor information to EEPROM.*
- [result\\_t XIMC\\_API get\\_motor\\_information](#) ([device\\_t](#) id, [motor\\_information\\_t](#) \*motor\_information)  
*Read motor information from EEPROM.*
- [result\\_t XIMC\\_API set\\_motor\\_settings](#) ([device\\_t](#) id, const [motor\\_settings\\_t](#) \*motor\_settings)  
*Set motor settings to EEPROM.*
- [result\\_t XIMC\\_API get\\_motor\\_settings](#) ([device\\_t](#) id, [motor\\_settings\\_t](#) \*motor\_settings)  
*Read motor settings from EEPROM.*
- [result\\_t XIMC\\_API set\\_encoder\\_information](#) ([device\\_t](#) id, const [encoder\\_information\\_t](#) \*encoder\_information)  
*Set encoder information to EEPROM.*
- [result\\_t XIMC\\_API get\\_encoder\\_information](#) ([device\\_t](#) id, [encoder\\_information\\_t](#) \*encoder\_information)  
*Read encoder information from EEPROM.*
- [result\\_t XIMC\\_API set\\_encoder\\_settings](#) ([device\\_t](#) id, const [encoder\\_settings\\_t](#) \*encoder\_settings)  
*Set encoder settings to EEPROM.*
- [result\\_t XIMC\\_API get\\_encoder\\_settings](#) ([device\\_t](#) id, [encoder\\_settings\\_t](#) \*encoder\_settings)  
*Read encoder settings from EEPROM.*
- [result\\_t XIMC\\_API set\\_hallsensor\\_information](#) ([device\\_t](#) id, const [hallsensor\\_information\\_t](#) \*hallsensor\_information)  
*Set hall sensor information to EEPROM.*



- [result\\_t XIMC\\_API get\\_hallsensor\\_information](#) ([device\\_t](#) id, [hallsensor\\_information\\_t](#) \*hallsensor\_information)  
*Read hall sensor information from EEPROM.*
- [result\\_t XIMC\\_API set\\_hallsensor\\_settings](#) ([device\\_t](#) id, const [hallsensor\\_settings\\_t](#) \*hallsensor\_settings)  
*Set hall sensor settings to EEPROM.*
- [result\\_t XIMC\\_API get\\_hallsensor\\_settings](#) ([device\\_t](#) id, [hallsensor\\_settings\\_t](#) \*hallsensor\_settings)  
*Read hall sensor settings from EEPROM.*
- [result\\_t XIMC\\_API set\\_gear\\_information](#) ([device\\_t](#) id, const [gear\\_information\\_t](#) \*gear\_information)  
*Set gear information to EEPROM.*
- [result\\_t XIMC\\_API get\\_gear\\_information](#) ([device\\_t](#) id, [gear\\_information\\_t](#) \*gear\_information)  
*Read gear information from EEPROM.*
- [result\\_t XIMC\\_API set\\_gear\\_settings](#) ([device\\_t](#) id, const [gear\\_settings\\_t](#) \*gear\_settings)  
*Set gear settings to EEPROM.*
- [result\\_t XIMC\\_API get\\_gear\\_settings](#) ([device\\_t](#) id, [gear\\_settings\\_t](#) \*gear\_settings)  
*Read gear settings from EEPROM.*
- [result\\_t XIMC\\_API set\\_accessories\\_settings](#) ([device\\_t](#) id, const [accessories\\_settings\\_t](#) \*accessories\_settings)  
*Set additional accessories information to EEPROM.*
- [result\\_t XIMC\\_API get\\_accessories\\_settings](#) ([device\\_t](#) id, [accessories\\_settings\\_t](#) \*accessories\_settings)  
*Read additional accessories information from EEPROM.*
- [result\\_t XIMC\\_API get\\_bootloader\\_version](#) ([device\\_t](#) id, unsigned int \*Major, unsigned int \*Minor, unsigned int \*Release)  
*Read controller's firmware version.*
- [result\\_t XIMC\\_API get\\_init\\_random](#) ([device\\_t](#) id, [init\\_random\\_t](#) \*init\_random)  
*Read random number from controller.*
- [result\\_t XIMC\\_API goto\\_firmware](#) ([device\\_t](#) id, uint8\_t \*ret)  
*Reboot to firmware.*
- [result\\_t XIMC\\_API has\\_firmware](#) (const char \*name, uint8\_t \*ret)  
*Check for firmware on device.*
- [result\\_t XIMC\\_API command\\_update\\_firmware](#) (const char \*name, const uint8\_t \*data, uint32\_t data\_size)  
*Update firmware.*
- [result\\_t XIMC\\_API write\\_key](#) (const char \*name, uint8\_t \*key)  
*Write controller key.*
- [result\\_t XIMC\\_API command\\_reset](#) ([device\\_t](#) id)  
*Reset controller.*
- [result\\_t XIMC\\_API command\\_clear\\_fram](#) ([device\\_t](#) id)  
*Clear controller FRAM.*

## Boards and drivers control

### Functions for searching and opening/closing devices

- typedef char \* [pchar](#)  
*Nevermind.*
- typedef void([XIMC\\_CALLCONV](#) \* [logging\\_callback\\_t](#))(int loglevel, const wchar\_t \*message, void \*user\_data)  
*Logging callback prototype.*
- [device\\_t XIMC\\_API open\\_device](#) (const char \*name)  
*Open a device with OS name name and return identifier of the device which can be used in calls.*
- [result\\_t XIMC\\_API close\\_device](#) ([device\\_t](#) \*id)  
*Close specified device.*
- [result\\_t XIMC\\_API probe\\_device](#) (const char \*name)  
*Check if a device with OS name name is XIMC device.*
- [device\\_enumeration\\_t XIMC\\_API enumerate\\_devices](#) (int enumerate\_flags, const char \*hints)  
*Enumerate all devices that looks like valid.*
- [result\\_t XIMC\\_API free\\_enumerate\\_devices](#) ([device\\_enumeration\\_t](#) device\_enumeration)



- Free memory returned by enumerate\_devices.*
- `int XIMC_API get_device_count (device_enumeration_t device_enumeration)`  
*Get device count.*
- `pchar XIMC_API get_device_name (device_enumeration_t device_enumeration, int device_index)`  
*Get device name from the device enumeration.*
- `result_t XIMC_API get_enumerate_device_serial (device_enumeration_t device_enumeration, int device_index, uint32_t *serial)`  
*Get device serial number from the device enumeration.*
- `result_t XIMC_API get_enumerate_device_information (device_enumeration_t device_enumeration, int device_index, device_information_t *device_information)`  
*Get device information from the device enumeration.*
- `result_t XIMC_API get_enumerate_device_controller_name (device_enumeration_t device_enumeration, int device_index, controller_name_t *controller_name)`  
*TODO.*
- `result_t XIMC_API get_enumerate_device_stage_name (device_enumeration_t device_enumeration, int device_index, stage_name_t *stage_name)`  
*TODO.*
- `result_t XIMC_API get_enumerate_device_network_information (device_enumeration_t device_enumeration, int device_index, device_network_information_t *device_network_information)`  
*TODO.*
- `result_t XIMC_API reset_locks ()`  
*Reset library locks in a case of deadlock.*
- `result_t XIMC_API ximc_fix_usbser_sys (const char *device_name)`  
*Fix for errors in Windows USB driver stack.*
- `void XIMC_API msec_sleep (unsigned int msec)`  
*Sleeps for a specified amount of time.*
- `void XIMC_API ximc_version (char *version)`  
*Returns a library version.*
- `void XIMC_API logging_callback_stderr_wide (int loglevel, const wchar_t *message, void *user_data)`  
*Simple callback for logging to stderr in wide chars.*
- `void XIMC_API logging_callback_stderr_narrow (int loglevel, const wchar_t *message, void *user_data)`  
*Simple callback for logging to stderr in narrow (single byte) chars.*
- `void XIMC_API set_logging_callback (logging_callback_t logging_callback, void *user_data)`  
*Sets a logging callback.*
- `result_t XIMC_API get_status (device_t id, status_t *status)`  
*Return device state.*
- `result_t XIMC_API get_status_calb (device_t id, status_calb_t *status, const calibration_t *calibration)`  
*Calibrated device state.*
- `result_t XIMC_API get_device_information (device_t id, device_information_t *device_information)`  
*Return device information.*

### 5.1.1 Detailed Description

Header file for libximc library.

### 5.1.2 Macro Definition Documentation

#### 5.1.2.1 `#define ALARM_ON_DRIVER_OVERHEATING 0x01`

If this flag is set enter Alarm state on driver overheat signal.

5.1.2.2 `#define BORDER_IS_ENCODER 0x01`

Borders are fixed by predetermined encoder values, if set; borders position on limit switches, if not set.

5.1.2.3 `#define BORDER_STOP_LEFT 0x02`

Motor should stop on left border.

5.1.2.4 `#define BORDER_STOP_RIGHT 0x04`

Motor should stop on right border.

5.1.2.5 `#define BORDERS_SWAP_MISSET_DETECTION 0x08`

Motor should stop on both borders.

Need to save motor then wrong border settings is set

5.1.2.6 `#define BRAKE_ENABLED 0x01`

Brake control is enabled, if this flag is set.

5.1.2.7 `#define BRAKE_ENG_PWROFF 0x02`

Brake turns off power of step motor, if this flag is set.

5.1.2.8 `#define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04`

Pushed left button corresponds to open contact, if this flag is set.

5.1.2.9 `#define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08`

Pushed right button corresponds to open contact, if this flag is set.

5.1.2.10 `#define CONTROL_MODE_BITS 0x03`

Bits to control engine by joystick or buttons.

5.1.2.11 `#define CONTROL_MODE_JOY 0x01`

Control by joystick.

5.1.2.12 `#define CONTROL_MODE_LR 0x02`

Control by left/right buttons.

5.1.2.13 `#define CONTROL_MODE_OFF 0x00`

Control is disabled.

5.1.2.14 `#define CTP_ALARM_ON_ERROR 0x04`

Set ALARM on mismatch, if flag set.

5.1.2.15 `#define CTP_BASE 0x02`

Position control is based on revolution sensor, if this flag is set; otherwise it is based on encoder.

5.1.2.16 `#define CTP_ENABLED 0x01`

Position control is enabled, if flag set.

5.1.2.17 `#define DRIVER_TYPE_DISCRETE_FET 0x01`

Driver with discrete FET keys.

Default option.

5.1.2.18 `#define DRIVER_TYPE_EXTERNAL 0x03`

External driver.

5.1.2.19 `#define DRIVER_TYPE_INTEGRATE 0x02`

Driver with integrated IC.

5.1.2.20 `#define EEPROM_PRECEDENCE 0x01`

If the flag is set settings from external EEPROM override controller settings.

5.1.2.21 `#define ENC_STATE_ABSENT 0x00`

Encoder is absent.

5.1.2.22 `#define ENC_STATE_MALFUNC 0x02`

Encoder is connected and malfunctioning.

5.1.2.23 `#define ENC_STATE_OK 0x04`

Encoder is connected and working properly.

5.1.2.24 `#define ENC_STATE_REVERS 0x03`

Encoder is connected and operational but counts in otyher direction.

5.1.2.25 `#define ENC_STATE_UNKNOWN 0x01`

Encoder state is unknown.

5.1.2.26 `#define ENDER_SW1_ACTIVE_LOW 0x02`

1 - Limit switch connected to pin SW1 is triggered by a low level on pin.

5.1.2.27 `#define ENDER_SW2_ACTIVE_LOW 0x04`

1 - Limit switch connected to pin SW2 is triggered by a low level on pin.

5.1.2.28 `#define ENDER_SWAP 0x01`

First limit switch on the right side, if set; otherwise on the left side.

5.1.2.29 `#define ENGINE_ACCEL_ON 0x10`

Acceleration enable flag.

If it set, motion begins with acceleration and ends with deceleration.

5.1.2.30 `#define ENGINE_ANTIPLAY 0x08`

Play compensation flag.

If it set, engine makes backlash (play) compensation procedure and reach the predetermined position accurately on low speed.

5.1.2.31 `#define ENGINE_LIMIT_CURR 0x40`

Maximum motor current limit enable flag (is only used with DC motor).

5.1.2.32 `#define ENGINE_LIMIT_RPM 0x80`

Maximum motor speed limit enable flag.

5.1.2.33 `#define ENGINE_LIMIT_VOLT 0x20`

Maximum motor voltage limit enable flag (is only used with DC motor).

5.1.2.34 `#define ENGINE_MAX_SPEED 0x04`

Max speed flag.

If it is set, engine uses maximum speed achievable with the present engine settings as nominal speed.

5.1.2.35 `#define ENGINE_REVERSE 0x01`

Reverse flag.

It determines motor shaft rotation direction that corresponds to feedback counts increasing. If not set (default), motor shaft rotation direction under positive voltage corresponds to feedback counts increasing and vice versa. Change it if you see that positive directions on motor and feedback are opposite.

5.1.2.36 `#define ENGINE_TYPE_2DC 0x02`

2 DC motors.

5.1.2.37 `#define ENGINE_TYPE_BRUSHLESS 0x05`

Brushless motor.

5.1.2.38 `#define ENGINE_TYPE_DC 0x01`

DC motor.

5.1.2.39 `#define ENGINE_TYPE_NONE 0x00`

A value that shouldn't be used.

5.1.2.40 `#define ENGINE_TYPE_STEP 0x03`

Step motor.

5.1.2.41 `#define ENGINE_TYPE_TEST 0x04`

Duty cycle are fixed.

Used only manufacturer.

5.1.2.42 `#define ENUMERATE_PROBE 0x01`

Check if a device with OS name name is XIMC device.

Be carefully with this flag because it sends some data to the device.

5.1.2.43 `#define EXTIO_SETUP_INVERT 0x02`

Interpret EXTIO states and fronts inverted if flag is set.

Falling front as input event and low logic level as active state.

5.1.2.44 `#define EXTIO_SETUP_MODE_IN_HOME 0x04`

Issue HOME command.

5.1.2.45 `#define EXTIO_SETUP_MODE_IN_MOVR 0x03`

Issue MOVR command with last used settings.

5.1.2.46 `#define EXTIO_SETUP_MODE_IN_NOP 0x00`

Do nothing.

5.1.2.47 `#define EXTIO_SETUP_MODE_IN_PWOF 0x02`

Issue PWOF command, powering off all engine windings.

5.1.2.48 `#define EXTIO_SETUP_MODE_IN_STOP 0x01`

Issue STOP command, ceasing the engine movement.

5.1.2.49 `#define EXTIO_SETUP_MODE_OUT_ALARM 0x30`

EXTIO pin stays active during Alarm state.

5.1.2.50 `#define EXTIO_SETUP_MODE_OUT_MOTOR_FOUND 0x50`

EXTIO pin stays active when motor is connected (first winding).

5.1.2.51 `#define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40`

EXTIO pin stays active when windings are powered.

5.1.2.52 `#define EXTIO_SETUP_MODE_OUT_MOVING 0x20`

EXTIO pin stays active during moving state.

5.1.2.53 `#define EXTIO_SETUP_MODE_OUT_OFF 0x00`

EXTIO pin always set in inactive state.

5.1.2.54 `#define EXTIO_SETUP_MODE_OUT_ON 0x10`

EXTIO pin always set in active state.

5.1.2.55 `#define EXTIO_SETUP_OUTPUT 0x01`

EXTIO works as output if flag is set, works as input otherwise.

5.1.2.56 `#define FEEDBACK_EMF 0x04`

Feedback by EMF.

5.1.2.57 `#define FEEDBACK_ENC_REVERSE 0x01`

Reverse count of encoder.

5.1.2.58 `#define FEEDBACK_ENCODER 0x01`

Feedback by encoder.

5.1.2.59 `#define FEEDBACK_ENCODERHALL 0x03`

Feedback by Hall detector.

5.1.2.60 `#define FEEDBACK_HALL_REVERSE 0x02`

Reverse count position on the Hall sensor.

5.1.2.61 `#define FEEDBACK_NONE 0x05`

Feedback is absent.

5.1.2.62 `#define H_BRIDGE_ALERT 0x04`

If this flag is set then turn off the power unit with a signal problem in one of the transistor bridge.

5.1.2.63 `#define HOME_DIR_FIRST 0x01`

Flag defines direction of 1st motion after execution of home command.

Direction is right, if set; otherwise left.

5.1.2.64 `#define HOME_DIR_SECOND 0x02`

Flag defines direction of 2nd motion.

Direction is right, if set; otherwise left.

5.1.2.65 `#define HOME_HALF_MV 0x08`

If the flag is set, the stop signals are ignored in start of second movement the first half-turn.

5.1.2.66 `#define HOME_MV_SEC_EN 0x04`

Use the second phase of calibration to the home position, if set; otherwise the second phase is skipped.

5.1.2.67 `#define HOME_STOP_FIRST_BITS 0x30`

Bits of the first stop selector.

5.1.2.68 `#define HOME_STOP_FIRST_LIM 0x30`

First motion stops by limit switch.

5.1.2.69 `#define HOME_STOP_FIRST_REV 0x10`

First motion stops by revolution sensor.

5.1.2.70 `#define HOME_STOP_FIRST_SYN 0x20`

First motion stops by synchronization input.

5.1.2.71 `#define HOME_STOP_SECOND_BITS 0xC0`

Bits of the second stop selector.

5.1.2.72 `#define HOME_STOP_SECOND_LIM 0xC0`

Second motion stops by limit switch.

5.1.2.73 `#define HOME_STOP_SECOND_REV 0x40`

Second motion stops by revolution sensor.

5.1.2.74 `#define HOME_STOP_SECOND_SYN 0x80`

Second motion stops by synchronization input.

5.1.2.75 `#define JOY_REVERSE 0x01`

Joystick action is reversed.

Joystick deviation to the upper values correspond to negative speeds and vice versa.

5.1.2.76 `#define LOW_UPWR_PROTECTION 0x02`

If this flag is set turn off motor when voltage is lower than LowUpwrOff.

5.1.2.77 `#define MICROSTEP_MODE_FRAC_128 0x08`

1/128 step mode.

5.1.2.78 `#define MICROSTEP_MODE_FRAC_16 0x05`

1/16 step mode.

5.1.2.79 `#define MICROSTEP_MODE_FRAC_2 0x02`

1/2 step mode.

5.1.2.80 `#define MICROSTEP_MODE_FRAC_256 0x09`

1/256 step mode.

5.1.2.81 `#define MICROSTEP_MODE_FRAC_32 0x06`

1/32 step mode.

5.1.2.82 `#define MICROSTEP_MODE_FRAC_4 0x03`

1/4 step mode.



5.1.2.83 `#define MICROSTEP_MODE_FRAC_64 0x07`

1/64 step mode.

5.1.2.84 `#define MICROSTEP_MODE_FRAC_8 0x04`

1/8 step mode.

5.1.2.85 `#define MICROSTEP_MODE_FULL 0x01`

Full step mode.

5.1.2.86 `#define MOVE_STATE_ANTIPLAY 0x04`

Motor is playing compensation, if flag set.

5.1.2.87 `#define MOVE_STATE_MOVING 0x01`

This flag indicates that controller is trying to move the motor.

5.1.2.88 `#define MOVE_STATE_TARGET_SPEED 0x02`

Target speed is reached, if flag set.

5.1.2.89 `#define MVCMD_ERROR 0x40`

Finish state (1 - move command have finished with an error, 0 - move command have finished correctly).

This flags is actual when MVCMD\_RUNNING signals movement finish.

5.1.2.90 `#define MVCMD_HOME 0x06`

Command home.

5.1.2.91 `#define MVCMD_LEFT 0x03`

Command left.

5.1.2.92 `#define MVCMD_LOFT 0x07`

Command loft.

5.1.2.93 `#define MVCMD_MOVE 0x01`

Command move.

5.1.2.94 `#define MVCMD_MOVR 0x02`

Command movr.

5.1.2.95 `#define MVCMD_NAME_BITS 0x3F`

Move command bit mask.

5.1.2.96 `#define MVCMD_RIGHT 0x04`

Command right.

5.1.2.97 `#define MVCMD_RUNNING 0x80`

Move command state (0 - move command have finished, 1 - move command is being executed).

5.1.2.98 `#define MVCMD_SSTP 0x08`

Command soft stop.

5.1.2.99 `#define MVCMD_STOP 0x05`

Command stop.

5.1.2.100 `#define MVCMD_UKNWN 0x00`

Unknown command.

5.1.2.101 `#define POWER_OFF_ENABLED 0x02`

Power off enabled after PowerOffDelay, if this flag is set.

5.1.2.102 `#define POWER_REDUCT_ENABLED 0x01`

Current reduction enabled after CurrReductDelay, if this flag is set.

5.1.2.103 `#define POWER_SMOOTH_CURRENT 0x04`

Current ramp-up/down is performed smoothly during current.set\_time, if this flag is set.

5.1.2.104 `#define PWR_STATE_MAX 0x05`

Motor windings are powered by maximum current driver can provide at this voltage.

5.1.2.105 `#define PWR_STATE_NORM 0x03`

Motor windings are powered by nominal current.

5.1.2.106 `#define PWR_STATE_OFF 0x01`

Motor windings are disconnected from the driver.

5.1.2.107 `#define PWR_STATE_REDUCT 0x04`

Motor windings are powered by reduced current to lower power consumption.

5.1.2.108 `#define PWR_STATE_UNKNOWN 0x00`

Unknown state, should never happen.

5.1.2.109 `#define REV_SENS_INV 0x08`

Sensor is active when it 0 and invert makes active level 1.

That is, if you do not invert, it is normal logic - 0 is the activation.

5.1.2.110 `#define SETPOS_IGNORE_ENCODER 0x02`

Will not reload encoder state if this flag is set.

5.1.2.111 `#define SETPOS_IGNORE_POSITION 0x01`

Will not reload position in steps/microsteps if this flag is set.

5.1.2.112 `#define STATE_ALARM 0x00040`

Controller is in alarm state indicating that something dangerous had happened.

Most commands are ignored in this state. To reset the flag a STOP command must be issued.

5.1.2.113 `#define STATE_BORDERS_SWAP_MISSET 0x08000`

Engine stuck at the wrong edge.

5.1.2.114 `#define STATE_BRAKE 0x0200`

State of Brake pin.

5.1.2.115 `#define STATE_BUTTON_LEFT 0x0008`

Button "left" state (1 if pressed).

5.1.2.116 `#define STATE_BUTTON_RIGHT 0x0004`

Button "right" state (1 if pressed).

5.1.2.117 `#define STATE_CONTR 0x0003F`

Flags of controller states.

5.1.2.118 `#define STATE_CONTROLLER_OVERHEAT 0x00200`

Controller overheat.

5.1.2.119 `#define STATE_CTP_ERROR 0x00080`

Control position error(is only used with stepper motor).

5.1.2.120 `#define STATE_DIG_SIGNAL 0xFFFF`

Flags of digital signals.

5.1.2.121 `#define STATE_EEPROM_CONNECTED 0x00010`

EEPROM with settings is connected.

5.1.2.122 `#define STATE_ENC_A 0x2000`

State of encoder A pin.

5.1.2.123 `#define STATE_ENC_B 0x4000`

State of encoder B pin.

5.1.2.124 `#define STATE_ERRC 0x00001`

Command error encountered.

5.1.2.125 `#define STATE_ERRD 0x00002`

Data integrity error encountered.

5.1.2.126 `#define STATE_ERRV 0x00004`

Value error encountered.

5.1.2.127 `#define STATE_GPIO_LEVEL 0x0020`

State of external GPIO pin.

5.1.2.128 `#define STATE_GPIO_PINOUT 0x0010`

External GPIO works as Out, if flag set; otherwise works as In.

5.1.2.129 `#define STATE_HALL_A 0x0040`

State of Hall.a pin.

5.1.2.130 `#define STATE_HALL_B 0x0080`

State of Hall.b pin.

5.1.2.131 `#define STATE_HALL_C 0x0100`

State of Hall.c pin.

5.1.2.132 `#define STATE_LEFT_EDGE 0x0002`

Engine stuck at the left edge.

5.1.2.133 `#define STATE_LOW_USB_VOLTAGE 0x02000`

USB voltage is insufficient for normal operation.

5.1.2.134 `#define STATE_OVERLOAD_POWER_CURRENT 0x00800`

Power current exceeds safe limit.

5.1.2.135 `#define STATE_OVERLOAD_POWER_VOLTAGE 0x00400`

Power voltage exceeds safe limit.

5.1.2.136 `#define STATE_OVERLOAD_USB_CURRENT 0x04000`

USB current exceeds safe limit.

5.1.2.137 `#define STATE_OVERLOAD_USB_VOLTAGE 0x01000`

USB voltage exceeds safe limit.

5.1.2.138 `#define STATE_POWER_OVERHEAT 0x00100`

Power driver overheat.

5.1.2.139 `#define STATE_REV_SENSOR 0x0400`

State of Revolution sensor pin.

5.1.2.140 `#define STATE_RIGHT_EDGE 0x0001`

Engine stuck at the right edge.

5.1.2.141 `#define STATE_SECUR 0x3FFC0`

Flags of security.

5.1.2.142 `#define STATE_SYNC_INPUT 0x0800`

State of Sync input pin.

5.1.2.143 `#define STATE_SYNC_OUTPUT 0x1000`

State of Sync output pin.

5.1.2.144 `#define SYNCIN_ENABLED 0x01`

Synchronization in mode is enabled, if this flag is set.

5.1.2.145 `#define SYNCIN_GOTOPOSITION 0x04`

The engine is go to position specified in Position and uPosition, if this flag is set.

And it is shift on the Position and uPosition, if this flag is unset

5.1.2.146 `#define SYNCIN_INVERT 0x02`

Trigger on falling edge if flag is set, on rising edge otherwise.

5.1.2.147 `#define SYNCOUT_ENABLED 0x01`

Synchronization out pin follows the synchronization logic, if set.

It governed by SYNCOUT\_STATE flag otherwise.

5.1.2.148 `#define SYNCOUT_IN_STEPS 0x08`

Use motor steps/encoder pulses instead of milliseconds for output pulse generation if the flag is set.

5.1.2.149 `#define SYNCOUT_INVERT 0x04`

Low level is active, if set, and high level is active otherwise.

5.1.2.150 `#define SYNCOUT_ONPERIOD 0x40`

Generate synchronization pulse every SyncOutPeriod encoder pulses.

5.1.2.151 `#define SYNCOUT_ONSTART 0x10`

Generate synchronization pulse when movement starts.

5.1.2.152 `#define SYNCOUT_ONSTOP 0x20`

Generate synchronization pulse when movement stops.

5.1.2.153 `#define SYNCOUT_STATE 0x02`

When output state is fixed by negative SYNCOUT\_ENABLED flag, the pin state is in accordance with this flag state.

5.1.2.154 `#define UART_PARITY_BITS 0x03`

Bits of the parity.

5.1.2.155 `#define WIND_A_STATE_ABSENT 0x00`

Winding A is disconnected.

5.1.2.156 `#define WIND_A_STATE_MALFUNC 0x02`

Winding A is short-circuited.

5.1.2.157 `#define WIND_A_STATE_OK 0x03`

Winding A is connected and working properly.

5.1.2.158 `#define WIND_A_STATE_UNKNOWN 0x01`

Winding A state is unknown.

5.1.2.159 `#define WIND_B_STATE_ABSENT 0x00`

Winding B is disconnected.

5.1.2.160 `#define WIND_B_STATE_MALFUNC 0x20`

Winding B is short-circuited.

5.1.2.161 `#define WIND_B_STATE_OK 0x30`

Winding B is connected and working properly.

5.1.2.162 `#define WIND_B_STATE_UNKNOWN 0x10`

Winding B state is unknown.

5.1.2.163 `#define XIMC_API`

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

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

### 5.1.3 Typedef Documentation

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

Logging callback prototype.

Parameters

<i>loglevel</i>	a loglevel
<i>message</i>	a message

### 5.1.4 Function Documentation

#### 5.1.4.1 **result\_t XIMC\_API** close\_device ( **device\_t** \* id )

Close specified device.

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

#### 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 )

This command adds one element of the FIFO commands that are executed when input clock pulse.

Each pulse synchronization or perform that action, which is described in SSNI, if the buffer is empty, or the oldest loaded into the buffer action to temporarily replace the speed and coordinate in SSNI. In the latter case this action is erased from the buffer. The number of remaining empty buffer elements can be found in the structure of GETS.

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

#### 5.1.4.3 **result\_t XIMC\_API** command\_clear\_fram ( **device\_t** id )

Clear controller FRAM.

Can be used by manufacturer only

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

#### 5.1.4.4 **result\_t XIMC\_API** command\_eeread\_settings ( **device\_t** id )

Read settings from controller's RAM to stage's EEPROM memory, which spontaneity connected to stage and it isn't change without it mechanical reconstruction.

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

#### 5.1.4.5 **result\_t XIMC\_API** command\_eesave\_settings ( **device\_t** id )

Save settings from controller's RAM to stage's EEPROM memory, which spontaneity connected to stage and it isn't change without it mechanical reconstruction.

Can be used by manufacturer only.

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------



5.1.4.6 **result\_t XIMC\_API** command\_home ( **device\_t** id )

The positive direction is to the right.

A value of zero reverses the direction of the direction of the flag, the set speed. Restriction imposed by the trailer, act the same, except that the limit switch contact does not stop. Limit the maximum speed, acceleration and deceleration function. 1) moves the motor according to the speed FastHome, uFastHome and flag HOME\_DIR\_FAST until limit switch, if the flag is set HOME\_STOP\_ENDS, until the signal from the input synchronization if the flag HOME\_STOP\_SYNC (as accurately as possible is important to catch the moment of operation limit switch) or until the signal is received from the speed sensor, if the flag HOME\_STOP\_REV\_SN 2) then moves according to the speed SlowHome, uSlowHome and flag HOME\_DIR\_SLOW until signal from the clock input, if the flag HOME\_MV\_SEC. If the flag HOME\_MV\_SEC reset skip this paragraph. 3) then move the motor according to the speed FastHome, uFastHome and flag HOME\_DIR\_SLOW a distance HomeDelta, uHomeDelta. description of flags and variable see in description for commands GHOM/SHOM

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

See Also

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

5.1.4.7 **result\_t XIMC\_API** command\_left ( **device\_t** id )

Start continous moving to the left.

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

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

Upon receiving the command "loft" the engine is shifted from the current point to a distance GENG :: Antiplay, then move to the same point.

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

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

Upon receiving the command "move" the engine starts to move with pre-set parameters (speed, acceleration, retention), to the point specified to the Position, uPosition.

For stepper motor uPosition sets the microstep for DC motor, this field is not used.

Parameters

<i>Position</i>	position to move.
<i>uPosition</i>	part of the position to move, microsteps. Range: -255..255.
<i>id</i>	an identifier of device

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

Upon receiving the command "movr" engine starts to move with pre-set parameters (speed, acceleration, hold), left or right (depending on the sign of DeltaPosition) by the number of pulses specified in the fields DeltaPosition, uDeltaPosition.

For stepper motor uDeltaPosition sets the microstep for DC motor, this field is not used.

Parameters

<i>DeltaPosition</i>	shift from initial position.
<i>uDeltaPosition</i>	part of the offset shift, microsteps. Range: -255..255.
<i>id</i>	an identifier of device

5.1.4.11 **result\_t XIMC\_API** command\_power\_off ( **device\_t** id )

Immediately power off motor regardless its state.

Shouldn't be used during motion as the motor could be power on again automatically to continue movement. The command is designed for manual motor power off. When automatic power off after stop is required, use power management system.

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

See Also

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

5.1.4.12 **result\_t XIMC\_API** command\_read\_settings ( **device\_t** id )

Read all settings from controller's flash memory to controller's RAM, replacing previous data in controller's RAM.

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

5.1.4.13 **result\_t XIMC\_API** command\_reset ( **device\_t** id )

Reset controller.

Can be used by manufacturer only

Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

5.1.4.14 **result\_t XIMC\_API** command\_right ( **device\_t** id )

Start continous moving to the right.

## Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

5.1.4.15 **result\_t XIMC\_API** command\_save\_settings ( **device\_t** id )

Save all settings from controller's RAM to controller's flash memory, replacing previous data in controller's flash memory.

## Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

5.1.4.16 **result\_t XIMC\_API** command\_sstp ( **device\_t** id )

soft stop engine.

The motor stops with deceleration speed.

## Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

5.1.4.17 **result\_t XIMC\_API** command\_stop ( **device\_t** id )

Immediately stop the engine, the transition to the STOP, mode key BREAK (winding short-circuited), the regime "retention" is deactivated for DC motors, keeping current in the windings for stepper motors (with Power management settings).

## Parameters

<i>id</i>	an identifier of device
-----------	-------------------------

5.1.4.18 **result\_t XIMC\_API** command\_update\_firmware ( const char \* name, const uint8\_t \* data, uint32\_t data\_size )

Update firmware.

Service command

## Parameters

<i>name</i>	a name of device
<i>data</i>	firmware byte stream
<i>data_size</i>	size of byte stream

5.1.4.19 **result\_t XIMC\_API** command\_zero ( **device\_t** id )

Sets the current position and the position in which the traffic moves by the move command and movr zero for all cases, except for movement to the target position.

In the latter case, set the zero current position and the target position counted so that the absolute position of the destination is the same. That is, if we were at 400 and moved to 500, then the command Zero makes the current position of 0, and the position of the destination - 100. Does not change the mode of movement that is if the motion

is carried, it continues, and if the engine is in the "hold", the type of retention remains.

Parameters

	<i>id</i>	an identifier of device
--	-----------	-------------------------

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

Enumerate all devices that looks like valid.

Parameters

in	<i>probe_flags</i>	enumerate devices flags
in	<i>hints</i>	extended information

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

Free memory returned by *enumerate\_devices*.

Parameters

in	<i>device_enumeration</i>	opaque pointer to an enumeration device data
----	---------------------------	--

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

Read additional accessories information from EEPROM.

Parameters

	<i>id</i>	an identifier of device
out	<i>accessories_settings</i>	structure contains information about additional accessories

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

Read analog data structure that contains raw analog data from ADC embedded on board.

This function used for device testing and deep recalibration by manufacturer only.

Parameters

	<i>id</i>	an identifier of device
out	<i>analog_data</i>	analog data coefficients

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

Read controller's firmware version.

## Parameters

	<i>id</i>	an identifier of device
out	<i>Major</i>	major version
out	<i>Minor</i>	minor version
out	<i>Release</i>	release version

5.1.4.25 **result\_t XIMC\_API** get\_brake\_settings ( **device\_t** id, **brake\_settings\_t** \* brake\_settings )

Read settings of brake control.

## Parameters

	<i>id</i>	an identifier of device
out	<i>brake_settings</i>	structure contains settings of brake control

5.1.4.26 **result\_t XIMC\_API** get\_chart\_data ( **device\_t** id, **chart\_data\_t** \* chart\_data )

Return device electrical parameters, useful for charts.

Useful function that fill structure with snapshot of controller voltages and currents.

See Also

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

## Parameters

	<i>id</i>	an identifier of device
out	<i>chart_data</i>	structure with snapshot of controller parameters.

5.1.4.27 **result\_t XIMC\_API** get\_control\_settings ( **device\_t** id, **control\_settings\_t** \* control\_settings )

Read settings of motor control.

When choosing CTL\_MODE = 1 switches motor control with the joystick. In this mode, the joystick to the maximum engine tends Move at MaxSpeed [i], where i = 0 if the previous use This mode is not selected another i. Buttons switch the room rate i. When CTL\_MODE = 2 is switched on motor control using the Left / right. When you click on the button motor starts to move in the appropriate direction at a speed MaxSpeed [0], at the end of time Timeout [i] motor move at a speed MaxSpeed [i+1]. at Transition from MaxSpeed [i] on MaxSpeed [i +1] to acceleration, as usual.

## Parameters

	<i>id</i>	an identifier of device
out	<i>control_settings</i>	structure contains settings motor control by joystick or buttons left/right.

5.1.4.28 **result\_t XIMC\_API** get\_controller\_name ( **device\_t** id, **controller\_name\_t** \* controller\_name )

Read user controller name and flags of setting from FRAM.

## Parameters

	<i>id</i>	an identifier of device
out	<i>controller_name</i>	structure contains previously set user controller name

5.1.4.29 **result\_t XIMC\_API** get\_ctp\_settings ( **device\_t** id, **ctp\_settings\_t** \* ctp\_settings )

Read settings of control position(is only used with stepper motor).

When controlling the step motor with encoder (CTP\_BASE 0) it is possible to detect the loss of steps. The controller knows the number of steps per revolution (GENG :: StepsPerRev) and the encoder resolution (GFBS :: IPT). When the control (flag CTP\_ENABLED), the controller stores the current position in the footsteps of SM and the current position of the encoder. Further, at each step of the position encoder is converted into steps and if the difference is greater CTPMinError, a flag STATE\_CTP\_ERROR. When controlling the step motor with speed sensor (CTP\_BASE 1), the position is controlled by him. The active edge of input clock controller stores the current value of steps. Further, at each turn checks how many steps shifted. When a mismatch CTPMinError a flag STATE\_CTP\_ERROR.

Parameters

	<i>id</i>	an identifier of device
out	<i>ctp_settings</i>	structure contains settings of control position

5.1.4.30 **result\_t XIMC\_API** get\_debug\_read ( **device\_t** id, **debug\_read\_t** \* debug\_read )

Read data from firmware for debug purpose.

Its use depends on context, firmware version and previous history.

Parameters

	<i>id</i>	an identifier of device
out	<i>debug_read</i>	Debug data.

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

Get device count.

Parameters

in	<i>device_enumeration</i>	opaque pointer to an enumeration device data
----	---------------------------	--

5.1.4.32 **result\_t XIMC\_API** get\_device\_information ( **device\_t** id, **device\_information\_t** \* device\_information )

Return device information.

All fields must point to allocated string buffers with at least 10 bytes. Works with both raw or initialized device.

Parameters

	<i>id</i>	an identifier of device
out	<i>device_information</i>	device information Device information.

See Also

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

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

Get device name from the device enumeration.

Returns *device\_index* device name.

Parameters

in	<i>device_enumeration</i>	opaque pointer to an enumeration device data
in	<i>device_index</i>	device index

5.1.4.34 **result\_t XIMC\_API** get\_edges\_settings ( **device\_t** id, **edges\_settings\_t** \* edges\_settings )

Read border and limit switches settings.

See Also

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

Parameters

	<i>id</i>	an identifier of device
out	<i>edges_settings</i>	edges settings, specify types of borders, motor behaviour and electrical behaviour of limit switches

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

Read encoder information from EEPROM.

Parameters

	<i>id</i>	an identifier of device
out	<i>encoder_information</i>	structure contains information about encoder

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

Read encoder settings from EEPROM.

Parameters

	<i>id</i>	an identifier of device
out	<i>encoder_settings</i>	structure contains encoder settings

5.1.4.37 **result\_t XIMC\_API** get\_engine\_settings ( **device\_t** id, **engine\_settings\_t** \* engine\_settings )

Read engine settings.

This function fill structure with set of useful motor settings stored in controller's memory. These settings specify motor shaft movement algorithm, list of limitations and rated characteristics.

See Also

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

Parameters

	<i>id</i>	an identifier of device
out	<i>engine_settings</i>	engine settings

5.1.4.38 **result\_t XIMC\_API** get\_entype\_settings ( **device\_t** id, **entype\_settings\_t** \* entype\_settings )

Return engine type and driver type.

Parameters

	<i>id</i>	an identifier of device
out	<i>EngineType</i>	engine type
out	<i>DriverType</i>	driver type

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

Get device information from the device enumeration.

Returns *device\_index* device serial number.

Parameters

in	<i>device_enumeration</i>	opaque pointer to an enumeration device data
in	<i>device_index</i>	device index
out	<i>device_information</i>	device information data

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

Get device serial number from the device enumeration.

Returns *device\_index* device serial number.

Parameters

in	<i>device_enumeration</i>	opaque pointer to an enumeration device data
in	<i>device_index</i>	device index
out	<i>serial</i>	device serial number

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

Read EXTIO settings.

This function reads a structure with a set of EXTIO settings from controller's memory.



See Also

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

Parameters

	<i>id</i>	an identifier of device
out	<i>extio_settings</i>	EXTIO settings

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

Read feedback settings.

Parameters

	<i>id</i>	an identifier of device
out	<i>IPS</i>	number of encoder pulses per shaft revolution. Range: 1..65535
out	<i>FeedbackType</i>	type of feedback
out	<i>FeedbackFlags</i>	flags of feedback

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

Read controller's firmware version.

Parameters

	<i>id</i>	an identifier of device
out	<i>Major</i>	major version
out	<i>Minor</i>	minor version
out	<i>Release</i>	release version

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

Read gear information from EEPROM.

Parameters

	<i>id</i>	an identifier of device
out	<i>gear_information</i>	structure contains information about step gearhead

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

Read gear settings from EEPROM.

Parameters

	<i>id</i>	an identifier of device
out	<i>gear_settings</i>	structure contains step gearhead settings

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

Read hall sensor information from EEPROM.

Parameters

	<i>id</i>	an identifier of device
out	<i>hallsensor_information</i>	structure contains information about hall sensor

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

Read hall sensor settings from EEPROM.

Parameters

	<i>id</i>	an identifier of device
out	<i>hallsensor_settings</i>	structure contains hall sensor settings

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

Read home settings.

This function fill structure with settings of calibrating position.

See Also

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

Parameters

	<i>id</i>	an identifier of device
out	<i>home_settings</i>	calibrating position settings

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

Read random number from controller.

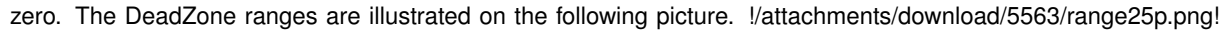
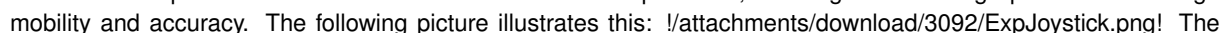
Parameters

	<i>id</i>	an identifier of device
out	<i>random</i>	sequence generated by the controller

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

Read settings of joystick.

If joystick position is outside DeadZone limits from the central position a movement with speed, defined by the joystick DeadZone edge to 100% deviation, begins. Joystick positions inside DeadZone limits correspond to zero speed (soft stop of motion) and positions beyond Low and High limits correspond MaxSpeed [i] or -MaxSpeed [i] (see command SCTL), where i = 0 by default and can be changed with left/right buttons (see command SCTL). If next

speed in list is zero (both integer and microstep parts), the button press is ignored. First speed in list shouldn't be zero. The DeadZone ranges are illustrated on the following picture.  The relationship between the deviation and the rate is exponential, allowing no switching speed combine high mobility and accuracy. The following picture illustrates this:  The nonlinearity parameter is adjustable. Setting it to zero makes deviation/speed relation linear.

Parameters

	<i>id</i>	an identifier of device
out	<i>joystick.settings</i>	structure contains joystick settings

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

Read motor information from EEPROM.

Parameters

	<i>id</i>	an identifier of device
out	<i>motor-information</i>	structure contains motor information

5.1.4.52 **result\_t XIMC\_API** get\_motor\_settings ( **device\_t** id, **motor.settings\_t** \* motor\_settings )

Read motor settings from EEPROM.

Parameters

	<i>id</i>	an identifier of device
out	<i>motor.settings</i>	structure contains motor settings

5.1.4.53 **result\_t XIMC\_API** get\_move\_settings ( **device\_t** id, **move.settings\_t** \* move\_settings )

Read command setup movement (speed, acceleration, threshold and etc).

Parameters

	<i>id</i>	an identifier of device
out	<i>move.settings</i>	structure contains move settings: speed, acceleration, deceleration etc.

5.1.4.54 **result\_t XIMC\_API** get\_pid\_settings ( **device\_t** id, **pid.settings\_t** \* pid\_settings )

Read PID settings.

This function fill structure with set of motor PID settings stored in controller's memory. These settings specify behaviour of PID routine for voltage. These factors are slightly different for different positioners. All boards are supplied with standart set of PID setting on controller's flash memory.

See Also

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

## Parameters

	<i>id</i>	an identifier of device
out	<i>pid_settings</i>	pid settings

5.1.4.55 **result\_t XIMC\_API** get\_position ( **device\_t** id, **get\_position\_t** \* the\_get\_position )

Reads the value position in steps and micro for stepper motor and encoder steps all engines.

## Parameters

	<i>id</i>	an identifier of device
out	<i>position</i>	structure contains move settings: speed, acceleration, deceleration etc.

5.1.4.56 **result\_t XIMC\_API** get\_power\_settings ( **device\_t** id, **power\_settings\_t** \* power\_settings )

Read settings of step motor power control.

Used with stepper motor only.

## Parameters

	<i>id</i>	an identifier of device
out	<i>power_settings</i>	structure contains settings of step motor power control

5.1.4.57 **result\_t XIMC\_API** get\_secure\_settings ( **device\_t** id, **secure\_settings\_t** \* secure\_settings )

Read protection settings.

## Parameters

	<i>id</i>	an identifier of device
out	<i>secure_settings</i>	critical parameter settings to protect the hardware

See Also

status\_t::flags

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

Read device serial number.

## Parameters

	<i>id</i>	an identifier of device
out	<i>serial</i>	serial number

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

Read stage information from EEPROM.

## Parameters

	<i>id</i>	an identifier of device
out	<i>stage_information</i>	structure contains stage information

5.1.4.60 **result\_t XIMC\_API** get\_stage\_name ( **device\_t** id, **stage\_name\_t** \* stage\_name )

Read user stage name from EEPROM.

## Parameters

	<i>id</i>	an identifier of device
out	<i>stage_name</i>	structure contains previously set user stage name

5.1.4.61 **result\_t XIMC\_API** get\_stage\_settings ( **device\_t** id, **stage\_settings\_t** \* stage\_settings )

Read stage settings from EEPROM.

## Parameters

	<i>id</i>	an identifier of device
out	<i>stage_settings</i>	structure contains stage settings

5.1.4.62 **result\_t XIMC\_API** get\_status ( **device\_t** id, **status\_t** \* status )

Return device state.

## Parameters

	<i>id</i>	an identifier of device
out	<i>status</i>	structure with snapshot of controller status Device state. Useful structure that contains current controller status, including speed, position and boolean flags.

See Also

[get\\_status](#)

5.1.4.63 **result\_t XIMC\_API** get\_status\_calb ( **device\_t** id, **status\_calb\_t** \* status, const **calibration\_t** \* calibration )

Calibrated device state.

Useful structure that contains current controller status, including speed, position and boolean flags.

See Also

[get\\_status](#)

5.1.4.64 **result\_t XIMC\_API** get\_sync\_in\_settings ( **device\_t** id, **sync\_in\_settings\_t** \* sync\_in\_settings )

Read input synchronization settings.

This function fill structure with set of input synchronization settings, modes, periods and flags, that specify behaviour of input synchronization. All boards are supplied with standart set of these settings.

See Also

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

Parameters

	<i>id</i>	an identifier of device
out	<i>sync_in_settings</i>	synchronization settings

5.1.4.65 **result\_t XIMC\_API** get\_sync\_out\_settings ( **device\_t** id, **sync\_out\_settings\_t** \* sync\_out\_settings )

Read output synchronization settings.

This function fill structure with set of output synchronization settings, modes, periods and flags, that specify behaviour of output synchronization. All boards are supplied with standart set of these settings.

See Also

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

Parameters

	<i>id</i>	an identifier of device
out	<i>sync_out_settings</i>	synchronization settings

5.1.4.66 **result\_t XIMC\_API** get\_uart\_settings ( **device\_t** id, **uart\_settings\_t** \* uart\_settings )

Read UART settings.

This function fill structure with UART settings.

See Also

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

Parameters

	<i>Speed</i>	UART speed
out	<i>uart_settings</i>	UART settings

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

Reboot to firmware.

Parameters

	<i>id</i>	an identifier of device
out	<i>ret</i>	RESULT_OK, if reboot to firmware is possible. Reboot is done after reply to this command. RESULT_NO_FIRMWARE, if firmware is not found. RESULT_ALREADY_IN_FIRMWARE, if this command was sent when controller is already in firmware.

5.1.4.68 **result\_t XIMC\_API** has\_firmware ( const char \* name, uint8\_t \* ret )

Check for firmware on device.

Parameters

	<i>name</i>	a name of device
out	<i>ret</i>	non-zero if firmware existed

5.1.4.69 void **XIMC\_API** logging\_callback\_stderr\_narrow ( int loglevel, const wchar\_t \* message, void \* user\_data )

Simple callback for logging to stderr in narrow (single byte) chars.

Parameters

<i>loglevel</i>	a loglevel
<i>message</i>	a message

5.1.4.70 void **XIMC\_API** logging\_callback\_stderr\_wide ( int loglevel, const wchar\_t \* message, void \* user\_data )

Simple callback for logging to stderr in wide chars.

Parameters

<i>loglevel</i>	a loglevel
<i>message</i>	a message

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

Sleeps for a specified amount of time.

Parameters

<i>msec</i>	time in milliseconds
-------------	----------------------

5.1.4.72 **device\_t XIMC\_API** open\_device ( const char \* name )

Open a device with OS name *name* and return identifier of the device which can be used in calls.

Parameters

in	<i>name</i>	- a device name - e.g. COM3 or /dev/tty.s123
----	-------------	--

5.1.4.73 **result\_t XIMC\_API** probe\_device ( const char \* name )

Check if a device with OS name *name* is XIMC device.

Be carefully with this call because it sends some data to the device.

## Parameters

<i>in</i>	<i>name</i>	- a device name
-----------	-------------	-----------------

5.1.4.74 **result\_t XIMC\_API** service\_command\_updf ( **device\_t** id )

Command puts the controller to update the firmware.

After receiving this command, the firmware board sets a flag (for loader), sends echo reply and restarts the controller.

5.1.4.75 **result\_t XIMC\_API** set\_accessories\_settings ( **device\_t** id, const **accessories\_settings\_t** \* accessories\_settings )

Set additional accessories information to EEPROM.

Can be used by manufacturer only.

## Parameters

	<i>id</i>	an identifier of device
<i>in</i>	<i>accessories_settings</i>	structure contains information about additional accessories

5.1.4.76 **result\_t XIMC\_API** set\_brake\_settings ( **device\_t** id, const **brake\_settings\_t** \* brake\_settings )

Set settings of brake control.

## Parameters

	<i>id</i>	an identifier of device
<i>in</i>	<i>brake_settings</i>	structure contains settings of brake control

5.1.4.77 **result\_t XIMC\_API** set\_control\_settings ( **device\_t** id, const **control\_settings\_t** \* control\_settings )

Set settings of motor control.

When choosing CTL\_MODE = 1 switches motor control with the joystick. In this mode, the joystick to the maximum engine tends Move at MaxSpeed [i], where i = 0 if the previous use This mode is not selected another i. Buttons switch the room rate i. When CTL\_MODE = 2 is switched on motor control using the Left / right. When you click on the button motor starts to move in the appropriate direction at a speed MaxSpeed [0], at the end of time Timeout [i] motor move at a speed MaxSpeed [i+1]. at Transition from MaxSpeed [i] on MaxSpeed [i + 1] to acceleration, as usual.

## Parameters

	<i>id</i>	an identifier of device
<i>in</i>	<i>control_settings</i>	structure contains settings motor control by joystick or buttons left/right.

5.1.4.78 **result\_t XIMC\_API** set\_controller\_name ( **device\_t** id, const **controller\_name\_t** \* controller\_name )

Write user controller name and flags of setting from FRAM.



## Parameters

	<i>id</i>	an identifier of device
in	<i>controller_name</i>	structure contains previously set user controller name

5.1.4.79 **result\_t XIMC\_API** set\_ctp\_settings ( **device\_t** id, const **ctp\_settings\_t** \* ctp\_settings )

Set settings of control position(is only used with stepper motor).

When controlling the step motor with encoder (CTP.BASE 0) it is possible to detect the loss of steps. The controller knows the number of steps per revolution (GENG :: StepsPerRev) and the encoder resolution (GFBS :: IPT). When the control (flag CTP.ENABLED), the controller stores the current position in the footsteps of SM and the current position of the encoder. Further, at each step of the position encoder is converted into steps and if the difference is greater CTPMinError, a flag STATE\_CTP\_ERROR. When controlling the step motor with speed sensor (CTP.BASE 1), the position is controlled by him. The active edge of input clock controller stores the current value of steps. Further, at each turn checks how many steps shifted. When a mismatch CTPMinError a flag STATE\_CTP\_ERROR.

## Parameters

	<i>id</i>	an identifier of device
in	<i>ctp_settings</i>	structure contains settings of control position

5.1.4.80 **result\_t XIMC\_API** set\_debug\_write ( **device\_t** id, const **debug\_write\_t** \* debug\_write )

Write data to firmware for debug purpose.

## Parameters

	<i>id</i>	an identifier of device
in	<i>debug_write</i>	Debug data.

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

Set border and limit switches settings.

See Also

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

## Parameters

	<i>id</i>	an identifier of device
in	<i>edges_settings</i>	edges settings, specify types of borders, motor behaviour and electrical behaviour of limit switches

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

Set encoder information to EEPROM.

Can be used by manufacturer only.

## Parameters

	<i>id</i>	an identifier of device
in	<i>encoder_information</i>	structure contains information about encoder

5.1.4.83 **result\_t XIMC\_API** set\_encoder\_settings ( **device\_t** id, const **encoder\_settings\_t** \* encoder\_settings )

Set encoder settings to EEPROM.

Can be used by manufacturer only.

## Parameters

	<i>id</i>	an identifier of device
in	<i>encoder_settings</i>	structure contains encoder settings

5.1.4.84 **result\_t XIMC\_API** set\_engine\_settings ( **device\_t** id, const **engine\_settings\_t** \* engine\_settings )

Set engine settings.

This function send structure with set of engine settings to controller's memory. These settings specify motor shaft movement algorithm, list of limitations and rated characteristics. Use it when you change motor, encoder, positioner etc. Please note that wrong engine settings lead to device malfunction, can lead to irreversible damage of board.

See Also

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

## Parameters

	<i>id</i>	an identifier of device
in	<i>engine_settings</i>	engine settings

5.1.4.85 **result\_t XIMC\_API** set\_entype\_settings ( **device\_t** id, const **entype\_settings\_t** \* entype\_settings )

Set engine type and driver type.

## Parameters

	<i>id</i>	an identifier of device
in	<i>EngineType</i>	engine type
in	<i>DriverType</i>	driver type

5.1.4.86 **result\_t XIMC\_API** set\_extio\_settings ( **device\_t** id, const **extio\_settings\_t** \* extio\_settings )

Set EXTIO settings.

This function writes a structure with a set of EXTIO settings to controller's memory. By default input event are signalled through rising front and output states are signalled by high logic state.

See Also

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

## Parameters

	<i>id</i>	an identifier of device
in	<i>extio_settings</i>	EXTIO settings

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

Set feedback settings.

## Parameters

	<i>id</i>	an identifier of device
in	<i>IPS</i>	number of encoder pulses per shaft revolution. Range: 1..65535
in	<i>FeedbackType</i>	type of feedback
in	<i>FeedbackFlags</i>	flags of feedback

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

Set gear information to EEPROM.

Can be used by manufacturer only.

## Parameters

	<i>id</i>	an identifier of device
in	<i>gear_information</i>	structure contains information about step gearhead

5.1.4.89 **result\_t XIMC\_API** set\_gear\_settings ( **device\_t** id, const **gear\_settings\_t** \* gear\_settings )

Set gear settings to EEPROM.

Can be used by manufacturer only.

## Parameters

	<i>id</i>	an identifier of device
in	<i>gear_settings</i>	structure contains step gearhead settings

5.1.4.90 **result\_t XIMC\_API** set\_hallsensor\_information ( **device\_t** id, const **hallsensor\_information\_t** \* hallsensor\_information )

Set hall sensor information to EEPROM.

Can be used by manufacturer only.

## Parameters

	<i>id</i>	an identifier of device
in	<i>hallsensor-information</i>	structure contains information about hall sensor

5.1.4.91 **result\_t XIMC\_API** set\_hallsensor\_settings ( **device\_t** id, const **hallsensor\_settings\_t** \* hallsensor\_settings )

Set hall sensor settings to EEPROM.

Can be used by manufacturer only.

Parameters

	<i>id</i>	an identifier of device
<i>in</i>	<i>hallsensor_settings</i>	structure contains hall sensor settings

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

Set home settings.

This function send structure with calibrating position settings to controller's memory.

See Also

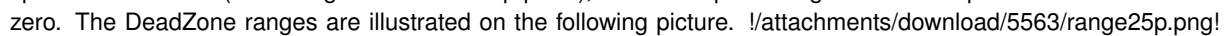
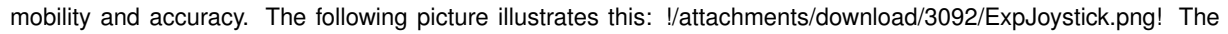
[home\\_settings\\_t](#)

Parameters

	<i>id</i>	an identifier of device
<i>in</i>	<i>home_settings</i>	calibrating position settings

5.1.4.93 **result\_t XIMC\_API** set\_joystick\_settings ( **device\_t** id, const **joystick\_settings\_t** \* joystick\_settings )

Set settings of joystick.

If joystick position is outside DeadZone limits from the central position a movement with speed, defined by the joystick DeadZone edge to 100% deviation, begins. Joystick positions inside DeadZone limits correspond to zero speed (soft stop of motion) and positions beyond Low and High limits correspond MaxSpeed [i] or -MaxSpeed [i] (see command SCTL), where i = 0 by default and can be changed with left/right buttons (see command SCTL). If next speed in list is zero (both integer and microstep parts), the button press is ignored. First speed in list shouldn't be zero. The DeadZone ranges are illustrated on the following picture.  The relationship between the deviation and the rate is exponential, allowing no switching speed combine high mobility and accuracy. The following picture illustrates this:  The nonlinearity parameter is adjustable. Setting it to zero makes deviation/speed relation linear.

Parameters

	<i>id</i>	an identifier of device
<i>in</i>	<i>joystick_settings</i>	structure contains joystick settings

5.1.4.94 **void XIMC\_API** set\_logging\_callback ( **logging\_callback\_t** logging\_callback, void \* user\_data )

Sets a logging callback.

Call resets a callback to default (stderr, syslog) if NULL passed.

## Parameters

<i>logging_callback</i>	a callback for log messages
-------------------------	-----------------------------

5.1.4.95 **result\_t XIMC\_API** set\_motor\_information ( **device\_t** id, const **motor\_information\_t** \* motor\_information )

Set motor information to EEPROM.

Can be used by manufacturer only.

## Parameters

	<i>id</i>	an identifier of device
in	<i>motor_information</i>	structure contains motor information

5.1.4.96 **result\_t XIMC\_API** set\_motor\_settings ( **device\_t** id, const **motor\_settings\_t** \* motor\_settings )

Set motor settings to EEPROM.

Can be used by manufacturer only.

## Parameters

	<i>id</i>	an identifier of device
in	<i>motor_settings</i>	structure contains motor information

5.1.4.97 **result\_t XIMC\_API** set\_move\_settings ( **device\_t** id, const **move\_settings\_t** \* move\_settings )

Set command setup movement (speed, acceleration, threshold and etc).

## Parameters

	<i>id</i>	an identifier of device
in	<i>move_settings</i>	structure contains move settings: speed, acceleration, deceleration etc.

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

Set PID settings.

This function send structure with set of PID factors to controller's memory. These settings specify behaviour of PID routine for voltage. These factors are slightly different for different positioners. All boards are supplied with standart set of PID setting on controller's flash memory. Please use it for loading new PID settings when you change positioner. Please note that wrong PID settings lead to device malfunction.

See Also

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

## Parameters

	<i>id</i>	an identifier of device
in	<i>pid_settings</i>	pid settings

5.1.4.99 **result\_t XIMC\_API** set\_position ( **device\_t** id, const **set\_position\_t** \* the\_set\_position )

Sets any position value in steps and micro for stepper motor and encoder steps of all engines.

It means, that changing main indicator of position.

Parameters

	<i>id</i>	an identifier of device
out	<i>position</i>	structure contains move settings: speed, acceleration, deceleration etc.

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

Set settings of step motor power control.

Used with stepper motor only.

Parameters

	<i>id</i>	an identifier of device
in	<i>power_settings</i>	structure contains settings of step motor power control

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

Set protection settings.

Parameters

	<i>id</i>	an identifier of device
	<i>secure_settings</i>	structure with secure data

See Also

status.t::flags

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

Write device serial number and hardware version to controller's flash memory.

Along with the new serial number and hardware version a "Key" is transmitted. The SN and hardware version are changed and saved when keys match. Can be used by manufacturer only.

Parameters

	<i>id</i>	an identifier of device
in	<i>serial</i>	number structure contains new serial number and secret key.

5.1.4.103 **result\_t XIMC\_API** set\_stage\_information ( **device\_t** id, const **stage\_information\_t** \* stage\_information )

Set stage information to EEPROM.

Can be used by manufacturer only.

## Parameters

	<i>id</i>	an identifier of device
in	<i>stage_information</i>	structure contains stage information

5.1.4.104 **result.t XIMC\_API** set\_stage\_name ( **device.t** id, const **stage\_name.t** \* stage\_name )

Write user stage name from EEPROM.

## Parameters

	<i>id</i>	an identifier of device
in	<i>stage_name</i>	structure contains previously set user stage name

5.1.4.105 **result.t XIMC\_API** set\_stage\_settings ( **device.t** id, const **stage\_settings.t** \* stage\_settings )

Set stage settings to EEPROM.

Can be used by manufacturer only

## Parameters

	<i>id</i>	an identifier of device
in	<i>stage_settings</i>	structure contains stage settings

5.1.4.106 **result.t XIMC\_API** set\_sync\_in\_settings ( **device.t** id, const **sync.in.settings.t** \* sync.in.settings )

Set input synchronization settings.

This function send structure with set of input synchronization settings, that specify behaviour of input synchronization, to controller's memory. All boards are supplied with standart set of these settings.

See Also

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

## Parameters

	<i>id</i>	an identifier of device
in	<i>sync.in.settings</i>	synchronization settings

5.1.4.107 **result.t XIMC\_API** set\_sync\_out\_settings ( **device.t** id, const **sync.out.settings.t** \* sync.out.settings )

Set output synchronization settings.

This function send structure with set of output synchronization settings, that specify behaviour of output synchronization, to controller's memory. All boards are supplied with standart set of these settings.

See Also

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

## Parameters

	<i>id</i>	an identifier of device
<i>in</i>	<i>sync_out_settings</i>	synchronization settings

5.1.4.108 **result.t XIMC\_API** `set_uart_settings ( device.t id, const uart.settings.t * uart_settings )`

Set UART settings.

This function send structure with UART settings to controller's memory.

See Also

[uart.settings.t](#)

## Parameters

	<i>Speed</i>	UART speed
<i>in</i>	<i>uart_settings</i>	UART settings

5.1.4.109 **result.t XIMC\_API** `write_key ( const char * name, uint8_t * key )`

Write controller key.

Can be used by manufacturer only

## Parameters

	<i>name</i>	a name of device
<i>in</i>	<i>key</i>	protection key. Range: 0..4294967295

5.1.4.110 **result.t XIMC\_API** `ximc_fix_usbser_sys ( const char * device_name )`

Fix for errors in Windows USB driver stack.

USB subsystem on Windows does not always work correctly. The following bugs are possible: the device cannot be opened at all, or the device can be opened and written to, but it will not respond with data. These errors can be fixed by device reconnection or removal-rescan in device manager. [ximc\\_fix\\_usbser\\_sys\(\)](#) is a shortcut function to do the remove-rescan process. You should call this function if libximc library cannot open the device which was not physically removed from the system or if the device does not respond.

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

Returns a library version.

## Parameters

<i>version</i>	a buffer to hold a version string, 32 bytes is enough
----------------	---



# Index

- A1Voltage
  - [analog\\_data.t](#), 11
- A1Voltage\_ADC
  - [analog\\_data.t](#), 11
- A2Voltage
  - [analog\\_data.t](#), 11
- A2Voltage\_ADC
  - [analog\\_data.t](#), 11
- ACurrent
  - [analog\\_data.t](#), 11
- ACurrent\_ADC
  - [analog\\_data.t](#), 11
- Accel
  - [move\\_settings\\_calb.t](#), 45
  - [move\\_settings.t](#), 46
- accessories\_settings.t, 8
  - LimitSwitchesSettings, 9
  - MBRatedCurrent, 9
  - MBRatedVoltage, 9
  - MBSSettings, 9
  - MBTorque, 9
  - MagneticBrakeInfo, 9
  - TSGrad, 9
  - TSMMax, 9
  - TSMMin, 9
  - TSSettings, 9
  - TemperatureSensorInfo, 9
- Accuracy
  - [sync\\_out\\_settings\\_calb.t](#), 62
  - [sync\\_out\\_settings.t](#), 63
- [analog\\_data.t](#), 10
  - A1Voltage, 11
  - A1Voltage\_ADC, 11
  - A2Voltage, 11
  - A2Voltage\_ADC, 11
  - ACurrent, 11
  - ACurrent\_ADC, 11
  - B1Voltage, 11
  - B1Voltage\_ADC, 12
  - B2Voltage, 12
  - B2Voltage\_ADC, 12
  - BCurrent, 12
  - BCurrent\_ADC, 12
  - FullCurrent, 12
  - FullCurrent\_ADC, 12
  - Joy, 12
  - Joy\_ADC, 12
  - L, 12
  - L5, 12
  - L5\_ADC, 12
  - Pot, 13
  - R, 13
  - SupVoltage, 13
  - SupVoltage\_ADC, 13
  - Temp, 13
  - Temp\_ADC, 13
- Antiplay
  - [engine\\_settings\\_calb.t](#), 27
  - [engine\\_settings.t](#), 29
- AntiplaySpeed
  - [move\\_settings\\_calb.t](#), 45
  - [move\\_settings.t](#), 46
- B1Voltage
  - [analog\\_data.t](#), 11
- B1Voltage\_ADC
  - [analog\\_data.t](#), 12
- B2Voltage
  - [analog\\_data.t](#), 12
- B2Voltage\_ADC
  - [analog\\_data.t](#), 12
- BCurrent
  - [analog\\_data.t](#), 12
- BCurrent\_ADC
  - [analog\\_data.t](#), 12
- BORDER\_IS\_ENCODER
  - [ximc.h](#), 85
- BORDER\_STOP\_LEFT
  - [ximc.h](#), 86
- BORDER\_STOP\_RIGHT
  - [ximc.h](#), 86
- BRAKE\_ENABLED
  - [ximc.h](#), 86
- BRAKE\_ENG\_PWROFF
  - [ximc.h](#), 86
- BorderFlags
  - [edges\\_settings\\_calb.t](#), 23
  - [edges\\_settings.t](#), 24
- [brake\\_settings.t](#), 13
  - BrakeFlags, 14
  - t1, 14
  - t2, 14
  - t3, 14
  - t4, 14
- BrakeFlags
  - [brake\\_settings.t](#), 14
- CONTROL\_MODE\_BITS
  - [ximc.h](#), 86

CONTROL\_MODE\_JOY  
     ximc.h, 86  
 CONTROL\_MODE\_LR  
     ximc.h, 86  
 CONTROL\_MODE\_OFF  
     ximc.h, 86  
 CTP\_ALARM\_ON\_ERROR  
     ximc.h, 86  
 CTP\_BASE  
     ximc.h, 87  
 CTP\_ENABLED  
     ximc.h, 87  
 CTPFlags  
     ctp\_settings.t, 20  
 CTPMinError  
     ctp\_settings.t, 20  
 calibration.t, 14  
 chart\_data.t, 15  
     DutyCycle, 15  
     Joy, 15  
     Pot, 15  
     WindingCurrentA, 16  
     WindingCurrentB, 16  
     WindingCurrentC, 16  
     WindingVoltageA, 16  
     WindingVoltageB, 16  
     WindingVoltageC, 16  
 close\_device  
     ximc.h, 100  
 ClutterTime  
     sync\_in\_settings.calb.t, 60  
     sync\_in\_settings.t, 61  
 CmdBufFreeSpace  
     status.calb.t, 56  
     status.t, 58  
 command\_add\_sync\_in\_action  
     ximc.h, 100  
 command\_add\_sync\_in\_action.calb.t, 16  
     Position, 16  
     Time, 16  
 command\_add\_sync\_in\_action.t, 17  
     Time, 17  
     uPosition, 17  
 command\_clear\_fram  
     ximc.h, 100  
 command\_eeread\_settings  
     ximc.h, 100  
 command\_eesave\_settings  
     ximc.h, 100  
 command\_home  
     ximc.h, 100  
 command\_left  
     ximc.h, 101  
 command\_loft  
     ximc.h, 101  
 command\_move  
     ximc.h, 101  
 command\_movr  
     ximc.h, 101  
 command\_power\_off  
     ximc.h, 102  
 command\_read\_settings  
     ximc.h, 102  
 command\_reset  
     ximc.h, 102  
 command\_right  
     ximc.h, 102  
 command\_save\_settings  
     ximc.h, 103  
 command\_sstp  
     ximc.h, 103  
 command\_stop  
     ximc.h, 103  
 command\_update\_firmware  
     ximc.h, 103  
 command\_zero  
     ximc.h, 103  
 control\_settings.calb.t, 17  
     Flags, 18  
     MaxClickTime, 18  
     MaxSpeed, 18  
     Timeout, 18  
 control\_settings.t, 18  
     Flags, 19  
     MaxClickTime, 19  
     MaxSpeed, 19  
     Timeout, 19  
     uDeltaPosition, 19  
     uMaxSpeed, 19  
 controller\_name.t, 19  
     ControllerName, 20  
     CtrlFlags, 20  
 ControllerName  
     controller\_name.t, 20  
 CriticalIpwr  
     secure\_settings.t, 49  
 CriticalIusb  
     secure\_settings.t, 49  
 CriticalIT  
     secure\_settings.t, 49  
 CriticalUpwr  
     secure\_settings.t, 49  
 CriticalUusb  
     secure\_settings.t, 49  
 ctp\_settings.t, 20  
     CTPFlags, 20  
     CTPMinError, 20  
 CtrlFlags  
     controller\_name.t, 20  
 CurPosition  
     status.calb.t, 56  
     status.t, 58  
 CurSpeed  
     status.calb.t, 56  
     status.t, 58  
 CurT

- status\_calb.t, 56
- status.t, 58
- CurrReductDelay
  - power\_settings.t, 48
- CurrentSetTime
  - power\_settings.t, 48
- DRIVER\_TYPE\_EXTERNAL
  - ximc.h, 87
- DeadZone
  - joystick\_settings.t, 40
- debug\_read.t, 21
  - DebugData, 21
- debug\_write.t, 21
  - DebugData, 22
- DebugData
  - debug\_read.t, 21
  - debug\_write.t, 22
- Decel
  - move\_settings\_calb.t, 45
  - move\_settings.t, 46
- DetentTorque
  - motor\_settings.t, 42
- device\_information.t, 22
  - Major, 22
  - Minor, 22
  - Release, 22
- device\_network\_information.t, 23
- DriverType
  - entype\_settings.t, 30
- DutyCycle
  - chart\_data.t, 15
- EEPROM\_PRECEDENCE
  - ximc.h, 87
- ENC\_STATE\_ABSENT
  - ximc.h, 87
- ENC\_STATE\_MALFUNC
  - ximc.h, 87
- ENC\_STATE\_OK
  - ximc.h, 87
- ENC\_STATE\_REVERS
  - ximc.h, 87
- ENC\_STATE\_UNKNOWN
  - ximc.h, 87
- ENDER\_SW1\_ACTIVE\_LOW
  - ximc.h, 87
- ENDER\_SW2\_ACTIVE\_LOW
  - ximc.h, 88
- ENDER\_SWAP
  - ximc.h, 88
- ENGINE\_ACCEL\_ON
  - ximc.h, 88
- ENGINE\_ANTIPLAY
  - ximc.h, 88
- ENGINE\_LIMIT\_CURR
  - ximc.h, 88
- ENGINE\_LIMIT\_RPM
  - ximc.h, 88
- ENGINE\_LIMIT\_VOLT
  - ximc.h, 88
- ENGINE\_MAX\_SPEED
  - ximc.h, 88
- ENGINE\_REVERSE
  - ximc.h, 88
- ENGINE\_TYPE\_2DC
  - ximc.h, 88
- ENGINE\_TYPE\_DC
  - ximc.h, 89
- ENGINE\_TYPE\_NONE
  - ximc.h, 89
- ENGINE\_TYPE\_STEP
  - ximc.h, 89
- ENGINE\_TYPE\_TEST
  - ximc.h, 89
- ENUMERATE\_PROBE
  - ximc.h, 89
- EXTIO\_SETUP\_INVERT
  - ximc.h, 89
- EXTIO\_SETUP\_OUTPUT
  - ximc.h, 90
- EXTIOModeFlags
  - extio\_settings.t, 31
- EXTIOSetupFlags
  - extio\_settings.t, 31
- edges\_settings\_calb.t, 23
  - BorderFlags, 23
  - EnderFlags, 23
  - LeftBorder, 23
  - RightBorder, 23
- edges\_settings.t, 24
  - BorderFlags, 24
  - EnderFlags, 24
  - LeftBorder, 24
  - RightBorder, 24
  - uLeftBorder, 25
  - uRightBorder, 25
- Efficiency
  - gear\_settings.t, 33
- EncPosition
  - get\_position\_calb.t, 34
  - get\_position.t, 35
  - set\_position\_calb.t, 51
  - set\_position.t, 52
  - status\_calb.t, 56
  - status.t, 58
- EncSts
  - status\_calb.t, 56
  - status.t, 58
- encoder\_information.t, 25
  - Manufacturer, 25
  - PartNumber, 25
- encoder\_settings.t, 26
  - EncoderSettings, 26
  - MaxCurrentConsumption, 26
  - MaxOperatingFrequency, 26
  - SupplyVoltageMax, 26

- SupplyVoltageMin, 26
- EncoderSettings
  - encoder\_settings.t, 26
- EnderFlags
  - edges\_settings\_calb.t, 23
  - edges\_settings.t, 24
- engine\_settings\_calb.t, 27
  - Antiplay, 27
  - EngineFlags, 27
  - MicrostepMode, 27
  - NomCurrent, 27
  - NomSpeed, 27
  - NomVoltage, 27
  - StepsPerRev, 28
- engine\_settings.t, 28
  - Antiplay, 29
  - EngineFlags, 29
  - MicrostepMode, 29
  - NomCurrent, 29
  - NomSpeed, 29
  - NomVoltage, 29
  - StepsPerRev, 29
  - uNomSpeed, 29
- EngineFlags
  - engine\_settings\_calb.t, 27
  - engine\_settings.t, 29
- EngineType
  - entype\_settings.t, 30
- entype\_settings.t, 29
  - DriverType, 30
  - EngineType, 30
- enumerate\_devices
  - ximc.h, 104
- ExpFactor
  - joystick\_settings.t, 40
- extio\_settings.t, 30
  - EXTIOModeFlags, 31
  - EXTIOSetupFlags, 31
- FEEDBACK\_EMF
  - ximc.h, 90
- FEEDBACK\_ENC\_REVERSE
  - ximc.h, 90
- FEEDBACK\_ENCODER
  - ximc.h, 90
- FEEDBACK\_ENCODERHALL
  - ximc.h, 90
- FEEDBACK\_NONE
  - ximc.h, 91
- FastHome
  - home\_settings\_calb.t, 37
  - home\_settings.t, 38
- feedback\_settings.t, 31
  - FeedbackFlags, 31
  - FeedbackType, 31
  - HallSPR, 32
  - HallShift, 32
- FeedbackFlags
  - feedback\_settings.t, 31
- FeedbackType
  - feedback\_settings.t, 31
- Flags
  - control\_settings\_calb.t, 18
  - control\_settings.t, 19
  - secure\_settings.t, 49
  - status\_calb.t, 56
  - status.t, 58
- free\_enumerate\_devices
  - ximc.h, 104
- FullCurrent
  - analog\_data.t, 12
- FullCurrent\_ADC
  - analog\_data.t, 12
- GPIOFlags
  - status\_calb.t, 56
  - status.t, 58
- gear\_information.t, 32
  - Manufacturer, 32
  - PartNumber, 32
- gear\_settings.t, 32
  - Efficiency, 33
  - InputInertia, 33
  - MaxOutputBacklash, 33
  - RatedInputSpeed, 33
  - RatedInputTorque, 33
  - ReductionIn, 34
  - ReductionOut, 34
- get\_accessories\_settings
  - ximc.h, 104
- get\_analog\_data
  - ximc.h, 104
- get\_bootloader\_version
  - ximc.h, 104
- get\_brake\_settings
  - ximc.h, 105
- get\_chart\_data
  - ximc.h, 105
- get\_control\_settings
  - ximc.h, 105
- get\_controller\_name
  - ximc.h, 105
- get\_ctp\_settings
  - ximc.h, 106
- get\_debug\_read
  - ximc.h, 106
- get\_device\_count
  - ximc.h, 106
- get\_device\_information
  - ximc.h, 106
- get\_device\_name
  - ximc.h, 106
- get\_edges\_settings
  - ximc.h, 107
- get\_encoder\_information
  - ximc.h, 107
- get\_encoder\_settings
  - ximc.h, 107

- get\_engine\_settings
  - ximc.h, [107](#)
- get\_entype\_settings
  - ximc.h, [108](#)
- get\_enumerate\_device\_information
  - ximc.h, [108](#)
- get\_enumerate\_device\_serial
  - ximc.h, [108](#)
- get\_extio\_settings
  - ximc.h, [108](#)
- get\_feedback\_settings
  - ximc.h, [109](#)
- get\_firmware\_version
  - ximc.h, [109](#)
- get\_gear\_information
  - ximc.h, [109](#)
- get\_gear\_settings
  - ximc.h, [109](#)
- get\_hallsensor\_information
  - ximc.h, [109](#)
- get\_hallsensor\_settings
  - ximc.h, [110](#)
- get\_home\_settings
  - ximc.h, [110](#)
- get\_init\_random
  - ximc.h, [110](#)
- get\_joystick\_settings
  - ximc.h, [110](#)
- get\_motor\_information
  - ximc.h, [111](#)
- get\_motor\_settings
  - ximc.h, [111](#)
- get\_move\_settings
  - ximc.h, [111](#)
- get\_pid\_settings
  - ximc.h, [111](#)
- get\_position
  - ximc.h, [112](#)
- get\_position\_calb\_t, [34](#)
  - EncPosition, [34](#)
  - Position, [34](#)
- get\_position\_t, [34](#)
  - EncPosition, [35](#)
- get\_power\_settings
  - ximc.h, [112](#)
- get\_secure\_settings
  - ximc.h, [112](#)
- get\_serial\_number
  - ximc.h, [112](#)
- get\_stage\_information
  - ximc.h, [112](#)
- get\_stage\_name
  - ximc.h, [113](#)
- get\_stage\_settings
  - ximc.h, [113](#)
- get\_status
  - ximc.h, [113](#)
- get\_status\_calb
  - ximc.h, [113](#)
- get\_sync\_in\_settings
  - ximc.h, [113](#)
- get\_sync\_out\_settings
  - ximc.h, [114](#)
- get\_uart\_settings
  - ximc.h, [114](#)
- goto\_firmware
  - ximc.h, [114](#)
- H\_BRIDGE\_ALERT
  - ximc.h, [91](#)
- HOME\_DIR\_FIRST
  - ximc.h, [91](#)
- HOME\_DIR\_SECOND
  - ximc.h, [91](#)
- HOME\_HALF\_MV
  - ximc.h, [91](#)
- HOME\_MV\_SEC\_EN
  - ximc.h, [91](#)
- HOME\_STOP\_FIRST\_LIM
  - ximc.h, [91](#)
- HOME\_STOP\_FIRST\_REV
  - ximc.h, [91](#)
- HOME\_STOP\_FIRST\_SYN
  - ximc.h, [91](#)
- HallSPR
  - feedback\_settings\_t, [32](#)
- HallShift
  - feedback\_settings\_t, [32](#)
- hallsensor\_information\_t, [35](#)
  - Manufacturer, [35](#)
  - PartNumber, [35](#)
- hallsensor\_settings\_t, [35](#)
  - MaxCurrentConsumption, [36](#)
  - MaxOperatingFrequency, [36](#)
  - SupplyVoltageMax, [36](#)
  - SupplyVoltageMin, [36](#)
- has\_firmware
  - ximc.h, [114](#)
- HoldCurrent
  - power\_settings\_t, [48](#)
- home\_settings\_calb\_t, [36](#)
  - FastHome, [37](#)
  - HomeDelta, [37](#)
  - HomeFlags, [37](#)
  - SlowHome, [37](#)
- home\_settings\_t, [37](#)
  - FastHome, [38](#)
  - HomeDelta, [38](#)
  - HomeFlags, [38](#)
  - SlowHome, [38](#)
  - uFastHome, [38](#)
  - uHomeDelta, [38](#)
  - uSlowHome, [38](#)
- HomeDelta
  - home\_settings\_calb\_t, [37](#)
  - home\_settings\_t, [38](#)
- HomeFlags

- home\_settings\_calb.t, 37
- home\_settings.t, 38
- HorizontalLoadCapacity
  - stage\_settings.t, 54
- init\_random.t, 38
  - key, 39
- InputInertia
  - gear\_settings.t, 33
- lpwr
  - status\_calb.t, 56
  - status.t, 59
- lusb
  - status\_calb.t, 56
  - status.t, 59
- JOY\_REVERSE
  - ximc.h, 92
- Joy
  - analog\_data.t, 12
  - chart\_data.t, 15
- Joy\_ADC
  - analog\_data.t, 12
- JoyCenter
  - joystick\_settings.t, 40
- JoyFlags
  - joystick\_settings.t, 40
- JoyHighEnd
  - joystick\_settings.t, 40
- JoyLowEnd
  - joystick\_settings.t, 40
- joystick\_settings.t, 39
  - DeadZone, 40
  - ExpFactor, 40
  - JoyCenter, 40
  - JoyFlags, 40
  - JoyHighEnd, 40
  - JoyLowEnd, 40
- Key
  - serial\_number.t, 50
- key
  - init\_random.t, 39
- L
  - analog\_data.t, 12
- L5
  - analog\_data.t, 12
- L5\_ADC
  - analog\_data.t, 12
- LOW\_UPWR\_PROTECTION
  - ximc.h, 92
- LeadScrewPitch
  - stage\_settings.t, 54
- LeftBorder
  - edges\_settings\_calb.t, 23
  - edges\_settings.t, 24
- LimitSwitchesSettings
  - accessories\_settings.t, 9
- logging\_callback\_stderr\_narrow
  - ximc.h, 115
- logging\_callback\_stderr\_wide
  - ximc.h, 115
- logging\_callback.t
  - ximc.h, 99
- LowUpwrOff
  - secure\_settings.t, 49
- MBRatedCurrent
  - accessories\_settings.t, 9
- MBRatedVoltage
  - accessories\_settings.t, 9
- MBSettings
  - accessories\_settings.t, 9
- MBTorque
  - accessories\_settings.t, 9
- MICROSTEP\_MODE\_FULL
  - ximc.h, 93
- MOVE\_STATE\_ANTIPLAY
  - ximc.h, 93
- MOVE\_STATE\_MOVING
  - ximc.h, 93
- MVCMD\_ERROR
  - ximc.h, 93
- MVCMD\_HOME
  - ximc.h, 93
- MVCMD\_LEFT
  - ximc.h, 93
- MVCMD\_LOFT
  - ximc.h, 93
- MVCMD\_MOVE
  - ximc.h, 93
- MVCMD\_MOVR
  - ximc.h, 93
- MVCMD\_NAME\_BITS
  - ximc.h, 93
- MVCMD\_RIGHT
  - ximc.h, 94
- MVCMD\_RUNNING
  - ximc.h, 94
- MVCMD\_SSTP
  - ximc.h, 94
- MVCMD\_STOP
  - ximc.h, 94
- MVCMD\_UKNWN
  - ximc.h, 94
- MagneticBrakeInfo
  - accessories\_settings.t, 9
- Major
  - device\_information.t, 22
  - serial\_number.t, 50
- Manufacturer
  - encoder\_information.t, 25
  - gear\_information.t, 32
  - hallsensor\_information.t, 35
  - motor\_information.t, 41
  - stage\_information.t, 52
- MaxClickTime

- control\_settings\_calb\_t, 18
- control\_settings\_t, 19
- MaxCurrent
  - motor\_settings\_t, 42
- MaxCurrentConsumption
  - encoder\_settings\_t, 26
  - hallsensor\_settings\_t, 36
  - stage\_settings\_t, 54
- MaxCurrentTime
  - motor\_settings\_t, 42
- MaxOperatingFrequency
  - encoder\_settings\_t, 26
  - hallsensor\_settings\_t, 36
- MaxOutputBacklash
  - gear\_settings\_t, 33
- MaxSpeed
  - control\_settings\_calb\_t, 18
  - control\_settings\_t, 19
  - motor\_settings\_t, 43
  - stage\_settings\_t, 54
- MechanicalTimeConstant
  - motor\_settings\_t, 43
- MicrostepMode
  - engine\_settings\_calb\_t, 27
  - engine\_settings\_t, 29
- MinimumUusb
  - secure\_settings\_t, 49
- Minor
  - device\_information\_t, 22
  - serial\_number\_t, 50
- motor\_information\_t, 40
  - Manufacturer, 41
  - PartNumber, 41
- motor\_settings\_t, 41
  - DetentTorque, 42
  - MaxCurrent, 42
  - MaxCurrentTime, 42
  - MaxSpeed, 43
  - MechanicalTimeConstant, 43
  - MotorType, 43
  - NoLoadCurrent, 43
  - NoLoadSpeed, 43
  - NominalCurrent, 43
  - NominalPower, 43
  - NominalSpeed, 43
  - NominalTorque, 43
  - NominalVoltage, 44
  - Phases, 44
  - Poles, 44
  - RotorInertia, 44
  - SpeedConstant, 44
  - SpeedTorqueGradient, 44
  - StallTorque, 44
  - TorqueConstant, 44
  - WindingInductance, 44
  - WindingResistance, 45
- MotorType
  - motor\_settings\_t, 43
- move\_settings\_calb\_t, 45
  - Accel, 45
  - AntiplaySpeed, 45
  - Decel, 45
  - Speed, 45
- move\_settings\_t, 45
  - Accel, 46
  - AntiplaySpeed, 46
  - Decel, 46
  - Speed, 46
  - uAntiplaySpeed, 46
  - uSpeed, 46
- MoveSts
  - status\_calb\_t, 56
  - status\_t, 59
- msec\_sleep
  - ximc.h, 115
- MvCmdSts
  - status\_calb\_t, 56
  - status\_t, 59
- NoLoadCurrent
  - motor\_settings\_t, 43
- NoLoadSpeed
  - motor\_settings\_t, 43
- NomCurrent
  - engine\_settings\_calb\_t, 27
  - engine\_settings\_t, 29
- NomSpeed
  - engine\_settings\_calb\_t, 27
  - engine\_settings\_t, 29
- NomVoltage
  - engine\_settings\_calb\_t, 27
  - engine\_settings\_t, 29
- NominalCurrent
  - motor\_settings\_t, 43
- NominalPower
  - motor\_settings\_t, 43
- NominalSpeed
  - motor\_settings\_t, 43
- NominalTorque
  - motor\_settings\_t, 43
- NominalVoltage
  - motor\_settings\_t, 44
- open\_device
  - ximc.h, 115
- POWER\_OFF\_ENABLED
  - ximc.h, 94
- POWER\_REDUCT\_ENABLED
  - ximc.h, 94
- POWER\_SMOOTH\_CURRENT
  - ximc.h, 94
- PWR\_STATE\_MAX
  - ximc.h, 94
- PWR\_STATE\_NORM
  - ximc.h, 94
- PWR\_STATE\_OFF

- ximc.h, [94](#)
- PWR\_STATE\_REDUCT
  - ximc.h, [94](#)
- PWR\_STATE\_UNKNOWN
  - ximc.h, [95](#)
- PWRSts
  - status\_calb\_t, [56](#)
  - status\_t, [59](#)
- PartNumber
  - encoder\_information\_t, [25](#)
  - gear\_information\_t, [32](#)
  - hallsensor\_information\_t, [35](#)
  - motor\_information\_t, [41](#)
  - stage\_information\_t, [52](#)
- Phases
  - motor\_settings\_t, [44](#)
- pid\_settings\_t, [47](#)
- Poles
  - motor\_settings\_t, [44](#)
- PosFlags
  - set\_position\_calb\_t, [51](#)
  - set\_position\_t, [52](#)
- Position
  - command\_add\_sync\_in\_action\_calb\_t, [16](#)
  - get\_position\_calb\_t, [34](#)
  - set\_position\_calb\_t, [51](#)
  - sync\_in\_settings\_calb\_t, [60](#)
- PositionerName
  - stage\_name\_t, [53](#)
- Pot
  - analog\_data\_t, [13](#)
  - chart\_data\_t, [15](#)
- power\_settings\_t, [47](#)
  - CurrReductDelay, [48](#)
  - CurrentSetTime, [48](#)
  - HoldCurrent, [48](#)
  - PowerFlags, [48](#)
  - PowerOffDelay, [48](#)
- PowerFlags
  - power\_settings\_t, [48](#)
- PowerOffDelay
  - power\_settings\_t, [48](#)
- probe\_device
  - ximc.h, [115](#)
- R
  - analog\_data\_t, [13](#)
- REV\_SENS\_INV
  - ximc.h, [95](#)
- RatedInputSpeed
  - gear\_settings\_t, [33](#)
- RatedInputTorque
  - gear\_settings\_t, [33](#)
- ReductionIn
  - gear\_settings\_t, [34](#)
- ReductionOut
  - gear\_settings\_t, [34](#)
- Release
  - device\_information\_t, [22](#)
- serial\_number\_t, [50](#)
- RightBorder
  - edges\_settings\_calb\_t, [23](#)
  - edges\_settings\_t, [24](#)
- RotorInertia
  - motor\_settings\_t, [44](#)
- SN
  - serial\_number\_t, [50](#)
- STATE\_ALARM
  - ximc.h, [95](#)
- STATE\_BRAKE
  - ximc.h, [95](#)
- STATE\_BUTTON\_LEFT
  - ximc.h, [95](#)
- STATE\_BUTTON\_RIGHT
  - ximc.h, [95](#)
- STATE\_CONTR
  - ximc.h, [95](#)
- STATE\_CTP\_ERROR
  - ximc.h, [95](#)
- STATE\_DIG\_SIGNAL
  - ximc.h, [96](#)
- STATE\_ENC\_A
  - ximc.h, [96](#)
- STATE\_ENC\_B
  - ximc.h, [96](#)
- STATE\_ERRC
  - ximc.h, [96](#)
- STATE\_ERRD
  - ximc.h, [96](#)
- STATE\_ERRV
  - ximc.h, [96](#)
- STATE\_GPIO\_LEVEL
  - ximc.h, [96](#)
- STATE\_GPIO\_PINOUT
  - ximc.h, [96](#)
- STATE\_HALL\_A
  - ximc.h, [96](#)
- STATE\_HALL\_B
  - ximc.h, [96](#)
- STATE\_HALL\_C
  - ximc.h, [96](#)
- STATE\_LEFT\_EDGE
  - ximc.h, [97](#)
- STATE\_POWER\_OVERHEAT
  - ximc.h, [97](#)
- STATE\_REV\_SENSOR
  - ximc.h, [97](#)
- STATE\_RIGHT\_EDGE
  - ximc.h, [97](#)
- STATE\_SECUR
  - ximc.h, [97](#)
- STATE\_SYNC\_INPUT
  - ximc.h, [97](#)
- STATE\_SYNC\_OUTPUT
  - ximc.h, [97](#)
- SYNCIN\_ENABLED
  - ximc.h, [98](#)



SYNCIN\_GOTOPOSITION  
     ximc.h, 98  
 SYNCIN\_INVERT  
     ximc.h, 98  
 SYNCOUT\_ENABLED  
     ximc.h, 98  
 SYNCOUT\_IN\_STEPS  
     ximc.h, 98  
 SYNCOUT\_INVERT  
     ximc.h, 98  
 SYNCOUT\_ONPERIOD  
     ximc.h, 98  
 SYNCOUT\_ONSTART  
     ximc.h, 98  
 SYNCOUT\_ONSTOP  
     ximc.h, 98  
 SYNCOUT\_STATE  
     ximc.h, 98  
 secure\_settings.t, 48  
     Criticalpwr, 49  
     Criticalusb, 49  
     CriticalT, 49  
     CriticalUpwr, 49  
     CriticalUusb, 49  
     Flags, 49  
     LowUpwrOff, 49  
     MinimumUusb, 49  
 serial\_number.t, 50  
     Key, 50  
     Major, 50  
     Minor, 50  
     Release, 50  
     SN, 50  
 service\_command.updf  
     ximc.h, 116  
 set\_accessories\_settings  
     ximc.h, 116  
 set\_brake\_settings  
     ximc.h, 116  
 set\_control\_settings  
     ximc.h, 116  
 set\_controller\_name  
     ximc.h, 116  
 set\_ctp\_settings  
     ximc.h, 117  
 set\_debug\_write  
     ximc.h, 117  
 set\_edges\_settings  
     ximc.h, 117  
 set\_encoder\_information  
     ximc.h, 117  
 set\_encoder\_settings  
     ximc.h, 118  
 set\_engine\_settings  
     ximc.h, 118  
 set\_entype\_settings  
     ximc.h, 118  
 set\_extio\_settings  
     ximc.h, 118  
 set\_feedback\_settings  
     ximc.h, 119  
 set\_gear\_information  
     ximc.h, 119  
 set\_gear\_settings  
     ximc.h, 119  
 set\_hallsensor\_information  
     ximc.h, 119  
 set\_hallsensor\_settings  
     ximc.h, 119  
 set\_home\_settings  
     ximc.h, 120  
 set\_joystick\_settings  
     ximc.h, 120  
 set\_logging\_callback  
     ximc.h, 120  
 set\_motor\_information  
     ximc.h, 121  
 set\_motor\_settings  
     ximc.h, 121  
 set\_move\_settings  
     ximc.h, 121  
 set\_pid\_settings  
     ximc.h, 121  
 set\_position  
     ximc.h, 121  
 set\_position\_calb.t, 51  
     EncPosition, 51  
     PosFlags, 51  
     Position, 51  
 set\_position.t, 51  
     EncPosition, 52  
     PosFlags, 52  
 set\_power\_settings  
     ximc.h, 122  
 set\_secure\_settings  
     ximc.h, 122  
 set\_serial\_number  
     ximc.h, 122  
 set\_stage\_information  
     ximc.h, 122  
 set\_stage\_name  
     ximc.h, 123  
 set\_stage\_settings  
     ximc.h, 123  
 set\_sync\_in\_settings  
     ximc.h, 123  
 set\_sync\_out\_settings  
     ximc.h, 123  
 set\_uart\_settings  
     ximc.h, 124  
 SlowHome  
     home\_settings\_calb.t, 37  
     home\_settings.t, 38  
 Speed  
     move\_settings\_calb.t, 45  
     move\_settings.t, 46

- sync\_in\_settings\_calb.t, 60
  - sync\_in\_settings.t, 61
- SpeedConstant
  - motor\_settings.t, 44
- SpeedTorqueGradient
  - motor\_settings.t, 44
- stage\_information.t, 52
  - Manufacturer, 52
  - PartNumber, 52
- stage\_name.t, 52
  - PositionerName, 53
- stage\_settings.t, 53
  - HorizontalLoadCapacity, 54
  - LeadScrewPitch, 54
  - MaxCurrentConsumption, 54
  - MaxSpeed, 54
  - SupplyVoltageMax, 54
  - SupplyVoltageMin, 54
  - TravelRange, 54
  - Units, 54
  - VerticalLoadCapacity, 55
- StallTorque
  - motor\_settings.t, 44
- status\_calb.t, 55
  - CmdBufFreeSpace, 56
  - CurPosition, 56
  - CurSpeed, 56
  - CurT, 56
  - EncPosition, 56
  - EncSts, 56
  - Flags, 56
  - GPIOFlags, 56
  - Ipwr, 56
  - Iusb, 56
  - MoveSts, 56
  - MvCmdSts, 56
  - PWRSts, 56
  - Upwr, 57
  - Uusb, 57
  - WindSts, 57
- status.t, 57
  - CmdBufFreeSpace, 58
  - CurPosition, 58
  - CurSpeed, 58
  - CurT, 58
  - EncPosition, 58
  - EncSts, 58
  - Flags, 58
  - GPIOFlags, 58
  - Ipwr, 59
  - Iusb, 59
  - MoveSts, 59
  - MvCmdSts, 59
  - PWRSts, 59
  - uCurPosition, 59
  - uCurSpeed, 59
  - Upwr, 59
  - Uusb, 59
  - WindSts, 59
- StepsPerRev
  - engine\_settings\_calb.t, 28
  - engine\_settings.t, 29
- SupVoltage
  - analog\_data.t, 13
- SupVoltage\_ADC
  - analog\_data.t, 13
- SupplyVoltageMax
  - encoder\_settings.t, 26
  - hallsensor\_settings.t, 36
  - stage\_settings.t, 54
- SupplyVoltageMin
  - encoder\_settings.t, 26
  - hallsensor\_settings.t, 36
  - stage\_settings.t, 54
- sync\_in\_settings\_calb.t, 60
  - ClutterTime, 60
  - Position, 60
  - Speed, 60
  - SyncInFlags, 60
- sync\_in\_settings.t, 60
  - ClutterTime, 61
  - Speed, 61
  - SyncInFlags, 61
  - uPosition, 61
  - uSpeed, 61
- sync\_out\_settings\_calb.t, 61
  - Accuracy, 62
  - SyncOutFlags, 62
  - SyncOutPeriod, 62
  - SyncOutPulseSteps, 62
- sync\_out\_settings.t, 62
  - Accuracy, 63
  - SyncOutFlags, 63
  - SyncOutPeriod, 63
  - SyncOutPulseSteps, 63
  - uAccuracy, 63
- SyncInFlags
  - sync\_in\_settings\_calb.t, 60
  - sync\_in\_settings.t, 61
- SyncOutFlags
  - sync\_out\_settings\_calb.t, 62
  - sync\_out\_settings.t, 63
- SyncOutPeriod
  - sync\_out\_settings\_calb.t, 62
  - sync\_out\_settings.t, 63
- SyncOutPulseSteps
  - sync\_out\_settings\_calb.t, 62
  - sync\_out\_settings.t, 63
- t1
  - brake\_settings.t, 14
- t2
  - brake\_settings.t, 14
- t3
  - brake\_settings.t, 14
- t4
  - brake\_settings.t, 14

- TSGrad
  - accessories\_settings.t, 9
- TSMaX
  - accessories\_settings.t, 9
- TSMIn
  - accessories\_settings.t, 9
- TSSettings
  - accessories\_settings.t, 9
- Temp
  - analog\_data.t, 13
- Temp\_ADC
  - analog\_data.t, 13
- TemperatureSensorInfo
  - accessories\_settings.t, 9
- Time
  - command\_add\_sync\_in\_action.calb.t, 16
  - command\_add\_sync\_in\_action.t, 17
- Timeout
  - control\_settings.calb.t, 18
  - control\_settings.t, 19
- TorqueConstant
  - motor\_settings.t, 44
- TravelRange
  - stage\_settings.t, 54
- UART\_PARITY\_BITS
  - ximc.h, 98
- UARTSetupFlags
  - uart\_settings.t, 64
- uAccuracy
  - sync\_out\_settings.t, 63
- uAntiplaySpeed
  - move\_settings.t, 46
- uCurPosition
  - status.t, 59
- uCurSpeed
  - status.t, 59
- uDeltaPosition
  - control\_settings.t, 19
- uFastHome
  - home\_settings.t, 38
- uHomeDelta
  - home\_settings.t, 38
- uLeftBorder
  - edges\_settings.t, 25
- uMaxSpeed
  - control\_settings.t, 19
- uNomSpeed
  - engine\_settings.t, 29
- uPosition
  - command\_add\_sync\_in\_action.t, 17
  - sync\_in\_settings.t, 61
- uRightBorder
  - edges\_settings.t, 25
- uSlowHome
  - home\_settings.t, 38
- uSpeed
  - move\_settings.t, 46
  - sync\_in\_settings.t, 61
- uart\_settings.t, 63
  - UARTSetupFlags, 64
- Units
  - stage\_settings.t, 54
- Upwr
  - status\_calb.t, 57
  - status.t, 59
- Uusb
  - status\_calb.t, 57
  - status.t, 59
- VerticalLoadCapacity
  - stage\_settings.t, 55
- WIND\_A\_STATE\_ABSENT
  - ximc.h, 98
- WIND\_A\_STATE\_OK
  - ximc.h, 99
- WIND\_B\_STATE\_ABSENT
  - ximc.h, 99
- WIND\_B\_STATE\_OK
  - ximc.h, 99
- WindSts
  - status\_calb.t, 57
  - status.t, 59
- WindingCurrentA
  - chart\_data.t, 16
- WindingCurrentB
  - chart\_data.t, 16
- WindingCurrentC
  - chart\_data.t, 16
- WindingInductance
  - motor\_settings.t, 44
- WindingResistance
  - motor\_settings.t, 45
- WindingVoltageA
  - chart\_data.t, 16
- WindingVoltageB
  - chart\_data.t, 16
- WindingVoltageC
  - chart\_data.t, 16
- write\_key
  - ximc.h, 124
- XIMC\_API
  - ximc.h, 99
- ximc.h, 65
  - BORDER\_IS\_ENCODER, 85
  - BORDER\_STOP\_LEFT, 86
  - BORDER\_STOP\_RIGHT, 86
  - BRAKE\_ENABLED, 86
  - BRAKE\_ENG\_PWROFF, 86
  - CONTROL\_MODE\_BITS, 86
  - CONTROL\_MODE\_JOY, 86
  - CONTROL\_MODE\_LR, 86
  - CONTROL\_MODE\_OFF, 86
  - CTP\_ALARM\_ON\_ERROR, 86
  - CTP\_BASE, 87
  - CTP\_ENABLED, 87

close\_device, 100  
command\_add\_sync\_in\_action, 100  
command\_clear\_fram, 100  
command\_eeread\_settings, 100  
command\_eesave\_settings, 100  
command\_home, 100  
command\_left, 101  
command\_loft, 101  
command\_move, 101  
command\_movr, 101  
command\_power\_off, 102  
command\_read\_settings, 102  
command\_reset, 102  
command\_right, 102  
command\_save\_settings, 103  
command\_sstp, 103  
command\_stop, 103  
command\_update\_firmware, 103  
command\_zero, 103  
EEPROM.PRECEDENCE, 87  
ENC\_STATE\_ABSENT, 87  
ENC\_STATE\_MALFUNC, 87  
ENC\_STATE\_OK, 87  
ENC\_STATE\_REVERSES, 87  
ENC\_STATE\_UNKNOWN, 87  
ENDER\_SWAP, 88  
ENGINE\_ACCEL\_ON, 88  
ENGINE\_ANTIPLAY, 88  
ENGINE\_LIMIT\_CURR, 88  
ENGINE\_LIMIT\_RPM, 88  
ENGINE\_LIMIT\_VOLT, 88  
ENGINE\_MAX\_SPEED, 88  
ENGINE\_REVERSE, 88  
ENGINE\_TYPE\_2DC, 88  
ENGINE\_TYPE\_DC, 89  
ENGINE\_TYPE\_NONE, 89  
ENGINE\_TYPE\_STEP, 89  
ENGINE\_TYPE\_TEST, 89  
ENUMERATE\_PROBE, 89  
EXTIO\_SETUP\_INVERT, 89  
EXTIO\_SETUP\_OUTPUT, 90  
enumerate\_devices, 104  
FEEDBACK\_EMF, 90  
FEEDBACK\_ENCODER, 90  
FEEDBACK\_ENCODERHALL, 90  
FEEDBACK\_NONE, 91  
free\_enumerate\_devices, 104  
get\_accessories\_settings, 104  
get\_analog\_data, 104  
get\_bootloader\_version, 104  
get\_brake\_settings, 105  
get\_chart\_data, 105  
get\_control\_settings, 105  
get\_controller\_name, 105  
get\_ctp\_settings, 106  
get\_debug\_read, 106  
get\_device\_count, 106  
get\_device\_information, 106  
get\_device\_name, 106  
get\_edges\_settings, 107  
get\_encoder\_information, 107  
get\_encoder\_settings, 107  
get\_engine\_settings, 107  
get\_entype\_settings, 108  
get\_enumerate\_device\_information, 108  
get\_enumerate\_device\_serial, 108  
get\_extio\_settings, 108  
get\_feedback\_settings, 109  
get\_firmware\_version, 109  
get\_gear\_information, 109  
get\_gear\_settings, 109  
get\_hallsensor\_information, 109  
get\_hallsensor\_settings, 110  
get\_home\_settings, 110  
get\_init\_random, 110  
get\_joystick\_settings, 110  
get\_motor\_information, 111  
get\_motor\_settings, 111  
get\_move\_settings, 111  
get\_pid\_settings, 111  
get\_position, 112  
get\_power\_settings, 112  
get\_secure\_settings, 112  
get\_serial\_number, 112  
get\_stage\_information, 112  
get\_stage\_name, 113  
get\_stage\_settings, 113  
get\_status, 113  
get\_status\_calb, 113  
get\_sync\_in\_settings, 113  
get\_sync\_out\_settings, 114  
get\_uart\_settings, 114  
goto\_firmware, 114  
H\_BRIDGE\_ALERT, 91  
HOME\_DIR\_FIRST, 91  
HOME\_DIR\_SECOND, 91  
HOME\_HALF\_MV, 91  
HOME\_MV\_SEC\_EN, 91  
has\_firmware, 114  
JOY\_REVERSE, 92  
LOW\_UPWR\_PROTECTION, 92  
logging\_callback\_stderr\_narrow, 115  
logging\_callback\_stderr\_wide, 115  
logging\_callback\_t, 99  
MICROSTEP\_MODE\_FULL, 93  
MOVE\_STATE\_ANTIPLAY, 93  
MOVE\_STATE\_MOVING, 93  
MVCMD\_ERROR, 93  
MVCMD\_HOME, 93  
MVCMD\_LEFT, 93  
MVCMD\_LOFT, 93  
MVCMD\_MOVE, 93  
MVCMD\_MOVR, 93  
MVCMD\_NAME\_BITS, 93  
MVCMD\_RIGHT, 94  
MVCMD\_RUNNING, 94

MVCMD\_SSTP, [94](#)  
MVCMD\_STOP, [94](#)  
MVCMD\_UKNWN, [94](#)  
msec.sleep, [115](#)  
open\_device, [115](#)  
POWER\_OFF\_ENABLED, [94](#)  
PWR\_STATE\_MAX, [94](#)  
PWR\_STATE\_NORM, [94](#)  
PWR\_STATE\_OFF, [94](#)  
PWR\_STATE\_REDUCT, [94](#)  
PWR\_STATE\_UNKNOWN, [95](#)  
probe\_device, [115](#)  
REV\_SENS\_INV, [95](#)  
STATE\_ALARM, [95](#)  
STATE\_BRAKE, [95](#)  
STATE\_BUTTON\_LEFT, [95](#)  
STATE\_BUTTON\_RIGHT, [95](#)  
STATE\_CONTR, [95](#)  
STATE\_CTP\_ERROR, [95](#)  
STATE\_DIG\_SIGNAL, [96](#)  
STATE\_ENC\_A, [96](#)  
STATE\_ENC\_B, [96](#)  
STATE\_ERRC, [96](#)  
STATE\_ERRD, [96](#)  
STATE\_ERRV, [96](#)  
STATE\_GPIO\_LEVEL, [96](#)  
STATE\_GPIO\_PINOUT, [96](#)  
STATE\_HALL\_A, [96](#)  
STATE\_HALL\_B, [96](#)  
STATE\_HALL\_C, [96](#)  
STATE\_LEFT\_EDGE, [97](#)  
STATE\_REV\_SENSOR, [97](#)  
STATE\_RIGHT\_EDGE, [97](#)  
STATE\_SECUR, [97](#)  
STATE\_SYNC\_INPUT, [97](#)  
STATE\_SYNC\_OUTPUT, [97](#)  
SYNCIN\_ENABLED, [98](#)  
SYNCIN\_GOTOPOSITION, [98](#)  
SYNCIN\_INVERT, [98](#)  
SYNCOUT\_ENABLED, [98](#)  
SYNCOUT\_IN\_STEPS, [98](#)  
SYNCOUT\_INVERT, [98](#)  
SYNCOUT\_ONPERIOD, [98](#)  
SYNCOUT\_ONSTART, [98](#)  
SYNCOUT\_ONSTOP, [98](#)  
SYNCOUT\_STATE, [98](#)  
service\_command\_updf, [116](#)  
set\_accessories\_settings, [116](#)  
set\_brake\_settings, [116](#)  
set\_control\_settings, [116](#)  
set\_controller\_name, [116](#)  
set\_ctp\_settings, [117](#)  
set\_debug\_write, [117](#)  
set\_edges\_settings, [117](#)  
set\_encoder\_information, [117](#)  
set\_encoder\_settings, [118](#)  
set\_engine\_settings, [118](#)  
set\_entype\_settings, [118](#)  
set\_extio\_settings, [118](#)  
set\_feedback\_settings, [119](#)  
set\_gear\_information, [119](#)  
set\_gear\_settings, [119](#)  
set\_hallsensor\_information, [119](#)  
set\_hallsensor\_settings, [119](#)  
set\_home\_settings, [120](#)  
set\_joystick\_settings, [120](#)  
set\_logging\_callback, [120](#)  
set\_motor\_information, [121](#)  
set\_motor\_settings, [121](#)  
set\_move\_settings, [121](#)  
set\_pid\_settings, [121](#)  
set\_position, [121](#)  
set\_power\_settings, [122](#)  
set\_secure\_settings, [122](#)  
set\_serial\_number, [122](#)  
set\_stage\_information, [122](#)  
set\_stage\_name, [123](#)  
set\_stage\_settings, [123](#)  
set\_sync\_in\_settings, [123](#)  
set\_sync\_out\_settings, [123](#)  
set\_uart\_settings, [124](#)  
UART\_PARITY\_BITS, [98](#)  
WIND\_A\_STATE\_OK, [99](#)  
WIND\_B\_STATE\_OK, [99](#)  
write\_key, [124](#)  
XIMC\_API, [99](#)  
ximc\_fix\_usbser\_sys, [124](#)  
ximc\_version, [124](#)  
ximc\_fix\_usbser\_sys  
    ximc.h, [124](#)  
ximc\_version  
    ximc.h, [124](#)