

libximc

3.1.1

Generated by Doxygen 1.8.13

# Contents

|          |                                                                                          |          |
|----------|------------------------------------------------------------------------------------------|----------|
| <b>1</b> | <b>libximc library</b>                                                                   | <b>1</b> |
| 1.1      | What the controller does . . . . .                                                       | 1        |
| 1.2      | What can do libximc library . . . . .                                                    | 1        |
| 1.3      | Assistance . . . . .                                                                     | 2        |
| <b>2</b> | <b>Introduction</b>                                                                      | <b>3</b> |
| 2.1      | About library . . . . .                                                                  | 3        |
| 2.1.1    | Supported OS and environment requirements: . . . . .                                     | 3        |
| <b>3</b> | <b>How to use with...</b>                                                                | <b>4</b> |
| 3.1      | Generic logging facility . . . . .                                                       | 7        |
| 3.2      | Required permissions . . . . .                                                           | 7        |
| 3.3      | C-profiles . . . . .                                                                     | 7        |
| 3.4      | Python-profiles . . . . .                                                                | 7        |
| <b>4</b> | <b>Working with user units</b>                                                           | <b>8</b> |
| 4.1      | The structure of the conversion units calibration_t . . . . .                            | 8        |
| 4.2      | Alternative functions for working with user units and data structures for them . . . . . | 8        |
| 4.3      | Coordinate correction table for more accurate positioning . . . . .                      | 9        |

|          |                                         |           |
|----------|-----------------------------------------|-----------|
| <b>5</b> | <b>Data Structure Documentation</b>     | <b>10</b> |
| 5.1      | accessories_settings_t Struct Reference | 10        |
| 5.1.1    | Detailed Description                    | 11        |
| 5.1.2    | Field Documentation                     | 11        |
| 5.1.2.1  | LimitSwitchesSettings                   | 11        |
| 5.1.2.2  | MagneticBrakeInfo                       | 11        |
| 5.1.2.3  | MBRatedCurrent                          | 11        |
| 5.1.2.4  | MBRatedVoltage                          | 11        |
| 5.1.2.5  | MBSettings                              | 12        |
| 5.1.2.6  | MBTorque                                | 12        |
| 5.1.2.7  | TemperatureSensorInfo                   | 12        |
| 5.1.2.8  | TSGrad                                  | 12        |
| 5.1.2.9  | TSMax                                   | 12        |
| 5.1.2.10 | TSMin                                   | 12        |
| 5.1.2.11 | TSSettings                              | 13        |
| 5.2      | analog_data_t Struct Reference          | 13        |
| 5.2.1    | Detailed Description                    | 14        |
| 5.2.2    | Field Documentation                     | 14        |
| 5.2.2.1  | A1Voltage                               | 14        |
| 5.2.2.2  | A1Voltage_ADC                           | 15        |
| 5.2.2.3  | A2Voltage                               | 15        |
| 5.2.2.4  | A2Voltage_ADC                           | 15        |
| 5.2.2.5  | ACurrent                                | 15        |
| 5.2.2.6  | ACurrent_ADC                            | 15        |
| 5.2.2.7  | B1Voltage                               | 15        |
| 5.2.2.8  | B1Voltage_ADC                           | 16        |
| 5.2.2.9  | B2Voltage                               | 16        |
| 5.2.2.10 | B2Voltage_ADC                           | 16        |
| 5.2.2.11 | BCurrent                                | 16        |

|          |                                         |    |
|----------|-----------------------------------------|----|
| 5.2.2.12 | BCurrent_ADC                            | 16 |
| 5.2.2.13 | Enc_Check                               | 16 |
| 5.2.2.14 | Enc_Check_ADC                           | 17 |
| 5.2.2.15 | FullCurrent                             | 17 |
| 5.2.2.16 | FullCurrent_ADC                         | 17 |
| 5.2.2.17 | Joy                                     | 17 |
| 5.2.2.18 | Joy_ADC                                 | 17 |
| 5.2.2.19 | L                                       | 17 |
| 5.2.2.20 | Pot                                     | 18 |
| 5.2.2.21 | R                                       | 18 |
| 5.2.2.22 | SupVoltage                              | 18 |
| 5.2.2.23 | SupVoltage_ADC                          | 18 |
| 5.2.2.24 | Temp                                    | 18 |
| 5.2.2.25 | Temp_ADC                                | 18 |
| 5.3      | brake_settings_t Struct Reference       | 19 |
| 5.3.1    | Detailed Description                    | 19 |
| 5.3.2    | Field Documentation                     | 19 |
| 5.3.2.1  | BrakeFlags                              | 19 |
| 5.3.2.2  | t1                                      | 19 |
| 5.3.2.3  | t2                                      | 20 |
| 5.3.2.4  | t3                                      | 20 |
| 5.3.2.5  | t4                                      | 20 |
| 5.4      | calibration_settings_t Struct Reference | 20 |
| 5.4.1    | Detailed Description                    | 21 |
| 5.4.2    | Field Documentation                     | 21 |
| 5.4.2.1  | CSS1_A                                  | 21 |
| 5.4.2.2  | CSS1_B                                  | 21 |
| 5.4.2.3  | CSS2_A                                  | 21 |
| 5.4.2.4  | CSS2_B                                  | 21 |

|         |                                          |    |
|---------|------------------------------------------|----|
| 5.4.2.5 | FullCurrent_A                            | 22 |
| 5.4.2.6 | FullCurrent_B                            | 22 |
| 5.5     | calibration_t Struct Reference           | 22 |
| 5.5.1   | Detailed Description                     | 22 |
| 5.5.2   | Field Documentation                      | 23 |
| 5.5.2.1 | A                                        | 23 |
| 5.6     | chart_data_t Struct Reference            | 23 |
| 5.6.1   | Detailed Description                     | 24 |
| 5.6.2   | Field Documentation                      | 24 |
| 5.6.2.1 | AveragedPowerRatio                       | 24 |
| 5.6.2.2 | Joy                                      | 25 |
| 5.6.2.3 | Pot                                      | 25 |
| 5.6.2.4 | WindingCurrentA                          | 25 |
| 5.6.2.5 | WindingCurrentB                          | 25 |
| 5.6.2.6 | WindingCurrentC                          | 25 |
| 5.6.2.7 | WindingVoltageA                          | 25 |
| 5.6.2.8 | WindingVoltageB                          | 26 |
| 5.6.2.9 | WindingVoltageC                          | 26 |
| 5.7     | control_settings_calb_t Struct Reference | 26 |
| 5.7.1   | Detailed Description                     | 26 |
| 5.7.2   | Field Documentation                      | 27 |
| 5.7.2.1 | Flags                                    | 27 |
| 5.7.2.2 | MaxClickTime                             | 27 |
| 5.7.2.3 | MaxSpeed                                 | 27 |
| 5.7.2.4 | Timeout                                  | 27 |
| 5.8     | control_settings_t Struct Reference      | 27 |
| 5.8.1   | Detailed Description                     | 28 |
| 5.8.2   | Field Documentation                      | 28 |
| 5.8.2.1 | Flags                                    | 28 |

|          |                                       |    |
|----------|---------------------------------------|----|
| 5.8.2.2  | MaxClickTime                          | 29 |
| 5.8.2.3  | MaxSpeed                              | 29 |
| 5.8.2.4  | Timeout                               | 29 |
| 5.8.2.5  | uDeltaPosition                        | 29 |
| 5.8.2.6  | uMaxSpeed                             | 29 |
| 5.9      | controller_name_t Struct Reference    | 29 |
| 5.9.1    | Detailed Description                  | 30 |
| 5.9.2    | Field Documentation                   | 30 |
| 5.9.2.1  | ControllerName                        | 30 |
| 5.9.2.2  | CtrlFlags                             | 30 |
| 5.10     | ctp_settings_t Struct Reference       | 30 |
| 5.10.1   | Detailed Description                  | 31 |
| 5.10.2   | Field Documentation                   | 31 |
| 5.10.2.1 | CTPFlags                              | 31 |
| 5.10.2.2 | CTPMinError                           | 31 |
| 5.11     | debug_read_t Struct Reference         | 31 |
| 5.11.1   | Detailed Description                  | 32 |
| 5.11.2   | Field Documentation                   | 32 |
| 5.11.2.1 | DebugData                             | 32 |
| 5.12     | debug_write_t Struct Reference        | 32 |
| 5.12.1   | Detailed Description                  | 32 |
| 5.12.2   | Field Documentation                   | 33 |
| 5.12.2.1 | DebugData                             | 33 |
| 5.13     | device_information_t Struct Reference | 33 |
| 5.13.1   | Detailed Description                  | 33 |
| 5.13.2   | Field Documentation                   | 33 |
| 5.13.2.1 | Major                                 | 34 |
| 5.13.2.2 | Minor                                 | 34 |
| 5.13.2.3 | Release                               | 34 |

|          |                                               |    |
|----------|-----------------------------------------------|----|
| 5.14     | device_network_information_t Struct Reference | 34 |
| 5.14.1   | Detailed Description                          | 34 |
| 5.15     | edges_settings_calb_t Struct Reference        | 35 |
| 5.15.1   | Detailed Description                          | 35 |
| 5.15.2   | Field Documentation                           | 35 |
| 5.15.2.1 | BorderFlags                                   | 35 |
| 5.15.2.2 | EnderFlags                                    | 35 |
| 5.15.2.3 | LeftBorder                                    | 36 |
| 5.15.2.4 | RightBorder                                   | 36 |
| 5.16     | edges_settings_t Struct Reference             | 36 |
| 5.16.1   | Detailed Description                          | 36 |
| 5.16.2   | Field Documentation                           | 37 |
| 5.16.2.1 | BorderFlags                                   | 37 |
| 5.16.2.2 | EnderFlags                                    | 37 |
| 5.16.2.3 | LeftBorder                                    | 37 |
| 5.16.2.4 | RightBorder                                   | 37 |
| 5.16.2.5 | uLeftBorder                                   | 37 |
| 5.16.2.6 | uRightBorder                                  | 38 |
| 5.17     | emf_settings_t Struct Reference               | 38 |
| 5.17.1   | Detailed Description                          | 38 |
| 5.17.2   | Field Documentation                           | 38 |
| 5.17.2.1 | BackEMFFlags                                  | 39 |
| 5.17.2.2 | Km                                            | 39 |
| 5.17.2.3 | L                                             | 39 |
| 5.17.2.4 | R                                             | 39 |
| 5.18     | encoder_information_t Struct Reference        | 39 |
| 5.18.1   | Detailed Description                          | 40 |
| 5.18.2   | Field Documentation                           | 40 |
| 5.18.2.1 | Manufacturer                                  | 40 |

|          |                                          |    |
|----------|------------------------------------------|----|
| 5.18.2.2 | PartNumber                               | 40 |
| 5.19     | encoder_settings_t Struct Reference      | 40 |
| 5.19.1   | Detailed Description                     | 41 |
| 5.19.2   | Field Documentation                      | 41 |
| 5.19.2.1 | EncoderSettings                          | 41 |
| 5.19.2.2 | MaxCurrentConsumption                    | 41 |
| 5.19.2.3 | MaxOperatingFrequency                    | 41 |
| 5.19.2.4 | SupplyVoltageMax                         | 41 |
| 5.19.2.5 | SupplyVoltageMin                         | 42 |
| 5.20     | engine_advanced_setup_t Struct Reference | 42 |
| 5.20.1   | Detailed Description                     | 42 |
| 5.20.2   | Field Documentation                      | 42 |
| 5.20.2.1 | stepcloseloop_Kp_high                    | 42 |
| 5.20.2.2 | stepcloseloop_Kp_low                     | 43 |
| 5.20.2.3 | stepcloseloop_Kw                         | 43 |
| 5.21     | engine_settings_calb_t Struct Reference  | 43 |
| 5.21.1   | Detailed Description                     | 44 |
| 5.21.2   | Field Documentation                      | 44 |
| 5.21.2.1 | Antiplay                                 | 44 |
| 5.21.2.2 | EngineFlags                              | 44 |
| 5.21.2.3 | MicrostepMode                            | 44 |
| 5.21.2.4 | NomCurrent                               | 45 |
| 5.21.2.5 | NomSpeed                                 | 45 |
| 5.21.2.6 | NomVoltage                               | 45 |
| 5.21.2.7 | PeakCurrent                              | 45 |
| 5.21.2.8 | StepsPerRev                              | 45 |
| 5.22     | engine_settings_t Struct Reference       | 45 |
| 5.22.1   | Detailed Description                     | 46 |
| 5.22.2   | Field Documentation                      | 46 |

|          |                                      |    |
|----------|--------------------------------------|----|
| 5.22.2.1 | Antiplay                             | 46 |
| 5.22.2.2 | EngineFlags                          | 47 |
| 5.22.2.3 | MicrostepMode                        | 47 |
| 5.22.2.4 | NomCurrent                           | 47 |
| 5.22.2.5 | NomSpeed                             | 47 |
| 5.22.2.6 | NomVoltage                           | 47 |
| 5.22.2.7 | PeakCurrent                          | 47 |
| 5.22.2.8 | StepsPerRev                          | 48 |
| 5.22.2.9 | uNomSpeed                            | 48 |
| 5.23     | entype_settings_t Struct Reference   | 48 |
| 5.23.1   | Detailed Description                 | 48 |
| 5.23.2   | Field Documentation                  | 48 |
| 5.23.2.1 | DriverType                           | 49 |
| 5.23.2.2 | EngineType                           | 49 |
| 5.24     | extended_settings_t Struct Reference | 49 |
| 5.24.1   | Detailed Description                 | 49 |
| 5.25     | extio_settings_t Struct Reference    | 49 |
| 5.25.1   | Detailed Description                 | 50 |
| 5.25.2   | Field Documentation                  | 50 |
| 5.25.2.1 | EXTIOModeFlags                       | 50 |
| 5.25.2.2 | EXTIOSetupFlags                      | 50 |
| 5.26     | feedback_settings_t Struct Reference | 50 |
| 5.26.1   | Detailed Description                 | 51 |
| 5.26.2   | Field Documentation                  | 51 |
| 5.26.2.1 | CountsPerTurn                        | 51 |
| 5.26.2.2 | FeedbackFlags                        | 51 |
| 5.26.2.3 | FeedbackType                         | 51 |
| 5.26.2.4 | IPS                                  | 51 |
| 5.27     | gear_information_t Struct Reference  | 52 |

|          |                                               |    |
|----------|-----------------------------------------------|----|
| 5.27.1   | Detailed Description                          | 52 |
| 5.27.2   | Field Documentation                           | 52 |
| 5.27.2.1 | Manufacturer                                  | 52 |
| 5.27.2.2 | PartNumber                                    | 52 |
| 5.28     | gear_settings_t Struct Reference              | 52 |
| 5.28.1   | Detailed Description                          | 53 |
| 5.28.2   | Field Documentation                           | 53 |
| 5.28.2.1 | Efficiency                                    | 53 |
| 5.28.2.2 | InputInertia                                  | 53 |
| 5.28.2.3 | MaxOutputBacklash                             | 54 |
| 5.28.2.4 | RatedInputSpeed                               | 54 |
| 5.28.2.5 | RatedInputTorque                              | 54 |
| 5.28.2.6 | ReductionIn                                   | 54 |
| 5.28.2.7 | ReductionOut                                  | 54 |
| 5.29     | get_position_calb_t Struct Reference          | 54 |
| 5.29.1   | Detailed Description                          | 55 |
| 5.29.2   | Field Documentation                           | 55 |
| 5.29.2.1 | EncPosition                                   | 55 |
| 5.29.2.2 | Position                                      | 55 |
| 5.30     | get_position_t Struct Reference               | 55 |
| 5.30.1   | Detailed Description                          | 56 |
| 5.30.2   | Field Documentation                           | 56 |
| 5.30.2.1 | EncPosition                                   | 56 |
| 5.30.2.2 | uPosition                                     | 56 |
| 5.31     | globally_unique_identifier_t Struct Reference | 56 |
| 5.31.1   | Detailed Description                          | 57 |
| 5.31.2   | Field Documentation                           | 57 |
| 5.31.2.1 | UniqueID0                                     | 57 |
| 5.31.2.2 | UniqueID1                                     | 57 |

|          |                                           |    |
|----------|-------------------------------------------|----|
| 5.31.2.3 | UniqueID2                                 | 57 |
| 5.31.2.4 | UniqueID3                                 | 58 |
| 5.32     | hallsensor_information_t Struct Reference | 58 |
| 5.32.1   | Detailed Description                      | 58 |
| 5.32.2   | Field Documentation                       | 58 |
| 5.32.2.1 | Manufacturer                              | 58 |
| 5.32.2.2 | PartNumber                                | 58 |
| 5.33     | hallsensor_settings_t Struct Reference    | 59 |
| 5.33.1   | Detailed Description                      | 59 |
| 5.33.2   | Field Documentation                       | 59 |
| 5.33.2.1 | MaxCurrentConsumption                     | 59 |
| 5.33.2.2 | MaxOperatingFrequency                     | 59 |
| 5.33.2.3 | SupplyVoltageMax                          | 60 |
| 5.33.2.4 | SupplyVoltageMin                          | 60 |
| 5.34     | home_settings_calb_t Struct Reference     | 60 |
| 5.34.1   | Detailed Description                      | 60 |
| 5.34.2   | Field Documentation                       | 60 |
| 5.34.2.1 | FastHome                                  | 61 |
| 5.34.2.2 | HomeDelta                                 | 61 |
| 5.34.2.3 | HomeFlags                                 | 61 |
| 5.34.2.4 | SlowHome                                  | 61 |
| 5.35     | home_settings_t Struct Reference          | 61 |
| 5.35.1   | Detailed Description                      | 62 |
| 5.35.2   | Field Documentation                       | 62 |
| 5.35.2.1 | FastHome                                  | 62 |
| 5.35.2.2 | HomeDelta                                 | 62 |
| 5.35.2.3 | HomeFlags                                 | 62 |
| 5.35.2.4 | SlowHome                                  | 62 |
| 5.35.2.5 | uFastHome                                 | 63 |

|          |                                      |    |
|----------|--------------------------------------|----|
| 5.35.2.6 | uHomeDelta                           | 63 |
| 5.35.2.7 | uSlowHome                            | 63 |
| 5.36     | init_random_t Struct Reference       | 63 |
| 5.36.1   | Detailed Description                 | 63 |
| 5.36.2   | Field Documentation                  | 64 |
| 5.36.2.1 | key                                  | 64 |
| 5.37     | joystick_settings_t Struct Reference | 64 |
| 5.37.1   | Detailed Description                 | 64 |
| 5.37.2   | Field Documentation                  | 65 |
| 5.37.2.1 | DeadZone                             | 65 |
| 5.37.2.2 | ExpFactor                            | 65 |
| 5.37.2.3 | JoyCenter                            | 65 |
| 5.37.2.4 | JoyFlags                             | 65 |
| 5.37.2.5 | JoyHighEnd                           | 65 |
| 5.37.2.6 | JoyLowEnd                            | 66 |
| 5.38     | measurements_t Struct Reference      | 66 |
| 5.38.1   | Detailed Description                 | 66 |
| 5.38.2   | Field Documentation                  | 66 |
| 5.38.2.1 | Error                                | 66 |
| 5.38.2.2 | Length                               | 67 |
| 5.39     | motor_information_t Struct Reference | 67 |
| 5.39.1   | Detailed Description                 | 67 |
| 5.39.2   | Field Documentation                  | 67 |
| 5.39.2.1 | Manufacturer                         | 67 |
| 5.39.2.2 | PartNumber                           | 68 |
| 5.40     | motor_settings_t Struct Reference    | 68 |
| 5.40.1   | Detailed Description                 | 69 |
| 5.40.2   | Field Documentation                  | 69 |
| 5.40.2.1 | DetentTorque                         | 69 |

|           |                                       |    |
|-----------|---------------------------------------|----|
| 5.40.2.2  | MaxCurrent                            | 69 |
| 5.40.2.3  | MaxCurrentTime                        | 70 |
| 5.40.2.4  | MaxSpeed                              | 70 |
| 5.40.2.5  | MechanicalTimeConstant                | 70 |
| 5.40.2.6  | MotorType                             | 70 |
| 5.40.2.7  | NoLoadCurrent                         | 70 |
| 5.40.2.8  | NoLoadSpeed                           | 70 |
| 5.40.2.9  | NominalCurrent                        | 71 |
| 5.40.2.10 | NominalPower                          | 71 |
| 5.40.2.11 | NominalSpeed                          | 71 |
| 5.40.2.12 | NominalTorque                         | 71 |
| 5.40.2.13 | NominalVoltage                        | 71 |
| 5.40.2.14 | Phases                                | 71 |
| 5.40.2.15 | Poles                                 | 72 |
| 5.40.2.16 | RotorInertia                          | 72 |
| 5.40.2.17 | SpeedConstant                         | 72 |
| 5.40.2.18 | SpeedTorqueGradient                   | 72 |
| 5.40.2.19 | StallTorque                           | 72 |
| 5.40.2.20 | TorqueConstant                        | 72 |
| 5.40.2.21 | WindingInductance                     | 73 |
| 5.40.2.22 | WindingResistance                     | 73 |
| 5.41      | move_settings_calb_t Struct Reference | 73 |
| 5.41.1    | Detailed Description                  | 73 |
| 5.41.2    | Field Documentation                   | 74 |
| 5.41.2.1  | Accel                                 | 74 |
| 5.41.2.2  | AntiplaySpeed                         | 74 |
| 5.41.2.3  | Decel                                 | 74 |
| 5.41.2.4  | MoveFlags                             | 74 |
| 5.41.2.5  | Speed                                 | 75 |

|          |                                                       |    |
|----------|-------------------------------------------------------|----|
| 5.42     | <a href="#">move_settings_t Struct Reference</a>      | 75 |
| 5.42.1   | <a href="#">Detailed Description</a>                  | 75 |
| 5.42.2   | <a href="#">Field Documentation</a>                   | 75 |
| 5.42.2.1 | <a href="#">Accel</a>                                 | 76 |
| 5.42.2.2 | <a href="#">AntiplaySpeed</a>                         | 76 |
| 5.42.2.3 | <a href="#">Decel</a>                                 | 76 |
| 5.42.2.4 | <a href="#">MoveFlags</a>                             | 76 |
| 5.42.2.5 | <a href="#">Speed</a>                                 | 76 |
| 5.42.2.6 | <a href="#">uAntiplaySpeed</a>                        | 76 |
| 5.42.2.7 | <a href="#">uSpeed</a>                                | 77 |
| 5.43     | <a href="#">network_settings_t Struct Reference</a>   | 77 |
| 5.43.1   | <a href="#">Detailed Description</a>                  | 77 |
| 5.43.2   | <a href="#">Field Documentation</a>                   | 77 |
| 5.43.2.1 | <a href="#">DefaultGateway</a>                        | 77 |
| 5.43.2.2 | <a href="#">DHCPEnabled</a>                           | 78 |
| 5.43.2.3 | <a href="#">IPv4Address</a>                           | 78 |
| 5.43.2.4 | <a href="#">SubnetMask</a>                            | 78 |
| 5.44     | <a href="#">nonvolatile_memory_t Struct Reference</a> | 78 |
| 5.44.1   | <a href="#">Detailed Description</a>                  | 78 |
| 5.44.2   | <a href="#">Field Documentation</a>                   | 78 |
| 5.44.2.1 | <a href="#">UserData</a>                              | 79 |
| 5.45     | <a href="#">password_settings_t Struct Reference</a>  | 79 |
| 5.45.1   | <a href="#">Detailed Description</a>                  | 79 |
| 5.45.2   | <a href="#">Field Documentation</a>                   | 79 |
| 5.45.2.1 | <a href="#">UserPassword</a>                          | 79 |
| 5.46     | <a href="#">pid_settings_t Struct Reference</a>       | 79 |
| 5.46.1   | <a href="#">Detailed Description</a>                  | 80 |
| 5.47     | <a href="#">power_settings_t Struct Reference</a>     | 80 |
| 5.47.1   | <a href="#">Detailed Description</a>                  | 81 |

|          |                                      |    |
|----------|--------------------------------------|----|
| 5.47.2   | Field Documentation                  | 81 |
| 5.47.2.1 | CurrentSetTime                       | 81 |
| 5.47.2.2 | CurrReductDelay                      | 81 |
| 5.47.2.3 | HoldCurrent                          | 81 |
| 5.47.2.4 | PowerFlags                           | 81 |
| 5.47.2.5 | PowerOffDelay                        | 82 |
| 5.48     | secure_settings_t Struct Reference   | 82 |
| 5.48.1   | Detailed Description                 | 82 |
| 5.48.2   | Field Documentation                  | 82 |
| 5.48.2.1 | CriticalIpwr                         | 83 |
| 5.48.2.2 | CriticalUsb                          | 83 |
| 5.48.2.3 | CriticalT                            | 83 |
| 5.48.2.4 | CriticalUpwr                         | 83 |
| 5.48.2.5 | CriticalUsb                          | 83 |
| 5.48.2.6 | Flags                                | 83 |
| 5.48.2.7 | LowUpwrOff                           | 84 |
| 5.48.2.8 | MinimumUsb                           | 84 |
| 5.49     | serial_number_t Struct Reference     | 84 |
| 5.49.1   | Detailed Description                 | 84 |
| 5.49.2   | Field Documentation                  | 84 |
| 5.49.2.1 | Key                                  | 85 |
| 5.49.2.2 | Major                                | 85 |
| 5.49.2.3 | Minor                                | 85 |
| 5.49.2.4 | Release                              | 85 |
| 5.49.2.5 | SN                                   | 85 |
| 5.50     | set_position_calb_t Struct Reference | 85 |
| 5.50.1   | Detailed Description                 | 86 |
| 5.50.2   | Field Documentation                  | 86 |
| 5.50.2.1 | EncPosition                          | 86 |

|          |                                      |    |
|----------|--------------------------------------|----|
| 5.50.2.2 | PosFlags                             | 86 |
| 5.50.2.3 | Position                             | 86 |
| 5.51     | set_position_t Struct Reference      | 87 |
| 5.51.1   | Detailed Description                 | 87 |
| 5.51.2   | Field Documentation                  | 87 |
| 5.51.2.1 | EncPosition                          | 87 |
| 5.51.2.2 | PosFlags                             | 87 |
| 5.51.2.3 | uPosition                            | 88 |
| 5.52     | stage_information_t Struct Reference | 88 |
| 5.52.1   | Detailed Description                 | 88 |
| 5.52.2   | Field Documentation                  | 88 |
| 5.52.2.1 | Manufacturer                         | 88 |
| 5.52.2.2 | PartNumber                           | 89 |
| 5.53     | stage_name_t Struct Reference        | 89 |
| 5.53.1   | Detailed Description                 | 89 |
| 5.53.2   | Field Documentation                  | 89 |
| 5.53.2.1 | PositionerName                       | 89 |
| 5.54     | stage_settings_t Struct Reference    | 89 |
| 5.54.1   | Detailed Description                 | 90 |
| 5.54.2   | Field Documentation                  | 90 |
| 5.54.2.1 | HorizontalLoadCapacity               | 90 |
| 5.54.2.2 | LeadScrewPitch                       | 91 |
| 5.54.2.3 | MaxCurrentConsumption                | 91 |
| 5.54.2.4 | MaxSpeed                             | 91 |
| 5.54.2.5 | SupplyVoltageMax                     | 91 |
| 5.54.2.6 | SupplyVoltageMin                     | 91 |
| 5.54.2.7 | TravelRange                          | 91 |
| 5.54.2.8 | Units                                | 92 |
| 5.54.2.9 | VerticalLoadCapacity                 | 92 |

|           |                                |    |
|-----------|--------------------------------|----|
| 5.55      | status_calb_t Struct Reference | 92 |
| 5.55.1    | Detailed Description           | 93 |
| 5.55.2    | Field Documentation            | 93 |
| 5.55.2.1  | CmdBufFreeSpace                | 93 |
| 5.55.2.2  | CurPosition                    | 93 |
| 5.55.2.3  | CurSpeed                       | 93 |
| 5.55.2.4  | CurT                           | 93 |
| 5.55.2.5  | EncPosition                    | 94 |
| 5.55.2.6  | EncSts                         | 94 |
| 5.55.2.7  | Flags                          | 94 |
| 5.55.2.8  | GPIOFlags                      | 94 |
| 5.55.2.9  | Ipwr                           | 94 |
| 5.55.2.10 | Iusb                           | 94 |
| 5.55.2.11 | MoveSts                        | 95 |
| 5.55.2.12 | MvCmdSts                       | 95 |
| 5.55.2.13 | PWRSts                         | 95 |
| 5.55.2.14 | Upwr                           | 95 |
| 5.55.2.15 | Uusb                           | 95 |
| 5.55.2.16 | WindSts                        | 95 |
| 5.56      | status_t Struct Reference      | 96 |
| 5.56.1    | Detailed Description           | 96 |
| 5.56.2    | Field Documentation            | 97 |
| 5.56.2.1  | CmdBufFreeSpace                | 97 |
| 5.56.2.2  | CurPosition                    | 97 |
| 5.56.2.3  | CurSpeed                       | 97 |
| 5.56.2.4  | CurT                           | 97 |
| 5.56.2.5  | EncPosition                    | 97 |
| 5.56.2.6  | EncSts                         | 98 |
| 5.56.2.7  | Flags                          | 98 |

|           |                                           |     |
|-----------|-------------------------------------------|-----|
| 5.56.2.8  | GPIOFlags                                 | 98  |
| 5.56.2.9  | Ipwr                                      | 98  |
| 5.56.2.10 | Iusb                                      | 98  |
| 5.56.2.11 | MoveSts                                   | 98  |
| 5.56.2.12 | MvCmdSts                                  | 99  |
| 5.56.2.13 | PWRSts                                    | 99  |
| 5.56.2.14 | uCurPosition                              | 99  |
| 5.56.2.15 | uCurSpeed                                 | 99  |
| 5.56.2.16 | Upwr                                      | 99  |
| 5.56.2.17 | Uusb                                      | 99  |
| 5.56.2.18 | WindSts                                   | 100 |
| 5.57      | sync_in_settings_calb_t Struct Reference  | 100 |
| 5.57.1    | Detailed Description                      | 100 |
| 5.57.2    | Field Documentation                       | 100 |
| 5.57.2.1  | ClutterTime                               | 100 |
| 5.57.2.2  | Position                                  | 101 |
| 5.57.2.3  | Speed                                     | 101 |
| 5.57.2.4  | SyncInFlags                               | 101 |
| 5.58      | sync_in_settings_t Struct Reference       | 101 |
| 5.58.1    | Detailed Description                      | 102 |
| 5.58.2    | Field Documentation                       | 102 |
| 5.58.2.1  | ClutterTime                               | 102 |
| 5.58.2.2  | Speed                                     | 102 |
| 5.58.2.3  | SyncInFlags                               | 102 |
| 5.58.2.4  | uPosition                                 | 102 |
| 5.58.2.5  | uSpeed                                    | 103 |
| 5.59      | sync_out_settings_calb_t Struct Reference | 103 |
| 5.59.1    | Detailed Description                      | 103 |
| 5.59.2    | Field Documentation                       | 103 |

|          |                                      |            |
|----------|--------------------------------------|------------|
| 5.59.2.1 | Accuracy                             | 104        |
| 5.59.2.2 | SyncOutFlags                         | 104        |
| 5.59.2.3 | SyncOutPeriod                        | 104        |
| 5.59.2.4 | SyncOutPulseSteps                    | 104        |
| 5.60     | sync_out_settings_t Struct Reference | 104        |
| 5.60.1   | Detailed Description                 | 105        |
| 5.60.2   | Field Documentation                  | 105        |
| 5.60.2.1 | Accuracy                             | 105        |
| 5.60.2.2 | SyncOutFlags                         | 105        |
| 5.60.2.3 | SyncOutPeriod                        | 105        |
| 5.60.2.4 | SyncOutPulseSteps                    | 105        |
| 5.60.2.5 | uAccuracy                            | 106        |
| 5.61     | uart_settings_t Struct Reference     | 106        |
| 5.61.1   | Detailed Description                 | 106        |
| 5.61.2   | Field Documentation                  | 106        |
| 5.61.2.1 | UARTSetupFlags                       | 106        |
| <b>6</b> | <b>File Documentation</b>            | <b>107</b> |
| 6.1      | ximc.h File Reference                | 107        |
| 6.1.1    | Detailed Description                 | 132        |
| 6.1.2    | Macro Definition Documentation       | 132        |
| 6.1.2.1  | ALARM_FLAGS_STICKING                 | 132        |
| 6.1.2.2  | ALARM_ON_BORDERS_SWAP_MISSET         | 132        |
| 6.1.2.3  | ALARM_ON_DRIVER_OVERHEATING          | 133        |
| 6.1.2.4  | BACK_EMF_INDUCTANCE_AUTO             | 133        |
| 6.1.2.5  | BACK_EMF_KM_AUTO                     | 133        |
| 6.1.2.6  | BACK_EMF_RESISTANCE_AUTO             | 133        |
| 6.1.2.7  | BORDER_IS_ENCODER                    | 133        |
| 6.1.2.8  | BORDER_STOP_LEFT                     | 133        |
| 6.1.2.9  | BORDER_STOP_RIGHT                    | 134        |

|          |                                          |     |
|----------|------------------------------------------|-----|
| 6.1.2.10 | BORDERS_SWAP_MISSET_DETECTION . . . . .  | 134 |
| 6.1.2.11 | BRAKE_ENABLED . . . . .                  | 134 |
| 6.1.2.12 | BRAKE_ENG_PWROFF . . . . .               | 134 |
| 6.1.2.13 | BRAKING_OVERVOLTAGE_PROTECTION . . . . . | 134 |
| 6.1.2.14 | CONTROL_BTN_LEFT_PUSHED_OPEN . . . . .   | 134 |
| 6.1.2.15 | CONTROL_BTN_RIGHT_PUSHED_OPEN . . . . .  | 135 |
| 6.1.2.16 | CONTROL_MODE_BITS . . . . .              | 135 |
| 6.1.2.17 | CONTROL_MODE_JOY . . . . .               | 135 |
| 6.1.2.18 | CONTROL_MODE_LR . . . . .                | 135 |
| 6.1.2.19 | CONTROL_MODE_OFF . . . . .               | 135 |
| 6.1.2.20 | CTP_ALARM_ON_ERROR . . . . .             | 135 |
| 6.1.2.21 | CTP_BASE . . . . .                       | 136 |
| 6.1.2.22 | CTP_ENABLED . . . . .                    | 136 |
| 6.1.2.23 | CTP_ERROR_CORRECTION . . . . .           | 136 |
| 6.1.2.24 | DRIVER_TYPE_EXTERNAL . . . . .           | 136 |
| 6.1.2.25 | DRIVER_TYPE_INTEGRATE . . . . .          | 136 |
| 6.1.2.26 | EEPROM_PRECEDENCE . . . . .              | 136 |
| 6.1.2.27 | ENC_STATE_ABSENT . . . . .               | 137 |
| 6.1.2.28 | ENC_STATE_MALFUNC . . . . .              | 137 |
| 6.1.2.29 | ENC_STATE_OK . . . . .                   | 137 |
| 6.1.2.30 | ENC_STATE_REVERS . . . . .               | 137 |
| 6.1.2.31 | ENC_STATE_UNKNOWN . . . . .              | 137 |
| 6.1.2.32 | ENDER_SW1_ACTIVE_LOW . . . . .           | 137 |
| 6.1.2.33 | ENDER_SW2_ACTIVE_LOW . . . . .           | 138 |
| 6.1.2.34 | ENDER_SWAP . . . . .                     | 138 |
| 6.1.2.35 | ENGINE_ACCEL_ON . . . . .                | 138 |
| 6.1.2.36 | ENGINE_ANTIPLAY . . . . .                | 138 |
| 6.1.2.37 | ENGINE_CURRENT_AS_RMS . . . . .          | 138 |
| 6.1.2.38 | ENGINE_LIMIT_CURR . . . . .              | 138 |

|          |                               |     |
|----------|-------------------------------|-----|
| 6.1.2.39 | ENGINE_LIMIT_RPM              | 139 |
| 6.1.2.40 | ENGINE_LIMIT_VOLT             | 139 |
| 6.1.2.41 | ENGINE_MAX_SPEED              | 139 |
| 6.1.2.42 | ENGINE_REVERSE                | 139 |
| 6.1.2.43 | ENGINE_TYPE_2DC               | 139 |
| 6.1.2.44 | ENGINE_TYPE_BRUSHLESS         | 139 |
| 6.1.2.45 | ENGINE_TYPE_DC                | 140 |
| 6.1.2.46 | ENGINE_TYPE_NONE              | 140 |
| 6.1.2.47 | ENGINE_TYPE_STEP              | 140 |
| 6.1.2.48 | ENGINE_TYPE_TEST              | 140 |
| 6.1.2.49 | ENGINE_USE_PEAK_CURRENT       | 140 |
| 6.1.2.50 | ENUMERATE_PROBE               | 140 |
| 6.1.2.51 | EXTIO_SETUP_INVERT            | 141 |
| 6.1.2.52 | EXTIO_SETUP_MODE_IN_ALARM     | 141 |
| 6.1.2.53 | EXTIO_SETUP_MODE_IN_BITS      | 141 |
| 6.1.2.54 | EXTIO_SETUP_MODE_IN_HOME      | 141 |
| 6.1.2.55 | EXTIO_SETUP_MODE_IN_MOVR      | 141 |
| 6.1.2.56 | EXTIO_SETUP_MODE_IN_NOP       | 141 |
| 6.1.2.57 | EXTIO_SETUP_MODE_IN_PWOF      | 142 |
| 6.1.2.58 | EXTIO_SETUP_MODE_IN_STOP      | 142 |
| 6.1.2.59 | EXTIO_SETUP_MODE_OUT_ALARM    | 142 |
| 6.1.2.60 | EXTIO_SETUP_MODE_OUT_BITS     | 142 |
| 6.1.2.61 | EXTIO_SETUP_MODE_OUT_MOTOR_ON | 142 |
| 6.1.2.62 | EXTIO_SETUP_MODE_OUT_MOVING   | 142 |
| 6.1.2.63 | EXTIO_SETUP_MODE_OUT_OFF      | 143 |
| 6.1.2.64 | EXTIO_SETUP_MODE_OUT_ON       | 143 |
| 6.1.2.65 | EXTIO_SETUP_OUTPUT            | 143 |
| 6.1.2.66 | FEEDBACK_EMF                  | 143 |
| 6.1.2.67 | FEEDBACK_ENC_ADAPTIVE_HOLDING | 143 |

|          |                                |     |
|----------|--------------------------------|-----|
| 6.1.2.68 | FEEDBACK_ENC_FILTER_BITS       | 143 |
| 6.1.2.69 | FEEDBACK_ENC_FILTER_MEDIUM     | 144 |
| 6.1.2.70 | FEEDBACK_ENC_FILTER_NONE       | 144 |
| 6.1.2.71 | FEEDBACK_ENC_FILTER_STRONG     | 144 |
| 6.1.2.72 | FEEDBACK_ENC_FILTER_WEAK       | 144 |
| 6.1.2.73 | FEEDBACK_ENC_REVERSE           | 144 |
| 6.1.2.74 | FEEDBACK_ENC_TYPE_AUTO         | 144 |
| 6.1.2.75 | FEEDBACK_ENC_TYPE_BITS         | 145 |
| 6.1.2.76 | FEEDBACK_ENC_TYPE_DIFFERENTIAL | 145 |
| 6.1.2.77 | FEEDBACK_ENC_TYPE_SINGLE_ENDED | 145 |
| 6.1.2.78 | FEEDBACK_ENCODER               | 145 |
| 6.1.2.79 | FEEDBACK_ENCODER_MEDIATED      | 145 |
| 6.1.2.80 | FEEDBACK_NONE                  | 145 |
| 6.1.2.81 | H_BRIDGE_ALERT                 | 146 |
| 6.1.2.82 | HOME_DIR_FIRST                 | 146 |
| 6.1.2.83 | HOME_DIR_SECOND                | 146 |
| 6.1.2.84 | HOME_HALF_MV                   | 146 |
| 6.1.2.85 | HOME_MV_SEC_EN                 | 146 |
| 6.1.2.86 | HOME_STOP_FIRST_BITS           | 146 |
| 6.1.2.87 | HOME_STOP_FIRST_LIM            | 147 |
| 6.1.2.88 | HOME_STOP_FIRST_REV            | 147 |
| 6.1.2.89 | HOME_STOP_FIRST_SYN            | 147 |
| 6.1.2.90 | HOME_STOP_SECOND_BITS          | 147 |
| 6.1.2.91 | HOME_STOP_SECOND_LIM           | 147 |
| 6.1.2.92 | HOME_STOP_SECOND_REV           | 147 |
| 6.1.2.93 | HOME_STOP_SECOND_SYN           | 148 |
| 6.1.2.94 | HOME_USE_FAST                  | 148 |
| 6.1.2.95 | JOY_REVERSE                    | 148 |
| 6.1.2.96 | LOW_UPWR_PROTECTION            | 148 |

|           |                         |     |
|-----------|-------------------------|-----|
| 6.1.2.97  | MICROSTEP_MODE_FRAC_128 | 148 |
| 6.1.2.98  | MICROSTEP_MODE_FRAC_16  | 148 |
| 6.1.2.99  | MICROSTEP_MODE_FRAC_2   | 149 |
| 6.1.2.100 | MICROSTEP_MODE_FRAC_256 | 149 |
| 6.1.2.101 | MICROSTEP_MODE_FRAC_32  | 149 |
| 6.1.2.102 | MICROSTEP_MODE_FRAC_4   | 149 |
| 6.1.2.103 | MICROSTEP_MODE_FRAC_64  | 149 |
| 6.1.2.104 | MICROSTEP_MODE_FRAC_8   | 149 |
| 6.1.2.105 | MICROSTEP_MODE_FULL     | 150 |
| 6.1.2.106 | MOVE_STATE_ANTIPLAY     | 150 |
| 6.1.2.107 | MOVE_STATE_MOVING       | 150 |
| 6.1.2.108 | MOVE_STATE_TARGET_SPEED | 150 |
| 6.1.2.109 | MVCMD_ERROR             | 150 |
| 6.1.2.110 | MVCMD_HOME              | 150 |
| 6.1.2.111 | MVCMD_LEFT              | 151 |
| 6.1.2.112 | MVCMD_LOFT              | 151 |
| 6.1.2.113 | MVCMD_MOVE              | 151 |
| 6.1.2.114 | MVCMD_MOVR              | 151 |
| 6.1.2.115 | MVCMD_NAME_BITS         | 151 |
| 6.1.2.116 | MVCMD_RIGHT             | 151 |
| 6.1.2.117 | MVCMD_RUNNING           | 152 |
| 6.1.2.118 | MVCMD_SSTP              | 152 |
| 6.1.2.119 | MVCMD_STOP              | 152 |
| 6.1.2.120 | MVCMD_UKNWN             | 152 |
| 6.1.2.121 | POWER_OFF_ENABLED       | 152 |
| 6.1.2.122 | POWER_REDUCT_ENABLED    | 152 |
| 6.1.2.123 | POWER_SMOOTH_CURRENT    | 153 |
| 6.1.2.124 | PWR_STATE_MAX           | 153 |
| 6.1.2.125 | PWR_STATE_NORM          | 153 |

|                                                 |     |
|-------------------------------------------------|-----|
| 6.1.2.126 PWR_STATE_OFF . . . . .               | 153 |
| 6.1.2.127 PWR_STATE_REDUCT . . . . .            | 153 |
| 6.1.2.128 PWR_STATE_UNKNOWN . . . . .           | 153 |
| 6.1.2.129 REV_SENS_INV . . . . .                | 154 |
| 6.1.2.130 RPM_DIV_1000 . . . . .                | 154 |
| 6.1.2.131 SETPOS_IGNORE_ENCODER . . . . .       | 154 |
| 6.1.2.132 SETPOS_IGNORE_POSITION . . . . .      | 154 |
| 6.1.2.133 STATE_ALARM . . . . .                 | 154 |
| 6.1.2.134 STATE_BORDERS_SWAP_MISSET . . . . .   | 154 |
| 6.1.2.135 STATE_BRAKE . . . . .                 | 155 |
| 6.1.2.136 STATE_BUTTON_LEFT . . . . .           | 155 |
| 6.1.2.137 STATE_BUTTON_RIGHT . . . . .          | 155 |
| 6.1.2.138 STATE_CONTR . . . . .                 | 155 |
| 6.1.2.139 STATE_CONTROLLER_OVERHEAT . . . . .   | 155 |
| 6.1.2.140 STATE_CTP_ERROR . . . . .             | 155 |
| 6.1.2.141 STATE_DIG_SIGNAL . . . . .            | 156 |
| 6.1.2.142 STATE_EEPROM_CONNECTED . . . . .      | 156 |
| 6.1.2.143 STATE_ENC_A . . . . .                 | 156 |
| 6.1.2.144 STATE_ENC_B . . . . .                 | 156 |
| 6.1.2.145 STATE_ENGINE_RESPONSE_ERROR . . . . . | 156 |
| 6.1.2.146 STATE_ERRC . . . . .                  | 156 |
| 6.1.2.147 STATE_ERRD . . . . .                  | 157 |
| 6.1.2.148 STATE_ERRV . . . . .                  | 157 |
| 6.1.2.149 STATE_EXTIO_ALARM . . . . .           | 157 |
| 6.1.2.150 STATE_GPIO_LEVEL . . . . .            | 157 |
| 6.1.2.151 STATE_GPIO_PINOUT . . . . .           | 157 |
| 6.1.2.152 STATE_IS_HOMED . . . . .              | 157 |
| 6.1.2.153 STATE_LEFT_EDGE . . . . .             | 158 |
| 6.1.2.154 STATE_LOW_USB_VOLTAGE . . . . .       | 158 |

|                                                  |     |
|--------------------------------------------------|-----|
| 6.1.2.155 STATE_OVERLOAD_POWER_CURRENT . . . . . | 158 |
| 6.1.2.156 STATE_OVERLOAD_POWER_VOLTAGE . . . . . | 158 |
| 6.1.2.157 STATE_OVERLOAD_USB_CURRENT . . . . .   | 158 |
| 6.1.2.158 STATE_OVERLOAD_USB_VOLTAGE . . . . .   | 158 |
| 6.1.2.159 STATE_POWER_OVERHEAT . . . . .         | 159 |
| 6.1.2.160 STATE_REV_SENSOR . . . . .             | 159 |
| 6.1.2.161 STATE_RIGHT_EDGE . . . . .             | 159 |
| 6.1.2.162 STATE_SECUR . . . . .                  | 159 |
| 6.1.2.163 STATE_SYNC_INPUT . . . . .             | 159 |
| 6.1.2.164 STATE_SYNC_OUTPUT . . . . .            | 159 |
| 6.1.2.165 STATE_WINDING_RES_MISMATCH . . . . .   | 160 |
| 6.1.2.166 SYNCIN_ENABLED . . . . .               | 160 |
| 6.1.2.167 SYNCIN_GOTOPOSITION . . . . .          | 160 |
| 6.1.2.168 SYNCIN_INVERT . . . . .                | 160 |
| 6.1.2.169 SYNCOUT_ENABLED . . . . .              | 160 |
| 6.1.2.170 SYNCOUT_IN_STEPS . . . . .             | 160 |
| 6.1.2.171 SYNCOUT_INVERT . . . . .               | 161 |
| 6.1.2.172 SYNCOUT_ONPERIOD . . . . .             | 161 |
| 6.1.2.173 SYNCOUT_ONSTART . . . . .              | 161 |
| 6.1.2.174 SYNCOUT_ONSTOP . . . . .               | 161 |
| 6.1.2.175 SYNCOUT_STATE . . . . .                | 161 |
| 6.1.2.176 UART_PARITY_BITS . . . . .             | 161 |
| 6.1.2.177 WIND_A_STATE_ABSENT . . . . .          | 162 |
| 6.1.2.178 WIND_A_STATE_MALFUNC . . . . .         | 162 |
| 6.1.2.179 WIND_A_STATE_OK . . . . .              | 162 |
| 6.1.2.180 WIND_A_STATE_UNKNOWN . . . . .         | 162 |
| 6.1.2.181 WIND_B_STATE_ABSENT . . . . .          | 162 |
| 6.1.2.182 WIND_B_STATE_MALFUNC . . . . .         | 162 |
| 6.1.2.183 WIND_B_STATE_OK . . . . .              | 163 |

|           |                                |     |
|-----------|--------------------------------|-----|
| 6.1.2.184 | WIND_B_STATE_UNKNOWN           | 163 |
| 6.1.2.185 | XIMC_API                       | 163 |
| 6.1.3     | Typedef Documentation          | 163 |
| 6.1.3.1   | calibration_t                  | 163 |
| 6.1.3.2   | logging_callback_t             | 164 |
| 6.1.4     | Function Documentation         | 164 |
| 6.1.4.1   | close_device()                 | 164 |
| 6.1.4.2   | command_clear_fram()           | 165 |
| 6.1.4.3   | command_eeread_settings()      | 165 |
| 6.1.4.4   | command_eesave_settings()      | 165 |
| 6.1.4.5   | command_home()                 | 166 |
| 6.1.4.6   | command_homezero()             | 166 |
| 6.1.4.7   | command_left()                 | 167 |
| 6.1.4.8   | command_loft()                 | 167 |
| 6.1.4.9   | command_move()                 | 167 |
| 6.1.4.10  | command_move_calb()            | 168 |
| 6.1.4.11  | command_movr()                 | 168 |
| 6.1.4.12  | command_movr_calb()            | 169 |
| 6.1.4.13  | command_power_off()            | 169 |
| 6.1.4.14  | command_read_robust_settings() | 170 |
| 6.1.4.15  | command_read_settings()        | 170 |
| 6.1.4.16  | command_reset()                | 170 |
| 6.1.4.17  | command_right()                | 170 |
| 6.1.4.18  | command_save_robust_settings() | 171 |
| 6.1.4.19  | command_save_settings()        | 171 |
| 6.1.4.20  | command_sstp()                 | 171 |
| 6.1.4.21  | command_start_measurements()   | 172 |
| 6.1.4.22  | command_stop()                 | 172 |
| 6.1.4.23  | command_update_firmware()      | 172 |

|          |                                                     |     |
|----------|-----------------------------------------------------|-----|
| 6.1.4.24 | <code>command_wait_for_stop()</code>                | 173 |
| 6.1.4.25 | <code>command_zero()</code>                         | 173 |
| 6.1.4.26 | <code>enumerate_devices()</code>                    | 174 |
| 6.1.4.27 | <code>free_enumerate_devices()</code>               | 176 |
| 6.1.4.28 | <code>get_accessories_settings()</code>             | 177 |
| 6.1.4.29 | <code>get_analog_data()</code>                      | 177 |
| 6.1.4.30 | <code>get_bootloader_version()</code>               | 177 |
| 6.1.4.31 | <code>get_brake_settings()</code>                   | 178 |
| 6.1.4.32 | <code>get_calibration_settings()</code>             | 178 |
| 6.1.4.33 | <code>get_chart_data()</code>                       | 179 |
| 6.1.4.34 | <code>get_control_settings()</code>                 | 179 |
| 6.1.4.35 | <code>get_control_settings_calb()</code>            | 179 |
| 6.1.4.36 | <code>get_controller_name()</code>                  | 180 |
| 6.1.4.37 | <code>get_ctp_settings()</code>                     | 180 |
| 6.1.4.38 | <code>get_debug_read()</code>                       | 181 |
| 6.1.4.39 | <code>get_device_count()</code>                     | 181 |
| 6.1.4.40 | <code>get_device_information()</code>               | 181 |
| 6.1.4.41 | <code>get_device_name()</code>                      | 182 |
| 6.1.4.42 | <code>get_edges_settings()</code>                   | 182 |
| 6.1.4.43 | <code>get_edges_settings_calb()</code>              | 183 |
| 6.1.4.44 | <code>get_emf_settings()</code>                     | 183 |
| 6.1.4.45 | <code>get_encoder_information()</code>              | 184 |
| 6.1.4.46 | <code>get_encoder_settings()</code>                 | 184 |
| 6.1.4.47 | <code>get_engine_advanced_setup()</code>            | 184 |
| 6.1.4.48 | <code>get_engine_settings()</code>                  | 185 |
| 6.1.4.49 | <code>get_engine_settings_calb()</code>             | 185 |
| 6.1.4.50 | <code>get_entype_settings()</code>                  | 186 |
| 6.1.4.51 | <code>get_enumerate_device_controller_name()</code> | 186 |
| 6.1.4.52 | <code>get_enumerate_device_information()</code>     | 186 |

|          |                                                            |     |
|----------|------------------------------------------------------------|-----|
| 6.1.4.53 | <a href="#">get_enumerate_device_network_information()</a> | 187 |
| 6.1.4.54 | <a href="#">get_enumerate_device_serial()</a>              | 187 |
| 6.1.4.55 | <a href="#">get_enumerate_device_stage_name()</a>          | 188 |
| 6.1.4.56 | <a href="#">get_extended_settings()</a>                    | 188 |
| 6.1.4.57 | <a href="#">get_extio_settings()</a>                       | 188 |
| 6.1.4.58 | <a href="#">get_feedback_settings()</a>                    | 189 |
| 6.1.4.59 | <a href="#">get_firmware_version()</a>                     | 189 |
| 6.1.4.60 | <a href="#">get_gear_information()</a>                     | 190 |
| 6.1.4.61 | <a href="#">get_gear_settings()</a>                        | 190 |
| 6.1.4.62 | <a href="#">get_globally_unique_identifier()</a>           | 190 |
| 6.1.4.63 | <a href="#">get_hallsensor_information()</a>               | 191 |
| 6.1.4.64 | <a href="#">get_hallsensor_settings()</a>                  | 191 |
| 6.1.4.65 | <a href="#">get_home_settings()</a>                        | 192 |
| 6.1.4.66 | <a href="#">get_home_settings_calb()</a>                   | 192 |
| 6.1.4.67 | <a href="#">get_init_random()</a>                          | 192 |
| 6.1.4.68 | <a href="#">get_joystick_settings()</a>                    | 193 |
| 6.1.4.69 | <a href="#">get_measurements()</a>                         | 193 |
| 6.1.4.70 | <a href="#">get_motor_information()</a>                    | 194 |
| 6.1.4.71 | <a href="#">get_motor_settings()</a>                       | 194 |
| 6.1.4.72 | <a href="#">get_move_settings()</a>                        | 194 |
| 6.1.4.73 | <a href="#">get_move_settings_calb()</a>                   | 195 |
| 6.1.4.74 | <a href="#">get_network_settings()</a>                     | 195 |
| 6.1.4.75 | <a href="#">get_nonvolatile_memory()</a>                   | 196 |
| 6.1.4.76 | <a href="#">get_password_settings()</a>                    | 196 |
| 6.1.4.77 | <a href="#">get_pid_settings()</a>                         | 196 |
| 6.1.4.78 | <a href="#">get_position()</a>                             | 197 |
| 6.1.4.79 | <a href="#">get_position_calb()</a>                        | 197 |
| 6.1.4.80 | <a href="#">get_power_settings()</a>                       | 197 |
| 6.1.4.81 | <a href="#">get_secure_settings()</a>                      | 198 |

|           |                                                  |     |
|-----------|--------------------------------------------------|-----|
| 6.1.4.82  | <a href="#">get_serial_number()</a>              | 198 |
| 6.1.4.83  | <a href="#">get_stage_information()</a>          | 198 |
| 6.1.4.84  | <a href="#">get_stage_name()</a>                 | 199 |
| 6.1.4.85  | <a href="#">get_stage_settings()</a>             | 199 |
| 6.1.4.86  | <a href="#">get_status()</a>                     | 199 |
| 6.1.4.87  | <a href="#">get_status_calb()</a>                | 200 |
| 6.1.4.88  | <a href="#">get_sync_in_settings()</a>           | 200 |
| 6.1.4.89  | <a href="#">get_sync_in_settings_calb()</a>      | 201 |
| 6.1.4.90  | <a href="#">get_sync_out_settings()</a>          | 201 |
| 6.1.4.91  | <a href="#">get_sync_out_settings_calb()</a>     | 202 |
| 6.1.4.92  | <a href="#">get_uart_settings()</a>              | 202 |
| 6.1.4.93  | <a href="#">goto_firmware()</a>                  | 203 |
| 6.1.4.94  | <a href="#">has_firmware()</a>                   | 203 |
| 6.1.4.95  | <a href="#">logging_callback_stderr_narrow()</a> | 203 |
| 6.1.4.96  | <a href="#">logging_callback_stderr_wide()</a>   | 204 |
| 6.1.4.97  | <a href="#">msec_sleep()</a>                     | 204 |
| 6.1.4.98  | <a href="#">open_device()</a>                    | 204 |
| 6.1.4.99  | <a href="#">probe_device()</a>                   | 205 |
| 6.1.4.100 | <a href="#">service_command_updf()</a>           | 205 |
| 6.1.4.101 | <a href="#">set_accessories_settings()</a>       | 205 |
| 6.1.4.102 | <a href="#">set_brake_settings()</a>             | 206 |
| 6.1.4.103 | <a href="#">set_calibration_settings()</a>       | 206 |
| 6.1.4.104 | <a href="#">set_control_settings()</a>           | 207 |
| 6.1.4.105 | <a href="#">set_control_settings_calb()</a>      | 207 |
| 6.1.4.106 | <a href="#">set_controller_name()</a>            | 208 |
| 6.1.4.107 | <a href="#">set_correction_table()</a>           | 208 |
| 6.1.4.108 | <a href="#">set_ctp_settings()</a>               | 209 |
| 6.1.4.109 | <a href="#">set_debug_write()</a>                | 209 |
| 6.1.4.110 | <a href="#">set_edges_settings()</a>             | 209 |

|           |                                              |     |
|-----------|----------------------------------------------|-----|
| 6.1.4.111 | <a href="#">set_edges_settings_calb()</a>    | 210 |
| 6.1.4.112 | <a href="#">set_emf_settings()</a>           | 210 |
| 6.1.4.113 | <a href="#">set_encoder_information()</a>    | 211 |
| 6.1.4.114 | <a href="#">set_encoder_settings()</a>       | 211 |
| 6.1.4.115 | <a href="#">set_engine_advanced_setup()</a>  | 211 |
| 6.1.4.116 | <a href="#">set_engine_settings()</a>        | 212 |
| 6.1.4.117 | <a href="#">set_engine_settings_calb()</a>   | 212 |
| 6.1.4.118 | <a href="#">set_entype_settings()</a>        | 213 |
| 6.1.4.119 | <a href="#">set_extended_settings()</a>      | 213 |
| 6.1.4.120 | <a href="#">set_extio_settings()</a>         | 213 |
| 6.1.4.121 | <a href="#">set_feedback_settings()</a>      | 214 |
| 6.1.4.122 | <a href="#">set_gear_information()</a>       | 214 |
| 6.1.4.123 | <a href="#">set_gear_settings()</a>          | 215 |
| 6.1.4.124 | <a href="#">set_hallsensor_information()</a> | 215 |
| 6.1.4.125 | <a href="#">set_hallsensor_settings()</a>    | 215 |
| 6.1.4.126 | <a href="#">set_home_settings()</a>          | 217 |
| 6.1.4.127 | <a href="#">set_home_settings_calb()</a>     | 217 |
| 6.1.4.128 | <a href="#">set_joystick_settings()</a>      | 218 |
| 6.1.4.129 | <a href="#">set_logging_callback()</a>       | 218 |
| 6.1.4.130 | <a href="#">set_motor_information()</a>      | 218 |
| 6.1.4.131 | <a href="#">set_motor_settings()</a>         | 219 |
| 6.1.4.132 | <a href="#">set_move_settings()</a>          | 219 |
| 6.1.4.133 | <a href="#">set_move_settings_calb()</a>     | 219 |
| 6.1.4.134 | <a href="#">set_network_settings()</a>       | 220 |
| 6.1.4.135 | <a href="#">set_nonvolatile_memory()</a>     | 220 |
| 6.1.4.136 | <a href="#">set_password_settings()</a>      | 221 |
| 6.1.4.137 | <a href="#">set_pid_settings()</a>           | 221 |
| 6.1.4.138 | <a href="#">set_position()</a>               | 221 |
| 6.1.4.139 | <a href="#">set_position_calb()</a>          | 222 |
| 6.1.4.140 | <a href="#">set_power_settings()</a>         | 222 |
| 6.1.4.141 | <a href="#">set_secure_settings()</a>        | 223 |
| 6.1.4.142 | <a href="#">set_serial_number()</a>          | 223 |
| 6.1.4.143 | <a href="#">set_stage_information()</a>      | 223 |
| 6.1.4.144 | <a href="#">set_stage_name()</a>             | 224 |
| 6.1.4.145 | <a href="#">set_stage_settings()</a>         | 224 |
| 6.1.4.146 | <a href="#">set_sync_in_settings()</a>       | 224 |
| 6.1.4.147 | <a href="#">set_sync_in_settings_calb()</a>  | 225 |
| 6.1.4.148 | <a href="#">set_sync_out_settings()</a>      | 225 |
| 6.1.4.149 | <a href="#">set_sync_out_settings_calb()</a> | 226 |
| 6.1.4.150 | <a href="#">set_uart_settings()</a>          | 226 |
| 6.1.4.151 | <a href="#">write_key()</a>                  | 227 |
| 6.1.4.152 | <a href="#">ximc_version()</a>               | 227 |



# Chapter 1

## libximc library

Documentation for libximc library.

Libximc is **thread safe**, cross-platform library for working with 8SMC4-USB and 8SMC5-USB controllers.

Full documentation about controllers is [there](#).

Full documentation about libximc API is available on the page [ximc.h](#).

The libximc library is now available on [PyPI](#), and can be installed directly using pip:

```
pip install libximc
```

This simplifies the use of the library in Python projects without the need for manual building.

### 1.1 What the controller does

- Supports input and output synchronization signals to ensure the joint operation of multiple devices within a complex system.
- Works with all compact stepper motors with a winding current of up to 3 A, without feedback, as well as with stepper motors equipped with an encoder in the feedback circuit, including a linear encoder on the positioner.
- Manages controller using ready-made [xilab software](#) or using examples which allow rapid development using C++, C#, .NET, Visual Basic, Xcode, Python, Matlab, Java and LabVIEW.

### 1.2 What can do libximc library

- Libximc manages controller using interfaces: USB 2.0, RS232 and Ethernet, also uses a common and proven virtual serial port interface, so you can work with motor control modules through this library under almost all operating systems, including Windows, Linux and macOS
- Libximc library supports plug/unplug on the fly. Each device can be controlled only by one program at once. **Multiple processes (programs) that control one device simultaneously are not allowed!**

#### Warning

Libximc library opens the controller in exclusive access mode. Any controller opened with libximc (Xi↔ Lab also uses this library) needs to be closed before it may be used by another process. So at first check that you have closed XILab or other software dealing with the controller before trying to reopen the controller.

Please read the [Introduction](#) to start work with library.

To use libximc in your project please consult with [How to use with...](#)

## 1.3 Assistance

Many thanks to everyone who sends us **errors** and **suggestions**. We appreciate your suggestions and try to make our product better!

## Chapter 2

# Introduction

### 2.1 About library

This document contains all the information about the libximc library, except for the building instructions, which you can find in the README.md file at the root of the GitHub repository: <https://github.com/↔Standa-Optomechanics/libximc>. It utilizes well known virtual COM-port interface, so you can use it on Windows, Linux, macOS for Intel and Apple Silicon (via Rosetta 2) including 64-bit versions. Multi-platform programming library supports plug/unplug on the fly.

**Each device can be controlled only by one program at once. Multiple processes (programs) that control one device simultaneously are not allowed.**

#### 2.1.1 Supported OS and environment requirements:

- macOS 10.15 or newer
- Windows 7 or newer
- Linux debian-based

## Chapter 3

### How to use with...

To acquire the first skills of using the library, a simple `testappeasy_C` test application has been created. Languages other than C are supported using calls with conversion of arguments of the `stdcall` type. A simple C test application is located in the `examples/test_C` directory, a C# project is located in `examples/test_CSharp`, on VB.NET - in `examples/test_VBNET`, for MATLAB - `examples/test_MATLAB`, for Java - `examples/test_Java`, for Python - `examples/test_Python`. Libraries, header files and other necessary files are located in the directories `'ximc/win32'`, `'ximc/win64'`, `'ximc/macosex'` and the like. The developer kit also includes already compiled examples: `testapp` and `testappeasy` x32 and x64 bits for Windows and only x64 bits for macOS, `test_CSharp`, `test_VBNET`, `test_Java` - cross-platform, `test_MATLAB` and `test_Python` do not require compilation.

#### Note

SDK requires Microsoft Visual C++ Redistributable Package (provided with SDK - `vcredist_x86` or `vcredist_x64`)

On Linux both the `libximc7_x.x.x` and `libximc7-dev_x.x.x` packages of the target architecture must be installed in the specified order. For install packages, you can use the `.deb` command: `dpkg -i filename.deb`, where `filename.deb` is the name of the package (packages in Debian have the extension `.deb`). You must run `dpkg` with superuser privileges (`root`).

`Testapp` can be built using `testapp.sln`. Library must be compiled with MS Visual C++ too, `mingw-library`. Make sure that Microsoft Visual C++ Redistributable Package is installed.

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

In case of the 8SMC4-USB-Eth1 Ethernet adapter usage it is necessary to set correct IP address of the Ethernet adapter in `testapp.c` file before build (see `enumerate_hints` variable).

`Testappeasy_C` and `testapp_C` can be built using `testappeasy_C.cbp` and `testapp_C.cbp` respectively. Library must be compiled with MS Visual C++ too, `mingw-library`. Make sure that Microsoft Visual C++ Redistributable Package is installed.

Open solution `examples/test_C/testappeasy_C/testappeasy_C.cbp` or `examples/test_C/testapp_C/testapp_C.cbp`, build and run from the IDE.

MinGW is a port of GCC to win32 platform. It's required to install MinGW package.

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.

In case of the 8SMC4-USB-Eth1 Ethernet adapter usage it is necessary to set correct IP address of the Ethernet adapter in testapp.c file before build (see enumerate\_hints variable).

First of all, you should create a library suitable for C++ Builder. **Visual C++ and Builder libraries are not compatible.** Invoke:

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

Then compile test application:

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

In case of the 8SMC4-USB-Eth1 Ethernet adapter usage it is necessary to set correct IP address of the Ethernet adapter in testapp.c file before build (see enumerate\_hints variable).

There is also an [unsupported example](#) of using libximc in a C++ Builder project

testapp should be built with XCode project testapp.xcodeproj. Library is a macOS framework, and in the example application it's bundled inside testapp.app

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

In case of the 8SMC4-USB-Eth1 Ethernet adapter usage it is necessary to set correct IP address of the Ethernet adapter in testapp.c file before build (see enumerate\_hints variable). There is also [an example of using the libximc library](#) in a C++ Builder project, **but it is not supported.**

Make sure that libximc (rpm or deb) is installed on your system. Installation of package should be performed with a package manager of operating system. On macOS 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 macOS 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 macOS 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.

In case of the 8SMC4-USB-Eth1 Ethernet adapter usage it is necessary to set correct IP address of the Ethernet adapter in `testapp.c` file before build (see `enumerate_hints` variable).

Wrapper assembly for `libximc.dll` is `ximc/winX/wrappers/csharp/ximcnet.dll`. It is provided with two different architectures. Tested on platforms .NET from 2.0 to 4.5.1

Test .NET applications for Visual Studio 2013 is located at `test_CSharp` (for C#) and `test_VBNET` (for VB.NET) respectively. Open solutions and build it.

In case of the 8SMC4-USB-Eth1 Ethernet adapter usage it is necessary to set correct IP address of the Ethernet adapter in `testapp.cs` or `testapp.vb` file (depending on programming language) before build (see `enumerate_hints` variable for C# or `enum_hints` variable for VB).

How to run example on Linux. Go to `examples/test_Java/compiled-winX/` and run:

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

How to run example on Windows. Go to `examples/test_Java/compiled-winX/`. Then run:

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

How to modify and recompile an example. Go to `examples/test_Java/compiled`. Sources are embedded in a `test_Java.jar`. Extract them:

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

Then rebuild sources:

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

or for Windows or macOS

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

Then build a jar:

```
jar cmf META-INF/MANIFEST.MF test_Java.jar ru
```

In case of the 8SMC4-USB-Eth1 Ethernet adapter usage it is necessary to set correct IP address of the Ethernet adapter in `TestJava.java` file before build (see `ENUM_HINTS` variable).

Change current directory to the `examples/test_Python/xxxtest`. NB: For `libximc` usage, the example uses the wrapper module `ximc/crossplatform/wrappers/python/libximc`.

To run:

```
python xxxx.py
```

In case of the 8SMC4-USB-Eth1 Ethernet adapter usage, it's necessary to set correct IP address of the Ethernet adapter in test\_Python.py file before launch (see enum\_hints variable).

Sample MATLAB program testximc.m is provided at the directory examples/test\_MATLAB. On Windows, copy ximc.h, libximc.dll, bindy.dll, xiwrapper.dll and contents of ximc/(win32,win64)/wrappers/matlab/ directory to the current directory.

Before launch:

On macOS: copy ximc/macosx/libximc.framework, ximc/macosx/wrappers/ximcm.h, ximc/ximc.h to the directory examples/test\_MATLAB. Install XCode compatible with MATLAB.

On Linux: install libximc\*deb and libximc-dev\*dev of target architecture. Then copy ximc/macosx/wrappers/ximcm.h to the directory examples/matlab. Install gcc compatible with MATLAB.

For XCode and gcc version compatibility check document <https://www.mathworks.com/content/dam/mathworks/mathworks/SystemRequirements-Release2014a-SupportedCompilers.pdf> or similar.

No preliminary setup is required for Windows.

Change current directory in the MATLAB to the examples/test\_MATLAB. Then launch in MATLAB prompt:

```
testximc
```

In case of the 8SMC4-USB-Eth1 Ethernet adapter usage it is necessary to set correct IP address of the Ethernet adapter in testximc.m file before launch (see enum\_hints variable).

## 3.1 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.

## 3.2 Required permissions

libximc generally does not require special permissions to work, it only needs read/write access to USB-serial ports on the system. An exception to this rule is a Windows-only "fix\_usbser\_sys()" function - it needs elevation and will produce null result if run as a regular user.

## 3.3 C-profiles

C-profiles are header files distributed with the libximc library. They enable one to set all controller settings for any of the supported stages with a single function call in a C/C++ program.

You may see how to use C-profiles in the example directory "examples/test\_C/testprofile\_C".

## 3.4 Python-profiles

Python-profiles are sets of configuration functions distributed with the libximc library. They allow to load the controller with settings of one of the supported stages using a single function call in a Python program.

You may see how to use Python-profiles in the example "examples/test\_Python/profiletest/testpythonprofile.py".

## Chapter 4

# Working with user units

In addition to working in basic units (steps, encoder value), the library allows you to work with user units. For this purpose are used:

- The structure of the conversion units [calibration\\_t](#)
- The functions of which have doubles for working with user units, data structures for these functions
- Coordinate correction table for more accurate positioning

### 4.1 The structure of the conversion units `calibration_t`

To specify conversion of the basic units in the user and back, [calibration\\_t](#) structure is used. With the help of coefficients A and MicrostepMode, specified in this structure, steps and microsteps which are integers are converted into the user value of the real type and back.

Conversion formulas:

- The conversion to user units.

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

- Conversion from user units.

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

### 4.2 Alternative functions for working with user units and data structures for them

Structures and functions for working with user units have the `_calb` postfix. The user using these functions can perform all actions in their own units without worrying about the computations of the controller. The data format of `_calb` structures is described in detail. For `_calb` functions particular descriptions are not used. They perform the same actions as the basic functions do. The difference between them and the basic functions is in the position, velocity, and acceleration of the data types defined as user-defined. If clarification for `_calb` functions is necessary, they are provided as notes in the description of the basic functions.

## 4.3 Coordinate correction table for more accurate positioning

Some functions for working with user units support coordinate transformation using a correction table. To load a table from a file, the [set\\_correction\\_table\(\)](#) function is used. Its description contains the functions and their data supporting correction.

### Note

For data fields which are corrected in case of loading of the table in the description of the field is written - corrected by the table.

File format:

- two columns separated by tabs;
- column headers are string;
- real type data, point is a separator;
- the first column is the coordinate, the second is the deviation caused by a mechanical error;
- the deviation between coordinates is calculated linearly;
- constant is equal to the deviation at the boundary beyond the range;
- maximum length of the table is 100 lines.

Sample file:

| X         | dX    |
|-----------|-------|
| 0         | 0     |
| 5.0       | 0.005 |
| 10.↔<br>0 | -0.01 |

## Chapter 5

# Data Structure Documentation

### 5.1 accessories\_settings\_t Struct Reference

Deprecated.

#### 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 (V).*
- 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 [TSMMax](#)  
*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.*

### 5.1.1 Detailed Description

Deprecated.

Additional accessories' information.

See also

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

### 5.1.2 Field Documentation

#### 5.1.2.1 LimitSwitchesSettings

`unsigned int LimitSwitchesSettings`

[Temperature sensor settings flags](#).

#### 5.1.2.2 MagneticBrakeInfo

`char MagneticBrakeInfo[25]`

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

#### 5.1.2.3 MBRatedCurrent

`float MBRatedCurrent`

Rated current for controlling the magnetic brake (A).

Data type: float.

#### 5.1.2.4 MBRatedVoltage

`float MBRatedVoltage`

Rated voltage for controlling the magnetic brake (V).

Data type: float.

## 5.1.2.5 MBSettings

unsigned int MBSettings

Magnetic brake settings flags.

## 5.1.2.6 MBTorque

float MBTorque

Retention moment (mN \* m).

Data type: float.

## 5.1.2.7 TemperatureSensorInfo

char TemperatureSensorInfo[25]

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

## 5.1.2.8 TSGrad

float TSGrad

The temperature gradient (V/degrees Celsius).

Data type: float.

## 5.1.2.9 TSMaX

float TSMaX

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

## 5.1.2.10 TSMin

float TSMin

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

## 5.1.2.11 TSSettings

unsigned int TSSettings

Temperature sensor settings flags.

## 5.2 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](#)  
"Supply voltage of H-bridge's MOSFETs" 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 [Enc\\_Check\\_ADC](#)  
Voltage on encoder check line, raw ADC data.
- unsigned int **deprecated0**
- int [A1Voltage](#)  
"Voltage on pin 1 winding A" calibrated data (in tens of mV).
- int [A2Voltage](#)  
"Voltage on pin 2 winding A" calibrated data (in tens of mV).
- int [B1Voltage](#)  
"Voltage on pin 1 winding B" calibrated data (in tens of mV).
- int [B2Voltage](#)  
"Voltage on pin 2 winding B" calibrated data (in tens of mV).
- int [SupVoltage](#)  
"Supply voltage on the top of H-bridge's MOSFETs" calibrated data (in tens of mV).

- int [ACurrent](#)  
*"Winding A current" calibrated data (in mA).*
- int [BCurrent](#)  
*"Winding B current" calibrated data (in mA).*
- int [FullCurrent](#)  
*"Full current" calibrated data (in mA).*
- int [Temp](#)  
*Temperature, calibrated data (in tenths of degrees Celsius).*
- int [Joy](#)  
*Joystick, calibrated data.*
- int [Pot](#)  
*Analog input, calibrated data.*
- int [Enc\\_Check](#)  
*Voltage on encoder check line, exponentially filtrated ADC codes.*
- unsigned int **deprecated1** [2]
- int [R](#)  
*Motor winding resistance in mOhms (is only used with stepper motors).*
- int [L](#)  
*Motor winding pseudo inductance in uH (is only used with stepper motors).*

### 5.2.1 Detailed Description

Analog data.

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

See also

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

### 5.2.2 Field Documentation

#### 5.2.2.1 A1Voltage

int A1Voltage

"Voltage on pin 1 winding A" calibrated data (in tens of mV).

## 5.2.2.2 A1Voltage\_ADC

```
unsigned int A1Voltage_ADC
```

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

## 5.2.2.3 A2Voltage

```
int A2Voltage
```

"Voltage on pin 2 winding A" calibrated data (in tens of mV).

## 5.2.2.4 A2Voltage\_ADC

```
unsigned int A2Voltage_ADC
```

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

## 5.2.2.5 ACurrent

```
int ACurrent
```

"Winding A current" calibrated data (in mA).

## 5.2.2.6 ACurrent\_ADC

```
unsigned int ACurrent_ADC
```

"Winding A current" raw data from ADC.

## 5.2.2.7 B1Voltage

```
int B1Voltage
```

"Voltage on pin 1 winding B" calibrated data (in tens of mV).

## 5.2.2.8 B1Voltage\_ADC

```
unsigned int B1Voltage_ADC
```

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

## 5.2.2.9 B2Voltage

```
int B2Voltage
```

"Voltage on pin 2 winding B" calibrated data (in tens of mV).

## 5.2.2.10 B2Voltage\_ADC

```
unsigned int B2Voltage_ADC
```

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

## 5.2.2.11 BCurrent

```
int BCurrent
```

"Winding B current" calibrated data (in mA).

## 5.2.2.12 BCurrent\_ADC

```
unsigned int BCurrent_ADC
```

"Winding B current" raw data from ADC.

## 5.2.2.13 Enc\_Check

```
int Enc_Check
```

Voltage on encoder check line, exponentially filtrated ADC codes.

Used to determine encoder type: single-ended or differential.

## 5.2.2.14 Enc\_Check\_ADC

```
unsigned int Enc_Check_ADC
```

Voltage on encoder check line, raw ADC data.

Used to determine encoder type: single-ended or differential.

## 5.2.2.15 FullCurrent

```
int FullCurrent
```

"Full current" calibrated data (in mA).

## 5.2.2.16 FullCurrent\_ADC

```
unsigned int FullCurrent_ADC
```

"Full current" raw data from ADC.

## 5.2.2.17 Joy

```
int Joy
```

Joystick, calibrated data.

Range: 0..10000

## 5.2.2.18 Joy\_ADC

```
unsigned int Joy_ADC
```

Joystick raw data from ADC.

## 5.2.2.19 L

```
int L
```

Motor winding pseudo inductance in uH (is only used with stepper motors).

## 5.2.2.20 Pot

```
int Pot
```

Analog input, calibrated data.

Range: 0..10000

## 5.2.2.21 R

```
int R
```

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

## 5.2.2.22 SupVoltage

```
int SupVoltage
```

"Supply voltage on the top of H-bridge's MOSFETs" calibrated data (in tens of mV).

## 5.2.2.23 SupVoltage\_ADC

```
unsigned int SupVoltage_ADC
```

"Supply voltage of H-bridge's MOSFETs" raw data from ADC.

## 5.2.2.24 Temp

```
int Temp
```

Temperature, calibrated data (in tenths of degrees Celsius).

## 5.2.2.25 Temp\_ADC

```
unsigned int Temp_ADC
```

Voltage from temperature sensor, raw data from ADC.

## 5.3 brake\_settings\_t Struct Reference

Brake settings.

### Data Fields

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

### 5.3.1 Detailed Description

Brake settings.

This structure contains brake control parameters.

See also

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

### 5.3.2 Field Documentation

#### 5.3.2.1 BrakeFlags

unsigned int BrakeFlags

[Brake settings flags.](#)

#### 5.3.2.2 t1

unsigned int t1

Time in ms between turning on motor power and turning off the brake.

## 5.3.2.3 t2

```
unsigned int t2
```

Time in ms between the brake turning off and moving readiness.

All moving commands will execute after this interval.

## 5.3.2.4 t3

```
unsigned int t3
```

Time in ms between motor stop and the brake turning on.

## 5.3.2.5 t4

```
unsigned int t4
```

Time in ms between turning on the brake and turning off motor power.

## 5.4 calibration\_settings\_t Struct Reference

Calibration settings.

### Data Fields

- float [CSS1.A](#)  
*Scaling factor for the analog measurements of the A winding current.*
- float [CSS1.B](#)  
*Offset for the analog measurements of the A winding current.*
- float [CSS2.A](#)  
*Scaling factor for the analog measurements of the B winding current.*
- float [CSS2.B](#)  
*Offset for the analog measurements of the B winding current.*
- float [FullCurrent.A](#)  
*Scaling factor for the analog measurements of the full current.*
- float [FullCurrent.B](#)  
*Offset for the analog measurements of the full current.*

### 5.4.1 Detailed Description

Calibration settings.

This structure contains calibration settings. These settings are used to convert bare ADC values to winding currents in mA and the full current in mA. Parameters are grouped into pairs, XXX\_A and XXX\_B, representing linear equation coefficients. The first one is the slope, the second one is the constant term. Thus,  $\text{XXX\_Current[mA]} = \text{XXX\_A[mA/ADC]} * \text{XXX\_ADC\_CODE[ADC]} + \text{XXX\_B[mA]}$ .

See also

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

### 5.4.2 Field Documentation

#### 5.4.2.1 CSS1\_A

float CSS1\_A

Scaling factor for the analog measurements of the A winding current.

#### 5.4.2.2 CSS1\_B

float CSS1\_B

Offset for the analog measurements of the A winding current.

#### 5.4.2.3 CSS2\_A

float CSS2\_A

Scaling factor for the analog measurements of the B winding current.

#### 5.4.2.4 CSS2\_B

float CSS2\_B

Offset for the analog measurements of the B winding current.

## 5.4.2.5 FullCurrent\_A

```
float FullCurrent_A
```

Scaling factor for the analog measurements of the full current.

## 5.4.2.6 FullCurrent\_B

```
float FullCurrent_B
```

Offset for the analog measurements of the full current.

## 5.5 calibration\_t Struct Reference

Calibration structure.

## Data Fields

- double [A](#)  
*Conversion coefficient equal to the number of millimeters (or other user units) per one step.*
- unsigned int [MicrostepMode](#)  
*Controller setting which is determine a step division mode.*

## 5.5.1 Detailed Description

Calibration structure.

Where to find all values for calculations?

- XILab (don't forget to load the profile for your positioner. The profile should match the full name of your positioner, e.g., 8MT173-25-MEn1.cfg):
  - In XILab settings, go to the "User units" tab. Divide the second number by the first one — this is your A coefficient.
  - In the "DC motor" / "BLDC motor" / "Stepper motor" tab (depending on your motor type), find the "Encoder counts per turn" field and use it in the formula for calculating coefficient B.
- Profile file (open with any text editor; make sure it matches the full name of your positioner, e.g., 8MT173-25-MEn1.cfg):
  - Find Step\_multiplier= and Unit\_multiplier=. Divide the second by the first — this is your A coefficient.
  - Find Encoder\_CPT= and use this value in the B coefficient formula instead of ENCODER\_COUNTS\_PER\_TURN.

How to calculate Speed, Accel, Decel, and AntiplaySpeed in user units when using a stepper motor with encoder or DC/BLDC motor?

1. Load the correct profile in XILab for your positioner (e.g., 8MT173-25-MEn1.cfg).
  2. Enable Feedback encoder mode if not already enabled.
  3. Enter speed in the "Working speed" field in user units.
  4. In the "User units" tab, disable the "User units" flag to see RPM.
  5. Multiply the RPM value from "Working speed" by coefficient B. Example:  $480 * 0.0000009375 = 0.00045$ . This value for the 8MT173-25-MEn1 in Encoder mode equals 2 mm/s.
- Acceleration, deceleration, and antiplay speed are calculated the same way.

## 5.5.2 Field Documentation

### 5.5.2.1 A

double A

Conversion coefficient equal to the number of millimeters (or other user units) per one step.

Must be non-zero and positive. Used for position and movement conversion.

- For stepper motor without encoder, only one coefficient A is used. Conversion formula:  $[\text{user\_unit} \leftrightarrow \text{unit/steps}]$ . Example: 800 steps = 1 mm, then  $A = 1/800 = 0.00125$
- When using a stepper motor with encoder or DC/BLDC motors, the position is set in counts, but speed, acceleration/deceleration, and antiplay speed are set in RPM, so two coefficients are required:
  - A. Conversion for position:  $[\text{user\_unit/counts}]$ . Example: 16000 counts = 1 mm, then  $A = 1/16000 = 0.0000625$
  - B. Conversion for speed, acceleration/deceleration, and antiplay speed:  $[60/\text{ENCODER\_COUNTS\_PER\_TURN} * A]$ . Example:  $60 / 4000 * 0.0000625 = 0.0000009375$

## 5.6 chart\_data\_t Struct Reference

Additional device state.

## Data Fields

- int [WindingVoltageA](#)  
*In case of a step motor, it contains the voltage across the winding A (in tens of mV); in case of a brushless motor, it contains the voltage on the first coil; in case of a DC motor, it contains the only winding current.*
- int [WindingVoltageB](#)  
*In case of a step motor, it contains the voltage across the winding B (in tens of mV); in case of a brushless motor, it contains the voltage on the second winding; and in case of a DC motor, this field is not used.*
- int [WindingVoltageC](#)  
*In case of a brushless motor, it contains the voltage on the third winding (in tens of mV); in the case of a step motor and a DC motor, the field is not used.*
- int [WindingCurrentA](#)  
*In case of a step motor, it contains the current in the winding A (in mA); in case of a brushless motor, it contains the current in the winding A; and in case of a DC motor, it contains the only winding current.*
- int [WindingCurrentB](#)  
*In case of a step motor, it contains the current in the winding B (in mA); in case of a brushless motor, it contains the current in the winding B; and in case of a DC motor, the field is not used.*
- int [WindingCurrentC](#)  
*In case of a brushless motor, it contains the current in the winding C (in mA); in case of a step motor and a DC motor, the field is not used.*
- unsigned int [Pot](#)  
*Analog input value, dimensionless.*
- unsigned int [Joy](#)  
*The joystick position, dimensionless.*
- int [AveragedPowerRatio](#)  
*The ratio of motor supplied power to nominal motor power, as a percentage.*

## 5.6.1 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](#)

## 5.6.2 Field Documentation

## 5.6.2.1 AveragedPowerRatio

int AveragedPowerRatio

The ratio of motor supplied power to nominal motor power, as a percentage.

## 5.6.2.2 Joy

```
unsigned int Joy
```

The joystick position, dimensionless.

Range: 0..10000

## 5.6.2.3 Pot

```
unsigned int Pot
```

Analog input value, dimensionless.

Range: 0..10000

## 5.6.2.4 WindingCurrentA

```
int WindingCurrentA
```

In case of a step motor, it contains the current in the winding A (in mA); in case of a brushless motor, it contains the current in the winding A; and in case of a DC motor, it contains the only winding current.

## 5.6.2.5 WindingCurrentB

```
int WindingCurrentB
```

In case of a step motor, it contains the current in the winding B (in mA); in case of a brushless motor, it contains the current in the winding B; and in case of a DC motor, the field is not used.

## 5.6.2.6 WindingCurrentC

```
int WindingCurrentC
```

In case of a brushless motor, it contains the current in the winding C (in mA); in case of a step motor and a DC motor, the field is not used.

## 5.6.2.7 WindingVoltageA

```
int WindingVoltageA
```

In case of a step motor, it contains the voltage across the winding A (in tens of mV); in case of a brushless motor, it contains the voltage on the first coil; in case of a DC motor, it contains the only winding current.

## 5.6.2.8 WindingVoltageB

```
int WindingVoltageB
```

In case of a step motor, it contains the voltage across the winding B (in tens of mV); in case of a brushless motor, it contains the voltage on the second winding; and in case of a DC motor, this field is not used.

## 5.6.2.9 WindingVoltageC

```
int WindingVoltageC
```

In case of a brushless motor, it contains the voltage on the third winding (in tens of mV); in the case of a step motor and a DC motor, the field is not used.

## 5.7 control\_settings\_calb\_t Struct Reference

User unit control settings.

### Data Fields

- float [MaxSpeed](#) [10]  
*Array of speeds used with the joystick and the button control.*
- unsigned int [Timeout](#) [9]  
*Timeout[i] is timeout in ms.*
- unsigned int [MaxClickTime](#)  
*Maximum click time (in ms).*
- unsigned int [Flags](#)  
*Control flags.*
- float [DeltaPosition](#)  
*Position shift (delta)*

### 5.7.1 Detailed Description

User unit control settings.

This structure contains control parameters.

In case of CTL\_MODE=1, the joystick motor control is enabled. In this mode, while the joystick is maximally displaced, the engine tends to move at MaxSpeed[i]. i=0 if another value hasn't been set at the previous usage. To change the speed index "i", use the buttons.

In case of CTL\_MODE=2, the motor is controlled by the left/right buttons. When you click on the button, the motor starts moving in the appropriate direction at a speed MaxSpeed[0]. After Timeout[i], the motor moves at speed MaxSpeed[i+1]. At the transition between MaxSpeed[i] and MaxSpeed[i+1] the motor just accelerates/decelerates as usual.

See also

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

### 5.7.2 Field Documentation

#### 5.7.2.1 Flags

`unsigned int Flags`

Control flags.

#### 5.7.2.2 MaxClickTime

`unsigned int MaxClickTime`

Maximum click time (in ms).

Until the expiration of this time, the first speed isn't applied.

#### 5.7.2.3 MaxSpeed

`float MaxSpeed[10]`

Array of speeds used with the joystick and the button control.

#### 5.7.2.4 Timeout

`unsigned int Timeout[9]`

Timeout[i] is timeout in ms.

After that, max\_speed[i+1] is applied. It's used with the button control only.

## 5.8 control\_settings\_t Struct Reference

Control settings.

## Data Fields

- unsigned int [MaxSpeed](#) [10]  
*Array of speeds (full step) used with the joystick and the button control.*
- unsigned int [uMaxSpeed](#) [10]  
*Array of speeds (in microsteps) used with the joystick and the button control.*
- unsigned int [Timeout](#) [9]  
*Timeout[i] is timeout in ms.*
- unsigned int [MaxClickTime](#)  
*Maximum click time (in ms).*
- unsigned int [Flags](#)  
*Control flags.*
- int [DeltaPosition](#)  
*Position Shift (delta) (full step)*
- int [uDeltaPosition](#)  
*Fractional part of the shift in micro steps.*

### 5.8.1 Detailed Description

Control settings.

This structure contains control parameters.

In case of CTL\_MODE=1, the joystick motor control is enabled. In this mode, while the joystick is maximally displaced, the engine tends to move at MaxSpeed[i]. i=0 if another value hasn't been set at the previous usage. To change the speed index "i", use the buttons.

In case of CTL\_MODE=2, the motor is controlled by the left/right buttons. When you click on the button, the motor starts moving in the appropriate direction at a speed MaxSpeed[0]. After Timeout[i], motor moves at speed MaxSpeed[i+1]. At the transition between MaxSpeed[i] and MaxSpeed[i+1] the motor just accelerates/decelerates as usual.

See also

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

### 5.8.2 Field Documentation

#### 5.8.2.1 Flags

unsigned int Flags

[Control flags.](#)

## 5.8.2.2 MaxClickTime

```
unsigned int MaxClickTime
```

Maximum click time (in ms).

Until the expiration of this time, the first speed isn't applied.

## 5.8.2.3 MaxSpeed

```
unsigned int MaxSpeed[10]
```

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

Range: 0..100000.

## 5.8.2.4 Timeout

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

Timeout[i] is timeout in ms.

After that, max\_speed[i+1] is applied. It's used with the button control only.

## 5.8.2.5 uDeltaPosition

```
int uDeltaPosition
```

Fractional part of the shift in micro steps.

It's used with a stepper motor only. The microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine\_settings).

## 5.8.2.6 uMaxSpeed

```
unsigned int uMaxSpeed[10]
```

Array of speeds (in microsteps) used with the joystick and the button control.

The microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine\_settings).

## 5.9 controller\_name\_t Struct Reference

Controller name and settings flags.

## Data Fields

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

### 5.9.1 Detailed Description

Controller name and settings flags.

See also

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

### 5.9.2 Field Documentation

#### 5.9.2.1 ControllerName

```
char ControllerName[17]
```

User controller name.

It may be set by the user. Max string length: 16 characters.

#### 5.9.2.2 CtrlFlags

```
unsigned int CtrlFlags
```

[Flags of internal controller settings.](#)

## 5.10 ctp\_settings\_t Struct Reference

Control position settings (used with stepper motor only)

## Data Fields

- unsigned int [CTPMinError](#)  
*The minimum difference between the SM position in steps and the encoder position that causes the setting of the STATE\_CTP\_ERROR flag.*
- unsigned int [CTPFlags](#)  
*Position control flags.*

### 5.10.1 Detailed Description

Control position settings (used with stepper motor only)

When controlling the step motor with the 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 is enabled (CTP\_ENABLED is set), the controller stores the current position in the steps of SM and the current position of the encoder. Next, the encoder position is converted into steps at each step, and if the difference between the current position in steps and the encoder position is greater than CTPMinError, the flag STATE\_CTP\_ERROR is set.

Alternatively, the stepper motor may be controlled with the speed sensor (CTP\_BASE 1). In this mode, at the active edges of the input clock, the controller stores the current value of steps. Then, at each revolution, the controller checks how many steps have been passed. When the difference is over the CTPMinError, the STATE\_CTP\_ERROR flag is set.

See also

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

### 5.10.2 Field Documentation

#### 5.10.2.1 CTPFlags

unsigned int CTPFlags

[Position control flags.](#)

#### 5.10.2.2 CTPMinError

unsigned int CTPMinError

The minimum difference between the SM position in steps and the encoder position that causes the setting of the STATE\_CTP\_ERROR flag.

Measured in steps.

## 5.11 debug\_read\_t Struct Reference

Debug data.

## Data Fields

- uint8\_t [DebugData](#) [128]  
*Arbitrary debug data.*

### 5.11.1 Detailed Description

Debug data.

These data are used for device debugging by the manufacturer.

See also

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

### 5.11.2 Field Documentation

#### 5.11.2.1 DebugData

uint8\_t DebugData[128]

Arbitrary debug data.

## 5.12 debug\_write\_t Struct Reference

Debug data.

## Data Fields

- uint8\_t [DebugData](#) [128]  
*Arbitrary debug data.*

### 5.12.1 Detailed Description

Debug data.

These data are used for device debugging by the manufacturer.

See also

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

### 5.12.2 Field Documentation

#### 5.12.2.1 DebugData

uint8\_t DebugData[128]

Arbitrary debug data.

## 5.13 device\_information\_t Struct Reference

Controller information structure.

### 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](#)  
*The minor number of the hardware version.*
- unsigned int [Release](#)  
*Release version.*

### 5.13.1 Detailed Description

Controller information structure.

See also

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

### 5.13.2 Field Documentation

## 5.13.2.1 Major

```
unsigned int Major
```

The major number of the hardware version.

## 5.13.2.2 Minor

```
unsigned int Minor
```

The minor number of the hardware version.

## 5.13.2.3 Release

```
unsigned int Release
```

Release version.

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

Device network information structure.

## Data Fields

- `uint32_t` [ipv4](#)  
*IPv4 address, passed in network byte order (big-endian byte order)*
- `char` [nodename](#) [16]  
*name of the Bindy node which hosts the device*
- `uint32_t` [axis.state](#)  
*flags representing device state*
- `char` [locker\\_username](#) [16]  
*name of the user who locked the device (if any)*
- `char` [locker\\_nodename](#) [16]  
*Bindy node name, which was used to lock the device (if any)*
- `time_t` [locked\\_time](#)  
*time the lock was acquired at (UTC, microseconds since the epoch)*

## 5.14.1 Detailed Description

Device network information structure.

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

User unit edges settings.

### 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.*

### 5.15.1 Detailed Description

User unit 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 may lead to device malfunction, which can cause irreversible damage to the board.

See also

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

### 5.15.2 Field Documentation

#### 5.15.2.1 BorderFlags

unsigned int BorderFlags

[Border flags.](#)

#### 5.15.2.2 EnderFlags

unsigned int EnderFlags

[Limit switches flags.](#)

## 5.15.2.3 LeftBorder

float LeftBorder

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

Corrected by the table.

## 5.15.2.4 RightBorder

float RightBorder

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

Corrected by the table.

## 5.16 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 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 microsteps.*

## 5.16.1 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 may lead to device malfunction, which can cause irreversible damage to the board.

See also

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

## 5.16.2 Field Documentation

### 5.16.2.1 BorderFlags

`unsigned int BorderFlags`

[Border flags.](#)

### 5.16.2.2 EnderFlags

`unsigned int EnderFlags`

[Limit switches flags.](#)

### 5.16.2.3 LeftBorder

`int LeftBorder`

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

### 5.16.2.4 RightBorder

`int RightBorder`

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

### 5.16.2.5 uLeftBorder

`int uLeftBorder`

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

The microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine\_settings).

## 5.16.2.6 uRightBorder

```
int uRightBorder
```

Right border position in microsteps.

Used with a stepper motor only. The microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine\_settings).

## 5.17 emf\_settings\_t Struct Reference

EMF settings.

### Data Fields

- float [L](#)  
*Motor winding inductance.*
- float [R](#)  
*Motor winding resistance.*
- float [Km](#)  
*Electromechanical ratio of the motor.*
- unsigned int [BackEMFFlags](#)  
*Flags of auto-detection of characteristics of windings of the engine.*

### 5.17.1 Detailed Description

EMF settings.

This structure contains the data for Electromechanical characteristics (EMF) of the motor. It determines the inductance, resistance, and Electromechanical coefficient of the motor. This data is stored in the flash memory of the controller. Please set new settings when you change the motor. Remember that improper EMF settings may damage the equipment.

See also

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

### 5.17.2 Field Documentation

## 5.17.2.1 BackEMFFlags

unsigned int BackEMFFlags

Flags of auto-detection of characteristics of windings of the engine.

## 5.17.2.2 Km

float Km

Electromechanical ratio of the motor.

## 5.17.2.3 L

float L

Motor winding inductance.

## 5.17.2.4 R

float R

Motor winding resistance.

## 5.18 encoder\_information\_t Struct Reference

Deprecated.

## Data Fields

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

### 5.18.1 Detailed Description

Deprecated.

Encoder information.

See also

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

### 5.18.2 Field Documentation

#### 5.18.2.1 Manufacturer

char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

#### 5.18.2.2 PartNumber

char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

## 5.19 encoder\_settings\_t Struct Reference

Deprecated.

### Data Fields

- float [MaxOperatingFrequency](#)  
*Maximum 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.*

### 5.19.1 Detailed Description

Deprecated.

Encoder settings.

See also

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

### 5.19.2 Field Documentation

#### 5.19.2.1 EncoderSettings

`unsigned int EncoderSettings`

[Encoder settings flags](#).

#### 5.19.2.2 MaxCurrentConsumption

`float MaxCurrentConsumption`

Max current consumption (mA).

Data type: float.

#### 5.19.2.3 MaxOperatingFrequency

`float MaxOperatingFrequency`

Maximum operation frequency (kHz).

Data type: float.

#### 5.19.2.4 SupplyVoltageMax

`float SupplyVoltageMax`

Maximum supply voltage (V).

Data type: float.

## 5.19.2.5 SupplyVoltageMin

```
float SupplyVoltageMin
```

Minimum supply voltage (V).

Data type: float.

## 5.20 engine\_advanced\_setup\_t Struct Reference

EAS settings.

## Data Fields

- unsigned int [stepcloseloop\\_Kw](#)  
*Manufacturer only.*
- unsigned int [stepcloseloop\\_Kp\\_low](#)  
*Manufacturer only.*
- unsigned int [stepcloseloop\\_Kp\\_high](#)  
*Manufacturer only.*

## 5.20.1 Detailed Description

EAS settings.

Manufacturer only. This structure is intended for setting parameters of algorithms that cannot be attributed to standard Kp, Ki, Kd, and L, R, Km.

See also

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

## 5.20.2 Field Documentation

## 5.20.2.1 stepcloseloop\_Kp\_high

```
unsigned int stepcloseloop_Kp_high
```

Manufacturer only.

Position feedback in the high-speed zone, range [0, 65535], default value 33.

## 5.20.2.2 stepcloseloop\_Kp\_low

unsigned int stepcloseloop\_Kp\_low

Manufacturer only.

Position feedback in the low-speed zone, range [0, 65535], default value 1000.

## 5.20.2.3 stepcloseloop\_Kw

unsigned int stepcloseloop\_Kw

Manufacturer only.

Mixing ratio of the actual and set speed, range [0, 100], default value 50.

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

Movement limitations and settings, related to the motor.

## Data Fields

- unsigned int [NomVoltage](#)  
*Rated voltage in tens of mV.*
- unsigned int [NomCurrent](#)  
*Rated current (in mA).*
- 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).*
- unsigned int [PeakCurrent](#)  
*Rated peak current (in mA).*

### 5.21.1 Detailed Description

Movement limitations and settings, related to the motor.

In user units.

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

See also

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

### 5.21.2 Field Documentation

#### 5.21.2.1 Antiplay

`float Antiplay`

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

Used if ENGINE\_ANTIPLAY flag is set.

#### 5.21.2.2 EngineFlags

`unsigned int EngineFlags`

[Flags of engine settings.](#)

#### 5.21.2.3 MicrostepMode

`unsigned int MicrostepMode`

[Flags of microstep mode.](#)

## 5.21.2.4 NomCurrent

```
unsigned int NomCurrent
```

Rated current (in mA).

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

## 5.21.2.5 NomSpeed

```
float NomSpeed
```

Nominal speed.

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

## 5.21.2.6 NomVoltage

```
unsigned int NomVoltage
```

Rated voltage in tens of mV.

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

## 5.21.2.7 PeakCurrent

```
unsigned int PeakCurrent
```

Rated peak current (in mA).

Controller is allowed to apply peak current while maintaining 10-second average current below nominal if USE\_PEAK\_CURRENT flag is set.

## 5.21.2.8 StepsPerRev

```
unsigned int StepsPerRev
```

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

Range: 1..65535.

## 5.22 engine\_settings\_t Struct Reference

Movement limitations and settings related to the motor.

## Data Fields

- unsigned int [NomVoltage](#)  
*Rated voltage in tens of mV.*
- unsigned int [NomCurrent](#)  
*Rated current (in mA).*
- unsigned int [NomSpeed](#)  
*Nominal (maximum) 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).*
- unsigned int [PeakCurrent](#)  
*Rated peak current (in mA).*

## 5.22.1 Detailed Description

Movement limitations and settings related to the motor.

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

See also

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

## 5.22.2 Field Documentation

## 5.22.2.1 Antiplay

int Antiplay

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

Used if ENGINE\_ANTIPLAY flag is set.

## 5.22.2.2 EngineFlags

unsigned int EngineFlags

[Flags of engine settings.](#)

## 5.22.2.3 MicrostepMode

unsigned int MicrostepMode

[Flags of microstep mode.](#)

## 5.22.2.4 NomCurrent

unsigned int NomCurrent

Rated current (in mA).

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

## 5.22.2.5 NomSpeed

unsigned int NomSpeed

Nominal (maximum) 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.

## 5.22.2.6 NomVoltage

unsigned int NomVoltage

Rated voltage in tens of mV.

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

## 5.22.2.7 PeakCurrent

unsigned int PeakCurrent

Rated peak current (in mA).

Controller is allowed to apply peak current while maintaining 10-second average current below nominal if USE\_PEAK\_CURRENT flag is set.

## 5.22.2.8 StepsPerRev

unsigned int StepsPerRev

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

Range: 1..65535.

## 5.22.2.9 uNomSpeed

unsigned int uNomSpeed

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

Microstep size and the range of valid values for this field depend on selected step division mode (see MicrostepMode field in engine\_settings).

## 5.23 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.*

## 5.23.1 Detailed Description

Engine type and driver type settings.

## Parameters

|                   |                           |
|-------------------|---------------------------|
| <i>id</i>         | An identifier of a device |
| <i>EngineType</i> | engine type               |
| <i>DriverType</i> | driver type               |

See also

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

## 5.23.2 Field Documentation

5.23.2.1 `DriverType`

`unsigned int DriverType`

[Flags of driver type.](#)

5.23.2.2 `EngineType`

`unsigned int EngineType`

[Flags of engine type.](#)

## 5.24 `extended_settings_t` Struct Reference

EST settings This data is stored in the controller's flash memory.

### Data Fields

- `unsigned int` **Param1**

## 5.24.1 Detailed Description

EST settings This data is stored in the controller's flash memory.

This structure is designed for the future. Currently, it is not in use.

See also

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

## 5.25 `extio_settings_t` Struct Reference

EXTIO settings.

### Data Fields

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

### 5.25.1 Detailed Description

EXTIO settings.

This structure contains all EXTIO settings. By default, input events are signaled through a rising front, and output states are signaled by a high logic state.

See also

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

### 5.25.2 Field Documentation

#### 5.25.2.1 EXTIOModeFlags

`unsigned int EXTIOModeFlags`

[External IO mode flags.](#)

#### 5.25.2.2 EXTIOSetupFlags

`unsigned int EXTIOSetupFlags`

[External IO setup flags.](#)

## 5.26 feedback\_settings\_t Struct Reference

Feedback settings.

### Data Fields

- unsigned int [IPS](#)  
*The number of encoder counts per shaft revolution.*
- unsigned int [FeedbackType](#)  
*Feedback type.*
- unsigned int [FeedbackFlags](#)  
*Describes feedback flags.*
- unsigned int [CountsPerTurn](#)  
*The number of encoder counts per shaft revolution.*

### 5.26.1 Detailed Description

Feedback settings.

This structure contains feedback settings.

See also

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

### 5.26.2 Field Documentation

#### 5.26.2.1 CountsPerTurn

`unsigned int CountsPerTurn`

The number of encoder counts per shaft revolution.

Range: 1..4294967295. To use the CountsPerTurn field, write 0 in the IPS field, otherwise the value from the IPS field will be used.

#### 5.26.2.2 FeedbackFlags

`unsigned int FeedbackFlags`

[Describes feedback flags.](#)

#### 5.26.2.3 FeedbackType

`unsigned int FeedbackType`

[Feedback type.](#)

#### 5.26.2.4 IPS

`unsigned int IPS`

The number of encoder counts per shaft revolution.

Range: 1..655535. The field is obsolete, it is recommended to write 0 to IPS and use the extended CountsPerTurn field. You may need to update the controller firmware to the latest version.

## 5.27 gear\_information\_t Struct Reference

Deprecated.

### Data Fields

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

### 5.27.1 Detailed Description

Deprecated.

Gear information.

See also

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

### 5.27.2 Field Documentation

#### 5.27.2.1 Manufacturer

char `Manufacturer`[17]

Manufacturer.

Max string length: 16 chars.

#### 5.27.2.2 PartNumber

char `PartNumber`[25]

Series and PartNumber.

Max string length: 24 chars.

## 5.28 gear\_settings\_t Struct Reference

Deprecated.

## Data Fields

- float [ReductionIn](#)  
*Input reduction coefficient.*
- float [ReductionOut](#)  
*Output reduction coefficient.*
- float [RatedInputTorque](#)  
*Maximum continuous torque ( $N * m$ ).*
- float [RatedInputSpeed](#)  
*Maximum speed on the input shaft (rpm).*
- float [MaxOutputBacklash](#)  
*Output backlash of the reduction gear (degree).*
- float [InputInertia](#)  
*Equivalent input gear inertia ( $g * cm^2$ ).*
- float [Efficiency](#)  
*Reduction gear efficiency (%).*

## 5.28.1 Detailed Description

Deprecated.

Gear settings.

See also

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

## 5.28.2 Field Documentation

## 5.28.2.1 Efficiency

float Efficiency

Reduction gear efficiency (%).

Data type: float.

## 5.28.2.2 InputInertia

float InputInertia

Equivalent input gear inertia ( $g * cm^2$ ).

Data type: float.

## 5.28.2.3 MaxOutputBacklash

float MaxOutputBacklash

Output backlash of the reduction gear (degree).

Data type: float.

## 5.28.2.4 RatedInputSpeed

float RatedInputSpeed

Maximum speed on the input shaft (rpm).

Data type: float.

## 5.28.2.5 RatedInputTorque

float RatedInputTorque

Maximum continuous torque (N \* m).

Data type: float.

## 5.28.2.6 ReductionIn

float ReductionIn

Input reduction coefficient.

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

## 5.28.2.7 ReductionOut

float ReductionOut

Output reduction coefficient.

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

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

Position information.

## Data Fields

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

### 5.29.1 Detailed Description

Position information.

A useful structure that contains position value in user units for stepper motor and encoder steps for all engines.

See also

[get\\_position](#)

### 5.29.2 Field Documentation

#### 5.29.2.1 `EncPosition`

`long_t EncPosition`

Encoder position.

#### 5.29.2.2 `Position`

`float Position`

The position in the engine.

Corrected by the table.

## 5.30 `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.*

### 5.30.1 Detailed Description

Position information.

A useful structure that contains position value in steps and microsteps for stepper motor and encoder steps for all engines.

See also

[get\\_position](#)

### 5.30.2 Field Documentation

#### 5.30.2.1 EncPosition

`long_t EncPosition`

Encoder position.

#### 5.30.2.2 uPosition

`int uPosition`

Microstep position is only used with stepper motors.

Microstep size and the range of valid values for this field depend on the selected step division mode (see MicrostepMode field in engine\_settings).

## 5.31 globally\_unique\_identifier\_t Struct Reference

Globally unique identifier.

## Data Fields

- unsigned int [UniqueID0](#)  
*Unique ID 0.*
- unsigned int [UniqueID1](#)  
*Unique ID 1.*
- unsigned int [UniqueID2](#)  
*Unique ID 2.*
- unsigned int [UniqueID3](#)  
*Unique ID 3.*

### 5.31.1 Detailed Description

Globally unique identifier.

Manufacturer only.

See also

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

### 5.31.2 Field Documentation

#### 5.31.2.1 UniqueID0

unsigned int UniqueID0

Unique ID 0.

#### 5.31.2.2 UniqueID1

unsigned int UniqueID1

Unique ID 1.

#### 5.31.2.3 UniqueID2

unsigned int UniqueID2

Unique ID 2.

## 5.31.2.4 UniqueID3

unsigned int UniqueID3

Unique ID 3.

## 5.32 hallsensor\_information\_t Struct Reference

Deprecated.

## Data Fields

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

## 5.32.1 Detailed Description

Deprecated.

Hall sensor information.

See also

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

## 5.32.2 Field Documentation

## 5.32.2.1 Manufacturer

char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

## 5.32.2.2 PartNumber

char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

## 5.33 hallsensor\_settings\_t Struct Reference

Deprecated.

### Data Fields

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

### 5.33.1 Detailed Description

Deprecated.

Hall sensor settings.

See also

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

### 5.33.2 Field Documentation

#### 5.33.2.1 MaxCurrentConsumption

float MaxCurrentConsumption

Maximum current consumption (mA).

Data type: float.

#### 5.33.2.2 MaxOperatingFrequency

float MaxOperatingFrequency

Maximum operation frequency (kHz).

Data type: float.

## 5.33.2.3 SupplyVoltageMax

float SupplyVoltageMax

Maximum supply voltage (V).

Data type: float.

## 5.33.2.4 SupplyVoltageMin

float SupplyVoltageMin

Minimum supply voltage (V).

Data type: float.

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

Position calibration settings which use user units.

## 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.*

## 5.34.1 Detailed Description

Position calibration settings which use user units.

This structure contains settings used in position calibrating. It specifies behavior of calibrating position.

See also

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

## 5.34.2 Field Documentation

## 5.34.2.1 FastHome

float FastHome

Speed used for first motion.

## 5.34.2.2 HomeDelta

float HomeDelta

Distance from break point.

## 5.34.2.3 HomeFlags

unsigned int HomeFlags

[Home settings flags.](#)

## 5.34.2.4 SlowHome

float SlowHome

Speed used for second motion.

## 5.35 home\_settings\_t Struct Reference

Position calibration settings.

## Data Fields

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

### 5.35.1 Detailed Description

Position calibration settings.

This structure contains settings used in position calibration. It specify behavior of calibration procedure.

See also

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

### 5.35.2 Field Documentation

#### 5.35.2.1 FastHome

`unsigned int FastHome`

Speed used for first motion (full steps).

Range: 0..100000.

#### 5.35.2.2 HomeDelta

`int HomeDelta`

Distance from break point (full steps).

#### 5.35.2.3 HomeFlags

`unsigned int HomeFlags`

[Home settings flags](#).

#### 5.35.2.4 SlowHome

`unsigned int SlowHome`

Speed used for second motion (full steps).

Range: 0..100000.

5.35.2.5 `uFastHome`

```
unsigned int uFastHome
```

Fractional part of the speed for first motion, microsteps.

The microstep size and the range of valid values for this field depend on the selected step division mode (see the `MicrostepMode` field in `engine_settings`).

5.35.2.6 `uHomeDelta`

```
int uHomeDelta
```

Fractional part of the delta distance, microsteps.

The microstep size and the range of valid values for this field depend on the selected step division mode (see the `MicrostepMode` field in `engine_settings`).

5.35.2.7 `uSlowHome`

```
unsigned int uSlowHome
```

Part of the speed for second motion, microsteps.

The microstep size and the range of valid values for this field depend on the selected step division mode (see the `MicrostepMode` field in `engine_settings`).

## 5.36 `init_random_t` Struct Reference

Random key.

### Data Fields

- `uint8_t key` [16]  
*Random key.*

### 5.36.1 Detailed Description

Random key.

Manufacturer only.

Structure that contains a random key. It is used in the encryption of `WKEY` and `SSER` command contents.

See also

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

### 5.36.2 Field Documentation

#### 5.36.2.1 key

uint8\_t key[16]

Random key.

## 5.37 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 upper end position.*
- unsigned int [ExpFactor](#)  
*Exponential nonlinearity factor.*
- unsigned int [DeadZone](#)  
*Joystick deviation from the central position in 0.1% units that do not result in a start of motion.*
- unsigned int [JoyFlags](#)  
*Joystick flags.*

### 5.37.1 Detailed Description

Joystick settings.

This structure contains joystick parameters. If joystick position falls outside DeadZone limits, a movement begins. The speed is defined by the joystick's position in the range from the DeadZone limit to the maximum deviation. Joystick positions inside DeadZone limits correspond to zero speed (a "soft stop" command is issued continuously), and positions beyond Low and High limits correspond to MaxSpeed[i] or -MaxSpeed[i] (see command SCTL), where i = 0 by default and can be changed with the left/right buttons (see command SCTL). If the next speed in the list is zero (both integer and microstep parts), the button press is ignored. The first speed in the list shouldn't be zero. DeadZone is defined in 0.1% units.

The relationship between the deviation and the rate is exponential, which allows for high mobility and accuracy without speed mode switching.

See also

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

## 5.37.2 Field Documentation

### 5.37.2.1 DeadZone

`unsigned int DeadZone`

Joystick deviation from the central position in 0.1% units that do not result in a start of motion.

Maximal deviation is  $\pm 25.5\%$  that is more than a half of the total joystick range.

### 5.37.2.2 ExpFactor

`unsigned int ExpFactor`

Exponential nonlinearity factor.

### 5.37.2.3 JoyCenter

`unsigned int JoyCenter`

Joystick center position.

Range: 0..10000.

### 5.37.2.4 JoyFlags

`unsigned int JoyFlags`

[Joystick flags](#).

### 5.37.2.5 JoyHighEnd

`unsigned int JoyHighEnd`

Joystick upper end position.

Range: 0..10000.

## 5.37.2.6 JoyLowEnd

unsigned int JoyLowEnd

Joystick lower end position.

Range: 0..10000.

## 5.38 measurements\_t Struct Reference

The structure contains a sequence of measured axis motion parameters - velocities and position errors.

### Data Fields

- int [Speed](#) [25]  
*Sequence of measured speeds (in encoder counts/s or microsteps/sec, depending on the motor type and control mode)*
- int [Error](#) [25]  
*Position error in microsteps (whole steps are recalculated considering the current step division mode) or encoder counts.*
- unsigned int [Length](#)  
*Actual sequence length.*

### 5.38.1 Detailed Description

The structure contains a sequence of measured axis motion parameters - velocities and position errors.

The time step between consecutive measurements is 1 ms.

See also

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

### 5.38.2 Field Documentation

#### 5.38.2.1 Error

int Error[25]

Position error in microsteps (whole steps are recalculated considering the current step division mode) or encoder counts.

## 5.38.2.2 Length

unsigned int Length

Actual sequence length.

The values contained in cells Length, Length+1, ...24 shall not be interpreted as measurement results.

## 5.39 motor\_information\_t Struct Reference

Deprecated.

## Data Fields

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

## 5.39.1 Detailed Description

Deprecated.

motor information.

See also

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

## 5.39.2 Field Documentation

## 5.39.2.1 Manufacturer

char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

## 5.39.2.2 PartNumber

```
char PartNumber[25]
```

Series and PartNumber.

Max string length: 24 chars.

## 5.40 motor\_settings\_t Struct Reference

Deprecated.

## 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 motors.*
- unsigned int [Phases](#)  
*Number of phases for BLDC motors.*
- float [NominalVoltage](#)  
*Nominal voltage on winding (B).*
- float [NominalCurrent](#)  
*Maximum direct current in winding for DC and BLDC engines, nominal current in windings for stepper motors (A).*
- float [NominalSpeed](#)  
*Not used.*
- float [NominalTorque](#)  
*Nominal torque (mN \* m).*
- float [NominalPower](#)  
*Nominal power (W).*
- float [WindingResistance](#)  
*Resistance of windings for DC engines, of each of two windings for stepper motors, or of each of three windings for BLDC engines (Ohm).*
- float [WindingInductance](#)  
*Inductance of windings for DC engines, inductance of each of two windings for stepper motors, or inductance of each of three windings for BLDC engines (mH).*
- float [RotorInertia](#)  
*Rotor inertia (g \* cm<sup>2</sup>).*
- 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 unpowered windings (mN \* m).*
- float [TorqueConstant](#)

*Torque constant that determines the proportionality constant between the maximum rotor torque and current flowing in the winding ( $\text{mN} \cdot \text{m} / \text{A}$ ).*

- float [SpeedConstant](#)

*Velocity constant, which determines the value or the amplitude of the induced voltage on the motion of DC or BLDC motors ( $\text{rpm} / \text{V}$ ) or stepper motors ( $\text{steps/s} / \text{V}$ ).*

- float [SpeedTorqueGradient](#)

*Speed torque gradient ( $\text{rpm} / \text{mN} \cdot \text{m}$ ).*

- float [MechanicalTimeConstant](#)

*Mechanical time constant ( $\text{ms}$ ).*

- float [MaxSpeed](#)

*The maximum speed for stepper motors ( $\text{steps/s}$ ) or DC and BLDC motors ( $\text{rpm}$ ).*

- float [MaxCurrent](#)

*The maximum current in the winding ( $\text{A}$ ).*

- float [MaxCurrentTime](#)

*Safe duration of overcurrent in the winding ( $\text{ms}$ ).*

- float [NoLoadCurrent](#)

*The current consumption in idle mode ( $\text{A}$ ).*

- float [NoLoadSpeed](#)

*Idle speed ( $\text{rpm}$ ).*

### 5.40.1 Detailed Description

Deprecated.

Physical characteristics and limitations of the motor.

See also

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

### 5.40.2 Field Documentation

#### 5.40.2.1 DetentTorque

`float DetentTorque`

Holding torque position with unpowered windings ( $\text{mN} \cdot \text{m}$ ).

Data type: float.

#### 5.40.2.2 MaxCurrent

`float MaxCurrent`

The maximum current in the winding ( $\text{A}$ ).

Data type: float.

## 5.40.2.3 MaxCurrentTime

```
float MaxCurrentTime
```

Safe duration of overcurrent in the winding (ms).

Data type: float.

## 5.40.2.4 MaxSpeed

```
float MaxSpeed
```

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

Data type: float.

## 5.40.2.5 MechanicalTimeConstant

```
float MechanicalTimeConstant
```

Mechanical time constant (ms).

Data type: float.

## 5.40.2.6 MotorType

```
unsigned int MotorType
```

[Motor Type flags.](#)

## 5.40.2.7 NoLoadCurrent

```
float NoLoadCurrent
```

The current consumption in idle mode (A).

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

## 5.40.2.8 NoLoadSpeed

```
float NoLoadSpeed
```

Idle speed (rpm).

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

## 5.40.2.9 NominalCurrent

```
float NominalCurrent
```

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

Data type: float.

## 5.40.2.10 NominalPower

```
float NominalPower
```

Nominal power (W).

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

## 5.40.2.11 NominalSpeed

```
float NominalSpeed
```

Not used.

Nominal speed (rpm). Used for DC and BLDC engines. Data type: float.

## 5.40.2.12 NominalTorque

```
float NominalTorque
```

Nominal torque (mN \* m).

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

## 5.40.2.13 NominalVoltage

```
float NominalVoltage
```

Nominal voltage on winding (V).

Data type: float

## 5.40.2.14 Phases

```
unsigned int Phases
```

Number of phases for BLDC motors.

## 5.40.2.15 Poles

```
unsigned int Poles
```

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

## 5.40.2.16 RotorInertia

```
float RotorInertia
```

Rotor inertia ( $\text{g} * \text{cm}^2$ ).

Data type: float.

## 5.40.2.17 SpeedConstant

```
float SpeedConstant
```

Velocity constant, which determines the value or the amplitude of the induced voltage on the motion of DC or BLDC motors ( $\text{rpm} / \text{V}$ ) or stepper motors ( $\text{steps/s} / \text{V}$ ).

Data type: float.

## 5.40.2.18 SpeedTorqueGradient

```
float SpeedTorqueGradient
```

Speed torque gradient ( $\text{rpm} / \text{mN} * \text{m}$ ).

Data type: float.

## 5.40.2.19 StallTorque

```
float StallTorque
```

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

Data type: float.

## 5.40.2.20 TorqueConstant

```
float TorqueConstant
```

Torque constant that determines the proportionality constant between the maximum rotor torque and current flowing in the winding ( $\text{mN} * \text{m} / \text{A}$ ).

Used mainly for DC motors. Data type: float.

## 5.40.2.21 WindingInductance

float WindingInductance

Inductance of windings for DC engines, inductance of each of two windings for stepper motors, or inductance of each of three windings for BLDC engines (mH).

Data type: float.

## 5.40.2.22 WindingResistance

float WindingResistance

Resistance of windings for DC engines, of each of two windings for stepper motors, or of each of three windings for BLDC engines (Ohm).

Data type: float.

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

User units move settings.

## Data Fields

- float [Speed](#)  
*Speed.*
- float [Accel](#)  
*Acceleration.*
- float [Decel](#)  
*Deceleration.*
- float [AntiplaySpeed](#)  
*Antiplay speed.*
- unsigned int [MoveFlags](#)  
*Flags of the motion parameters.*

## 5.41.1 Detailed Description

User units move settings.

See also

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

### 5.41.2 Field Documentation

#### 5.41.2.1 Accel

`float Accel`

Acceleration:

- For stepper motor without encoder — in steps per second<sup>2</sup>.
- When using encoder (including DC/BLDC) — in revolutions per minute (RPM).

#### 5.41.2.2 AntiplaySpeed

`float AntiplaySpeed`

Antiplay speed.

- For stepper motor without encoder — in steps per second.
- When using encoder (including DC/BLDC) — in revolutions per minute (RPM).

#### 5.41.2.3 Decel

`float Decel`

Deceleration.

- For stepper motor without encoder — in steps per second<sup>2</sup>.
- When using encoder (including DC/BLDC) — in revolutions per minute (RPM).

#### 5.41.2.4 MoveFlags

`unsigned int MoveFlags`

Flags of the motion parameters.

## 5.41.2.5 Speed

float Speed

Speed.

- For stepper motor without encoder — in steps per second.
- When using encoder (including DC/BLDC) — in revolutions per minute (RPM).

## 5.42 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 microstep fractions/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, microsteps/s.*
- unsigned int [MoveFlags](#)  
*Flags of the motion parameters.*

## 5.42.1 Detailed Description

Move settings.

See also

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

## 5.42.2 Field Documentation

## 5.42.2.1 Accel

```
unsigned int Accel
```

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

Range: 1..65535.

## 5.42.2.2 AntiplaySpeed

```
unsigned int AntiplaySpeed
```

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

Range: 0..100000.

## 5.42.2.3 Decel

```
unsigned int Decel
```

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

Range: 1..65535.

## 5.42.2.4 MoveFlags

```
unsigned int MoveFlags
```

[Flags of the motion parameters.](#)

## 5.42.2.5 Speed

```
unsigned int Speed
```

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

Range: 0..100000.

## 5.42.2.6 uAntiplaySpeed

```
unsigned int uAntiplaySpeed
```

Speed in antiplay mode, microsteps/s.

The microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine\_settings). Used with a stepper motor only.

## 5.42.2.7 uSpeed

```
unsigned int uSpeed
```

Target speed in microstep fractions/s.

The microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine\_settings). Used with a stepper motor only.

## 5.43 network\_settings\_t Struct Reference

Network settings.

## Data Fields

- unsigned int [DHCPEnabled](#)  
*Indicates the method to get the IP-address.*
- unsigned int [IPv4Address](#) [4]  
*IP-address of the device in format x.x.x.x.*
- unsigned int [SubnetMask](#) [4]  
*The mask of the subnet in format x.x.x.x.*
- unsigned int [DefaultGateway](#) [4]  
*Default value of the gateway in format x.x.x.x.*

## 5.43.1 Detailed Description

Network settings.

Manufacturer only. This structure contains network settings.

See also

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

## 5.43.2 Field Documentation

## 5.43.2.1 DefaultGateway

```
unsigned int DefaultGateway[4]
```

Default value of the gateway in format x.x.x.x.

## 5.43.2.2 DHCPEnabled

```
unsigned int DHCPEnabled
```

Indicates the method to get the IP-address.

It can be either 0 (static) or 1 (DHCP).

## 5.43.2.3 IPv4Address

```
unsigned int IPv4Address[4]
```

IP-address of the device in format x.x.x.x.

## 5.43.2.4 SubnetMask

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

The mask of the subnet in format x.x.x.x.

## 5.44 nonvolatile\_memory\_t Struct Reference

Structure contains user data to save into the FRAM.

## Data Fields

- unsigned int [UserData](#) [7]  
*User data.*

## 5.44.1 Detailed Description

Structure contains user data to save into the FRAM.

See also

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

## 5.44.2 Field Documentation

## 5.44.2.1 UserData

```
unsigned int UserData[7]
```

User data.

It may be set by the user. Each element of the array stores only 32 bits of user data. This is important on systems where an int type contains more than 4 bytes. For example, on all amd64 systems.

## 5.45 password\_settings\_t Struct Reference

The web-page user password.

## Data Fields

- char [UserPassword](#) [21]

*Password for the web-page that the user can change with a USB command or via web-page.*

## 5.45.1 Detailed Description

The web-page user password.

Manufacturer only. This structure contains the user password.

See also

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

## 5.45.2 Field Documentation

## 5.45.2.1 UserPassword

```
char UserPassword[21]
```

Password for the web-page that the user can change with a USB command or via web-page.

## 5.46 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.*
- float [Kpf](#)  
*Proportional gain for BLDC position PID routine.*
- float [Kif](#)  
*Integral gain for BLDC position PID routine.*
- float [Kdf](#)  
*Differential gain for BLDC position PID routine.*

## 5.46.1 Detailed Description

PID settings.

This structure contains factors for PID routine. It specifies the behavior of the voltage PID routine. These factors are slightly different for different positioners. All boards are supplied with the standard set of PID settings in the 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](#)

5.47 `power_settings_t` Struct Reference

Step motor power settings.

## Data Fields

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

### 5.47.1 Detailed Description

Step motor power settings.

See also

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

### 5.47.2 Field Documentation

#### 5.47.2.1 CurrentSetTime

```
unsigned int CurrentSetTime
```

Time in ms to reach the nominal current.

#### 5.47.2.2 CurrReductDelay

```
unsigned int CurrReductDelay
```

Time in ms from going to STOP state to the end of current reduction.

#### 5.47.2.3 HoldCurrent

```
unsigned int HoldCurrent
```

Holding current, as percent of the nominal current.

Range: 0..100.

#### 5.47.2.4 PowerFlags

```
unsigned int PowerFlags
```

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

5.47.2.5 `PowerOffDelay`

```
unsigned int PowerOffDelay
```

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

## 5.48 `secure_settings_t` Struct Reference

This structure contains protection parameters: critical electrical values, flags for protection algorithms.

### Data Fields

- unsigned int [LowUpwrOff](#)  
*Lower voltage limit to turn off the motor, in tens of mV.*
- unsigned int [Criticalpwr](#)  
*Maximum motor current which triggers ALARM state, in mA.*
- unsigned int [CriticalUpwr](#)  
*Maximum motor voltage which triggers ALARM state, in tens of mV.*
- unsigned int [CriticalT](#)  
*Maximum temperature, which triggers ALARM state, in tenths of degrees Celsius.*
- unsigned int [Criticalusb](#)  
*Deprecated.*
- unsigned int [CriticalUusb](#)  
*Deprecated.*
- unsigned int [MinimumUusb](#)  
*Deprecated.*
- unsigned int [Flags](#)  
*Flags of secure settings.*

### 5.48.1 Detailed Description

This structure contains protection parameters: critical electrical values, flags for protection algorithms.

See also

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

### 5.48.2 Field Documentation

5.48.2.1 `CriticalIpwr`

```
unsigned int CriticalIpwr
```

Maximum motor current which triggers ALARM state, in mA.

5.48.2.2 `CriticalIusb`

```
unsigned int CriticalIusb
```

Deprecated.

Maximum USB current which triggers ALARM state, in mA.

5.48.2.3 `CriticalT`

```
unsigned int CriticalT
```

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

5.48.2.4 `CriticalUpwr`

```
unsigned int CriticalUpwr
```

Maximum motor voltage which triggers ALARM state, in tens of mV.

5.48.2.5 `CriticalUusb`

```
unsigned int CriticalUusb
```

Deprecated.

Maximum USB voltage which triggers ALARM state, in tens of mV.

5.48.2.6 `Flags`

```
unsigned int Flags
```

[Flags of secure settings.](#)

## 5.48.2.7 LowUpwrOff

unsigned int LowUpwrOff

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

## 5.48.2.8 MinimumUusb

unsigned int MinimumUusb

Deprecated.

Minimum USB voltage which triggers ALARM state, in tens of mV.

## 5.49 serial\_number\_t Struct Reference

The structure contains a new serial number, hardware version, and valid key.

## Data Fields

- unsigned int [SN](#)  
*New board serial number.*
- uint8\_t [Key](#) [32]  
*Protection key (256 bit).*
- unsigned int [Major](#)  
*The major number of the hardware version.*
- unsigned int [Minor](#)  
*The minor number of the hardware version.*
- unsigned int [Release](#)  
*Number of edits this release of hardware.*

## 5.49.1 Detailed Description

The structure contains a new serial number, hardware version, and valid key.

The SN and hardware version are changed and saved when the transmitted key matches the stored key. It can be used by the manufacturer only. Used from the loader only.

See also

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

## 5.49.2 Field Documentation

## 5.49.2.1 Key

```
uint8_t Key[32]
```

Protection key (256 bit).

## 5.49.2.2 Major

```
unsigned int Major
```

The major number of the hardware version.

## 5.49.2.3 Minor

```
unsigned int Minor
```

The minor number of the hardware version.

## 5.49.2.4 Release

```
unsigned int Release
```

Number of edits this release of hardware.

## 5.49.2.5 SN

```
unsigned int SN
```

New board serial number.

5.50 `set_position_calb_t` Struct Reference

User unit position information.

## Data Fields

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

### 5.50.1 Detailed Description

User unit position information.

A useful structure that contains position value in steps and microsteps for stepper motor and encoder steps of all engines.

See also

[set\\_position](#)

### 5.50.2 Field Documentation

#### 5.50.2.1 `EncPosition`

`long_t EncPosition`

Encoder position.

#### 5.50.2.2 `PosFlags`

`unsigned int PosFlags`

[Position setting flags.](#)

#### 5.50.2.3 `Position`

`float Position`

The position in the engine.

## 5.51 `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.*

### 5.51.1 Detailed Description

Position information.

A useful structure that contains position value in steps and microsteps for stepper motor and encoder steps for all engines.

See also

[set\\_position](#)

### 5.51.2 Field Documentation

#### 5.51.2.1 `EncPosition`

`long_t` `EncPosition`

Encoder position.

#### 5.51.2.2 `PosFlags`

`unsigned int` `PosFlags`

Position setting flags.

## 5.51.2.3 uPosition

```
int uPosition
```

Microstep position is only used with stepper motors.

Microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine\_settings).

## 5.52 stage\_information\_t Struct Reference

Deprecated.

## Data Fields

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

## 5.52.1 Detailed Description

Deprecated.

Stage information.

See also

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

## 5.52.2 Field Documentation

## 5.52.2.1 Manufacturer

```
char Manufacturer[17]
```

Manufacturer.

Max string length: 16 chars.

## 5.52.2.2 PartNumber

```
char PartNumber[25]
```

Series and PartNumber.

Max string length: 24 chars.

## 5.53 stage\_name\_t Struct Reference

Stage username.

### Data Fields

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

### 5.53.1 Detailed Description

Stage username.

See also

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

### 5.53.2 Field Documentation

## 5.53.2.1 PositionerName

```
char PositionerName[17]
```

User's positioner name.

It can be set by a user. Max string length: 16 characters.

## 5.54 stage\_settings\_t Struct Reference

Deprecated.

## 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](#)  
*Maximum speed (Units/c).*
- float [TravelRange](#)  
*Travel range (Units).*
- float [SupplyVoltageMin](#)  
*Minimum supply voltage (V).*
- float [SupplyVoltageMax](#)  
*Maximum supply voltage (V).*
- float [MaxCurrentConsumption](#)  
*Maximum current consumption (A).*
- float [HorizontalLoadCapacity](#)  
*Horizontal load capacity (kg).*
- float [VerticalLoadCapacity](#)  
*Vertical load capacity (kg).*

## 5.54.1 Detailed Description

Deprecated.

Stage settings.

See also

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

## 5.54.2 Field Documentation

5.54.2.1 `HorizontalLoadCapacity`

`float HorizontalLoadCapacity`

Horizontal load capacity (kg).

Data type: float.

## 5.54.2.2 LeadScrewPitch

```
float LeadScrewPitch
```

Lead screw pitch (mm).

Data type: float.

## 5.54.2.3 MaxCurrentConsumption

```
float MaxCurrentConsumption
```

Maximum current consumption (A).

Data type: float.

## 5.54.2.4 MaxSpeed

```
float MaxSpeed
```

Maximum speed (Units/c).

Data type: float.

## 5.54.2.5 SupplyVoltageMax

```
float SupplyVoltageMax
```

Maximum supply voltage (V).

Data type: float.

## 5.54.2.6 SupplyVoltageMin

```
float SupplyVoltageMin
```

Minimum supply voltage (V).

Data type: float.

## 5.54.2.7 TravelRange

```
float TravelRange
```

Travel range (Units).

Data type: float.

## 5.54.2.8 Units

char Units[9]

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

Max string length: 8 chars.

## 5.54.2.9 VerticalLoadCapacity

float VerticalLoadCapacity

Vertical load capacity (kg).

Data type: float.

## 5.55 status\_calb\_t Struct Reference

User unit device's 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.*
- float [CurPosition](#)  
*Current position.*
- long\_t [EncPosition](#)  
*Current encoder position.*
- float [CurSpeed](#)  
*Motor shaft speed.*
- int [Ipwr](#)  
*Engine current, mA.*
- int [Upwr](#)  
*Power supply voltage, tens of mV.*
- int [Iusb](#)  
*USB current, mA.*
- int [Uusb](#)  
*USB voltage, tens of mV.*
- int [CurT](#)  
*Temperature, tenths of degrees Celsius.*
- unsigned int [Flags](#)  
*Status flags.*
- unsigned int [GPIOFlags](#)  
*Status flags of the GPIO outputs.*
- unsigned int [CmdBufFreeSpace](#)  
*This field is a service field.*

### 5.55.1 Detailed Description

User unit device's state.

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

See also

`get_status_impl`

### 5.55.2 Field Documentation

#### 5.55.2.1 CmdBufFreeSpace

```
unsigned int CmdBufFreeSpace
```

This field is a service field.

It shows the number of free synchronization chain buffer cells.

#### 5.55.2.2 CurPosition

```
float CurPosition
```

Current position.

Corrected by the table.

#### 5.55.2.3 CurSpeed

```
float CurSpeed
```

Motor shaft speed.

#### 5.55.2.4 CurT

```
int CurT
```

Temperature, tenths of degrees Celsius.

## 5.55.2.5 EncPosition

`long_t EncPosition`

Current encoder position.

## 5.55.2.6 EncSts

`unsigned int EncSts`

Encoder state.

## 5.55.2.7 Flags

`unsigned int Flags`

Status flags.

## 5.55.2.8 GPIOFlags

`unsigned int GPIOFlags`

Status flags of the GPIO outputs.

## 5.55.2.9 Ipwr

`int Ipwr`

Engine current, mA.

## 5.55.2.10 Iusb

`int Iusb`

USB current, mA.

## 5.55.2.11 MoveSts

unsigned int MoveSts

Flags of move state.

## 5.55.2.12 MvCmdSts

unsigned int MvCmdSts

Move command state.

## 5.55.2.13 PWRSts

unsigned int PWRSts

Flags of power state of stepper motor.

## 5.55.2.14 Upwr

int Upwr

Power supply voltage, tens of mV.

## 5.55.2.15 Uusb

int Uusb

USB voltage, tens of mV.

## 5.55.2.16 WindSts

unsigned int WindSts

Winding state.

## 5.56 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 microsteps.*
- long\_t [EncPosition](#)  
*Current encoder position.*
- int [CurSpeed](#)  
*Motor shaft speed in steps/s or rpm.*
- int [uCurSpeed](#)  
*Fractional part of motor shaft speed in microsteps.*
- int [Ipwr](#)  
*Engine current, mA.*
- int [Upwr](#)  
*Power supply voltage, tens of mV.*
- int [Iusb](#)  
*USB current, mA.*
- int [Uusb](#)  
*USB voltage, tens of mV.*
- int [CurT](#)  
*Temperature, tenths of degrees Celsius.*
- unsigned int [Flags](#)  
*Status flags.*
- unsigned int [GPIOFlags](#)  
*Status flags of the GPIO outputs.*
- unsigned int [CmdBufFreeSpace](#)  
*This field is a service field.*

### 5.56.1 Detailed Description

Device state.

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

See also

`get_status_impl`

## 5.56.2 Field Documentation

### 5.56.2.1 CmdBufFreeSpace

`unsigned int CmdBufFreeSpace`

This field is a service field.

It shows the number of free synchronization chain buffer cells.

### 5.56.2.2 CurPosition

`int CurPosition`

Current position.

### 5.56.2.3 CurSpeed

`int CurSpeed`

Motor shaft speed in steps/s or rpm.

### 5.56.2.4 CurT

`int CurT`

Temperature, tenths of degrees Celsius.

### 5.56.2.5 EncPosition

`long_t EncPosition`

Current encoder position.

## 5.56.2.6 EncSts

unsigned int EncSts

[Encoder state.](#)

## 5.56.2.7 Flags

unsigned int Flags

[Status flags.](#)

## 5.56.2.8 GPIOFlags

unsigned int GPIOFlags

[Status flags of the GPIO outputs.](#)

## 5.56.2.9 Ipwr

int Ipwr

Engine current, mA.

## 5.56.2.10 Iusb

int Iusb

USB current, mA.

## 5.56.2.11 MoveSts

unsigned int MoveSts

[Flags of move state.](#)

## 5.56.2.12 MvCmdSts

```
unsigned int MvCmdSts
```

Move command state.

## 5.56.2.13 PWRSts

```
unsigned int PWRSts
```

Flags of power state of stepper motor.

## 5.56.2.14 uCurPosition

```
int uCurPosition
```

Step motor shaft position in microsteps.

The microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine\_settings). Used with stepper motors only.

## 5.56.2.15 uCurSpeed

```
int uCurSpeed
```

Fractional part of motor shaft speed in microsteps.

The microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine\_settings). Used with stepper motors only.

## 5.56.2.16 Upwr

```
int Upwr
```

Power supply voltage, tens of mV.

## 5.56.2.17 Uusb

```
int Uusb
```

USB voltage, tens of mV.

5.56.2.18 WindSts

unsigned int WindSts

[Winding state](#).

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

User unit synchronization settings.

### Data Fields

- unsigned int [SyncInFlags](#)  
*Flags for synchronization input setup.*
- unsigned int [ClutterTime](#)  
*Input synchronization pulse dead time (us).*
- float [Position](#)  
*Desired position or shift.*
- float [Speed](#)  
*Target speed.*
- unsigned int **reserved0**

### 5.57.1 Detailed Description

User unit synchronization settings.

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

See also

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

### 5.57.2 Field Documentation

#### 5.57.2.1 ClutterTime

unsigned int ClutterTime

Input synchronization pulse dead time (us).

## 5.57.2.2 Position

float Position

Desired position or shift.

## 5.57.2.3 Speed

float Speed

Target speed.

## 5.57.2.4 SyncInFlags

unsigned int SyncInFlags

Flags for synchronization input setup.

## 5.58 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 (us).*
- int [Position](#)  
*Desired position or shift (full 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.*
- unsigned int **reserved0**

### 5.58.1 Detailed Description

Synchronization settings.

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

See also

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

### 5.58.2 Field Documentation

#### 5.58.2.1 ClutterTime

`unsigned int ClutterTime`

Input synchronization pulse dead time (us).

#### 5.58.2.2 Speed

`unsigned int Speed`

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

Range: 0..100000.

#### 5.58.2.3 SyncInFlags

`unsigned int SyncInFlags`

[Flags for synchronization input setup.](#)

#### 5.58.2.4 uPosition

`int uPosition`

The fractional part of a position or shift in microsteps.

It is used with a stepper motor. The microstep size and the range of valid values for this field depend on the selected step division mode (see the `MicrostepMode` field in `engine_settings`).

## 5.58.2.5 uSpeed

unsigned int uSpeed

Target speed in microsteps/s.

Microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine\_settings). Used a stepper motor only.

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

Synchronization settings which use user units.

## Data Fields

- unsigned int [SyncOutFlags](#)  
*Flags of synchronization output.*
- unsigned int [SyncOutPulseSteps](#)  
*This value specifies the duration of output pulse.*
- unsigned int [SyncOutPeriod](#)  
*This value specifies the 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, every point in which is treated as the target position.*

## 5.59.1 Detailed Description

Synchronization settings which use user units.

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

See also

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

## 5.59.2 Field Documentation

## 5.59.2.1 Accuracy

float Accuracy

This is the neighborhood around the target coordinates, every point in which is treated as the target position. Getting in these points cause the stop impulse.

## 5.59.2.2 SyncOutFlags

unsigned int SyncOutFlags

Flags of synchronization output.

## 5.59.2.3 SyncOutPeriod

unsigned int SyncOutPeriod

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

## 5.59.2.4 SyncOutPulseSteps

unsigned int SyncOutPulseSteps

This value specifies the 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.

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

Synchronization settings.

## Data Fields

- unsigned int [SyncOutFlags](#)  
*Flags of synchronization output.*
- unsigned int [SyncOutPulseSteps](#)  
*This value specifies the duration of output pulse.*
- unsigned int [SyncOutPeriod](#)  
*This value specifies the 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, every point in which is treated as the target position.*
- unsigned int [uAccuracy](#)  
*This is the neighborhood around the target coordinates in microsteps (used with a stepper motor only).*

### 5.60.1 Detailed Description

Synchronization settings.

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

See also

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

### 5.60.2 Field Documentation

#### 5.60.2.1 Accuracy

`unsigned int Accuracy`

This is the neighborhood around the target coordinates, every point in which is treated as the target position.

Getting in these points cause the stop impulse.

#### 5.60.2.2 SyncOutFlags

`unsigned int SyncOutFlags`

Flags of synchronization output.

#### 5.60.2.3 SyncOutPeriod

`unsigned int SyncOutPeriod`

This value specifies the number of encoder pulses or steps between two output synchronization pulses when `SYNCOUT_ONPERIOD` is set.

#### 5.60.2.4 SyncOutPulseSteps

`unsigned int SyncOutPulseSteps`

This value specifies the 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.

5.60.2.5 `uAccuracy`

```
unsigned int uAccuracy
```

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

The microstep size and the range of valid values for this field depend on the selected step division mode (see the `MicrostepMode` field in `engine_settings`).

## 5.61 `uart_settings_t` Struct Reference

UART settings.

### Data Fields

- unsigned int [Speed](#)  
*UART baudrate (in bauds)*
- unsigned int [UARTSetupFlags](#)  
*UART parity flags.*

### 5.61.1 Detailed Description

UART settings.

This structure contains UART settings.

See also

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

### 5.61.2 Field Documentation

#### 5.61.2.1 `UARTSetupFlags`

```
unsigned int UARTSetupFlags
```

[UART parity flags.](#)

## Chapter 6

# File Documentation

### 6.1 ximc.h File Reference

Header file for libximc library.

#### Data Structures

- struct [calibration\\_t](#)  
*Calibration structure.*
- struct [device\\_network\\_information\\_t](#)  
*Device network information structure.*
- struct [feedback\\_settings\\_t](#)  
*Feedback settings.*
- struct [home\\_settings\\_t](#)  
*Position calibration settings.*
- struct [home\\_settings\\_calb\\_t](#)  
*Position calibration settings which use user units.*
- struct [move\\_settings\\_t](#)  
*Move settings.*
- struct [move\\_settings\\_calb\\_t](#)  
*User units move settings.*
- struct [engine\\_settings\\_t](#)  
*Movement limitations and settings related to the motor.*
- struct [engine\\_settings\\_calb\\_t](#)  
*Movement limitations and settings, related to the motor.*
- 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 protection parameters: critical electrical values, flags for protection algorithms.*
- struct [edges\\_settings\\_t](#)  
*Edges settings.*
- struct [edges\\_settings\\_calb\\_t](#)

- *User unit edges settings.*
- struct [pid\\_settings\\_t](#)
  - *PID settings.*
- struct [sync\\_in\\_settings\\_t](#)
  - *Synchronization settings.*
- struct [sync\\_in\\_settings\\_calb\\_t](#)
  - *User unit synchronization settings.*
- struct [sync\\_out\\_settings\\_t](#)
  - *Synchronization settings.*
- struct [sync\\_out\\_settings\\_calb\\_t](#)
  - *Synchronization settings which use user units.*
- struct [extio\\_settings\\_t](#)
  - *EXTIO settings.*
- struct [brake\\_settings\\_t](#)
  - *Brake settings.*
- struct [control\\_settings\\_t](#)
  - *Control settings.*
- struct [control\\_settings\\_calb\\_t](#)
  - *User unit control settings.*
- struct [joystick\\_settings\\_t](#)
  - *Joystick settings.*
- struct [ctp\\_settings\\_t](#)
  - *Control position settings (used with stepper motor only)*
- struct [uart\\_settings\\_t](#)
  - *UART settings.*
- struct [network\\_settings\\_t](#)
  - *Network settings.*
- struct [password\\_settings\\_t](#)
  - *The web-page user password.*
- struct [calibration\\_settings\\_t](#)
  - *Calibration settings.*
- struct [controller\\_name\\_t](#)
  - *Controller name and settings flags.*
- struct [nonvolatile\\_memory\\_t](#)
  - *Structure contains user data to save into the FRAM.*
- struct [emf\\_settings\\_t](#)
  - *EMF settings.*
- struct [engine\\_advanced\\_setup\\_t](#)
  - *EAS settings.*
- struct [extended\\_settings\\_t](#)
  - *EST settings This data is stored in the controller's flash memory.*
- struct [get\\_position\\_t](#)
  - *Position information.*
- struct [get\\_position\\_calb\\_t](#)
  - *Position information.*
- struct [set\\_position\\_t](#)
  - *Position information.*
- struct [set\\_position\\_calb\\_t](#)
  - *User unit position information.*

- struct [status\\_t](#)  
*Device state.*
- struct [status\\_calb\\_t](#)  
*User unit device's state.*
- struct [measurements\\_t](#)  
*The structure contains a sequence of measured axis motion parameters - velocities and position errors.*
- struct [chart\\_data\\_t](#)  
*Additional device state.*
- struct [device\\_information\\_t](#)  
*Controller information structure.*
- struct [serial\\_number\\_t](#)  
*The structure contains a new serial number, hardware version, and valid key.*
- struct [analog\\_data\\_t](#)  
*Analog data.*
- struct [debug\\_read\\_t](#)  
*Debug data.*
- struct [debug\\_write\\_t](#)  
*Debug data.*
- struct [stage\\_name\\_t](#)  
*Stage username.*
- struct [stage\\_information\\_t](#)  
*Deprecated.*
- struct [stage\\_settings\\_t](#)  
*Deprecated.*
- struct [motor\\_information\\_t](#)  
*Deprecated.*
- struct [motor\\_settings\\_t](#)  
*Deprecated.*
- struct [encoder\\_information\\_t](#)  
*Deprecated.*
- struct [encoder\\_settings\\_t](#)  
*Deprecated.*
- struct [hallsensor\\_information\\_t](#)  
*Deprecated.*
- struct [hallsensor\\_settings\\_t](#)  
*Deprecated.*
- struct [gear\\_information\\_t](#)  
*Deprecated.*
- struct [gear\\_settings\\_t](#)  
*Deprecated.*
- struct [accessories\\_settings\\_t](#)  
*Deprecated.*
- struct [init\\_random\\_t](#)  
*Random key.*
- struct [globally\\_unique\\_identifier\\_t](#)  
*Globally unique identifier.*

## Macros

- `#define XIMC_API`  
*Library import macro.*
- `#define XIMC_CALLCONV`  
*Library calling convention macros.*
- `#define XIMC_RETTYPE void*`  
*Thread return type.*
- `#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_nodvice -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

This is a bit mask for bitwise operations.

- `#define ENUMERATE_PROBE 0x01`  
*Check if a device with an OS name is a XIMC device.*
- `#define ENUMERATE_ALL_COM 0x02`  
*Check all COM devices.*
- `#define ENUMERATE_NETWORK 0x04`  
*Check network devices.*

## Flags of move state

*This is a bit mask for bitwise operations. Specify move states.*

See also

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

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

### Flags of internal controller settings

*This is a bit mask for bitwise operations.*

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

*This is a bit mask for bitwise operations. Specify power states.*

See also

[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 the maximum current driver can provide at this voltage.*

### Status flags

*This is a bit mask for bitwise operations. Controller flags returned by device query. Contains boolean part of controller state. May be combined with bitwise OR.*

See also

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

- #define [STATE\\_CONTR](#) 0x0000003F  
*Flags of controller states.*
- #define [STATE\\_ERRC](#) 0x00000001  
*Command error encountered.*
- #define [STATE\\_ERRD](#) 0x00000002  
*Data integrity error encountered.*

- #define [STATE\\_ERRV](#) 0x0000004  
*Value error encountered.*
- #define [STATE\\_EEPROM\\_CONNECTED](#) 0x0000010  
*EEPROM with settings is connected.*
- #define [STATE\\_IS\\_HOMED](#) 0x0000020  
*Calibration performed.*
- #define [STATE\\_SECUR](#) 0x1B3FFC0  
*Security flags.*
- #define [STATE\\_ALARM](#) 0x0000040  
*The controller is in an alarm state, indicating that something dangerous has happened.*
- #define [STATE\\_CTP\\_ERROR](#) 0x0000080  
*Control position error (is only used with stepper motor).*
- #define [STATE\\_POWER\\_OVERHEAT](#) 0x0000100  
*Power driver overheat.*
- #define [STATE\\_CONTROLLER\\_OVERHEAT](#) 0x0000200  
*Controller overheat.*
- #define [STATE\\_OVERLOAD\\_POWER\\_VOLTAGE](#) 0x0000400  
*Power voltage exceeds safe limit.*
- #define [STATE\\_OVERLOAD\\_POWER\\_CURRENT](#) 0x0000800  
*Power current exceeds safe limit.*
- #define [STATE\\_OVERLOAD\\_USB\\_VOLTAGE](#) 0x0001000  
*Deprecated.*
- #define [STATE\\_LOW\\_USB\\_VOLTAGE](#) 0x0002000  
*Deprecated.*
- #define [STATE\\_OVERLOAD\\_USB\\_CURRENT](#) 0x0004000  
*Deprecated.*
- #define [STATE\\_BORDERS\\_SWAP\\_MISSET](#) 0x0008000  
*Engine stuck at the wrong edge.*
- #define [STATE\\_LOW\\_POWER\\_VOLTAGE](#) 0x0010000  
*Power voltage is lower than Low Voltage Protection limit.*
- #define [STATE\\_H\\_BRIDGE\\_FAULT](#) 0x0020000  
*Signal from the driver that fault happened.*
- #define [STATE\\_WINDING\\_RES\\_MISMATCH](#) 0x0100000  
*The difference between winding resistances is too large.*
- #define [STATE\\_ENCODER\\_FAULT](#) 0x0200000  
*Signal from the encoder that fault happened.*
- #define [STATE\\_ENGINE\\_RESPONSE\\_ERROR](#) 0x0800000  
*Error response of the engine control action.*
- #define [STATE\\_EXTIO\\_ALARM](#) 0x1000000  
*The error is caused by the external EXTIO input signal.*

### Status flags of the GPIO outputs

This is a bit mask for bitwise operations. GPIO state flags returned by device query. Contains boolean part of controller state. May be combined with bitwise OR.

See also

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

- #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 output if the flag is set; otherwise, it works as input.*  
• #define [STATE.GPIO\\_LEVEL](#) 0x0020
- *State of external GPIO 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

*This is a bit mask for bitwise operations. Encoder state returned by device query.*

Note

*The ENCD flag is only active in the None and EMF modes. In other modes, such as Encoder and Encoder Mediated, it is not used, as these modes cannot operate without an encoder. After the controller is powered on, the flag will be in the unknown state — a movement is required to determine the position. As the movement occurs, the flag's value may change.*

See also

[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 other direction.*
- #define [ENC.STATE\\_OK](#) 0x04  
*Encoder is connected and working properly.*

### Winding state

*This is a bit mask for bitwise operations. Motor winding state returned by device query.*

See also

[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

This is a bit mask for bitwise operations. 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

[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 has finished with an error, 0 - move command has finished correctly).*
- #define [MVCMD\\_RUNNING](#) 0x80  
*Move command state (0 - move command has finished, 1 - move command is being executed).*

### Flags of the motion parameters

This is a bit mask for bitwise operations. Specify the motor shaft movement algorithm and list of limitations. Flags returned by the query of [get\\_move\\_settings](#).

See also

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

- #define [RPM\\_DIV\\_1000](#) 0x01  
*This flag indicates that the operating speed specified in the command is set in milliRPM.*

### Flags of engine settings

This is a bit mask for bitwise operations. Specify the motor shaft movement algorithm and list of limitations. Flags returned by query of engine settings. May be combined with bitwise OR.

See also

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

- #define [ENGINE\\_REVERSE](#) 0x001  
*Reverse flag.*
- #define [ENGINE\\_CURRENT\\_AS\\_RMS](#) 0x002  
*Engine current meaning flag.*
- #define [ENGINE\\_MAX\\_SPEED](#) 0x004  
*Max speed flag.*
- #define [ENGINE\\_ANTIPLAY](#) 0x008  
*Play compensation flag.*
- #define [ENGINE\\_ACCEL\\_ON](#) 0x010  
*Acceleration enable flag.*
- #define [ENGINE\\_LIMIT\\_VOLT](#) 0x020  
*Maximum motor voltage limit enable flag (is only used with DC motor).*
- #define [ENGINE\\_LIMIT\\_CURR](#) 0x040  
*Maximum motor current limit enable flag (is only used with DC motor).*
- #define [ENGINE\\_LIMIT\\_RPM](#) 0x080  
*Maximum motor speed limit enable flag.*
- #define [ENGINE\\_USE\\_PEAK\\_CURRENT](#) 0x100  
*Enable peak current usage.*

### Flags of microstep mode

This is a bit mask for bitwise operations. Specify settings for microstep mode. Used 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

*This is a bit mask for bitwise operations. 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

*This is a bit mask for bitwise operations. 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\\_INTEGRATE](#) 0x02  
*Driver with integrated IC.*
- #define [DRIVER\\_TYPE\\_EXTERNAL](#) 0x03  
*External driver.*

### Flags of power settings of stepper motor

*This is a bit mask for bitwise operations. Flags returned by query of engine settings. Specify power settings. Flags returned by query of power settings.*

See also

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

- #define [POWER\\_REDUCT\\_ENABLED](#) 0x01  
*Current reduction is enabled after CurrReductDelay if this flag is set.*
- #define [POWER\\_OFF\\_ENABLED](#) 0x02  
*Power off is enabled after PowerOffDelay if this flag is set.*
- #define [POWER\\_SMOOTH\\_CURRENT](#) 0x04  
*Current ramp-up/down are performed smoothly during current\_set\_time if this flag is set.*

### Flags of secure settings

This is a bit mask for bitwise operations. Flags returned by query of engine settings. Specify secure settings. Flags returned by query of secure settings.

See also

[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 the alarm state on the driver overheat signal.*
- #define [LOW\\_UPWR\\_PROTECTION](#) 0x02  
*If this flag is set, turn off the motor when the voltage is lower than LowUpwrOff.*
- #define [H\\_BRIDGE\\_ALERT](#) 0x04  
*If this flag is set, then turn off the power unit when there is a signal problem in one of the transistor bridges.*
- #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 [BRAKING\\_OVERVOLTAGE\\_PROTECTION](#) 0x20  
*If this flag is set, the firmware will close ground switches of H-bridge to disconnect motor from power circuit on overvoltage.*
- #define [ALARM\\_WINDING\\_MISMATCH](#) 0x40  
*If this flag is set, enter Alarm state when windings mismatch.*
- #define [ALARM\\_ENGINE\\_RESPONSE](#) 0x80  
*If this flag is set, enter the Alarm state on response of the engine control action.*

### Position setting flags

This is a bit mask for bitwise operations. Flags used in setting 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.

This is a bit mask for bitwise operations.

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\\_EMF](#) 0x04  
*Feedback by EMF.*
- #define [FEEDBACK\\_NONE](#) 0x05  
*Feedback is absent.*
- #define [FEEDBACK\\_ENCODER\\_MEDIATED](#) 0x06  
*Feedback by encoder mediated by mechanical transmission (for example leadscrew).*

### Describes feedback flags.

*This is a bit mask for bitwise operations.*

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\\_ENC\\_ADAPTIVE\\_HOLDING](#) 0x02  
*Enables the adaptive holding algorithm.*
- #define [FEEDBACK\\_ENC\\_FILTER\\_NONE](#) 0x00  
*Disable the internal filter of the encoder signal.*
- #define [FEEDBACK\\_ENC\\_FILTER\\_WEAK](#) 0x10  
*Weak noise filtering: the maximum encoder signal frequency is 3 MHz.*
- #define [FEEDBACK\\_ENC\\_FILTER\\_MEDIUM](#) 0x20  
*Medium noise filtering: the maximum encoder signal frequency is 1 MHz.*
- #define [FEEDBACK\\_ENC\\_FILTER\\_STRONG](#) 0x30  
*Strong noise filtering: the maximum encoder signal frequency is 300 kHz.*
- #define [FEEDBACK\\_ENC\\_FILTER\\_BITS](#) 0x30  
*Bits responsible for setting the internal filter of the encoder signal.*
- #define [FEEDBACK\\_ENC\\_TYPE\\_AUTO](#) 0x00  
*Auto detect encoder type.*
- #define [FEEDBACK\\_ENC\\_TYPE\\_SINGLE\\_ENDED](#) 0x40  
*Single-ended encoder.*
- #define [FEEDBACK\\_ENC\\_TYPE\\_DIFFERENTIAL](#) 0x80  
*Differential encoder.*
- #define [FEEDBACK\\_ENC\\_TYPE\\_BITS](#) 0xC0  
*Bits of the encoder type.*

### Flags for synchronization input setup

*This is a bit mask for bitwise operations.*

See also

[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 the flag is set, on rising edge otherwise.*
- #define [SYNCIN\\_GOTOPOSITION](#) 0x04  
*The engine is going to the position specified in Position and uPosition if this flag is set.*

### Flags of synchronization output

*This is a bit mask for bitwise operations.*

See also

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

- #define [SYNCOUT\\_ENABLED](#) 0x01  
*The synchronization out pin follows the synchronization logic if the flag is set.*
- #define [SYNCOUT\\_STATE](#) 0x02  
*When the output state is fixed by the negative SYNCOUT\_ENABLED flag, the pin state is in accordance with this flag state.*
- #define [SYNCOUT\\_INVERT](#) 0x04  
*The low level is active if the flag is set.*
- #define [SYNCOUT\\_IN\\_STEPS](#) 0x08  
*Use motor steps or encoder pulses instead of milliseconds for output pulse generation if the flag is set.*
- #define [SYNCOUT\\_ONSTART](#) 0x10  
*Generate a synchronization pulse when movement starts.*
- #define [SYNCOUT\\_ONSTOP](#) 0x20  
*Generate a synchronization pulse when movement stops.*
- #define [SYNCOUT\\_ONPERIOD](#) 0x40  
*Generate a synchronization pulse every SyncOutPeriod encoder pulses.*

### External IO setup flags

*This is a bit mask for bitwise operations.*

See also

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

### External IO mode flags

*This is a bit mask for bitwise operations.*

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\\_BITS](#) 0x0F  
*Bits of the behavior selector when the signal on input goes to the active state.*
- #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\\_IN\\_ALARM](#) 0x05  
*Set Alarm when the signal goes to the active state.*
- #define [EXTIO\\_SETUP\\_MODE\\_OUT\\_BITS](#) 0xF0  
*Bits of the output behavior selection.*
- #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 the alarm state.*
- #define [EXTIO\\_SETUP\\_MODE\\_OUT\\_MOTOR\\_ON](#) 0x40  
*EXTIO pin stays active when windings are powered.*

### Border flags

This is a bit mask for bitwise operations. Specify types of borders and motor behavior 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 are placed on limit switches, if not set.*
- #define [BORDER\\_STOP\\_LEFT](#) 0x02  
*The motor should stop on the left border.*
- #define [BORDER\\_STOP\\_RIGHT](#) 0x04  
*The motor should stop on the right border.*
- #define [BORDERS\\_SWAP\\_MISSET\\_DETECTION](#) 0x08  
*The motor should stop on both borders.*

### Limit switches flags

This is a bit mask for bitwise operations. Specify electrical behavior 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

This is a bit mask for bitwise operations. Specify behavior 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 the stepper motor power off if this flag is set.*

### Control flags

This is a bit mask for bitwise operations. 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 the 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 the 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

This is a bit mask for bitwise operations. 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

This is a bit mask for bitwise operations. Specify control position settings. 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  
*The position control is enabled if the flag is set.*
- #define [CTP\\_BASE](#) 0x02  
*The position control is based on the revolution sensor if this flag is set; otherwise, it is based on the encoder.*
- #define [CTP\\_ALARM\\_ON\\_ERROR](#) 0x04  
*Set ALARM on mismatch if the flag is set.*
- #define [REV\\_SENS\\_INV](#) 0x08  
*Typically, the sensor is active when it is at 0, and inversion makes active at 1.*
- #define [CTP\\_ERROR\\_CORRECTION](#) 0x10  
*Correct errors that appear when slippage occurs if the flag is set.*

### Home settings flags

This is a bit mask for bitwise operations. Specify home command behavior. May be combined with bitwise OR.

See also

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

- #define [HOME\\_DIR\\_FIRST](#) 0x001  
*The flag defines the direction of the 1st motion after execution of the home command.*
- #define [HOME\\_DIR\\_SECOND](#) 0x002  
*The flag defines the direction of the 2nd motion.*
- #define [HOME\\_MV\\_SEC\\_EN](#) 0x004  
*Use the second phase of calibration to the home position, if set; otherwise the second phase is skipped.*
- #define [HOME\\_HALF\\_MV](#) 0x008  
*If the flag is set, the stop signals are ignored during the first half-turn of the second movement.*
- #define [HOME\\_STOP\\_FIRST\\_BITS](#) 0x030  
*Bits of the first stop selector.*
- #define [HOME\\_STOP\\_FIRST\\_REV](#) 0x010  
*First motion stops by revolution sensor.*
- #define [HOME\\_STOP\\_FIRST\\_SYN](#) 0x020  
*First motion stops by synchronization input.*
- #define [HOME\\_STOP\\_FIRST\\_LIM](#) 0x030  
*First motion stops by limit switch.*

- #define [HOME\\_STOP\\_SECOND\\_BITS](#) 0x0C0  
*Bits of the second stop selector.*
- #define [HOME\\_STOP\\_SECOND\\_REV](#) 0x040  
*Second motion stops by revolution sensor.*
- #define [HOME\\_STOP\\_SECOND\\_SYN](#) 0x080  
*Second motion stops by synchronization input.*
- #define [HOME\\_STOP\\_SECOND\\_LIM](#) 0x0C0  
*Second motion stops by limit switch.*
- #define [HOME\\_USE\\_FAST](#) 0x100  
*Use the fast algorithm of calibration to the home position, if set; otherwise the traditional algorithm.*

### UART parity flags

*This is a bit mask for bitwise operations.*

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

*This is a bit mask for bitwise operations.*

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

*This is a bit mask for bitwise operations.*

See also

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

- #define [ENCSET\\_DIFFERENTIAL\\_OUTPUT](#) 0x001  
*If the flag is set, the encoder has differential output, otherwise single-ended output.*
- #define [ENCSET\\_PUSHPULL\\_OUTPUT](#) 0x004  
*If the flag is set the encoder has push-pull output, otherwise open drain output.*
- #define [ENCSET\\_INDEXCHANNEL\\_PRESENT](#) 0x010  
*If the flag is set, the encoder has an extra indexed channel.*
- #define [ENCSET\\_REVOLUTIONSENSOR\\_PRESENT](#) 0x040  
*If the flag is set, the encoder has the revolution sensor.*
- #define [ENCSET\\_REVOLUTIONSENSOR\\_ACTIVE\\_HIGH](#) 0x100  
*If the flag is set, the revolution sensor's active state is high logic state; otherwise, the active state is low logic state.*

### Magnetic brake settings flags

*This is a bit mask for bitwise operations.*

See also

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

- #define [MB\\_AVAILABLE](#) 0x01  
*If the 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

*This is a bit mask for bitwise operations.*

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

### Flags of auto-detection of characteristics of windings of the engine.

*This is a bit mask for bitwise operations.*

See also

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

- #define [BACK\\_EMF\\_INDUCTANCE\\_AUTO](#) 0x01  
*Flag of auto-detection of inductance of windings of the engine.*
- #define [BACK\\_EMF\\_RESISTANCE\\_AUTO](#) 0x02  
*Flag of auto-detection of resistance of windings of the engine.*
- #define [BACK\\_EMF\\_KM\\_AUTO](#) 0x04  
*Flag of auto-detection of electromechanical coefficient of the engine.*

## 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 structure.*
- typedef struct [device\\_network\\_information\\_t](#) [device\\_network\\_information\\_t](#)  
*Device network information structure.*

## Functions

### Controller settings setup

Read and write functions for almost all controller settings.

- [result\\_t](#) [XIMC\\_API](#) [set\\_feedback\\_settings](#) ([device\\_t](#) id, const [feedback\\_settings\\_t](#) \*feedback\_settings)  
*Feedback settings.*
- [result\\_t](#) [XIMC\\_API](#) [get\\_feedback\\_settings](#) ([device\\_t](#) id, [feedback\\_settings\\_t](#) \*feedback\_settings)  
*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)  
*Set user unit home settings.*
- [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)  
*Read user unit home settings.*
- [result\\_t](#) [XIMC\\_API](#) [set\\_move\\_settings](#) ([device\\_t](#) id, const [move\\_settings\\_t](#) \*move\_settings)  
*Movement settings set command (speed, acceleration, threshold, etc.).*
- [result\\_t](#) [XIMC\\_API](#) [set\\_move\\_settings\\_calb](#) ([device\\_t](#) id, const [move\\_settings\\_calb\\_t](#) \*move\_settings\_calb, const [calibration\\_t](#) \*calibration)  
*User unit movement settings set command (speed, acceleration, threshold, etc.).*
- [result\\_t](#) [XIMC\\_API](#) [get\\_move\\_settings](#) ([device\\_t](#) id, [move\\_settings\\_t](#) \*move\_settings)

- Movement settings read command (speed, acceleration, threshold, etc.).*
- [result\\_t XIMC\\_API get\\_move\\_settings\\_calb](#) ([device\\_t](#) id, [move\\_settings\\_calb\\_t](#) \*move\_settings\_calb, [const calibration\\_t](#) \*calibration)
- User unit movement settings read command (speed, acceleration, threshold, etc.).*
- [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)
- Set user unit engine settings.*
- [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)
- Read user unit engine settings.*
- [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)
- Set border and limit switches settings in user units.*
- [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)
- Read border and limit switches settings in user units.*
- [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)
- Set input user unit synchronization settings.*
- [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)
- Read input user unit synchronization settings.*
- [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)
- Set output user unit synchronization settings.*
- [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)  
*Read output user unit synchronization settings.*
- [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 brake control settings.*
- [result\\_t XIMC\\_API get\\_brake\\_settings](#) ([device\\_t](#) id, [brake\\_settings\\_t](#) \*brake\_settings)  
*Read break control settings.*
- [result\\_t XIMC\\_API set\\_control\\_settings](#) ([device\\_t](#) id, const [control\\_settings\\_t](#) \*control\_settings)  
*Set motor control settings.*
- [result\\_t XIMC\\_API set\\_control\\_settings\\_calb](#) ([device\\_t](#) id, const [control\\_settings\\_calb\\_t](#) \*control\_↵  
settings\_calb, const [calibration\\_t](#) \*calibration)  
*Set motor control settings with user units.*
- [result\\_t XIMC\\_API get\\_control\\_settings](#) ([device\\_t](#) id, [control\\_settings\\_t](#) \*control\_settings)  
*Read motor control settings.*
- [result\\_t XIMC\\_API get\\_control\\_settings\\_calb](#) ([device\\_t](#) id, [control\\_settings\\_calb\\_t](#) \*control\_settings\_↵  
\_calb, const [calibration\\_t](#) \*calibration)  
*Read motor control settings with user units.*
- [result\\_t XIMC\\_API set\\_joystick\\_settings](#) ([device\\_t](#) id, const [joystick\\_settings\\_t](#) \*joystick\_settings)  
*Set joystick position.*
- [result\\_t XIMC\\_API get\\_joystick\\_settings](#) ([device\\_t](#) id, [joystick\\_settings\\_t](#) \*joystick\_settings)  
*Read joystick settings.*
- [result\\_t XIMC\\_API set\\_ctp\\_settings](#) ([device\\_t](#) id, const [ctp\\_settings\\_t](#) \*ctp\_settings)  
*Set control position settings (used with stepper motor only).*
- [result\\_t XIMC\\_API get\\_ctp\\_settings](#) ([device\\_t](#) id, [ctp\\_settings\\_t](#) \*ctp\_settings)  
*Read control position settings (used with stepper motor only).*
- [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\\_network\\_settings](#) ([device\\_t](#) id, const [network\\_settings\\_t](#) \*network\_settings)  
*Set network settings.*
- [result\\_t XIMC\\_API get\\_network\\_settings](#) ([device\\_t](#) id, [network\\_settings\\_t](#) \*network\_settings)  
*Read network settings.*
- [result\\_t XIMC\\_API set\\_password\\_settings](#) ([device\\_t](#) id, const [password\\_settings\\_t](#) \*password\_↵  
settings)  
*Sets the password.*
- [result\\_t XIMC\\_API get\\_password\\_settings](#) ([device\\_t](#) id, [password\\_settings\\_t](#) \*password\_settings)  
*Read the password.*
- [result\\_t XIMC\\_API set\\_calibration\\_settings](#) ([device\\_t](#) id, const [calibration\\_settings\\_t](#) \*calibration\_↵  
settings)  
*Set calibration settings.*
- [result\\_t XIMC\\_API get\\_calibration\\_settings](#) ([device\\_t](#) id, [calibration\\_settings\\_t](#) \*calibration\_settings)  
*Read calibration settings.*
- [result\\_t XIMC\\_API set\\_controller\\_name](#) ([device\\_t](#) id, const [controller\\_name\\_t](#) \*controller\_name)  
*Write user's controller name and internal settings to the FRAM.*
- [result\\_t XIMC\\_API get\\_controller\\_name](#) ([device\\_t](#) id, [controller\\_name\\_t](#) \*controller\_name)  
*Read user's controller name and internal settings from the FRAM.*
- [result\\_t XIMC\\_API set\\_nonvolatile\\_memory](#) ([device\\_t](#) id, const [nonvolatile\\_memory\\_t](#) \*nonvolatile\_↵  
\_memory)  
*Write user data into the FRAM.*
- [result\\_t XIMC\\_API get\\_nonvolatile\\_memory](#) ([device\\_t](#) id, [nonvolatile\\_memory\\_t](#) \*nonvolatile\_↵  
memory)  
*Read user data from FRAM.*

- [result\\_t XIMC\\_API set\\_emf\\_settings](#) ([device\\_t](#) id, const [emf\\_settings\\_t](#) \*emf\_settings)  
*Set electromechanical coefficients.*
- [result\\_t XIMC\\_API get\\_emf\\_settings](#) ([device\\_t](#) id, [emf\\_settings\\_t](#) \*emf\_settings)  
*Read electromechanical settings.*
- [result\\_t XIMC\\_API set\\_engine\\_advanced\\_setup](#) ([device\\_t](#) id, const [engine\\_advanced\\_setup\\_t](#) \*engine\_advanced\_setup)  
*Set engine advanced settings.*
- [result\\_t XIMC\\_API get\\_engine\\_advanced\\_setup](#) ([device\\_t](#) id, [engine\\_advanced\\_setup\\_t](#) \*engine\_advanced\_setup)  
*Read engine advanced settings.*
- [result\\_t XIMC\\_API set\\_extended\\_settings](#) ([device\\_t](#) id, const [extended\\_settings\\_t](#) \*extended\_settings)  
*Set extended settings.*
- [result\\_t XIMC\\_API get\\_extended\\_settings](#) ([device\\_t](#) id, [extended\\_settings\\_t](#) \*extended\_settings)  
*Read extended settings.*

### Group of commands movement control

- [result\\_t XIMC\\_API command\\_stop](#) ([device\\_t](#) id)  
*Immediately stops the engine, moves it to the STOP state, and sets switches to BREAK mode (windings are short-circuited).*
- [result\\_t XIMC\\_API command\\_power\\_off](#) ([device\\_t](#) id)  
*Immediately power off the motor regardless its state.*
- [result\\_t XIMC\\_API command\\_move](#) ([device\\_t](#) id, int Position, int uPosition)  
*Move to position.*
- [result\\_t XIMC\\_API command\\_move\\_calb](#) ([device\\_t](#) id, float Position, const [calibration\\_t](#) \*calibration)  
*Move to position using user units.*
- [result\\_t XIMC\\_API command\\_movr](#) ([device\\_t](#) id, int DeltaPosition, int uDeltaPosition)  
*Shift by a set offset.*
- [result\\_t XIMC\\_API command\\_movr\\_calb](#) ([device\\_t](#) id, float DeltaPosition, const [calibration\\_t](#) \*calibration)  
*Shift by a set offset using user units.*
- [result\\_t XIMC\\_API command\\_home](#) ([device\\_t](#) id)  
*Moving to home position.*
- [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 position to a distance Antiplay defined in engine settings.*
- [result\\_t XIMC\\_API command\\_sstp](#) ([device\\_t](#) id)  
*Soft stop the engine.*
- [result\\_t XIMC\\_API get\\_position](#) ([device\\_t](#) id, [get\\_position\\_t](#) \*the\_get\_position)  
*Reads the value position in steps and microsteps for stepper motor and encoder steps for all engines.*
- [result\\_t XIMC\\_API get\\_position\\_calb](#) ([device\\_t](#) id, [get\\_position\\_calb\\_t](#) \*the\_get\_position\_calb, const [calibration\\_t](#) \*calibration)  
*Reads position value in user units for stepper motor and encoder steps for all engines.*
- [result\\_t XIMC\\_API set\\_position](#) ([device\\_t](#) id, const [set\\_position\\_t](#) \*the\_set\_position)  
*Sets position in steps and microsteps for stepper motor.*
- [result\\_t XIMC\\_API set\\_position\\_calb](#) ([device\\_t](#) id, const [set\\_position\\_calb\\_t](#) \*the\_set\_position\_calb, const [calibration\\_t](#) \*calibration)  
*Sets any position value and encoder value of all engines.*
- [result\\_t XIMC\\_API command\\_zero](#) ([device\\_t](#) id)  
*Sets the current position to 0.*

### Group of save settings and load settings commands

- [result\\_t XIMC\\_API command\\_save\\_settings \(device\\_t id\)](#)  
*Save all settings from the controller's RAM to the controller's flash memory, replacing previous data in the flash memory.*
- [result\\_t XIMC\\_API command\\_read\\_settings \(device\\_t id\)](#)  
*Read all settings from the controller's flash memory to the controller's RAM, replacing previous data in the RAM.*
- [result\\_t XIMC\\_API command\\_save\\_robust\\_settings \(device\\_t id\)](#)  
*Save important settings (calibration coefficients, etc.) from the controller's RAM to the controller's flash memory, replacing previous data in the flash memory.*
- [result\\_t XIMC\\_API command\\_read\\_robust\\_settings \(device\\_t id\)](#)  
*Read important settings (calibration coefficients, etc.) from the controller's flash memory to the controller's RAM, replacing previous data in the RAM.*
- [result\\_t XIMC\\_API command\\_eesave\\_settings \(device\\_t id\)](#)  
*Save settings from the controller's RAM to the stage's EEPROM.*
- [result\\_t XIMC\\_API command\\_eeread\\_settings \(device\\_t id\)](#)  
*Read settings from the stage's EEPROM to the controller's RAM.*
- [result\\_t XIMC\\_API command\\_start\\_measurements \(device\\_t id\)](#)  
*Start measurements and buffering of speed and the speed error (target speed minus real speed).*
- [result\\_t XIMC\\_API get\\_measurements \(device\\_t id, measurements\\_t \\*measurements\)](#)  
*A command to read the data buffer to build a speed graph and a speed error graph.*
- [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 the controller's firmware version.*
- [result\\_t XIMC\\_API service\\_command\\_updf \(device\\_t id\)](#)  
*The command switches the controller to update the firmware state.*

### 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 the controller's flash memory.*
- [result\\_t XIMC\\_API get\\_analog\\_data \(device\\_t id, analog\\_data\\_t \\*analog\\_data\)](#)  
*Read the analog data structure that contains raw analog data from the embedded ADC.*
- [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.*

### A group of EEPROM commands

- [result\\_t XIMC\\_API set\\_stage\\_name \(device\\_t id, const stage\\_name\\_t \\*stage\\_name\)](#)  
*Write the user's stage name to EEPROM.*
- [result\\_t XIMC\\_API get\\_stage\\_name \(device\\_t id, stage\\_name\\_t \\*stage\\_name\)](#)  
*Read the user's stage name from the EEPROM.*
- [result\\_t XIMC\\_API set\\_stage\\_information \(device\\_t id, const stage\\_information\\_t \\*stage\\_↵ information\)](#)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_stage\\_information \(device\\_t id, stage\\_information\\_t \\*stage\\_information\)](#)  
*Deprecated.*
- [result\\_t XIMC\\_API set\\_stage\\_settings \(device\\_t id, const stage\\_settings\\_t \\*stage\\_settings\)](#)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_stage\\_settings \(device\\_t id, stage\\_settings\\_t \\*stage\\_settings\)](#)  
*Deprecated.*

- [result\\_t XIMC\\_API set\\_motor\\_information](#) ([device\\_t](#) id, const [motor\\_information\\_t](#) \*motor\_↵  
information)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_motor\\_information](#) ([device\\_t](#) id, [motor\\_information\\_t](#) \*motor\_information)  
*Deprecated.*
- [result\\_t XIMC\\_API set\\_motor\\_settings](#) ([device\\_t](#) id, const [motor\\_settings\\_t](#) \*motor\_settings)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_motor\\_settings](#) ([device\\_t](#) id, [motor\\_settings\\_t](#) \*motor\_settings)  
*Deprecated.*
- [result\\_t XIMC\\_API set\\_encoder\\_information](#) ([device\\_t](#) id, const [encoder\\_information\\_t](#) \*encoder\_↵  
information)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_encoder\\_information](#) ([device\\_t](#) id, [encoder\\_information\\_t](#) \*encoder\_↵  
information)  
*Deprecated.*
- [result\\_t XIMC\\_API set\\_encoder\\_settings](#) ([device\\_t](#) id, const [encoder\\_settings\\_t](#) \*encoder\_settings)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_encoder\\_settings](#) ([device\\_t](#) id, [encoder\\_settings\\_t](#) \*encoder\_settings)  
*Deprecated.*
- [result\\_t XIMC\\_API set\\_hallsensor\\_information](#) ([device\\_t](#) id, const [hallsensor\\_information\\_↵  
t](#) \*hallsensor\_information)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_hallsensor\\_information](#) ([device\\_t](#) id, [hallsensor\\_information\\_t](#) \*hallsensor\_↵  
information)  
*Deprecated.*
- [result\\_t XIMC\\_API set\\_hallsensor\\_settings](#) ([device\\_t](#) id, const [hallsensor\\_settings\\_t](#) \*hallsensor\_↵  
settings)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_hallsensor\\_settings](#) ([device\\_t](#) id, [hallsensor\\_settings\\_t](#) \*hallsensor\_settings)  
*Deprecated.*
- [result\\_t XIMC\\_API set\\_gear\\_information](#) ([device\\_t](#) id, const [gear\\_information\\_t](#) \*gear\_information)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_gear\\_information](#) ([device\\_t](#) id, [gear\\_information\\_t](#) \*gear\_information)  
*Deprecated.*
- [result\\_t XIMC\\_API set\\_gear\\_settings](#) ([device\\_t](#) id, const [gear\\_settings\\_t](#) \*gear\_settings)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_gear\\_settings](#) ([device\\_t](#) id, [gear\\_settings\\_t](#) \*gear\_settings)  
*Deprecated.*
- [result\\_t XIMC\\_API set\\_accessories\\_settings](#) ([device\\_t](#) id, const [accessories\\_settings\\_t](#) \*accessories\_↵  
\_settings)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_accessories\\_settings](#) ([device\\_t](#) id, [accessories\\_settings\\_t](#) \*accessories\_↵  
settings)  
*Deprecated.*
- [result\\_t XIMC\\_API get\\_bootloader\\_version](#) ([device\\_t](#) id, unsigned int \*Major, unsigned int \*Minor,  
unsigned int \*Release)  
*Read the controller's bootloader version.*
- [result\\_t XIMC\\_API get\\_init\\_random](#) ([device\\_t](#) id, [init\\_random\\_t](#) \*init\_random)  
*Read a random number from the controller.*
- [result\\_t XIMC\\_API get\\_globally\\_unique\\_identifier](#) ([device\\_t](#) id, [globally\\_unique\\_identifier\\_t](#) \*globally\_↵  
\_unique\_identifier)  
*This value is unique to each individual device, but is not a random value.*
- [result\\_t XIMC\\_API goto\\_firmware](#) ([device\\_t](#) id, uint8\_t \*ret)  
*Reboot to firmware.*
- [result\\_t XIMC\\_API has\\_firmware](#) (const char \*uri, uint8\_t \*ret)  
*Check for firmware on device.*

- [result\\_t XIMC\\_API command\\_update\\_firmware](#) (const char \*uri, const uint8\_t \*data, uint32\_t data↔\_size)  
*Update firmware.*
- [result\\_t XIMC\\_API write\\_key](#) (const char \*uri, 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 \*uri)  
*Open a device with OS uri 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 set\\_correction\\_table](#) (device\_t id, const char \*namefile)  
*Command of loading a correction table from a text file.*
- [result\\_t XIMC\\_API probe\\_device](#) (const char \*uri)  
*Check if a device with OS uri uri is XIMC device.*
- [device\\_enumeration\\_t XIMC\\_API enumerate\\_devices](#) (int enumerate\_flags, const char \*hints)  
*Search and list of available devices.*
- [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)  
*Get controller name from the device enumeration.*
- [result\\_t XIMC\\_API get\\_enumerate\\_device\\_stage\\_name](#) (device\_enumeration\_t device\_enumeration, int device\_index, [stage\\_name\\_t](#) \*stage\_name)  
*Get stage name from the device enumeration.*
- [result\\_t XIMC\\_API get\\_enumerate\\_device\\_network\\_information](#) (device\_enumeration\_t device↔enumeration, int device\_index, [device\\_network\\_information\\_t](#) \*device\_network\_information)  
*Get device network information from the device enumeration.*

- [result\\_t XIMC\\_API reset\\_locks](#) ()  
*Resets the error of incorrect data transmission.*
- 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)  
*Return device state.*
- [result\\_t XIMC\\_API get\\_device\\_information](#) ([device\\_t](#) id, [device\\_information\\_t](#) \*device\_information)  
*Return device information.*
- [result\\_t XIMC\\_API command\\_wait\\_for\\_stop](#) ([device\\_t](#) id, uint32\_t refresh\_interval\_ms)  
*Wait for stop.*
- [result\\_t XIMC\\_API command\\_homezero](#) ([device\\_t](#) id)  
*Make home command, wait until it is finished and make zero command.*

### 6.1.1 Detailed Description

Header file for libximc library.

### 6.1.2 Macro Definition Documentation

#### 6.1.2.1 ALARM\_FLAGS\_STICKING

```
#define ALARM_FLAGS_STICKING 0x10
```

If this flag is set, only a STOP command can turn all alarms to 0.

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

```
#define ALARM_ON_BORDERS_SWAP_MISSET 0x08
```

If this flag is set, enter Alarm state on borders swap misset.

## 6.1.2.3 ALARM\_ON\_DRIVER\_OVERHEATING

```
#define ALARM_ON_DRIVER_OVERHEATING 0x01
```

If this flag is set, enter the alarm state on the driver overheat signal.

## 6.1.2.4 BACK\_EMF\_INDUCTANCE\_AUTO

```
#define BACK_EMF_INDUCTANCE_AUTO 0x01
```

Flag of auto-detection of inductance of windings of the engine.

## 6.1.2.5 BACK\_EMF\_KM\_AUTO

```
#define BACK_EMF_KM_AUTO 0x04
```

Flag of auto-detection of electromechanical coefficient of the engine.

## 6.1.2.6 BACK\_EMF\_RESISTANCE\_AUTO

```
#define BACK_EMF_RESISTANCE_AUTO 0x02
```

Flag of auto-detection of resistance of windings of the engine.

## 6.1.2.7 BORDER\_IS\_ENCODER

```
#define BORDER_IS_ENCODER 0x01
```

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

## 6.1.2.8 BORDER\_STOP\_LEFT

```
#define BORDER_STOP_LEFT 0x02
```

The motor should stop on the left border.

## 6.1.2.9 BORDER\_STOP\_RIGHT

```
#define BORDER_STOP_RIGHT 0x04
```

The motor should stop on the right border.

## 6.1.2.10 BORDERS\_SWAP\_MISSET\_DETECTION

```
#define BORDERS_SWAP_MISSET_DETECTION 0x08
```

The motor should stop on both borders.

Protects the motor when wrong border settings are set.

## 6.1.2.11 BRAKE\_ENABLED

```
#define BRAKE_ENABLED 0x01
```

Brake control is enabled if this flag is set.

## 6.1.2.12 BRAKE\_ENG\_PWROFF

```
#define BRAKE_ENG_PWROFF 0x02
```

Brake turns the stepper motor power off if this flag is set.

## 6.1.2.13 BRAKING\_OVERVOLTAGE\_PROTECTION

```
#define BRAKING_OVERVOLTAGE_PROTECTION 0x20
```

If this flag is set, the firmware will close ground switches of H-bridge to disconnect motor from power circuit on overvoltage.

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

```
#define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04
```

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

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

```
#define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08
```

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

## 6.1.2.16 CONTROL\_MODE\_BITS

```
#define CONTROL_MODE_BITS 0x03
```

Bits to control the engine by joystick or buttons.

## 6.1.2.17 CONTROL\_MODE\_JOY

```
#define CONTROL_MODE_JOY 0x01
```

Control by joystick.

## 6.1.2.18 CONTROL\_MODE\_LR

```
#define CONTROL_MODE_LR 0x02
```

Control by left/right buttons.

## 6.1.2.19 CONTROL\_MODE\_OFF

```
#define CONTROL_MODE_OFF 0x00
```

Control is disabled.

## 6.1.2.20 CTP\_ALARM\_ON\_ERROR

```
#define CTP_ALARM_ON_ERROR 0x04
```

Set ALARM on mismatch if the flag is set.

## 6.1.2.21 CTP\_BASE

```
#define CTP_BASE 0x02
```

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

## 6.1.2.22 CTP\_ENABLED

```
#define CTP_ENABLED 0x01
```

The position control is enabled if the flag is set.

## 6.1.2.23 CTP\_ERROR\_CORRECTION

```
#define CTP_ERROR_CORRECTION 0x10
```

Correct errors that appear when slippage occurs if the flag is set.

It works only with the encoder. Incompatible with the flag CTP\_ALARM\_ON\_ERROR.

## 6.1.2.24 DRIVER\_TYPE\_EXTERNAL

```
#define DRIVER_TYPE_EXTERNAL 0x03
```

External driver.

Support for this feature depends on the modification of the controller.

## 6.1.2.25 DRIVER\_TYPE\_INTEGRATE

```
#define DRIVER_TYPE_INTEGRATE 0x02
```

Driver with integrated IC.

## 6.1.2.26 EEPROM\_PRECEDENCE

```
#define EEPROM_PRECEDENCE 0x01
```

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

## 6.1.2.27 ENC\_STATE\_ABSENT

```
#define ENC_STATE_ABSENT 0x00
```

Encoder is absent.

## 6.1.2.28 ENC\_STATE\_MALFUNC

```
#define ENC_STATE_MALFUNC 0x02
```

Encoder is connected and malfunctioning.

## 6.1.2.29 ENC\_STATE\_OK

```
#define ENC_STATE_OK 0x04
```

Encoder is connected and working properly.

## 6.1.2.30 ENC\_STATE\_REVERS

```
#define ENC_STATE_REVERS 0x03
```

Encoder is connected and operational but counts in other direction.

## 6.1.2.31 ENC\_STATE\_UNKNOWN

```
#define ENC_STATE_UNKNOWN 0x01
```

Encoder state is unknown.

## 6.1.2.32 ENDER\_SW1\_ACTIVE\_LOW

```
#define ENDER_SW1_ACTIVE_LOW 0x02
```

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

## 6.1.2.33 ENDER\_SW2\_ACTIVE\_LOW

```
#define ENDER_SW2_ACTIVE_LOW 0x04
```

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

## 6.1.2.34 ENDER\_SWAP

```
#define ENDER_SWAP 0x01
```

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

## 6.1.2.35 ENGINE\_ACCEL\_ON

```
#define ENGINE_ACCEL_ON 0x010
```

Acceleration enable flag.

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

## 6.1.2.36 ENGINE\_ANTIPLAY

```
#define ENGINE_ANTIPLAY 0x008
```

Play compensation flag.

If it is set, the engine makes backlash (play) compensation and reaches the predetermined position accurately at low speed.

## 6.1.2.37 ENGINE\_CURRENT\_AS\_RMS

```
#define ENGINE_CURRENT_AS_RMS 0x002
```

Engine current meaning flag.

If the flag is unset, then the engine's current value is interpreted as the maximum amplitude value. If the flag is set, then the engine current value is interpreted as the root-mean-square current value (for stepper) or as the current value calculated from the maximum heat dissipation (BLDC).

## 6.1.2.38 ENGINE\_LIMIT\_CURR

```
#define ENGINE_LIMIT_CURR 0x040
```

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

## 6.1.2.39 ENGINE\_LIMIT\_RPM

```
#define ENGINE_LIMIT_RPM 0x080
```

Maximum motor speed limit enable flag.

## 6.1.2.40 ENGINE\_LIMIT\_VOLT

```
#define ENGINE_LIMIT_VOLT 0x020
```

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

## 6.1.2.41 ENGINE\_MAX\_SPEED

```
#define ENGINE_MAX_SPEED 0x004
```

Max speed flag.

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

## 6.1.2.42 ENGINE\_REVERSE

```
#define ENGINE_REVERSE 0x001
```

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.

## 6.1.2.43 ENGINE\_TYPE\_2DC

```
#define ENGINE_TYPE_2DC 0x02
```

2 DC motors.

## 6.1.2.44 ENGINE\_TYPE\_BRUSHLESS

```
#define ENGINE_TYPE_BRUSHLESS 0x05
```

Brushless motor.

## 6.1.2.45 ENGINE\_TYPE\_DC

```
#define ENGINE_TYPE_DC 0x01
```

DC motor.

## 6.1.2.46 ENGINE\_TYPE\_NONE

```
#define ENGINE_TYPE_NONE 0x00
```

A value that shouldn't be used.

## 6.1.2.47 ENGINE\_TYPE\_STEP

```
#define ENGINE_TYPE_STEP 0x03
```

Step motor.

## 6.1.2.48 ENGINE\_TYPE\_TEST

```
#define ENGINE_TYPE_TEST 0x04
```

Duty cycle are fixed.

Used only manufacturer.

## 6.1.2.49 ENGINE\_USE\_PEAK\_CURRENT

```
#define ENGINE_USE_PEAK_CURRENT 0x100
```

Enable peak current usage.

## 6.1.2.50 ENUMERATE\_PROBE

```
#define ENUMERATE_PROBE 0x01
```

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

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

## 6.1.2.51 EXTIO\_SETUP\_INVERT

```
#define EXTIO_SETUP_INVERT 0x02
```

Interpret EXTIO state inverted if the flag is set.

A falling front is treated as an input event and a low logic level as an active state.

## 6.1.2.52 EXTIO\_SETUP\_MODE\_IN\_ALARM

```
#define EXTIO_SETUP_MODE_IN_ALARM 0x05
```

Set Alarm when the signal goes to the active state.

## 6.1.2.53 EXTIO\_SETUP\_MODE\_IN\_BITS

```
#define EXTIO_SETUP_MODE_IN_BITS 0x0F
```

Bits of the behavior selector when the signal on input goes to the active state.

## 6.1.2.54 EXTIO\_SETUP\_MODE\_IN\_HOME

```
#define EXTIO_SETUP_MODE_IN_HOME 0x04
```

Issue HOME command.

## 6.1.2.55 EXTIO\_SETUP\_MODE\_IN\_MOVR

```
#define EXTIO_SETUP_MODE_IN_MOVR 0x03
```

Issue MOVR command with last used settings.

## 6.1.2.56 EXTIO\_SETUP\_MODE\_IN\_NOP

```
#define EXTIO_SETUP_MODE_IN_NOP 0x00
```

Do nothing.

## 6.1.2.57 EXTIO\_SETUP\_MODE\_IN\_PWOF

```
#define EXTIO_SETUP_MODE_IN_PWOF 0x02
```

Issue PWOF command, powering off all engine windings.

## 6.1.2.58 EXTIO\_SETUP\_MODE\_IN\_STOP

```
#define EXTIO_SETUP_MODE_IN_STOP 0x01
```

Issue STOP command, ceasing the engine movement.

## 6.1.2.59 EXTIO\_SETUP\_MODE\_OUT\_ALARM

```
#define EXTIO_SETUP_MODE_OUT_ALARM 0x30
```

EXTIO pin stays active during the alarm state.

## 6.1.2.60 EXTIO\_SETUP\_MODE\_OUT\_BITS

```
#define EXTIO_SETUP_MODE_OUT_BITS 0xF0
```

Bits of the output behavior selection.

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

```
#define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40
```

EXTIO pin stays active when windings are powered.

## 6.1.2.62 EXTIO\_SETUP\_MODE\_OUT\_MOVING

```
#define EXTIO_SETUP_MODE_OUT_MOVING 0x20
```

EXTIO pin stays active during moving state.

## 6.1.2.63 EXTIO\_SETUP\_MODE\_OUT\_OFF

```
#define EXTIO_SETUP_MODE_OUT_OFF 0x00
```

EXTIO pin always set in inactive state.

## 6.1.2.64 EXTIO\_SETUP\_MODE\_OUT\_ON

```
#define EXTIO_SETUP_MODE_OUT_ON 0x10
```

EXTIO pin always set in active state.

## 6.1.2.65 EXTIO\_SETUP\_OUTPUT

```
#define EXTIO_SETUP_OUTPUT 0x01
```

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

## 6.1.2.66 FEEDBACK\_EMF

```
#define FEEDBACK_EMF 0x04
```

Feedback by EMF.

## 6.1.2.67 FEEDBACK\_ENC\_ADAPTIVE\_HOLDING

```
#define FEEDBACK_ENC_ADAPTIVE_HOLDING 0x02
```

Enables the adaptive holding algorithm.

## 6.1.2.68 FEEDBACK\_ENC\_FILTER\_BITS

```
#define FEEDBACK_ENC_FILTER_BITS 0x30
```

Bits responsible for setting the internal filter of the encoder signal.

## 6.1.2.69 FEEDBACK\_ENC\_FILTER\_MEDIUM

```
#define FEEDBACK_ENC_FILTER_MEDIUM 0x20
```

Medium noise filtering: the maximum encoder signal frequency is 1 MHz.

## 6.1.2.70 FEEDBACK\_ENC\_FILTER\_NONE

```
#define FEEDBACK_ENC_FILTER_NONE 0x00
```

Disable the internal filter of the encoder signal.

## 6.1.2.71 FEEDBACK\_ENC\_FILTER\_STRONG

```
#define FEEDBACK_ENC_FILTER_STRONG 0x30
```

Strong noise filtering: the maximum encoder signal frequency is 300 kHz.

## 6.1.2.72 FEEDBACK\_ENC\_FILTER\_WEAK

```
#define FEEDBACK_ENC_FILTER_WEAK 0x10
```

Weak noise filtering: the maximum encoder signal frequency is 3 MHz.

## 6.1.2.73 FEEDBACK\_ENC\_REVERSE

```
#define FEEDBACK_ENC_REVERSE 0x01
```

Reverse count of encoder.

## 6.1.2.74 FEEDBACK\_ENC\_TYPE\_AUTO

```
#define FEEDBACK_ENC_TYPE_AUTO 0x00
```

Auto detect encoder type.

## 6.1.2.75 FEEDBACK\_ENC\_TYPE\_BITS

```
#define FEEDBACK_ENC_TYPE_BITS 0xC0
```

Bits of the encoder type.

## 6.1.2.76 FEEDBACK\_ENC\_TYPE\_DIFFERENTIAL

```
#define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80
```

Differential encoder.

## 6.1.2.77 FEEDBACK\_ENC\_TYPE\_SINGLE\_ENDED

```
#define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40
```

Single-ended encoder.

## 6.1.2.78 FEEDBACK\_ENCODER

```
#define FEEDBACK_ENCODER 0x01
```

Feedback by encoder.

## 6.1.2.79 FEEDBACK\_ENCODER\_MEDIATED

```
#define FEEDBACK_ENCODER_MEDIATED 0x06
```

Feedback by encoder mediated by mechanical transmission (for example leadscrew).

## 6.1.2.80 FEEDBACK\_NONE

```
#define FEEDBACK_NONE 0x05
```

Feedback is absent.

## 6.1.2.81 H\_BRIDGE\_ALERT

```
#define H_BRIDGE_ALERT 0x04
```

If this flag is set, then turn off the power unit when there is a signal problem in one of the transistor bridges.

## 6.1.2.82 HOME\_DIR\_FIRST

```
#define HOME_DIR_FIRST 0x001
```

The flag defines the direction of the 1st motion after execution of the home command.

The direction is to the right if the flag is set, and to the left otherwise.

## 6.1.2.83 HOME\_DIR\_SECOND

```
#define HOME_DIR_SECOND 0x002
```

The flag defines the direction of the 2nd motion.

The direction is to the right if the flag is set, and to the left otherwise.

## 6.1.2.84 HOME\_HALF\_MV

```
#define HOME_HALF_MV 0x008
```

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

## 6.1.2.85 HOME\_MV\_SEC\_EN

```
#define HOME_MV_SEC_EN 0x004
```

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

## 6.1.2.86 HOME\_STOP\_FIRST\_BITS

```
#define HOME_STOP_FIRST_BITS 0x030
```

Bits of the first stop selector.

## 6.1.2.87 HOME\_STOP\_FIRST\_LIM

```
#define HOME_STOP_FIRST_LIM 0x030
```

First motion stops by limit switch.

## 6.1.2.88 HOME\_STOP\_FIRST\_REV

```
#define HOME_STOP_FIRST_REV 0x010
```

First motion stops by revolution sensor.

## 6.1.2.89 HOME\_STOP\_FIRST\_SYN

```
#define HOME_STOP_FIRST_SYN 0x020
```

First motion stops by synchronization input.

## 6.1.2.90 HOME\_STOP\_SECOND\_BITS

```
#define HOME_STOP_SECOND_BITS 0x0C0
```

Bits of the second stop selector.

## 6.1.2.91 HOME\_STOP\_SECOND\_LIM

```
#define HOME_STOP_SECOND_LIM 0x0C0
```

Second motion stops by limit switch.

## 6.1.2.92 HOME\_STOP\_SECOND\_REV

```
#define HOME_STOP_SECOND_REV 0x040
```

Second motion stops by revolution sensor.

## 6.1.2.93 HOME\_STOP\_SECOND\_SYN

```
#define HOME_STOP_SECOND_SYN 0x080
```

Second motion stops by synchronization input.

## 6.1.2.94 HOME\_USE\_FAST

```
#define HOME_USE_FAST 0x100
```

Use the fast algorithm of calibration to the home position, if set; otherwise the traditional algorithm.

## 6.1.2.95 JOY\_REVERSE

```
#define JOY_REVERSE 0x01
```

Joystick action is reversed.

The joystick deviation to the upper values corresponds to negative speed and vice versa.

## 6.1.2.96 LOW\_UPWR\_PROTECTION

```
#define LOW_UPWR_PROTECTION 0x02
```

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

## 6.1.2.97 MICROSTEP\_MODE\_FRAC\_128

```
#define MICROSTEP_MODE_FRAC_128 0x08
```

1/128-step mode.

## 6.1.2.98 MICROSTEP\_MODE\_FRAC\_16

```
#define MICROSTEP_MODE_FRAC_16 0x05
```

1/16-step mode.

## 6.1.2.99 MICROSTEP\_MODE\_FRAC\_2

```
#define MICROSTEP_MODE_FRAC_2 0x02
```

1/2-step mode.

## 6.1.2.100 MICROSTEP\_MODE\_FRAC\_256

```
#define MICROSTEP_MODE_FRAC_256 0x09
```

1/256-step mode.

## 6.1.2.101 MICROSTEP\_MODE\_FRAC\_32

```
#define MICROSTEP_MODE_FRAC_32 0x06
```

1/32-step mode.

## 6.1.2.102 MICROSTEP\_MODE\_FRAC\_4

```
#define MICROSTEP_MODE_FRAC_4 0x03
```

1/4-step mode.

## 6.1.2.103 MICROSTEP\_MODE\_FRAC\_64

```
#define MICROSTEP_MODE_FRAC_64 0x07
```

1/64-step mode.

## 6.1.2.104 MICROSTEP\_MODE\_FRAC\_8

```
#define MICROSTEP_MODE_FRAC_8 0x04
```

1/8-step mode.

## 6.1.2.105 MICROSTEP\_MODE\_FULL

```
#define MICROSTEP_MODE_FULL 0x01
```

Full step mode.

## 6.1.2.106 MOVE\_STATE\_ANTIPLAY

```
#define MOVE_STATE_ANTIPLAY 0x04
```

Motor is playing compensation if the flag is set.

## 6.1.2.107 MOVE\_STATE\_MOVING

```
#define MOVE_STATE_MOVING 0x01
```

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

Don't use this flag to wait for the completion of the movement command. Use the MVCMD\_RUNNING flag from the MvCmdSts field instead.

## 6.1.2.108 MOVE\_STATE\_TARGET\_SPEED

```
#define MOVE_STATE_TARGET_SPEED 0x02
```

Target speed is reached if the flag is set.

## 6.1.2.109 MVCMD\_ERROR

```
#define MVCMD_ERROR 0x40
```

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

This flag makes sense when MVCMD\_RUNNING signals movement completion.

## 6.1.2.110 MVCMD\_HOME

```
#define MVCMD_HOME 0x06
```

Command home.

## 6.1.2.111 MVCMD\_LEFT

```
#define MVCMD_LEFT 0x03
```

Command left.

## 6.1.2.112 MVCMD\_LOFT

```
#define MVCMD_LOFT 0x07
```

Command loft.

## 6.1.2.113 MVCMD\_MOVE

```
#define MVCMD_MOVE 0x01
```

Command move.

## 6.1.2.114 MVCMD\_MOVR

```
#define MVCMD_MOVR 0x02
```

Command movr.

## 6.1.2.115 MVCMD\_NAME\_BITS

```
#define MVCMD_NAME_BITS 0x3F
```

Move command bit mask.

## 6.1.2.116 MVCMD\_RIGHT

```
#define MVCMD_RIGHT 0x04
```

Command rigt.

## 6.1.2.117 MVCMD\_RUNNING

```
#define MVCMD_RUNNING 0x80
```

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

## 6.1.2.118 MVCMD\_SSTP

```
#define MVCMD_SSTP 0x08
```

Command soft stop.

## 6.1.2.119 MVCMD\_STOP

```
#define MVCMD_STOP 0x05
```

Command stop.

## 6.1.2.120 MVCMD\_UKNWN

```
#define MVCMD_UKNWN 0x00
```

Unknown command.

## 6.1.2.121 POWER\_OFF\_ENABLED

```
#define POWER_OFF_ENABLED 0x02
```

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

## 6.1.2.122 POWER\_REDUCT\_ENABLED

```
#define POWER_REDUCT_ENABLED 0x01
```

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

## 6.1.2.123 POWER\_SMOOTH\_CURRENT

```
#define POWER_SMOOTH_CURRENT 0x04
```

Current ramp-up/down are performed smoothly during `current_set_time` if this flag is set.

## 6.1.2.124 PWR\_STATE\_MAX

```
#define PWR_STATE_MAX 0x05
```

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

## 6.1.2.125 PWR\_STATE\_NORM

```
#define PWR_STATE_NORM 0x03
```

Motor windings are powered by nominal current.

## 6.1.2.126 PWR\_STATE\_OFF

```
#define PWR_STATE_OFF 0x01
```

Motor windings are disconnected from the driver.

## 6.1.2.127 PWR\_STATE\_REDUCT

```
#define PWR_STATE_REDUCT 0x04
```

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

## 6.1.2.128 PWR\_STATE\_UNKNOWN

```
#define PWR_STATE_UNKNOWN 0x00
```

Unknown state, should never happen.

## 6.1.2.129 REV\_SENS\_INV

```
#define REV_SENS_INV 0x08
```

Typically, the sensor is active when it is at 0, and inversion makes active at 1.

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

## 6.1.2.130 RPM\_DIV\_1000

```
#define RPM_DIV_1000 0x01
```

This flag indicates that the operating speed specified in the command is set in milliRPM.

Applicable only for ENCODER feedback mode and only for BLDC motors.

## 6.1.2.131 SETPOS\_IGNORE\_ENCODER

```
#define SETPOS_IGNORE_ENCODER 0x02
```

Will not reload encoder state if this flag is set.

## 6.1.2.132 SETPOS\_IGNORE\_POSITION

```
#define SETPOS_IGNORE_POSITION 0x01
```

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

## 6.1.2.133 STATE\_ALARM

```
#define STATE_ALARM 0x0000040
```

The controller is in an alarm state, indicating that something dangerous has happened.

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

## 6.1.2.134 STATE\_BORDERS\_SWAP\_MISSET

```
#define STATE_BORDERS_SWAP_MISSET 0x0008000
```

Engine stuck at the wrong edge.

## 6.1.2.135 STATE\_BRAKE

```
#define STATE_BRAKE 0x0200
```

State of Brake pin.

Flag "1" - if the pin state brake is not powered (brake is clamped), "0" - if the pin state brake is powered (brake is unclamped).

## 6.1.2.136 STATE\_BUTTON\_LEFT

```
#define STATE_BUTTON_LEFT 0x0008
```

Button "left" state (1 if pressed).

## 6.1.2.137 STATE\_BUTTON\_RIGHT

```
#define STATE_BUTTON_RIGHT 0x0004
```

Button "right" state (1 if pressed).

## 6.1.2.138 STATE\_CONTR

```
#define STATE_CONTR 0x000003F
```

Flags of controller states.

## 6.1.2.139 STATE\_CONTROLLER\_OVERHEAT

```
#define STATE_CONTROLLER_OVERHEAT 0x0000200
```

Controller overheat.

## 6.1.2.140 STATE\_CTP\_ERROR

```
#define STATE_CTP_ERROR 0x0000080
```

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

The flag is set when the encoder position and step position are too far apart.

## 6.1.2.141 STATE\_DIG\_SIGNAL

```
#define STATE_DIG_SIGNAL 0xFFFF
```

Flags of digital signals.

## 6.1.2.142 STATE\_EEPROM\_CONNECTED

```
#define STATE_EEPROM_CONNECTED 0x0000010
```

EEPROM with settings is connected.

The built-in stage profile is uploaded from the EEPROM memory chip if the EEPROM\_PRECEDENCE flag is set, allowing you to connect various stages to the controller with automatic setup.

## 6.1.2.143 STATE\_ENC\_A

```
#define STATE_ENC_A 0x2000
```

State of encoder A pin.

## 6.1.2.144 STATE\_ENC\_B

```
#define STATE_ENC_B 0x4000
```

State of encoder B pin.

## 6.1.2.145 STATE\_ENGINE\_RESPONSE\_ERROR

```
#define STATE_ENGINE_RESPONSE_ERROR 0x0800000
```

Error response of the engine control action.

Motor control algorithm failure means that it can't make the correct decisions with the feedback data it receives. A single failure may be caused by a mechanical problem. A repeating failure can be caused by incorrect motor settings.

## 6.1.2.146 STATE\_ERRC

```
#define STATE_ERRC 0x0000001
```

Command error encountered.

The command received is not in the list of controller known commands. The most possible reason is the outdated firmware.

## 6.1.2.147 STATE\_ERRD

```
#define STATE_ERRD 0x0000002
```

Data integrity error encountered.

The data inside the command and its CRC code do not correspond. Therefore, the data can't be considered valid. This error may be caused by EMI in the UART/RS-232 interface.

## 6.1.2.148 STATE\_ERRV

```
#define STATE_ERRV 0x0000004
```

Value error encountered.

The values in the command can't be applied without correction because they fall outside the valid range. Corrected values were used instead of the original ones.

## 6.1.2.149 STATE\_EXTIO\_ALARM

```
#define STATE_EXTIO_ALARM 0x1000000
```

The error is caused by the external EXTIO input signal.

## 6.1.2.150 STATE\_GPIO\_LEVEL

```
#define STATE_GPIO_LEVEL 0x0020
```

State of external GPIO pin.

## 6.1.2.151 STATE\_GPIO\_PINOUT

```
#define STATE_GPIO_PINOUT 0x0010
```

External GPIO works as output if the flag is set; otherwise, it works as input.

## 6.1.2.152 STATE\_IS\_HOMED

```
#define STATE_IS_HOMED 0x0000020
```

Calibration performed.

This means that the relative position scale is calibrated against a hardware absolute position sensor, like a limit switch. Drops after loss of calibration, like harsh stops and possibly skipped steps.

## 6.1.2.153 STATE\_LEFT\_EDGE

```
#define STATE_LEFT_EDGE 0x0002
```

Engine stuck at the left edge.

## 6.1.2.154 STATE\_LOW\_USB\_VOLTAGE

```
#define STATE_LOW_USB_VOLTAGE 0x0002000
```

Deprecated.

USB voltage is insufficient for normal operation. This field is left for compatibility. Software should not rely on the value of this field.

## 6.1.2.155 STATE\_OVERLOAD\_POWER\_CURRENT

```
#define STATE_OVERLOAD_POWER_CURRENT 0x0000800
```

Power current exceeds safe limit.

## 6.1.2.156 STATE\_OVERLOAD\_POWER\_VOLTAGE

```
#define STATE_OVERLOAD_POWER_VOLTAGE 0x0000400
```

Power voltage exceeds safe limit.

## 6.1.2.157 STATE\_OVERLOAD\_USB\_CURRENT

```
#define STATE_OVERLOAD_USB_CURRENT 0x0004000
```

Deprecated.

USB current exceeds safe limit. This field is left for compatibility. Software should not rely on the value of this field.

## 6.1.2.158 STATE\_OVERLOAD\_USB\_VOLTAGE

```
#define STATE_OVERLOAD_USB_VOLTAGE 0x0001000
```

Deprecated.

USB voltage exceeds safe limit. This field is left for compatibility. Software should not rely on the value of this field.

## 6.1.2.159 STATE\_POWER\_OVERHEAT

```
#define STATE_POWER_OVERHEAT 0x0000100
```

Power driver overheat.

Motor control is disabled until some cooldown occurs. This should not happen with boxed versions of the controller. This may happen with the bare-board version of the controller with a custom radiator. Redesign your radiator.

## 6.1.2.160 STATE\_REV\_SENSOR

```
#define STATE_REV_SENSOR 0x0400
```

State of Revolution sensor pin.

## 6.1.2.161 STATE\_RIGHT\_EDGE

```
#define STATE_RIGHT_EDGE 0x0001
```

Engine stuck at the right edge.

## 6.1.2.162 STATE\_SECUR

```
#define STATE_SECUR 0x1B3FFC0
```

Security flags.

## 6.1.2.163 STATE\_SYNC\_INPUT

```
#define STATE_SYNC_INPUT 0x0800
```

State of Sync input pin.

## 6.1.2.164 STATE\_SYNC\_OUTPUT

```
#define STATE_SYNC_OUTPUT 0x1000
```

State of Sync output pin.

## 6.1.2.165 STATE\_WINDING\_RES\_MISMATCH

```
#define STATE_WINDING_RES_MISMATCH 0x0100000
```

The difference between winding resistances is too large.

This usually happens with a damaged stepper motor with partially short-circuited windings.

## 6.1.2.166 SYNCIN\_ENABLED

```
#define SYNCIN_ENABLED 0x01
```

Synchronization in mode is enabled if this flag is set.

## 6.1.2.167 SYNCIN\_GOTOPOSITION

```
#define SYNCIN_GOTOPOSITION 0x04
```

The engine is going to the position specified in Position and uPosition if this flag is set.

And it is shifting on the Position and uPosition if this flag is unset.

## 6.1.2.168 SYNCIN\_INVERT

```
#define SYNCIN_INVERT 0x02
```

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

## 6.1.2.169 SYNCOUT\_ENABLED

```
#define SYNCOUT_ENABLED 0x01
```

The synchronization out pin follows the synchronization logic if the flag is set.

Otherwise, it is governed by the SYNCOUT\_STATE flag.

## 6.1.2.170 SYNCOUT\_IN\_STEPS

```
#define SYNCOUT_IN_STEPS 0x08
```

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

## 6.1.2.171 SYNCOUT\_INVERT

```
#define SYNCOUT_INVERT 0x04
```

The low level is active if the flag is set.

Otherwise, the high level is active.

## 6.1.2.172 SYNCOUT\_ONPERIOD

```
#define SYNCOUT_ONPERIOD 0x40
```

Generate a synchronization pulse every SyncOutPeriod encoder pulses.

## 6.1.2.173 SYNCOUT\_ONSTART

```
#define SYNCOUT_ONSTART 0x10
```

Generate a synchronization pulse when movement starts.

## 6.1.2.174 SYNCOUT\_ONSTOP

```
#define SYNCOUT_ONSTOP 0x20
```

Generate a synchronization pulse when movement stops.

## 6.1.2.175 SYNCOUT\_STATE

```
#define SYNCOUT_STATE 0x02
```

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

## 6.1.2.176 UART\_PARITY\_BITS

```
#define UART_PARITY_BITS 0x03
```

Bits of the parity.

## 6.1.2.177 WIND\_A\_STATE\_ABSENT

```
#define WIND_A_STATE_ABSENT 0x00
```

Winding A is disconnected.

## 6.1.2.178 WIND\_A\_STATE\_MALFUNC

```
#define WIND_A_STATE_MALFUNC 0x02
```

Winding A is short-circuited.

## 6.1.2.179 WIND\_A\_STATE\_OK

```
#define WIND_A_STATE_OK 0x03
```

Winding A is connected and working properly.

## 6.1.2.180 WIND\_A\_STATE\_UNKNOWN

```
#define WIND_A_STATE_UNKNOWN 0x01
```

Winding A state is unknown.

## 6.1.2.181 WIND\_B\_STATE\_ABSENT

```
#define WIND_B_STATE_ABSENT 0x00
```

Winding B is disconnected.

## 6.1.2.182 WIND\_B\_STATE\_MALFUNC

```
#define WIND_B_STATE_MALFUNC 0x20
```

Winding B is short-circuited.

## 6.1.2.183 WIND\_B.STATE\_OK

```
#define WIND_B.STATE_OK 0x30
```

Winding B is connected and working properly.

## 6.1.2.184 WIND\_B.STATE\_UNKNOWN

```
#define WIND_B.STATE_UNKNOWN 0x10
```

Winding B state is unknown.

## 6.1.2.185 XIMC\_API

```
#define XIMC_API
```

Library import macro.

Macros allows to automatically import function from shared library. It automatically expands to `__declspec(dllimport)` on `msvc` when including header file.

## 6.1.3 Typedef Documentation

## 6.1.3.1 calibration\_t

```
typedef struct calibration_t calibration_t
```

Calibration structure.

Where to find all values for calculations?

- XILab (don't forget to load the profile for your positioner. The profile should match the full name of your positioner, e.g., 8MT173-25-MEn1.cfg):
  - In XILab settings, go to the "User units" tab. Divide the second number by the first one — this is your A coefficient.
  - In the "DC motor" / "BLDC motor" / "Stepper motor" tab (depending on your motor type), find the "Encoder counts per turn" field and use it in the formula for calculating coefficient B.
- Profile file (open with any text editor; make sure it matches the full name of your positioner, e.g., 8MT173-25-MEn1.cfg):
  - Find `Step_multiplier=` and `Unit_multiplier=`. Divide the second by the first — this is your A coefficient.

- Find Encoder\_CPT= and use this value in the B coefficient formula instead of ENCODER\_CO↵UNTS\_PER\_TURN.

How to calculate Speed, Accel, Decel, and AntiplaySpeed in user units when using a stepper motor with encoder or DC/BLDC motor?

1. Load the correct profile in XILab for your positioner (e.g., 8MT173-25-MEn1.cfg).
  2. Enable Feedback encoder mode if not already enabled.
  3. Enter speed in the "Working speed" field in user units.
  4. In the "User units" tab, disable the "User units" flag to see RPM.
  5. Multiply the RPM value from "Working speed" by coefficient B. Example:  $480 * 0.0000009375 = 0.00045$ . This value for the 8MT173-25-MEn1 in Encoder mode equals 2 mm/s.
- Acceleration, deceleration, and antiplay speed are calculated the same way.

#### 6.1.3.2 logging\_callback\_t

```
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  |

### 6.1.4 Function Documentation

#### 6.1.4.1 close\_device()

```
result_t XIMC_API close_device (
    device_t * id )
```

Close specified device.

Parameters

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

Note

The id parameter in this function is a C pointer, unlike most library functions that use this parameter

#### 6.1.4.2 command\_clear\_fram()

```
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 |
|-----------|-------------------------|

#### 6.1.4.3 command\_eeread\_settings()

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

Read settings from the stage's EEPROM to the controller's RAM.

This operation is performed automatically at the connection of the stage with an EEPROM to the controller.  
Can be used by the manufacturer only.

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

#### 6.1.4.4 command\_eesave\_settings()

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

Save settings from the controller's RAM to the stage's EEPROM.

Can be used by the manufacturer only.

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

6.1.4.5 `command_home()`

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

Moving to home position.

Moving algorithm:

- 1) Moves the motor according to the speed FastHome, uFastHome and flag HOME\_DIR\_FAST until the limit switch if the HOME\_STOP\_ENDS flag is set. Or moves the motor until the input synchronization signal occurs if the flag HOME\_STOP\_SYNC is set. Or moves until the revolution sensor signal occurs if the flag HOME\_STOP\_REV\_SN is set.
- 2) Then moves according to the speed SlowHome, uSlowHome and flag HOME\_DIR\_SLOW until the input clock signal occurs if the flag HOME\_MV\_SEC is set. If the flag HOME\_MV\_SEC is reset, skip this step.
- 3) Then shifts the motor according to the speed FastHome, uFastHome and the flag HOME\_DIR\_SLOW by HomeDelta distance, uHomeDelta.

See GHOM/SHOM commands' description for details on home flags.

Moving settings can be set by `set_home_settings/set_home_settings_calb`.

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

See also

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

6.1.4.6 `command_homezero()`

```
result_t XIMC_API command_homezero (
    device_t id )
```

Make home command, wait until it is finished and make zero command.

This is a convinient way to calibrate zero position.

Parameters

|     |            |                                                                                                                                             |
|-----|------------|---------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>id</i>  | an identifier of device                                                                                                                     |
| out | <i>ret</i> | RESULT_OK if controller has finished home & zero correctly or result of first controller query that returned anything other than RESULT_OK. |

6.1.4.7 `command_left()`

```
result_t XIMC_API command_left (
    device_t id )
```

Start continuous moving to the left.

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

6.1.4.8 `command_loft()`

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

Upon receiving the command "loft", the engine is shifted from the current position to a distance Antiplay defined in engine settings.

Then moves to the initial position.

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

6.1.4.9 `command_move()`

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

Move to position.

Upon receiving the command " move" the engine starts to move with pre-set parameters (speed, acceleration, retention), to the point specified by Position and uPosition. uPosition sets the microstep position of a stepper motor. In the case of DC motor, this field is ignored.

Parameters

|                      |                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>id</i>            | An identifier of a device                                                                                                                                                      |
| <i>Position</i>      | position to move.                                                                                                                                                              |
| <i>uPosition</i>     | the fractional part of the position to move, in microsteps. The microstep size and the range of valid values for this field depend on the selected step division mode (see the |
| Generated by Doxygen | MicrostepMode field in engine_settings).                                                                                                                                       |

6.1.4.10 `command_move_calb()`

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

Move to position using user units.

Upon receiving the command "move" the engine starts to move with preset parameters (speed, acceleration, retention), to the point specified by Position.

Parameters

|                    |                           |
|--------------------|---------------------------|
| <i>id</i>          | An identifier of a device |
| <i>Position</i>    | position to move.         |
| <i>calibration</i> | user unit settings        |

Note

The parameter Position is adjusted by the correction table.

6.1.4.11 `command_movr()`

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

Shift by a set offset.

Upon receiving the command "movr", the engine starts to move with preset parameters (speed, acceleration, hold) left or right (depending on the sign of DeltaPosition). It moves by the number of steps specified in the fields DeltaPosition and uDeltaPosition. uDeltaPosition sets the microstep offset for a stepper motor. In the case of a DC motor, this field is ignored.

Parameters

|                       |                                                                                                                                                                                                                     |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>DeltaPosition</i>  | shift from initial position.                                                                                                                                                                                        |
| <i>uDeltaPosition</i> | the fractional part of the offset shift, in microsteps. The microstep size and the range of valid values for this field depend on the selected step division mode (see the MicrostepMode field in engine_settings). |
| <i>id</i>             | An identifier of a device                                                                                                                                                                                           |

6.1.4.12 `command_movr_calb()`

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

Shift by a set offset using user units.

Upon receiving the command "movr", the engine starts to move with preset parameters (speed, acceleration, hold) left or right (depending on the sign of DeltaPosition). It moves by the distance specified in the field DeltaPosition.

Parameters

|                      |                              |
|----------------------|------------------------------|
| <i>DeltaPosition</i> | shift from initial position. |
| <i>id</i>            | An identifier of a device    |
| <i>user</i>          | unit calibration settings    |

Note

The final coordinate is calculated using DeltaPosition and adjusted by the correction table. However, the correction cannot be done if the motor moves. movr sets the target position equal to the current target position, shifted by delta. But the library can't determine the current target position while moving. So there is no possibility of calculating the final position and correcting it with the correction table.

6.1.4.13 `command_power_off()`

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

Immediately power off the motor regardless its state.

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

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

See also

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

6.1.4.14 `command_read_robust_settings()`

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

Read important settings (calibration coefficients, etc.) from the controller's flash memory to the controller's RAM, replacing previous data in the RAM.

Manufacturer only.

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

6.1.4.15 `command_read_settings()`

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

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

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

6.1.4.16 `command_reset()`

```
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 |
|-----------|-------------------------|

6.1.4.17 `command_right()`

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

Start continuous moving to the right.

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

#### 6.1.4.18 `command_save_robust_settings()`

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

Save important settings (calibration coefficients, etc.) from the controller's RAM to the controller's flash memory, replacing previous data in the flash memory.

Manufacturer only.

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

#### 6.1.4.19 `command_save_settings()`

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

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

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

#### 6.1.4.20 `command_sstp()`

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

Soft stop the engine.

The motor is slowing down with the deceleration specified in `move_settings`.

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

#### 6.1.4.21 `command_start_measurements()`

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

Start measurements and buffering of speed and the speed error (target speed minus real speed).

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

#### 6.1.4.22 `command_stop()`

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

Immediately stops the engine, moves it to the STOP state, and sets switches to BREAK mode (windings are short-circuited).

The holding regime is deactivated for DC motors, keeping current in the windings for stepper motors (to control it, see Power management settings).

When this command is called, the ALARM flag is reset.

Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

#### 6.1.4.23 `command_update_firmware()`

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

Update firmware.

Manufacturer only. Service command

## Parameters

|                  |                      |
|------------------|----------------------|
| <i>uri</i>       | a uri of device      |
| <i>data</i>      | firmware byte stream |
| <i>data_size</i> | size of byte stream  |

6.1.4.24 `command_wait_for_stop()`

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

Wait for stop.

## Parameters

|     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>id</i>                  | an identifier of device                                                                                                                                                                                                                                                                                                                                                                                         |
|     | <i>refresh_interval_ms</i> | Status refresh interval. The function waits this number of milliseconds between <code>get_status</code> requests to the controller. Recommended value of this parameter is 10 ms. Use values of less than 3 ms only when necessary - small refresh interval values do not significantly increase response time of the function, but they create substantially more traffic in controller-computer data channel. |
| out | <i>ret</i>                 | RESULT_OK if controller has stopped and result of the first <code>get_status</code> command which returned anything other than RESULT_OK otherwise.                                                                                                                                                                                                                                                             |

6.1.4.25 `command_zero()`

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

Sets the current position to 0.

Sets the target position of the move command and the movr command to zero for all cases except for movement to the target position. In the latter case, the target position is calculated so that the absolute position of the destination stays the same. For example, if we were at 400 and moved to 500, then the command Zero makes the current position 0 and the position of the destination 100. It does not change the mode of movement. If the motion is in progress, it continues, and if the engine is in the "hold" state, the type of retention remains.

## Parameters

|           |                           |
|-----------|---------------------------|
| <i>id</i> | An identifier of a device |
|-----------|---------------------------|

## 6.1.4.26 enumerate\_devices()

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

Search and list of available devices.

By default, it creates a list of devices connected to this computer and presented as COM ports. Additionally, you can enable the search for network devices. Devices found in the local network will be included in the same list.

To obtain information from the collected list, use the corresponding functions with the `get_enumerate_` prefix and the received `device_enumeration` identifier.

After finishing working with the list of found devices, you should free up memory using the `free_enumerate_devices()` function.

Parameters

|    |                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | <i>enumerate_flags</i> | <p>a set of flags that specify search modes. Flags can be used together via bitwise "OR":</p> <ul style="list-style-type: none"> <li>• <code>ENUMERATE_NETWORK</code> - enables searching for network devices. If the flag is set, network devices will be added to the general list. If the flag is not set, the list will only include devices connected to this computer.</li> <li>• <code>ENUMERATE_ALL_COM</code> - when enabled, queries all COM port devices in the system. When disabled, queries only devices whose names match the XIMC device mask ("XIMC Motor Controller" in Windows, <code>/dev/ximc/</code> and <code>/dev/ttyACM/</code> on Linux/Mac).</li> <li>• <code>ENUMERATE_PROBE</code> - enables checking of devices and collecting additional information (serial number, version, model, name...). If this flag is set, only devices that are guaranteed to be open will be added to the list, but devices connected via RS232 converters may not be included. If the flag is not set, the list will include more devices (in particular, devices explicitly listed in <code>hints</code> will be included), but availability and compatibility with this library is not guaranteed.</li> </ul> |
|----|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Parameters

|    |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | <i>hints</i> | <p>additional information to improve the search efficiency. It makes sense to use in case of complex network configuration, when automatic search may not find everything. Format - string "key1=value1\nkey2=value2". Unknown keys are ignored. One key can have several values, which are listed separated by commas: key=value1,value2,value3. Valid keys:</p> <ul style="list-style-type: none"> <li>• <code>addr</code> - list of URLs of network controllers or servers with connected controllers. The field is used together with the <code>ENUMERATE_NETWORK</code> flag. Protocols and address formats: <ul style="list-style-type: none"> <li>- <code>xi-tcp://&lt;ip-address&gt;</code> - network controllers and controllers connected via Ethernet-RS232 converters. If the <code>ENUMERATE_PROBE</code> flag is not set, all listed devices will be included in the list.</li> <li>- <code>xi-net://&lt;ip-address&gt;</code> - network multi-axis systems, xi-net servers. The request for information about the availability of controllers at these addresses will be made regardless of the results of the automatic network search procedure.</li> </ul> </li> <li>• <code>adapter_addr</code> - list of IP addresses of local network adapters through which the search should be performed. If the key is missing or no adapters are specified, the search is performed on all adapters.</li> </ul> |
|----|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

`device_enumeration` - device list identifier. Used to obtain information about devices using functions with `get_enumerate_` prefixes.

## Examples of use:

```
// Search for local devices without checking
device_enumeration_t device_enumeration = enumerate.devices(0, "");
// Search for local devices with verification
device_enumeration_t device_enumeration = enumerate.devices(ENUMERATE_PROBE, "");
// Fully automatic search for local and network devices
device_enumeration_t device_enumeration = enumerate.devices(ENUMERATE_NETWORK, "");
// Search for local and network devices
// using the local computer adapter with the address 192.168.0.100
// and explicit requests to the network controller with the address 192.168.0.11
// and the xi-net server with the address 192.168.0.10
device_enumeration_t device_enumeration = enumerate.devices(
ENUMERATE_NETWORK,
"addr=192.168.0.10,xi-tcp://192.168.0.11\nadapter_addr=192.168.0.100"
);
```

## 6.1.4.27 free\_enumerate\_devices()

```
result_t XIMC_API free_enumerate_devices (  
    device_enumeration_t device_enumeration )
```

Free memory returned by *enumerate\_devices*.

Parameters

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

#### 6.1.4.28 get\_accessories\_settings()

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

Deprecated.

Read additional accessory information from the EEPROM.

Parameters

|     |                             |                                                             |
|-----|-----------------------------|-------------------------------------------------------------|
|     | <i>id</i>                   | An identifier of a device                                   |
| out | <i>accessories_settings</i> | structure contains information about additional accessories |

#### 6.1.4.29 get\_analog\_data()

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

Read the analog data structure that contains raw analog data from the embedded ADC.

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

Parameters

|     |                    |                           |
|-----|--------------------|---------------------------|
|     | <i>id</i>          | An identifier of a device |
| out | <i>analog_data</i> | analog data coefficients  |

#### 6.1.4.30 get\_bootloader\_version()

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

Read the controller's bootloader version.

## Parameters

|     |                |                           |
|-----|----------------|---------------------------|
|     | <i>id</i>      | An identifier of a device |
| out | <i>Major</i>   | major version             |
| out | <i>Minor</i>   | minor version             |
| out | <i>Release</i> | release version           |

## 6.1.4.31 get\_brake\_settings()

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

Read break control settings.

## Parameters

|     |                       |                                              |
|-----|-----------------------|----------------------------------------------|
|     | <i>id</i>             | An identifier of a device                    |
| out | <i>brake_settings</i> | structure contains settings of brake control |

## 6.1.4.32 get\_calibration\_settings()

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

Read calibration settings.

Manufacturer only. This function reads the structure with calibration settings. These settings are used to convert bare ADC values to winding currents in mA and the full current in mA. Parameters are grouped into pairs, XXX\_A and XXX\_B, representing linear equation coefficients. The first one is the slope, the second one is the constant term. Thus,  $XXX\_Current[mA] = XXX\_A[mA/ADC] * XXX\_ADC\_CODE[ADC] + XXX\_B[mA]$ .

See also

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

## Parameters

|     |                             |                           |
|-----|-----------------------------|---------------------------|
|     | <i>id</i>                   | An identifier of a device |
| out | <i>calibration_settings</i> | calibration settings      |

## 6.1.4.33 get\_chart\_data()

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

Return device electrical parameters, useful for charts.

A useful function that fills the structure with a snapshot of the controller voltages and currents.

See also

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

Parameters

|     |                   |                                                     |
|-----|-------------------|-----------------------------------------------------|
|     | <i>id</i>         | An identifier of a device                           |
| out | <i>chart_data</i> | structure with a snapshot of controller parameters. |

## 6.1.4.34 get\_control\_settings()

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

Read motor control settings.

In case of CTL\_MODE=1, joystick motor control is enabled. In this mode, while the joystick is maximally displaced, the engine tends to move at MaxSpeed[i]. i=0 if another value hasn't been set at the previous usage. To change the speed index "i", use the buttons.

In case of CTL\_MODE=2, the motor is controlled by the left/right buttons. When you click on the button, the motor starts moving in the appropriate direction at a speed MaxSpeed[0]. After Timeout[i], motor moves at speed MaxSpeed[i+1]. At the transition between MaxSpeed[i] and MaxSpeed[i+1] the motor just accelerates/decelerates as usual.

Parameters

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

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

```
result_t XIMC_API get_control_settings_calb (
    device_t id,
```

```
control_settings_calb_t * control_settings_calb,
const calibration_t * calibration )
```

Read motor control settings with user units.

In case of CTL\_MODE=1, the joystick motor control is enabled. In this mode, while the joystick is maximally displaced, the engine tends to move at MaxSpeed[i]. i=0 if another value hasn't been set at the previous usage. To change the speed index "i", use the buttons.

In case of CTL\_MODE=2, the motor is controlled by the left/right buttons. When you click on the button, the motor starts moving in the appropriate direction at a speed MaxSpeed[0]. After Timeout[i], motor moves at speed MaxSpeed[i+1]. At the transition between MaxSpeed[i] and MaxSpeed[i+1] the motor just accelerates/decelerates as usual.

Parameters

|     |                              |                                                      |
|-----|------------------------------|------------------------------------------------------|
|     | <i>id</i>                    | An identifier of a device                            |
| out | <i>control_settings_calb</i> | structure contains user unit motor control settings. |
|     | <i>calibration</i>           | user unit settings                                   |

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

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

Read user's controller name and internal settings from the FRAM.

Parameters

|     |                        |                                                        |
|-----|------------------------|--------------------------------------------------------|
|     | <i>id</i>              | An identifier of a device                              |
| out | <i>controller_name</i> | structure contains previously set user controller name |

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

```
result_t XIMC_API get_ctp_settings (
    device_t id,
    ctp_settings_t * ctp_settings )
```

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

When controlling the step motor with an 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 is enabled (CTP\_ENABLED is set), the controller stores the current position in the steps of SM and the current position of the encoder. Next, the encoder position is converted into

steps at each step, and if the difference between the current position in steps and the encoder position is greater than CTPMinError, the flag STATE\_CTP\_ERROR is set.

Alternatively, the stepper motor may be controlled with the speed sensor (CTP\_BASE 1). In this mode, at the active edges of the input clock, the controller stores the current value of steps. Then, at each revolution, the controller checks how many steps have been passed. When the difference is over the CTPMinError, the STATE\_CTP\_ERROR flag is set.

Parameters

|     |                     |                                               |
|-----|---------------------|-----------------------------------------------|
|     | <i>id</i>           | An identifier of a device                     |
| out | <i>ctp_settings</i> | structure contains position control settings. |

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

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

Read data from firmware for debug purpose.

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

Parameters

|     |                   |                           |
|-----|-------------------|---------------------------|
|     | <i>id</i>         | An identifier of a device |
| out | <i>debug_read</i> | Debug data.               |

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

```
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 |
|----|---------------------------|----------------------------------------------|

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

```
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](#)

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

```
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                                 |

#### 6.1.4.42 get\_edges\_settings()

```
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 a device                                                                  |
| out | <i>edges_settings</i> | edges settings, types of borders, motor behavior and electrical behavior of limit switches |

## 6.1.4.43 get\_edges\_settings\_calb()

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

Read border and limit switches settings in user units.

See also

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

## Parameters

|     |                            |                                                                                            |
|-----|----------------------------|--------------------------------------------------------------------------------------------|
|     | <i>id</i>                  | An identifier of a device                                                                  |
| out | <i>edges_settings_calb</i> | edges settings, types of borders, motor behavior and electrical behavior of limit switches |
|     | <i>calibration</i>         | user unit settings                                                                         |

## Note

Attention! Some parameters of the `edges_settings_calb` structure are corrected by the coordinate correction table.

## 6.1.4.44 get\_emf\_settings()

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

Read electromechanical settings.

The settings are different for different stepper motors.

See also

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

Parameters

|     |                     |                           |
|-----|---------------------|---------------------------|
|     | <i>id</i>           | An identifier of a device |
| out | <i>emf_settings</i> | EMF settings              |

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

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

Deprecated.

Read encoder information from the EEPROM.

Parameters

|     |                            |                                              |
|-----|----------------------------|----------------------------------------------|
|     | <i>id</i>                  | An identifier of a device                    |
| out | <i>encoder_information</i> | structure contains information about encoder |

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

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

Deprecated.

Read encoder settings from the EEPROM.

Parameters

|     |                         |                                     |
|-----|-------------------------|-------------------------------------|
|     | <i>id</i>               | An identifier of a device           |
| out | <i>encoder_settings</i> | structure contains encoder settings |

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

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

Read engine advanced settings.

Manufacturer only.

See also

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

Parameters

|     |                              |                           |
|-----|------------------------------|---------------------------|
|     | <i>id</i>                    | An identifier of a device |
| out | <i>engine_advanced_setup</i> | EAS settings              |

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

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

Read engine settings.

This function reads the structure containing a set of useful motor settings stored in the 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 a device |
| out | <i>engine_settings</i> | engine settings           |

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

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

Read user unit engine settings.

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

See also

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

## Parameters

|     |                             |                           |
|-----|-----------------------------|---------------------------|
|     | <i>id</i>                   | An identifier of a device |
| out | <i>engine_settings_calb</i> | engine settings           |
|     | <i>calibration</i>          | user unit settings        |

## 6.1.4.50 get\_entype\_settings()

```
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 a device                                    |
| out | <i>entype_settings</i> | structure contains motor type and power driver type settings |

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

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

Get controller name from the device enumeration.

Returns *device\_index* device controller name.

## Parameters

|     |                           |                                              |
|-----|---------------------------|----------------------------------------------|
| in  | <i>device_enumeration</i> | opaque pointer to an enumeration device data |
| in  | <i>device_index</i>       | device index                                 |
| out | <i>controller_name</i>    | controller name                              |

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

```
result_t XIMC_API get_enumerate_device_information (
    device_enumeration_t device_enumeration,
```

```
int device_index,  
device_information_t * device_information )
```

Get device information from the device enumeration.

Returns *device\_index* device information.

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                      |

#### 6.1.4.53 get\_enumerate\_device\_network\_information()

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

Get device network information from the device enumeration.

Returns *device\_index* device network information.

Parameters

|     |                                   |                                              |
|-----|-----------------------------------|----------------------------------------------|
| in  | <i>device_enumeration</i>         | opaque pointer to an enumeration device data |
| in  | <i>device_index</i>               | device index                                 |
| out | <i>device_network_information</i> | device network information data              |

#### 6.1.4.54 get\_enumerate\_device\_serial()

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

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                         |

6.1.4.55 `get_enumerate_device_stage_name()`

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

Get stage name from the device enumeration.

Returns *device\_index* device stage name.

Parameters

|     |                           |                                              |
|-----|---------------------------|----------------------------------------------|
| in  | <i>device_enumeration</i> | opaque pointer to an enumeration device data |
| in  | <i>device_index</i>       | device index                                 |
| out | <i>stage_name</i>         | stage name                                   |

6.1.4.56 `get_extended_settings()`

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

Read extended settings.

Currently, it is not in use.

See also

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

Parameters

|     |                          |                           |
|-----|--------------------------|---------------------------|
|     | <i>id</i>                | An identifier of a device |
| out | <i>extended_settings</i> | EST settings              |

6.1.4.57 `get_extio_settings()`

```
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 the controller's memory.

See also

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

Parameters

|     |                       |                           |
|-----|-----------------------|---------------------------|
|     | <i>id</i>             | An identifier of a device |
| out | <i>extio_settings</i> | EXTIO settings            |

#### 6.1.4.58 get\_feedback\_settings()

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

Feedback settings.

Parameters

|     |                      |                                                                                                                                                                                                                                            |
|-----|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>id</i>            | An identifier of a device                                                                                                                                                                                                                  |
| out | <i>IPS</i>           | number of encoder counts per shaft revolution. Range: 1..65535. The field is obsolete, it is recommended to write 0 to IPS and use the extended CountsPerTurn field. You may need to update the controller firmware to the latest version. |
| out | <i>FeedbackType</i>  | type of feedback                                                                                                                                                                                                                           |
| out | <i>FeedbackFlags</i> | flags of feedback                                                                                                                                                                                                                          |
| out | <i>CountsPerTurn</i> | number of encoder counts per shaft revolution. Range: 1..4294967295. To use the CountsPerTurn field, write 0 in the IPS field, otherwise the value from the IPS field will be used.                                                        |

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

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

Read the controller's firmware version.

## Parameters

|     |                |                           |
|-----|----------------|---------------------------|
|     | <i>id</i>      | An identifier of a device |
| out | <i>Major</i>   | major version             |
| out | <i>Minor</i>   | minor version             |
| out | <i>Release</i> | release version           |

## 6.1.4.60 get\_gear\_information()

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

Deprecated.

Read gear information from the EEPROM.

## Parameters

|     |                         |                                                    |
|-----|-------------------------|----------------------------------------------------|
|     | <i>id</i>               | An identifier of a device                          |
| out | <i>gear_information</i> | structure contains information about step gearhead |

## 6.1.4.61 get\_gear\_settings()

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

Deprecated.

Read gear settings from the EEPROM.

## Parameters

|     |                      |                                           |
|-----|----------------------|-------------------------------------------|
|     | <i>id</i>            | An identifier of a device                 |
| out | <i>gear_settings</i> | structure contains step gearhead settings |

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

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

This value is unique to each individual device, but is not a random value.

Manufacturer only. This unique device identifier can be used to initiate secure boot processes or as a serial number for USB or other end applications.

Parameters

|     |                                   |                                                                                     |
|-----|-----------------------------------|-------------------------------------------------------------------------------------|
|     | <i>id</i>                         | An identifier of a device                                                           |
| out | <i>globally_unique_identifier</i> | the result of fields 0-3 concatenated defines the unique 128-bit device identifier. |

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

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

Deprecated.

Read hall sensor information from the EEPROM.

Parameters

|     |                               |                                                  |
|-----|-------------------------------|--------------------------------------------------|
|     | <i>id</i>                     | An identifier of a device                        |
| out | <i>hallsensor_information</i> | structure contains information about hall sensor |

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

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

Deprecated.

Read hall sensor settings from the EEPROM.

Parameters

|     |                            |                                         |
|-----|----------------------------|-----------------------------------------|
|     | <i>id</i>                  | An identifier of a device               |
| out | <i>hallsensor_settings</i> | structure contains hall sensor settings |

## 6.1.4.65 get\_home\_settings()

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

Read home settings.

This function reads the structure with home position settings.

See also

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

Parameters

|     |                      |                               |
|-----|----------------------|-------------------------------|
|     | <i>id</i>            | An identifier of a device     |
| out | <i>home_settings</i> | calibrating position settings |

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

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

Read user unit home settings.

This function reads the structure with home position settings.

See also

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

Parameters

|     |                           |                               |
|-----|---------------------------|-------------------------------|
|     | <i>id</i>                 | An identifier of a device     |
| out | <i>home_settings_calb</i> | calibrating position settings |
|     | <i>calibration</i>        | user unit settings            |

## 6.1.4.67 get\_init\_random()

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

Read a random number from the controller.

Manufacturer only.

Parameters

|     |                    |                                             |
|-----|--------------------|---------------------------------------------|
|     | <i>id</i>          | An identifier of a device                   |
| out | <i>init_random</i> | random sequence generated by the controller |

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

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

Read joystick settings.

If joystick position falls outside DeadZone limits, a movement begins. The speed is defined by the joystick's position in the range from the DeadZone limit to the maximum deviation. Joystick positions inside DeadZone limits correspond to zero speed (a "soft stop" command is issued continuously), and positions beyond Low and High limits correspond to MaxSpeed[i] or -MaxSpeed[i] (see command SCTL), where  $i = 0$  by default and can be changed with the left/right buttons (see command SCTL). If the next speed in the list is zero (both integer and microstep parts), the button press is ignored. The first speed in the list shouldn't be zero. DeadZone is defined in 0.1% units. See the Joystick control section on [https://doc.xisupport.com/en/8smc5-usb/8SMCn-USB/Technical\\_specification/Additional\\_features/Joystick\\_control.html](https://doc.xisupport.com/en/8smc5-usb/8SMCn-USB/Technical_specification/Additional_features/Joystick_control.html) for more information.

Parameters

|     |                          |                                      |
|-----|--------------------------|--------------------------------------|
|     | <i>id</i>                | An identifier of a device            |
| out | <i>joystick_settings</i> | structure contains joystick settings |

#### 6.1.4.69 get\_measurements()

```
result_t XIMC_API get_measurements (
    device_t id,
    measurements_t * measurements )
```

A command to read the data buffer to build a speed graph and a speed error graph.

Filling the buffer starts with the command "start\_measurements". The buffer holds 25 points; the points are taken with a period of 1 ms. To create a robust system, read data every 20 ms. If the buffer is full, it is recommended to repeat the readings every 5 ms until the buffer again becomes filled with 20 points.

To stop measurements just stop reading data. After buffer overflow measurements will stop automatically.

See also

[measurements\\_t](#)

Parameters

|     |                     |                                       |
|-----|---------------------|---------------------------------------|
|     | <i>id</i>           | An identifier of a device             |
| out | <i>measurements</i> | structure with buffer and its length. |

#### 6.1.4.70 get\_motor\_information()

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

Deprecated.

Read motor information from the EEPROM.

Parameters

|     |                          |                                      |
|-----|--------------------------|--------------------------------------|
|     | <i>id</i>                | An identifier of a device            |
| out | <i>motor_information</i> | structure contains motor information |

#### 6.1.4.71 get\_motor\_settings()

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

Deprecated.

Read motor settings from the EEPROM.

Parameters

|     |                       |                                   |
|-----|-----------------------|-----------------------------------|
|     | <i>id</i>             | An identifier of a device         |
| out | <i>motor_settings</i> | structure contains motor settings |

#### 6.1.4.72 get\_move\_settings()

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

Movement settings read command (speed, acceleration, threshold, etc.).

## Parameters

|     |                      |                                                                          |
|-----|----------------------|--------------------------------------------------------------------------|
|     | <i>id</i>            | An identifier of a device                                                |
| out | <i>move_settings</i> | structure contains move settings: speed, acceleration, deceleration etc. |

## 6.1.4.73 get\_move\_settings\_calb()

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

User unit movement settings read command (speed, acceleration, threshold, etc.).

## Parameters

|     |                           |                                                                          |
|-----|---------------------------|--------------------------------------------------------------------------|
|     | <i>id</i>                 | An identifier of a device                                                |
| out | <i>move_settings_calb</i> | structure contains move settings: speed, acceleration, deceleration etc. |
|     | <i>calibration</i>        | user unit settings                                                       |

## 6.1.4.74 get\_network\_settings()

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

Read network settings.

Manufacturer only. This function returns the current network settings.

See also

`net_settings_t`

## Parameters

|                          |                                         |
|--------------------------|-----------------------------------------|
| <i>DHCPEnabled</i>       | DHCP enabled (1) or not (0)             |
| <i>IPv4Address[4]</i>    | Array[4] with an IP address             |
| <i>SubnetMask[4]</i>     | Array[4] with a subnet mask address     |
| <i>DefaultGateway[4]</i> | Array[4] with a default gateway address |

6.1.4.75 `get_nonvolatile_memory()`

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

Read user data from FRAM.

Parameters

|     |                           |                                              |
|-----|---------------------------|----------------------------------------------|
|     | <i>id</i>                 | An identifier of a device                    |
| out | <i>nonvolatile_memory</i> | structure contains previously set user data. |

6.1.4.76 `get_password_settings()`

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

Read the password.

Manufacturer only. This function reads the user password for the device's web-page.

See also

`pwd_settings_t`

Parameters

|                         |                       |
|-------------------------|-----------------------|
| <i>UserPassword[20]</i> | Password for web-page |
|-------------------------|-----------------------|

6.1.4.77 `get_pid_settings()`

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

Read PID settings.

This function reads the structure containing a set of motor PID settings stored in the controller's memory. These settings specify the behavior of the PID routine for the positioner. These factors are slightly different for different positioners. All boards are supplied with the standard set of PID settings in the controller's flash memory.

See also

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

## Parameters

|     |                     |                           |
|-----|---------------------|---------------------------|
|     | <i>id</i>           | An identifier of a device |
| out | <i>pid_settings</i> | PID settings              |

## 6.1.4.78 get\_position()

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

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

## Parameters

|     |                         |                                    |
|-----|-------------------------|------------------------------------|
|     | <i>id</i>               | An identifier of a device          |
| out | <i>the_get_position</i> | structure contains motor position. |

## 6.1.4.79 get\_position\_calb()

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

Reads position value in user units for stepper motor and encoder steps for all engines.

## Parameters

|     |                              |                                    |
|-----|------------------------------|------------------------------------|
|     | <i>id</i>                    | An identifier of a device          |
| out | <i>the_get_position_calb</i> | structure contains motor position. |
|     | <i>calibration</i>           | user unit settings                 |

## Note

Attention! Some parameters of the `get_position_calb` structure are corrected by the coordinate correction table.

## 6.1.4.80 get\_power\_settings()

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

Read settings of step motor power control.

Used with a stepper motor only.

Parameters

|     |                       |                                                         |
|-----|-----------------------|---------------------------------------------------------|
|     | <i>id</i>             | An identifier of a device                               |
| out | <i>power_settings</i> | structure contains settings of step motor power control |

#### 6.1.4.81 get\_secure\_settings()

```
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 a device                           |
| out | <i>secure_settings</i> | critical parameter settings to protect the hardware |

See also

status\_t::flags

#### 6.1.4.82 get\_serial\_number()

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

Read device serial number.

Parameters

|     |                     |                           |
|-----|---------------------|---------------------------|
|     | <i>id</i>           | An identifier of a device |
| out | <i>SerialNumber</i> | serial number             |

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

```
result_t XIMC_API get_stage_information (
```

```

device_t id,
stage_information_t * stage_information )

```

Deprecated.

Read stage information from the EEPROM.

Parameters

|     |                          |                                      |
|-----|--------------------------|--------------------------------------|
|     | <i>id</i>                | An identifier of a device            |
| out | <i>stage_information</i> | structure contains stage information |

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

```

result_t XIMC_API get_stage_name (
    device_t id,
    stage_name_t * stage_name )

```

Read the user's stage name from the EEPROM.

Parameters

|     |                   |                                                          |
|-----|-------------------|----------------------------------------------------------|
|     | <i>id</i>         | An identifier of a device                                |
| out | <i>stage_name</i> | structure contains the previously set user's stage name. |

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

```

result_t XIMC_API get_stage_settings (
    device_t id,
    stage_settings_t * stage_settings )

```

Deprecated.

Read stage settings from the EEPROM.

Parameters

|     |                       |                                   |
|-----|-----------------------|-----------------------------------|
|     | <i>id</i>             | An identifier of a device         |
| out | <i>stage_settings</i> | structure contains stage settings |

#### 6.1.4.86 get\_status()

```

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](#)

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

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

Return device state.

Parameters

|     |                    |                                                                                                                                                    |
|-----|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>id</i>          | an identifier of device                                                                                                                            |
| out | <i>status</i>      | structure with snapshot of controller status                                                                                                       |
|     | <i>calibration</i> | user unit settings Calibrated device state. Useful structure that contains current controller status, including speed, position and boolean flags. |

See also

[get\\_status](#)

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

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

Read input synchronization settings.

This function reads the structure with a set of input synchronization settings, modes, periods and flags that specify the behavior of input synchronization. All boards are supplied with the standard set of these settings.

See also

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

Parameters

|     |                         |                           |
|-----|-------------------------|---------------------------|
|     | <i>id</i>               | An identifier of a device |
| out | <i>sync_in_settings</i> | synchronization settings  |

#### 6.1.4.89 `get_sync_in_settings_calb()`

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

Read input user unit synchronization settings.

This function reads the structure with a set of input synchronization settings, modes, periods and flags that specify the behavior of input synchronization. All boards are supplied with the standard set of these settings.

See also

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

Parameters

|     |                              |                           |
|-----|------------------------------|---------------------------|
|     | <i>id</i>                    | An identifier of a device |
| out | <i>sync_in_settings_calb</i> | synchronization settings  |
|     | <i>calibration</i>           | user unit settings        |

#### 6.1.4.90 `get_sync_out_settings()`

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

Read output synchronization settings.

This function reads the structure containing a set of output synchronization settings, modes, periods and flags that specify the behavior of output synchronization. All boards are supplied with the standard set of these settings.

See also

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

Parameters

|     |                          |                           |
|-----|--------------------------|---------------------------|
|     | <i>id</i>                | An identifier of a device |
| out | <i>sync_out_settings</i> | synchronization settings  |

#### 6.1.4.91 get\_sync\_out\_settings\_calb()

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

Read output user unit synchronization settings.

This function reads the structure containing a set of output synchronization settings, modes, periods and flags that specify the behavior of output synchronization. All boards are supplied with the standard set of these settings.

See also

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

Parameters

|     |                               |                           |
|-----|-------------------------------|---------------------------|
|     | <i>id</i>                     | An identifier of a device |
| out | <i>sync_out_settings_calb</i> | synchronization settings  |
|     | <i>calibration</i>            | user unit settings        |

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

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

Read UART settings.

This function reads the structure containing UART settings.

See also

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

Parameters

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

#### 6.1.4.93 goto\_firmware()

```
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. |

#### 6.1.4.94 has\_firmware()

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

Check for firmware on device.

Parameters

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

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

```
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  |

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

```
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  |

#### 6.1.4.97 msec\_sleep()

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

Sleeps for a specified amount of time.

Parameters

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

#### 6.1.4.98 open\_device()

```
device_t XIMC_API open_device (
    const char * uri )
```

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

## Parameters

|    |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | <i>uri</i> | - a device URI. Device URI has a form of "xi-com:port" or "xi-net://host/serial" or "xi-emu:///abs_path_to_file". For POSIX systems one can omit root-slash in <i>abs_path_to_file</i> ; for example, "xi-emu:///home/user/virt_controller.bin". In case of USB-COM port, the "port" is the OS device URI. For example, "xi-com:\\\\\\\\.\\\\COM3" in Windows (note that double-backslash will be transformed to single-backslash) or "xi-com:///dev/ttyACM0" in Linux/Mac. In case of network device, the "host" is an IPv4 address or fully qualified domain URI (FQDN), "serial" is the device serial number in hexadecimal system. For example, "xi-net://192.168.0.1/00001234" or "xi-net://hostname.com/89ABCDEF". In case of TCP protocol, use "xi-tcp://<ip/host>:<port>". For example, "xi-tcp://192.168.0.1:1818". In case of virtual device, the "abs_file_to_file" is the full path to the virtual device's file. If it doesn't exist, then it is created and initialized with default values. For example, "xi-emu:///C:/dir/file.bin" in Windows or "xi-emu:///home/user/file.bin" in Linux/Mac. |
|----|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 6.1.4.99 probe\_device()

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

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

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

## Parameters

|    |            |                |
|----|------------|----------------|
| in | <i>uri</i> | - a device uri |
|----|------------|----------------|

## 6.1.4.100 service\_command\_updf()

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

The command switches the controller to update the firmware state.

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

## 6.1.4.101 set\_accessories\_settings()

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

Deprecated.

Set additional accessories' information to the EEPROM. Can be used by the manufacturer only.

## Parameters

|    |                             |                                                             |
|----|-----------------------------|-------------------------------------------------------------|
|    | <i>id</i>                   | An identifier of a device                                   |
| in | <i>accessories_settings</i> | structure contains information about additional accessories |

## 6.1.4.102 set\_brake\_settings()

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

Set brake control settings.

## Parameters

|    |                       |                                           |
|----|-----------------------|-------------------------------------------|
|    | <i>id</i>             | An identifier of a device                 |
| in | <i>brake_settings</i> | structure contains brake control settings |

## 6.1.4.103 set\_calibration\_settings()

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

Set calibration settings.

Manufacturer only. This function sends the structure with calibration settings to the controller's memory. These settings are used to convert bare ADC values to winding currents in mA and the full current in mA. Parameters are grouped into pairs, XXX\_A and XXX\_B, representing linear equation coefficients. The first one is the slope, the second one is the constant term. Thus,  $XXX\_Current[mA] = XXX\_A[mA/ADC] * X_{ADC\_CODE[ADC]} + XXX\_B[mA]$ .

See also

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

## Parameters

|    |                             |                           |
|----|-----------------------------|---------------------------|
|    | <i>id</i>                   | An identifier of a device |
| in | <i>calibration_settings</i> | calibration settings      |

## 6.1.4.104 set\_control\_settings()

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

Set motor control settings.

In case of CTL\_MODE=1, joystick motor control is enabled. In this mode, the joystick is maximally displaced, the engine tends to move at MaxSpeed[i]. i=0 if another value hasn't been set at the previous usage. To change the speed index "i", use the buttons.

In case of CTL\_MODE=2, the motor is controlled by the left/right buttons. When you click on the button, the motor starts moving in the appropriate direction at a speed MaxSpeed[0]. After Timeout[i], motor moves at speed MaxSpeed[i+1]. At the transition between MaxSpeed[i] and MaxSpeed[i+1] the motor just accelerates/decelerates as usual.

Parameters

|    |                         |                                            |
|----|-------------------------|--------------------------------------------|
|    | <i>id</i>               | An identifier of a device                  |
| in | <i>control_settings</i> | structure contains motor control settings. |

## 6.1.4.105 set\_control\_settings\_calb()

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

Set motor control settings with user units.

In case of CTL\_MODE=1, joystick motor control is enabled. In this mode, while the joystick is maximally displaced, the engine tends to move at MaxSpeed[i]. i=0 if another value hasn't been set at the previous usage. To change the speed index "i", use the buttons.

In case of CTL\_MODE=2, the motor is controlled by the left/right buttons. When you click on the button, the motor starts moving in the appropriate direction at a speed MaxSpeed[0]. After Timeout[i], motor moves at speed MaxSpeed[i+1]. At the transition between MaxSpeed[i] and MaxSpeed[i+1] the motor just accelerates/decelerates as usual.

Parameters

|    |                              |                                            |
|----|------------------------------|--------------------------------------------|
|    | <i>id</i>                    | An identifier of a device                  |
| in | <i>control_settings_calb</i> | structure contains motor control settings. |
|    | <i>calibration</i>           | user unit settings                         |

6.1.4.106 `set_controller_name()`

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

Write user's controller name and internal settings to the FRAM.

Parameters

|    |                        |                                                              |
|----|------------------------|--------------------------------------------------------------|
|    | <i>id</i>              | An identifier of a device                                    |
| in | <i>controller_name</i> | structure contains the previously set user's controller name |

6.1.4.107 `set_correction_table()`

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

Command of loading a correction table from a text file.

The correction table is used for position correction in case of mechanical inaccuracies. It works for some parameters in `_calb` commands.

Parameters

|    |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | <i>id</i>       | an identifier the device                                                                                                                                                                                                                                                                                                                                                                                                                        |
| in | <i>namefile</i> | - the file name must be either a full path or a relative path. If the file name is set to NULL, the correction table will be cleared. File format: two tab-separated columns. Column headers are strings. Data is real, the dot is a delimiter. The first column is a coordinate. The second one is the deviation caused by a mechanical error. The maximum length of a table is 100 rows. Coordinate column must be sorted in ascending order. |

See also

[command\\_move](#)  
[get\\_position\\_calb](#)  
[get\\_position\\_calb\\_t](#)  
[get\\_status\\_calb](#)  
[status\\_calb\\_t](#)  
[get\\_edges\\_settings\\_calb](#)  
[set\\_edges\\_settings\\_calb](#)  
[edges\\_settings\\_calb\\_t](#)

## 6.1.4.108 set\_ctp\_settings()

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

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

When controlling the step motor with the 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 is enabled (CTP\_ENABLED is set), the controller stores the current position in the steps of SM and the current position of the encoder. Next, the encoder position is converted into steps at each step, and if the difference between the current position in steps and the encoder position is greater than CTPMinError, the flag STATE\_CTP\_ERROR is set.

Alternatively, the stepper motor may be controlled with the speed sensor (CTP\_BASE 1). In this mode, at the active edges of the input clock, the controller stores the current value of steps. Then, at each revolution, the controller checks how many steps have been passed. When the difference is over the CTPMinError, the STATE\_CTP\_ERROR flag is set.

Parameters

|    |                     |                                               |
|----|---------------------|-----------------------------------------------|
|    | <i>id</i>           | An identifier of a device                     |
| in | <i>ctp_settings</i> | structure contains position control settings. |

## 6.1.4.109 set\_debug\_write()

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

Write data to firmware for debug purpose.

Manufacturer only.

Parameters

|    |                    |                           |
|----|--------------------|---------------------------|
|    | <i>id</i>          | An identifier of a device |
| in | <i>debug_write</i> | Debug data.               |

## 6.1.4.110 set\_edges\_settings()

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

Set border and limit switches settings.

See also

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

Parameters

|    |                       |                                                                                                    |
|----|-----------------------|----------------------------------------------------------------------------------------------------|
|    | <i>id</i>             | An identifier of a device                                                                          |
| in | <i>edges_settings</i> | edges settings, specify types of borders, motor behavior and electrical behavior of limit switches |

#### 6.1.4.111 set\_edges\_settings\_calb()

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

Set border and limit switches settings in user units.

See also

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

Parameters

|    |                            |                                                                                                    |
|----|----------------------------|----------------------------------------------------------------------------------------------------|
|    | <i>id</i>                  | An identifier of a device                                                                          |
| in | <i>edges_settings_calb</i> | edges settings, specify types of borders, motor behavior and electrical behavior of limit switches |
|    | <i>calibration</i>         | user unit settings                                                                                 |

Note

Attention! Some parameters of the edges\_settings\_calb structure are corrected by the coordinate correction table.

#### 6.1.4.112 set\_emf\_settings()

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

Set electromechanical coefficients.

The settings are different for different stepper motors. Please set new settings when you change the motor.

See also

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

Parameters

|    |                     |                           |
|----|---------------------|---------------------------|
|    | <i>id</i>           | An identifier of a device |
| in | <i>emf_settings</i> | EMF settings              |

#### 6.1.4.113 set\_encoder\_information()

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

Deprecated.

Set encoder information to the EEPROM. Can be used by the manufacturer only.

Parameters

|    |                            |                                              |
|----|----------------------------|----------------------------------------------|
|    | <i>id</i>                  | An identifier of a device                    |
| in | <i>encoder_information</i> | structure contains information about encoder |

#### 6.1.4.114 set\_encoder\_settings()

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

Deprecated.

Set encoder settings to the EEPROM. Can be used by the manufacturer only.

Parameters

|    |                         |                                     |
|----|-------------------------|-------------------------------------|
|    | <i>id</i>               | An identifier of a device           |
| in | <i>encoder_settings</i> | structure contains encoder settings |

#### 6.1.4.115 set\_engine\_advanced\_setup()

```
result_t XIMC_API set_engine_advanced_setup (
    device_t id,
    const engine_advanced_setup_t * engine_advanced_setup )
```

Set engine advanced settings.

Manufacturer only.

See also

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

Parameters

|    |                              |                           |
|----|------------------------------|---------------------------|
|    | <i>id</i>                    | An identifier of a device |
| in | <i>engine_advanced_setup</i> | EAS settings              |

#### 6.1.4.116 set\_engine\_settings()

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

Set engine settings.

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

See also

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

Parameters

|    |                        |                           |
|----|------------------------|---------------------------|
|    | <i>id</i>              | An identifier of a device |
| in | <i>engine_settings</i> | engine settings           |

#### 6.1.4.117 set\_engine\_settings\_calb()

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

Set user unit engine settings.

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

See also

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

## Parameters

|    |                             |                           |
|----|-----------------------------|---------------------------|
|    | <i>id</i>                   | An identifier of a device |
| in | <i>engine_settings_calb</i> | engine settings           |
|    | <i>calibration</i>          | user unit settings        |

## 6.1.4.118 set\_entype\_settings()

```
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 a device                                    |
| in | <i>entype_settings</i> | structure contains motor type and power driver type settings |

## 6.1.4.119 set\_extended\_settings()

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

Set extended settings.

Currently, it is not in use.

See also

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

## Parameters

|    |                          |                           |
|----|--------------------------|---------------------------|
|    | <i>id</i>                | An identifier of a device |
| in | <i>extended_settings</i> | EST settings              |

## 6.1.4.120 set\_extio\_settings()

```
result_t XIMC_API set_extio_settings (
```

```
device_t id,
const extio_settings_t * extio_settings )
```

Set EXTIO settings.

This function sends the structure with a set of EXTIO settings to the controller's memory. By default, input events are signaled through a rising front, and output states are signaled by a high logic state.

See also

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

Parameters

|    |                       |                           |
|----|-----------------------|---------------------------|
|    | <i>id</i>             | An identifier of a device |
| in | <i>extio_settings</i> | EXTIO settings            |

#### 6.1.4.121 set\_feedback\_settings()

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

Feedback settings.

Parameters

|    |                      |                                                                                                                                                                                                                                            |
|----|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | <i>id</i>            | An identifier of a device                                                                                                                                                                                                                  |
| in | <i>IPS</i>           | number of encoder counts per shaft revolution. Range: 1..65535. The field is obsolete, it is recommended to write 0 to IPS and use the extended CountsPerTurn field. You may need to update the controller firmware to the latest version. |
| in | <i>FeedbackType</i>  | type of feedback                                                                                                                                                                                                                           |
| in | <i>FeedbackFlags</i> | flags of feedback                                                                                                                                                                                                                          |
| in | <i>CountsPerTurn</i> | number of encoder counts per shaft revolution. Range: 1..4294967295. To use the CountsPerTurn field, write 0 in the IPS field, otherwise the value from the IPS field will be used.                                                        |

#### 6.1.4.122 set\_gear\_information()

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

Deprecated.

Set gear information to the EEPROM. Can be used by the manufacturer only.

Parameters

|    |                         |                                                    |
|----|-------------------------|----------------------------------------------------|
|    | <i>id</i>               | An identifier of a device                          |
| in | <i>gear_information</i> | structure contains information about step gearhead |

#### 6.1.4.123 set\_gear\_settings()

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

Deprecated.

Set gear settings to the EEPROM. Can be used by the manufacturer only.

Parameters

|    |                      |                                           |
|----|----------------------|-------------------------------------------|
|    | <i>id</i>            | An identifier of a device                 |
| in | <i>gear_settings</i> | structure contains step gearhead settings |

#### 6.1.4.124 set\_hallsensor\_information()

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

Deprecated.

Set hall sensor information to the EEPROM. Can be used by the manufacturer only.

Parameters

|    |                               |                                                  |
|----|-------------------------------|--------------------------------------------------|
|    | <i>id</i>                     | An identifier of a device                        |
| in | <i>hallsensor_information</i> | structure contains information about hall sensor |

#### 6.1.4.125 set\_hallsensor\_settings()

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

Deprecated.

Set hall sensor settings to the EEPROM. Can be used by the manufacturer only.

## Parameters

|    |                            |                                         |
|----|----------------------------|-----------------------------------------|
|    | <i>id</i>                  | An identifier of a device               |
| in | <i>hallsensor_settings</i> | structure contains hall sensor settings |

## 6.1.4.126 set\_home\_settings()

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

Set home settings.

This function sends home position structure to the controller's memory.

See also

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

## Parameters

|    |                      |                               |
|----|----------------------|-------------------------------|
|    | <i>id</i>            | An identifier of a device     |
| in | <i>home_settings</i> | calibrating position settings |

## 6.1.4.127 set\_home\_settings\_calb()

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

Set user unit home settings.

This function sends home position structure to the controller's memory.

See also

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

## Parameters

|    |                           |                               |
|----|---------------------------|-------------------------------|
|    | <i>id</i>                 | An identifier of a device     |
| in | <i>home_settings_calb</i> | calibrating position settings |
|    | <i>calibration</i>        | user unit settings            |

## 6.1.4.128 set\_joystick\_settings()

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

Set joystick position.

If joystick position falls outside DeadZone limits, a movement begins. The speed is defined by the joystick's position in the range from the DeadZone limit to the maximum deviation. Joystick positions inside DeadZone limits correspond to zero speed (a "soft stop" command is issued continuously), and positions beyond Low and High limits correspond to MaxSpeed[i] or -MaxSpeed[i] (see command SCTL), where  $i = 0$  by default and can be changed with the left/right buttons (see command SCTL). If the next speed in the list is zero (both integer and microstep parts), the button press is ignored. The first speed in the list shouldn't be zero. DeadZone is defined in 0.1% units. See the "Joystick control" section for more information.

Parameters

|    |                          |                                      |
|----|--------------------------|--------------------------------------|
|    | <i>id</i>                | An identifier of a device            |
| in | <i>joystick_settings</i> | structure contains joystick settings |

## 6.1.4.129 set\_logging\_callback()

```
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 |
|-------------------------|-----------------------------|

## 6.1.4.130 set\_motor\_information()

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

Deprecated.

Set motor information to the EEPROM. Can be used by the manufacturer only.

## Parameters

|    |                          |                                      |
|----|--------------------------|--------------------------------------|
|    | <i>id</i>                | An identifier of a device            |
| in | <i>motor_information</i> | structure contains motor information |

## 6.1.4.131 set\_motor\_settings()

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

Deprecated.

Set motor settings to the EEPROM. Can be used by the manufacturer only.

## Parameters

|    |                       |                                      |
|----|-----------------------|--------------------------------------|
|    | <i>id</i>             | An identifier of a device            |
| in | <i>motor_settings</i> | structure contains motor information |

## 6.1.4.132 set\_move\_settings()

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

Movement settings set command (speed, acceleration, threshold, etc.).

## Parameters

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

## 6.1.4.133 set\_move\_settings\_calb()

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

User unit movement settings set command (speed, acceleration, threshold, etc.).

## Parameters

|    |                           |                                                                          |
|----|---------------------------|--------------------------------------------------------------------------|
|    | <i>id</i>                 | An identifier of a device                                                |
| in | <i>move_settings_calb</i> | structure contains move settings: speed, acceleration, deceleration etc. |
|    | <i>calibration</i>        | user unit settings                                                       |

## 6.1.4.134 set\_network\_settings()

```
result_t XIMC_API set_network_settings (
    device_t id,
    const network_settings_t * network_settings )
```

Set network settings.

Manufacturer only. This function sets the desired network settings.

See also

net\_settings\_t

## Parameters

|                          |                                         |
|--------------------------|-----------------------------------------|
| <i>DHCPEnabled</i>       | DHCP enabled (1) or not (0)             |
| <i>IPv4Address[4]</i>    | Array[4] with an IP address             |
| <i>SubnetMask[4]</i>     | Array[4] with a subnet mask address     |
| <i>DefaultGateway[4]</i> | Array[4] with a default gateway address |

## 6.1.4.135 set\_nonvolatile\_memory()

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

Write user data into the FRAM.

## Parameters

|    |                           |                           |
|----|---------------------------|---------------------------|
|    | <i>id</i>                 | An identifier of a device |
| in | <i>nonvolatile_memory</i> | user data.                |

## 6.1.4.136 set\_password\_settings()

```
result_t XIMC_API set_password_settings (
    device_t id,
    const password_settings_t * password_settings )
```

Sets the password.

Manufacturer only. This function sets the user password for the device's web-page.

See also

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

Parameters

|                         |                       |
|-------------------------|-----------------------|
| <i>UserPassword[20]</i> | Password for web-page |
|-------------------------|-----------------------|

## 6.1.4.137 set\_pid\_settings()

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

Set PID settings.

This function sends the structure with a set of PID factors to the controller's memory. These settings specify the behavior of the PID routine for the positioner. These factors are slightly different for different positioners. All boards are supplied with the standard set of PID settings in the 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 a device |
| in | <i>pid_settings</i> | PID settings              |

## 6.1.4.138 set\_position()

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

Sets position in steps and microsteps for stepper motor.

Sets encoder position for all engines.

Parameters

|     |                         |                                    |
|-----|-------------------------|------------------------------------|
|     | <i>id</i>               | An identifier of a device          |
| out | <i>the_set_position</i> | structure contains motor position. |

#### 6.1.4.139 set\_position\_calb()

```
result_t XIMC_API set_position_calb (
    device_t id,
    const set_position_calb_t * the_set_position_calb,
    const calibration_t * calibration )
```

Sets any position value and encoder value of all engines.

In user units.

Parameters

|     |                              |                                    |
|-----|------------------------------|------------------------------------|
|     | <i>id</i>                    | An identifier of a device          |
| out | <i>the_set_position_calb</i> | structure contains motor position. |
|     | <i>calibration</i>           | user unit settings                 |

#### 6.1.4.140 set\_power\_settings()

```
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 a stepper motor only.

Parameters

|    |                       |                                                         |
|----|-----------------------|---------------------------------------------------------|
|    | <i>id</i>             | An identifier of a device                               |
| in | <i>power_settings</i> | structure contains settings of step motor power control |

## 6.1.4.141 set\_secure\_settings()

```
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 a device  |
| <i>secure_settings</i> | structure with secure data |

See also

status\_t::flags

## 6.1.4.142 set\_serial\_number()

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

Write device serial number and hardware version to the 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 the manufacturer only. Used from the loader only.

Parameters

|    |                      |                                                      |
|----|----------------------|------------------------------------------------------|
|    | <i>id</i>            | An identifier of a device                            |
| in | <i>serial_number</i> | structure contains new serial number and secret key. |

## 6.1.4.143 set\_stage\_information()

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

Deprecated.

Set stage information to the EEPROM. Can be used by the manufacturer only.

Parameters

|    |                          |                                      |
|----|--------------------------|--------------------------------------|
|    | <i>id</i>                | An identifier of a device            |
| in | <i>stage_information</i> | structure contains stage information |

#### 6.1.4.144 set\_stage\_name()

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

Write the user's stage name to EEPROM.

Parameters

|    |                   |                                                          |
|----|-------------------|----------------------------------------------------------|
|    | <i>id</i>         | An identifier of a device                                |
| in | <i>stage_name</i> | structure contains the previously set user's stage name. |

#### 6.1.4.145 set\_stage\_settings()

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

Deprecated.

Set stage settings to the EEPROM. Can be used by the manufacturer only

Parameters

|    |                       |                                   |
|----|-----------------------|-----------------------------------|
|    | <i>id</i>             | An identifier of a device         |
| in | <i>stage_settings</i> | structure contains stage settings |

#### 6.1.4.146 set\_sync\_in\_settings()

```
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 sends the structure with a set of input synchronization settings that specify the behavior of input synchronization to the controller's memory. All boards are supplied with the standard set of these settings.

See also

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

Parameters

|    |                         |                           |
|----|-------------------------|---------------------------|
|    | <i>id</i>               | An identifier of a device |
| in | <i>sync_in_settings</i> | synchronization settings  |

#### 6.1.4.147 set\_sync\_in\_settings\_calb()

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

Set input user unit synchronization settings.

This function sends the structure with a set of input synchronization settings that specify the behavior of input synchronization to the controller's memory. All boards are supplied with the standard set of these settings.

See also

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

Parameters

|    |                              |                           |
|----|------------------------------|---------------------------|
|    | <i>id</i>                    | An identifier of a device |
| in | <i>sync_in_settings_calb</i> | synchronization settings  |
|    | <i>calibration</i>           | user unit settings        |

#### 6.1.4.148 set\_sync\_out\_settings()

```
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 sends the structure with a set of output synchronization settings that specify the behavior of output synchronization to the controller's memory. All boards are supplied with the standard set of these settings.

See also

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

## Parameters

|    |                          |                           |
|----|--------------------------|---------------------------|
|    | <i>id</i>                | An identifier of a device |
| in | <i>sync_out_settings</i> | synchronization settings  |

## 6.1.4.149 set\_sync\_out\_settings\_calb()

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

Set output user unit synchronization settings.

This function sends the structure with a set of output synchronization settings that specify the behavior of output synchronization to the controller's memory. All boards are supplied with the standard set of these settings.

See also

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

## Parameters

|    |                               |                           |
|----|-------------------------------|---------------------------|
|    | <i>id</i>                     | An identifier of a device |
| in | <i>sync_out_settings_calb</i> | synchronization settings  |
|    | <i>calibration</i>            | user unit settings        |

## 6.1.4.150 set\_uart\_settings()

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

Set UART settings.

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

See also

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

Parameters

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

#### 6.1.4.151 write\_key()

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

Write controller key.

Can be used by manufacturer only

Parameters

|    |            |                                      |
|----|------------|--------------------------------------|
|    | <i>uri</i> | a uri of device                      |
| in | <i>key</i> | protection key. Range: 0..4294967295 |

#### 6.1.4.152 ximc\_version()

```
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

## A

- calibration.t, [23](#)
- A1Voltage
  - analog\_data.t, [14](#)
- A1Voltage\_ADC
  - analog\_data.t, [14](#)
- A2Voltage
  - analog\_data.t, [15](#)
- A2Voltage\_ADC
  - analog\_data.t, [15](#)
- ACurrent
  - analog\_data.t, [15](#)
- ACurrent\_ADC
  - analog\_data.t, [15](#)
- ALARM\_FLAGS\_STICKING
  - ximc.h, [132](#)
- ALARM\_ON\_BORDERS\_SWAP\_MISSET
  - ximc.h, [132](#)
- ALARM\_ON\_DRIVER\_OVERHEATING
  - ximc.h, [132](#)
- Accel
  - move\_settings\_calb.t, [74](#)
  - move\_settings.t, [75](#)
- accessories\_settings.t, [10](#)
  - LimitSwitchesSettings, [11](#)
  - MBRatedCurrent, [11](#)
  - MBRatedVoltage, [11](#)
  - MBSettings, [11](#)
  - MBTorque, [12](#)
  - MagneticBrakeInfo, [11](#)
  - TSGrad, [12](#)
  - TSMMax, [12](#)
  - TSMIn, [12](#)
  - TSSettings, [12](#)
  - TemperatureSensorInfo, [12](#)
- Accuracy
  - sync\_out\_settings\_calb.t, [103](#)
  - sync\_out\_settings.t, [105](#)
- analog\_data.t, [13](#)
  - A1Voltage, [14](#)
  - A1Voltage\_ADC, [14](#)
  - A2Voltage, [15](#)
  - A2Voltage\_ADC, [15](#)
  - ACurrent, [15](#)
  - ACurrent\_ADC, [15](#)
  - B1Voltage, [15](#)
  - B1Voltage\_ADC, [15](#)
  - B2Voltage, [16](#)
  - B2Voltage\_ADC, [16](#)
  - BCurrent, [16](#)
  - BCurrent\_ADC, [16](#)
  - Enc\_Check, [16](#)
  - Enc\_Check\_ADC, [16](#)
  - FullCurrent, [17](#)
  - FullCurrent\_ADC, [17](#)
  - Joy, [17](#)
  - Joy\_ADC, [17](#)
  - L, [17](#)
  - Pot, [17](#)
  - R, [18](#)
  - SupVoltage, [18](#)
  - SupVoltage\_ADC, [18](#)
  - Temp, [18](#)
  - Temp\_ADC, [18](#)
- Antiplay
  - engine\_settings\_calb.t, [44](#)
  - engine\_settings.t, [46](#)
- AntiplaySpeed
  - move\_settings\_calb.t, [74](#)
  - move\_settings.t, [76](#)
- AveragedPowerRatio
  - chart\_data.t, [24](#)
- B1Voltage
  - analog\_data.t, [15](#)
- B1Voltage\_ADC
  - analog\_data.t, [15](#)
- B2Voltage
  - analog\_data.t, [16](#)
- B2Voltage\_ADC
  - analog\_data.t, [16](#)
- BACK\_EMF\_INDUCTANCE\_AUTO
  - ximc.h, [133](#)
- BACK\_EMF\_KM\_AUTO
  - ximc.h, [133](#)
- BACK\_EMF\_RESISTANCE\_AUTO
  - ximc.h, [133](#)
- BCurrent
  - analog\_data.t, [16](#)
- BCurrent\_ADC
  - analog\_data.t, [16](#)
- BORDER\_IS\_ENCODER
  - ximc.h, [133](#)
- BORDER\_STOP\_LEFT
  - ximc.h, [133](#)

BORDER\_STOP\_RIGHT  
ximc.h, [133](#)

BORDERS\_SWAP\_MISSET\_DETECTION  
ximc.h, [134](#)

BRAKE\_ENABLED  
ximc.h, [134](#)

BRAKE\_ENG\_PWROFF  
ximc.h, [134](#)

BRAKING\_OVERVOLTAGE\_PROTECTION  
ximc.h, [134](#)

BackEMFFlags  
emf\_settings\_t, [38](#)

BorderFlags  
edges\_settings\_calb\_t, [35](#)  
edges\_settings\_t, [37](#)

brake\_settings\_t, [19](#)  
BrakeFlags, [19](#)  
t1, [19](#)  
t2, [19](#)  
t3, [20](#)  
t4, [20](#)

BrakeFlags  
brake\_settings\_t, [19](#)

CONTROL\_BTN\_LEFT\_PUSHED\_OPEN  
ximc.h, [134](#)

CONTROL\_BTN\_RIGHT\_PUSHED\_OPEN  
ximc.h, [134](#)

CONTROL\_MODE\_BITS  
ximc.h, [135](#)

CONTROL\_MODE\_JOY  
ximc.h, [135](#)

CONTROL\_MODE\_LR  
ximc.h, [135](#)

CONTROL\_MODE\_OFF  
ximc.h, [135](#)

CSS1\_A  
calibration\_settings\_t, [21](#)

CSS1\_B  
calibration\_settings\_t, [21](#)

CSS2\_A  
calibration\_settings\_t, [21](#)

CSS2\_B  
calibration\_settings\_t, [21](#)

CTP\_ALARM\_ON\_ERROR  
ximc.h, [135](#)

CTP\_BASE  
ximc.h, [135](#)

CTP\_ENABLED  
ximc.h, [136](#)

CTP\_ERROR\_CORRECTION  
ximc.h, [136](#)

CTPFlags  
ctp\_settings\_t, [31](#)

CTPMinError  
ctp\_settings\_t, [31](#)

calibration\_settings\_t, [20](#)  
CSS1\_A, [21](#)  
CSS1\_B, [21](#)  
CSS2\_A, [21](#)  
CSS2\_B, [21](#)  
FullCurrent\_A, [21](#)  
FullCurrent\_B, [22](#)

calibration\_t, [22](#)  
A, [23](#)  
ximc.h, [163](#)

chart\_data\_t, [23](#)  
AveragedPowerRatio, [24](#)  
Joy, [24](#)  
Pot, [25](#)  
WindingCurrentA, [25](#)  
WindingCurrentB, [25](#)  
WindingCurrentC, [25](#)  
WindingVoltageA, [25](#)  
WindingVoltageB, [25](#)  
WindingVoltageC, [26](#)

close\_device  
ximc.h, [164](#)

ClutterTime  
sync\_in\_settings\_calb\_t, [100](#)  
sync\_in\_settings\_t, [102](#)

CmdBufFreeSpace  
status\_calb\_t, [93](#)  
status\_t, [97](#)

command\_clear\_fram  
ximc.h, [165](#)

command\_eeread\_settings  
ximc.h, [165](#)

command\_eesave\_settings  
ximc.h, [165](#)

command\_home  
ximc.h, [166](#)

command\_homezero  
ximc.h, [166](#)

command\_left  
ximc.h, [167](#)

command\_loft  
ximc.h, [167](#)

command\_move  
ximc.h, [167](#)

command\_move\_calb  
ximc.h, [168](#)

command\_movr  
ximc.h, [168](#)

command\_movr\_calb  
ximc.h, [168](#)

command\_power\_off  
ximc.h, [169](#)

command\_read\_robust\_settings  
ximc.h, [169](#)

command\_read\_settings  
ximc.h, [170](#)

- command\_reset
  - ximc.h, [170](#)
- command\_right
  - ximc.h, [170](#)
- command\_save\_robust\_settings
  - ximc.h, [171](#)
- command\_save\_settings
  - ximc.h, [171](#)
- command\_sstp
  - ximc.h, [171](#)
- command\_start\_measurements
  - ximc.h, [172](#)
- command\_stop
  - ximc.h, [172](#)
- command\_update\_firmware
  - ximc.h, [172](#)
- command\_wait\_for\_stop
  - ximc.h, [173](#)
- command\_zero
  - ximc.h, [173](#)
- control\_settings\_calb\_t, [26](#)
  - Flags, [27](#)
  - MaxClickTime, [27](#)
  - MaxSpeed, [27](#)
  - Timeout, [27](#)
- control\_settings\_t, [27](#)
  - Flags, [28](#)
  - MaxClickTime, [28](#)
  - MaxSpeed, [29](#)
  - Timeout, [29](#)
  - uDeltaPosition, [29](#)
  - uMaxSpeed, [29](#)
- controller\_name\_t, [29](#)
  - ControllerName, [30](#)
  - CtrlFlags, [30](#)
- ControllerName
  - controller\_name\_t, [30](#)
- CountsPerTurn
  - feedback\_settings\_t, [51](#)
- CriticalIpwr
  - secure\_settings\_t, [82](#)
- CriticalIusb
  - secure\_settings\_t, [83](#)
- CriticalUpwr
  - secure\_settings\_t, [83](#)
- CriticalUusb
  - secure\_settings\_t, [83](#)
- CriticalT
  - secure\_settings\_t, [83](#)
- ctp\_settings\_t, [30](#)
  - CTPFlags, [31](#)
  - CTPMinError, [31](#)
- CtrlFlags
  - controller\_name\_t, [30](#)
- CurPosition
  - status\_calb\_t, [93](#)
  - status\_t, [97](#)
- CurSpeed
  - status\_calb\_t, [93](#)
  - status\_t, [97](#)
- CurrReductDelay
  - power\_settings\_t, [81](#)
- CurrentSetTime
  - power\_settings\_t, [81](#)
- CurT
  - status\_calb\_t, [93](#)
  - status\_t, [97](#)
- DHCPEnabled
  - network\_settings\_t, [77](#)
- DRIVER\_TYPE\_EXTERNAL
  - ximc.h, [136](#)
- DRIVER\_TYPE\_INTEGRATE
  - ximc.h, [136](#)
- DeadZone
  - joystick\_settings\_t, [65](#)
- debug\_read\_t, [31](#)
  - DebugData, [32](#)
- debug\_write\_t, [32](#)
  - DebugData, [33](#)
- DebugData
  - debug\_read\_t, [32](#)
  - debug\_write\_t, [33](#)
- Decel
  - move\_settings\_calb\_t, [74](#)
  - move\_settings\_t, [76](#)
- DefaultGateway
  - network\_settings\_t, [77](#)
- DetentTorque
  - motor\_settings\_t, [69](#)
- device\_information\_t, [33](#)
  - Major, [33](#)
  - Minor, [34](#)
  - Release, [34](#)
- device\_network\_information\_t, [34](#)
- DriverType
  - entype\_settings\_t, [48](#)
- EEPROM\_PRECEDENCE
  - ximc.h, [136](#)
- ENC\_STATE\_ABSENT
  - ximc.h, [136](#)
- ENC\_STATE\_MALFUNC
  - ximc.h, [137](#)
- ENC\_STATE\_OK
  - ximc.h, [137](#)
- ENC\_STATE\_REVERS
  - ximc.h, [137](#)
- ENC\_STATE\_UNKNOWN
  - ximc.h, [137](#)
- ENDER\_SW1\_ACTIVE\_LOW
  - ximc.h, [137](#)
- ENDER\_SW2\_ACTIVE\_LOW

- ximc.h, [137](#)
- ENDER\_SWAP
  - ximc.h, [138](#)
- ENGINE\_ACCEL\_ON
  - ximc.h, [138](#)
- ENGINE\_ANTIPLAY
  - ximc.h, [138](#)
- ENGINE\_CURRENT\_AS\_RMS
  - ximc.h, [138](#)
- ENGINE\_LIMIT\_CURR
  - ximc.h, [138](#)
- ENGINE\_LIMIT\_RPM
  - ximc.h, [138](#)
- ENGINE\_LIMIT\_VOLT
  - ximc.h, [139](#)
- ENGINE\_MAX\_SPEED
  - ximc.h, [139](#)
- ENGINE\_REVERSE
  - ximc.h, [139](#)
- ENGINE\_TYPE\_2DC
  - ximc.h, [139](#)
- ENGINE\_TYPE\_BRUSHLESS
  - ximc.h, [139](#)
- ENGINE\_TYPE\_DC
  - ximc.h, [139](#)
- ENGINE\_TYPE\_NONE
  - ximc.h, [140](#)
- ENGINE\_TYPE\_STEP
  - ximc.h, [140](#)
- ENGINE\_TYPE\_TEST
  - ximc.h, [140](#)
- ENGINE\_USE\_PEAK\_CURRENT
  - ximc.h, [140](#)
- ENUMERATE\_PROBE
  - ximc.h, [140](#)
- EXTIO\_SETUP\_INVERT
  - ximc.h, [140](#)
- EXTIO\_SETUP\_MODE\_IN\_ALARM
  - ximc.h, [141](#)
- EXTIO\_SETUP\_MODE\_IN\_BITS
  - ximc.h, [141](#)
- EXTIO\_SETUP\_MODE\_IN\_HOME
  - ximc.h, [141](#)
- EXTIO\_SETUP\_MODE\_IN\_MOVR
  - ximc.h, [141](#)
- EXTIO\_SETUP\_MODE\_IN\_NOP
  - ximc.h, [141](#)
- EXTIO\_SETUP\_MODE\_IN\_PWOF
  - ximc.h, [141](#)
- EXTIO\_SETUP\_MODE\_IN\_STOP
  - ximc.h, [142](#)
- EXTIO\_SETUP\_MODE\_OUT\_ALARM
  - ximc.h, [142](#)
- EXTIO\_SETUP\_MODE\_OUT\_BITS
  - ximc.h, [142](#)
- EXTIO\_SETUP\_MODE\_OUT\_MOTOR\_ON
  - ximc.h, [142](#)
- EXTIO\_SETUP\_MODE\_OUT\_MOVING
  - ximc.h, [142](#)
- EXTIO\_SETUP\_MODE\_OUT\_OFF
  - ximc.h, [142](#)
- EXTIO\_SETUP\_MODE\_OUT\_ON
  - ximc.h, [143](#)
- EXTIO\_SETUP\_OUTPUT
  - ximc.h, [143](#)
- EXTIOModeFlags
  - extio\_settings\_t, [50](#)
- EXTIOSetupFlags
  - extio\_settings\_t, [50](#)
- edges\_settings\_calb\_t, [35](#)
  - BorderFlags, [35](#)
  - EnderFlags, [35](#)
  - LeftBorder, [35](#)
  - RightBorder, [36](#)
- edges\_settings\_t, [36](#)
  - BorderFlags, [37](#)
  - EnderFlags, [37](#)
  - LeftBorder, [37](#)
  - RightBorder, [37](#)
  - uLeftBorder, [37](#)
  - uRightBorder, [37](#)
- Efficiency
  - gear\_settings\_t, [53](#)
- emf\_settings\_t, [38](#)
  - BackEMFFlags, [38](#)
  - Km, [39](#)
  - L, [39](#)
  - R, [39](#)
- Enc\_Check
  - analog\_data\_t, [16](#)
- Enc\_Check\_ADC
  - analog\_data\_t, [16](#)
- EncPosition
  - get\_position\_calb\_t, [55](#)
  - get\_position\_t, [56](#)
  - set\_position\_calb\_t, [86](#)
  - set\_position\_t, [87](#)
  - status\_calb\_t, [93](#)
  - status\_t, [97](#)
- EncSts
  - status\_calb\_t, [94](#)
  - status\_t, [97](#)
- encoder\_information\_t, [39](#)
  - Manufacturer, [40](#)
  - PartNumber, [40](#)
- encoder\_settings\_t, [40](#)
  - EncoderSettings, [41](#)
  - MaxCurrentConsumption, [41](#)
  - MaxOperatingFrequency, [41](#)
  - SupplyVoltageMax, [41](#)
  - SupplyVoltageMin, [41](#)
- EncoderSettings

- encoder\_settings\_t, [41](#)
- EnderFlags
  - edges\_settings\_calb\_t, [35](#)
  - edges\_settings\_t, [37](#)
- engine\_advanced\_setup\_t, [42](#)
  - stepcloseloop\_Kp\_high, [42](#)
  - stepcloseloop\_Kp\_low, [42](#)
  - stepcloseloop\_Kw, [43](#)
- engine\_settings\_calb\_t, [43](#)
  - Antiplay, [44](#)
  - EngineFlags, [44](#)
  - MicrostepMode, [44](#)
  - NomCurrent, [44](#)
  - NomSpeed, [45](#)
  - NomVoltage, [45](#)
  - PeakCurrent, [45](#)
  - StepsPerRev, [45](#)
- engine\_settings\_t, [45](#)
  - Antiplay, [46](#)
  - EngineFlags, [46](#)
  - MicrostepMode, [47](#)
  - NomCurrent, [47](#)
  - NomSpeed, [47](#)
  - NomVoltage, [47](#)
  - PeakCurrent, [47](#)
  - StepsPerRev, [47](#)
  - uNomSpeed, [48](#)
- EngineFlags
  - engine\_settings\_calb\_t, [44](#)
  - engine\_settings\_t, [46](#)
- EngineType
  - entype\_settings\_t, [49](#)
- entype\_settings\_t, [48](#)
  - DriverType, [48](#)
  - EngineType, [49](#)
- enumerate\_devices
  - ximc.h, [173](#)
- Error
  - measurements\_t, [66](#)
- ExpFactor
  - joystick\_settings\_t, [65](#)
- extended\_settings\_t, [49](#)
- extio\_settings\_t, [49](#)
  - EXTIOModeFlags, [50](#)
  - EXTIOSetupFlags, [50](#)
- FEEDBACK\_EMF
  - ximc.h, [143](#)
- FEEDBACK\_ENC\_ADAPTIVE\_HOLDING
  - ximc.h, [143](#)
- FEEDBACK\_ENC\_FILTER\_BITS
  - ximc.h, [143](#)
- FEEDBACK\_ENC\_FILTER\_MEDIUM
  - ximc.h, [143](#)
- FEEDBACK\_ENC\_FILTER\_NONE
  - ximc.h, [144](#)
- FEEDBACK\_ENC\_FILTER\_STRONG
  - ximc.h, [144](#)
- FEEDBACK\_ENC\_FILTER\_WEAK
  - ximc.h, [144](#)
- FEEDBACK\_ENC\_REVERSE
  - ximc.h, [144](#)
- FEEDBACK\_ENC\_TYPE\_AUTO
  - ximc.h, [144](#)
- FEEDBACK\_ENC\_TYPE\_BITS
  - ximc.h, [144](#)
- FEEDBACK\_ENC\_TYPE\_DIFFERENTIAL
  - ximc.h, [145](#)
- FEEDBACK\_ENC\_TYPE\_SINGLE\_ENDED
  - ximc.h, [145](#)
- FEEDBACK\_ENCODER\_MEDIATED
  - ximc.h, [145](#)
- FEEDBACK\_ENCODER
  - ximc.h, [145](#)
- FEEDBACK\_NONE
  - ximc.h, [145](#)
- FastHome
  - home\_settings\_calb\_t, [60](#)
  - home\_settings\_t, [62](#)
- feedback\_settings\_t, [50](#)
  - CountsPerTurn, [51](#)
  - FeedbackFlags, [51](#)
  - FeedbackType, [51](#)
  - IPS, [51](#)
- FeedbackFlags
  - feedback\_settings\_t, [51](#)
- FeedbackType
  - feedback\_settings\_t, [51](#)
- Flags
  - control\_settings\_calb\_t, [27](#)
  - control\_settings\_t, [28](#)
  - secure\_settings\_t, [83](#)
  - status\_calb\_t, [94](#)
  - status\_t, [98](#)
- free\_enumerate\_devices
  - ximc.h, [175](#)
- FullCurrent
  - analog\_data\_t, [17](#)
- FullCurrent\_ADC
  - analog\_data\_t, [17](#)
- FullCurrent\_A
  - calibration\_settings\_t, [21](#)
- FullCurrent\_B
  - calibration\_settings\_t, [22](#)
- GPIOFlags
  - status\_calb\_t, [94](#)
  - status\_t, [98](#)
- gear\_information\_t, [52](#)
  - Manufacturer, [52](#)
  - PartNumber, [52](#)
- gear\_settings\_t, [52](#)

- Efficiency, [53](#)
- InputInertia, [53](#)
- MaxOutputBacklash, [53](#)
- RatedInputSpeed, [54](#)
- RatedInputTorque, [54](#)
- ReductionIn, [54](#)
- ReductionOut, [54](#)
- get\_accessories\_settings
  - [ximc.h, 177](#)
- get\_analog\_data
  - [ximc.h, 177](#)
- get\_bootloader\_version
  - [ximc.h, 177](#)
- get\_brake\_settings
  - [ximc.h, 178](#)
- get\_calibration\_settings
  - [ximc.h, 178](#)
- get\_chart\_data
  - [ximc.h, 178](#)
- get\_control\_settings
  - [ximc.h, 179](#)
- get\_control\_settings\_calb
  - [ximc.h, 179](#)
- get\_controller\_name
  - [ximc.h, 180](#)
- get\_ctp\_settings
  - [ximc.h, 180](#)
- get\_debug\_read
  - [ximc.h, 181](#)
- get\_device\_count
  - [ximc.h, 181](#)
- get\_device\_information
  - [ximc.h, 181](#)
- get\_device\_name
  - [ximc.h, 182](#)
- get\_edges\_settings
  - [ximc.h, 182](#)
- get\_edges\_settings\_calb
  - [ximc.h, 183](#)
- get\_emf\_settings
  - [ximc.h, 183](#)
- get\_encoder\_information
  - [ximc.h, 184](#)
- get\_encoder\_settings
  - [ximc.h, 184](#)
- get\_engine\_advanced\_setup
  - [ximc.h, 184](#)
- get\_engine\_settings
  - [ximc.h, 185](#)
- get\_engine\_settings\_calb
  - [ximc.h, 185](#)
- get\_entype\_settings
  - [ximc.h, 186](#)
- get\_enumerate\_device\_controller\_name
  - [ximc.h, 186](#)
- get\_enumerate\_device\_information
  - [ximc.h, 186](#)
- get\_enumerate\_device\_network\_information
  - [ximc.h, 187](#)
- get\_enumerate\_device\_serial
  - [ximc.h, 187](#)
- get\_enumerate\_device\_stage\_name
  - [ximc.h, 188](#)
- get\_extended\_settings
  - [ximc.h, 188](#)
- get\_extio\_settings
  - [ximc.h, 188](#)
- get\_feedback\_settings
  - [ximc.h, 189](#)
- get\_firmware\_version
  - [ximc.h, 189](#)
- get\_gear\_information
  - [ximc.h, 190](#)
- get\_gear\_settings
  - [ximc.h, 190](#)
- get\_globally\_unique\_identifier
  - [ximc.h, 190](#)
- get\_hallsensor\_information
  - [ximc.h, 191](#)
- get\_hallsensor\_settings
  - [ximc.h, 191](#)
- get\_home\_settings
  - [ximc.h, 191](#)
- get\_home\_settings\_calb
  - [ximc.h, 192](#)
- get\_init\_random
  - [ximc.h, 192](#)
- get\_joystick\_settings
  - [ximc.h, 193](#)
- get\_measurements
  - [ximc.h, 193](#)
- get\_motor\_information
  - [ximc.h, 194](#)
- get\_motor\_settings
  - [ximc.h, 194](#)
- get\_move\_settings
  - [ximc.h, 194](#)
- get\_move\_settings\_calb
  - [ximc.h, 195](#)
- get\_network\_settings
  - [ximc.h, 195](#)
- get\_nonvolatile\_memory
  - [ximc.h, 195](#)
- get\_password\_settings
  - [ximc.h, 196](#)
- get\_pid\_settings
  - [ximc.h, 196](#)
- get\_position
  - [ximc.h, 197](#)
- get\_position\_calb
  - [ximc.h, 197](#)
- get\_position\_calb\_t, [54](#)

- EncPosition, [55](#)
- Position, [55](#)
- get\_position\_t, [55](#)
  - EncPosition, [56](#)
  - uPosition, [56](#)
- get\_power\_settings
  - ximc.h, [197](#)
- get\_secure\_settings
  - ximc.h, [198](#)
- get\_serial\_number
  - ximc.h, [198](#)
- get\_stage\_information
  - ximc.h, [198](#)
- get\_stage\_name
  - ximc.h, [199](#)
- get\_stage\_settings
  - ximc.h, [199](#)
- get\_status
  - ximc.h, [199](#)
- get\_status\_calb
  - ximc.h, [200](#)
- get\_sync\_in\_settings
  - ximc.h, [200](#)
- get\_sync\_in\_settings\_calb
  - ximc.h, [201](#)
- get\_sync\_out\_settings
  - ximc.h, [201](#)
- get\_sync\_out\_settings\_calb
  - ximc.h, [202](#)
- get\_uart\_settings
  - ximc.h, [202](#)
- globally\_unique\_identifier\_t, [56](#)
  - UniqueID0, [57](#)
  - UniqueID1, [57](#)
  - UniqueID2, [57](#)
  - UniqueID3, [57](#)
- goto\_firmware
  - ximc.h, [203](#)
- H\_BRIDGE\_ALERT
  - ximc.h, [145](#)
- HOME\_DIR\_FIRST
  - ximc.h, [146](#)
- HOME\_DIR\_SECOND
  - ximc.h, [146](#)
- HOME\_HALF\_MV
  - ximc.h, [146](#)
- HOME\_MV\_SEC\_EN
  - ximc.h, [146](#)
- HOME\_STOP\_FIRST\_BITS
  - ximc.h, [146](#)
- HOME\_STOP\_FIRST\_LIM
  - ximc.h, [146](#)
- HOME\_STOP\_FIRST\_REV
  - ximc.h, [147](#)
- HOME\_STOP\_FIRST\_SYN
  - ximc.h, [147](#)
- HOME\_STOP\_SECOND\_BITS
  - ximc.h, [147](#)
- HOME\_STOP\_SECOND\_LIM
  - ximc.h, [147](#)
- HOME\_STOP\_SECOND\_REV
  - ximc.h, [147](#)
- HOME\_STOP\_SECOND\_SYN
  - ximc.h, [147](#)
- HOME\_USE\_FAST
  - ximc.h, [148](#)
- hallsensor\_information\_t, [58](#)
  - Manufacturer, [58](#)
  - PartNumber, [58](#)
- hallsensor\_settings\_t, [59](#)
  - MaxCurrentConsumption, [59](#)
  - MaxOperatingFrequency, [59](#)
  - SupplyVoltageMax, [59](#)
  - SupplyVoltageMin, [60](#)
- has\_firmware
  - ximc.h, [203](#)
- HoldCurrent
  - power\_settings\_t, [81](#)
- home\_settings\_calb\_t, [60](#)
  - FastHome, [60](#)
  - HomeDelta, [61](#)
  - HomeFlags, [61](#)
  - SlowHome, [61](#)
- home\_settings\_t, [61](#)
  - FastHome, [62](#)
  - HomeDelta, [62](#)
  - HomeFlags, [62](#)
  - SlowHome, [62](#)
  - uFastHome, [62](#)
  - uHomeDelta, [63](#)
  - uSlowHome, [63](#)
- HomeDelta
  - home\_settings\_calb\_t, [61](#)
  - home\_settings\_t, [62](#)
- HomeFlags
  - home\_settings\_calb\_t, [61](#)
  - home\_settings\_t, [62](#)
- HorizontalLoadCapacity
  - stage\_settings\_t, [90](#)
- IPS
  - feedback\_settings\_t, [51](#)
- IPv4Address
  - network\_settings\_t, [78](#)
- init\_random\_t, [63](#)
  - key, [64](#)
- InputInertia
  - gear\_settings\_t, [53](#)
- lpwr
  - status\_calb\_t, [94](#)
  - status\_t, [98](#)

- lusb
  - status\_calb\_t, [94](#)
  - status\_t, [98](#)
- JOY\_REVERSE
  - ximc.h, [148](#)
- Joy
  - analog\_data\_t, [17](#)
  - chart\_data\_t, [24](#)
- Joy\_ADC
  - analog\_data\_t, [17](#)
- JoyCenter
  - joystick\_settings\_t, [65](#)
- JoyFlags
  - joystick\_settings\_t, [65](#)
- JoyHighEnd
  - joystick\_settings\_t, [65](#)
- JoyLowEnd
  - joystick\_settings\_t, [65](#)
- joystick\_settings\_t, [64](#)
  - DeadZone, [65](#)
  - ExpFactor, [65](#)
  - JoyCenter, [65](#)
  - JoyFlags, [65](#)
  - JoyHighEnd, [65](#)
  - JoyLowEnd, [65](#)
- Key
  - serial\_number\_t, [84](#)
- key
  - init\_random\_t, [64](#)
- Km
  - emf\_settings\_t, [39](#)
- L
  - analog\_data\_t, [17](#)
  - emf\_settings\_t, [39](#)
- LOW\_UPWR\_PROTECTION
  - ximc.h, [148](#)
- LeadScrewPitch
  - stage\_settings\_t, [90](#)
- LeftBorder
  - edges\_settings\_calb\_t, [35](#)
  - edges\_settings\_t, [37](#)
- Length
  - measurements\_t, [66](#)
- LimitSwitchesSettings
  - accessories\_settings\_t, [11](#)
- logging\_callback\_stderr\_narrow
  - ximc.h, [203](#)
- logging\_callback\_stderr\_wide
  - ximc.h, [204](#)
- logging\_callback\_t
  - ximc.h, [164](#)
- LowUpwrOff
  - secure\_settings\_t, [83](#)
- MBRatedCurrent
  - accessories\_settings\_t, [11](#)
- MBRatedVoltage
  - accessories\_settings\_t, [11](#)
- MBSettings
  - accessories\_settings\_t, [11](#)
- MBTorque
  - accessories\_settings\_t, [12](#)
- MICROSTEP\_MODE\_FRAC\_128
  - ximc.h, [148](#)
- MICROSTEP\_MODE\_FRAC\_16
  - ximc.h, [148](#)
- MICROSTEP\_MODE\_FRAC\_2
  - ximc.h, [148](#)
- MICROSTEP\_MODE\_FRAC\_256
  - ximc.h, [149](#)
- MICROSTEP\_MODE\_FRAC\_32
  - ximc.h, [149](#)
- MICROSTEP\_MODE\_FRAC\_4
  - ximc.h, [149](#)
- MICROSTEP\_MODE\_FRAC\_64
  - ximc.h, [149](#)
- MICROSTEP\_MODE\_FRAC\_8
  - ximc.h, [149](#)
- MICROSTEP\_MODE\_FULL
  - ximc.h, [149](#)
- MOVE\_STATE\_ANTIPLAY
  - ximc.h, [150](#)
- MOVE\_STATE\_MOVING
  - ximc.h, [150](#)
- MOVE\_STATE\_TARGET\_SPEED
  - ximc.h, [150](#)
- MVCMD\_ERROR
  - ximc.h, [150](#)
- MVCMD\_HOME
  - ximc.h, [150](#)
- MVCMD\_LEFT
  - ximc.h, [150](#)
- MVCMD\_LOFT
  - ximc.h, [151](#)
- MVCMD\_MOVE
  - ximc.h, [151](#)
- MVCMD\_MOVR
  - ximc.h, [151](#)
- MVCMD\_NAME\_BITS
  - ximc.h, [151](#)
- MVCMD\_RIGHT
  - ximc.h, [151](#)
- MVCMD\_RUNNING
  - ximc.h, [151](#)
- MVCMD\_SSTP
  - ximc.h, [152](#)
- MVCMD\_STOP
  - ximc.h, [152](#)
- MVCMD\_UKNWN
  - ximc.h, [152](#)

- MagneticBrakeInfo
  - accessories\_settings\_t, [11](#)
- Major
  - device\_information\_t, [33](#)
  - serial\_number\_t, [85](#)
- Manufacturer
  - encoder\_information\_t, [40](#)
  - gear\_information\_t, [52](#)
  - hallsensor\_information\_t, [58](#)
  - motor\_information\_t, [67](#)
  - stage\_information\_t, [88](#)
- MaxClickTime
  - control\_settings\_calb\_t, [27](#)
  - control\_settings\_t, [28](#)
- MaxCurrent
  - motor\_settings\_t, [69](#)
- MaxCurrentConsumption
  - encoder\_settings\_t, [41](#)
  - hallsensor\_settings\_t, [59](#)
  - stage\_settings\_t, [91](#)
- MaxCurrentTime
  - motor\_settings\_t, [69](#)
- MaxOperatingFrequency
  - encoder\_settings\_t, [41](#)
  - hallsensor\_settings\_t, [59](#)
- MaxOutputBacklash
  - gear\_settings\_t, [53](#)
- MaxSpeed
  - control\_settings\_calb\_t, [27](#)
  - control\_settings\_t, [29](#)
  - motor\_settings\_t, [70](#)
  - stage\_settings\_t, [91](#)
- measurements\_t, [66](#)
  - Error, [66](#)
  - Length, [66](#)
- MechanicalTimeConstant
  - motor\_settings\_t, [70](#)
- MicrostepMode
  - engine\_settings\_calb\_t, [44](#)
  - engine\_settings\_t, [47](#)
- MinimumUusb
  - secure\_settings\_t, [84](#)
- Minor
  - device\_information\_t, [34](#)
  - serial\_number\_t, [85](#)
- motor\_information\_t, [67](#)
  - Manufacturer, [67](#)
  - PartNumber, [67](#)
- motor\_settings\_t, [68](#)
  - DetentTorque, [69](#)
  - MaxCurrent, [69](#)
  - MaxCurrentTime, [69](#)
  - MaxSpeed, [70](#)
  - MechanicalTimeConstant, [70](#)
  - MotorType, [70](#)
  - NoLoadCurrent, [70](#)
  - NoLoadSpeed, [70](#)
  - NominalCurrent, [70](#)
  - NominalPower, [71](#)
  - NominalSpeed, [71](#)
  - NominalTorque, [71](#)
  - NominalVoltage, [71](#)
  - Phases, [71](#)
  - Poles, [71](#)
  - RotorInertia, [72](#)
  - SpeedConstant, [72](#)
  - SpeedTorqueGradient, [72](#)
  - StallTorque, [72](#)
  - TorqueConstant, [72](#)
  - WindingInductance, [72](#)
  - WindingResistance, [73](#)
- MotorType
  - motor\_settings\_t, [70](#)
- move\_settings\_calb\_t, [73](#)
  - Accel, [74](#)
  - AntiplaySpeed, [74](#)
  - Decel, [74](#)
  - MoveFlags, [74](#)
  - Speed, [74](#)
- move\_settings\_t, [75](#)
  - Accel, [75](#)
  - AntiplaySpeed, [76](#)
  - Decel, [76](#)
  - MoveFlags, [76](#)
  - Speed, [76](#)
  - uAntiplaySpeed, [76](#)
  - uSpeed, [76](#)
- MoveFlags
  - move\_settings\_calb\_t, [74](#)
  - move\_settings\_t, [76](#)
- MoveSts
  - status\_calb\_t, [94](#)
  - status\_t, [98](#)
- msec\_sleep
  - ximc.h, [204](#)
- MvCmdSts
  - status\_calb\_t, [95](#)
  - status\_t, [98](#)
- network\_settings\_t, [77](#)
  - DHCPEnabled, [77](#)
  - DefaultGateway, [77](#)
  - IPv4Address, [78](#)
  - SubnetMask, [78](#)
- NoLoadCurrent
  - motor\_settings\_t, [70](#)
- NoLoadSpeed
  - motor\_settings\_t, [70](#)
- NomCurrent
  - engine\_settings\_calb\_t, [44](#)
  - engine\_settings\_t, [47](#)
- NomSpeed

- engine\_settings\_calb\_t, [45](#)
- engine\_settings\_t, [47](#)
- NomVoltage
  - engine\_settings\_calb\_t, [45](#)
  - engine\_settings\_t, [47](#)
- NominalCurrent
  - motor\_settings\_t, [70](#)
- NominalPower
  - motor\_settings\_t, [71](#)
- NominalSpeed
  - motor\_settings\_t, [71](#)
- NominalTorque
  - motor\_settings\_t, [71](#)
- NominalVoltage
  - motor\_settings\_t, [71](#)
- nonvolatile\_memory\_t, [78](#)
- UserData, [78](#)
- open\_device
  - ximc.h, [204](#)
- POWER\_OFF\_ENABLED
  - ximc.h, [152](#)
- POWER\_REDUCT\_ENABLED
  - ximc.h, [152](#)
- POWER\_SMOOTH\_CURRENT
  - ximc.h, [152](#)
- PWR\_STATE\_MAX
  - ximc.h, [153](#)
- PWR\_STATE\_NORM
  - ximc.h, [153](#)
- PWR\_STATE\_OFF
  - ximc.h, [153](#)
- PWR\_STATE\_REDUCT
  - ximc.h, [153](#)
- PWR\_STATE\_UNKNOWN
  - ximc.h, [153](#)
- PWRSts
  - status\_calb\_t, [95](#)
  - status\_t, [99](#)
- PartNumber
  - encoder\_information\_t, [40](#)
  - gear\_information\_t, [52](#)
  - hallsensor\_information\_t, [58](#)
  - motor\_information\_t, [67](#)
  - stage\_information\_t, [88](#)
- password\_settings\_t, [79](#)
- UserPassword, [79](#)
- PeakCurrent
  - engine\_settings\_calb\_t, [45](#)
  - engine\_settings\_t, [47](#)
- Phases
  - motor\_settings\_t, [71](#)
- pid\_settings\_t, [79](#)
- Poles
  - motor\_settings\_t, [71](#)
- PosFlags
  - set\_position\_calb\_t, [86](#)
  - set\_position\_t, [87](#)
- Position
  - get\_position\_calb\_t, [55](#)
  - set\_position\_calb\_t, [86](#)
  - sync\_in\_settings\_calb\_t, [100](#)
- PositionerName
  - stage\_name\_t, [89](#)
- Pot
  - analog\_data\_t, [17](#)
  - chart\_data\_t, [25](#)
- power\_settings\_t, [80](#)
- CurrReductDelay, [81](#)
- CurrentSetTime, [81](#)
- HoldCurrent, [81](#)
- PowerFlags, [81](#)
- PowerOffDelay, [81](#)
- PowerFlags
  - power\_settings\_t, [81](#)
- PowerOffDelay
  - power\_settings\_t, [81](#)
- probe\_device
  - ximc.h, [205](#)
- R
  - analog\_data\_t, [18](#)
  - emf\_settings\_t, [39](#)
- REV\_SENS\_INV
  - ximc.h, [153](#)
- RPM\_DIV\_1000
  - ximc.h, [154](#)
- RatedInputSpeed
  - gear\_settings\_t, [54](#)
- RatedInputTorque
  - gear\_settings\_t, [54](#)
- ReductionIn
  - gear\_settings\_t, [54](#)
- ReductionOut
  - gear\_settings\_t, [54](#)
- Release
  - device\_information\_t, [34](#)
  - serial\_number\_t, [85](#)
- RightBorder
  - edges\_settings\_calb\_t, [36](#)
  - edges\_settings\_t, [37](#)
- RotorInertia
  - motor\_settings\_t, [72](#)
- SETPOS\_IGNORE\_ENCODER
  - ximc.h, [154](#)
- SETPOS\_IGNORE\_POSITION
  - ximc.h, [154](#)
- STATE\_ALARM
  - ximc.h, [154](#)
- STATE\_BORDERS\_SWAP\_MISSET
  - ximc.h, [154](#)
- STATE\_BRAKE

- ximc.h, [154](#)
- STATE\_BUTTON\_LEFT
  - ximc.h, [155](#)
- STATE\_BUTTON\_RIGHT
  - ximc.h, [155](#)
- STATE\_CONTROLLER\_OVERHEAT
  - ximc.h, [155](#)
- STATE\_CONTR
  - ximc.h, [155](#)
- STATE\_CTP\_ERROR
  - ximc.h, [155](#)
- STATE\_DIG\_SIGNAL
  - ximc.h, [155](#)
- STATE\_EEPROM\_CONNECTED
  - ximc.h, [156](#)
- STATE\_ENC\_A
  - ximc.h, [156](#)
- STATE\_ENC\_B
  - ximc.h, [156](#)
- STATE\_ENGINE\_RESPONSE\_ERROR
  - ximc.h, [156](#)
- STATE\_ERRC
  - ximc.h, [156](#)
- STATE\_ERRD
  - ximc.h, [156](#)
- STATE\_ERRV
  - ximc.h, [157](#)
- STATE\_EXTIO\_ALARM
  - ximc.h, [157](#)
- STATE\_GPIO\_LEVEL
  - ximc.h, [157](#)
- STATE\_GPIO\_PINOUT
  - ximc.h, [157](#)
- STATE\_IS\_HOMED
  - ximc.h, [157](#)
- STATE\_LEFT\_EDGE
  - ximc.h, [157](#)
- STATE\_LOW\_USB\_VOLTAGE
  - ximc.h, [158](#)
- STATE\_OVERLOAD\_POWER\_CURRENT
  - ximc.h, [158](#)
- STATE\_OVERLOAD\_POWER\_VOLTAGE
  - ximc.h, [158](#)
- STATE\_OVERLOAD\_USB\_CURRENT
  - ximc.h, [158](#)
- STATE\_OVERLOAD\_USB\_VOLTAGE
  - ximc.h, [158](#)
- STATE\_POWER\_OVERHEAT
  - ximc.h, [158](#)
- STATE\_REV\_SENSOR
  - ximc.h, [159](#)
- STATE\_RIGHT\_EDGE
  - ximc.h, [159](#)
- STATE\_SECUR
  - ximc.h, [159](#)
- STATE\_SYNC\_INPUT
  - ximc.h, [159](#)
- STATE\_SYNC\_OUTPUT
  - ximc.h, [159](#)
- STATE\_WINDING\_RES\_MISMATCH
  - ximc.h, [159](#)
- SYNCIN\_ENABLED
  - ximc.h, [160](#)
- SYNCIN\_GOTOPOSITION
  - ximc.h, [160](#)
- SYNCIN\_INVERT
  - ximc.h, [160](#)
- SYNCOUT\_ENABLED
  - ximc.h, [160](#)
- SYNCOUT\_IN\_STEPS
  - ximc.h, [160](#)
- SYNCOUT\_INVERT
  - ximc.h, [160](#)
- SYNCOUT\_ONPERIOD
  - ximc.h, [161](#)
- SYNCOUT\_ONSTART
  - ximc.h, [161](#)
- SYNCOUT\_ONSTOP
  - ximc.h, [161](#)
- SYNCOUT\_STATE
  - ximc.h, [161](#)
- secure\_settings\_t, [82](#)
  - Criticalpwr, [82](#)
  - Criticalusb, [83](#)
  - CriticalUpwr, [83](#)
  - CriticalUusb, [83](#)
  - CriticalT, [83](#)
  - Flags, [83](#)
  - LowUpwrOff, [83](#)
  - MinimumUusb, [84](#)
- serial\_number\_t, [84](#)
  - Key, [84](#)
  - Major, [85](#)
  - Minor, [85](#)
  - Release, [85](#)
  - SN, [85](#)
- service\_command\_updf
  - ximc.h, [205](#)
- set\_accessories\_settings
  - ximc.h, [205](#)
- set\_brake\_settings
  - ximc.h, [206](#)
- set\_calibration\_settings
  - ximc.h, [206](#)
- set\_control\_settings
  - ximc.h, [206](#)
- set\_control\_settings\_calb
  - ximc.h, [207](#)
- set\_controller\_name
  - ximc.h, [207](#)
- set\_correction\_table
  - ximc.h, [208](#)

- set\_ctp\_settings
  - ximc.h, [208](#)
- set\_debug\_write
  - ximc.h, [209](#)
- set\_edges\_settings
  - ximc.h, [209](#)
- set\_edges\_settings\_calb
  - ximc.h, [210](#)
- set\_emf\_settings
  - ximc.h, [210](#)
- set\_encoder\_information
  - ximc.h, [211](#)
- set\_encoder\_settings
  - ximc.h, [211](#)
- set\_engine\_advanced\_setup
  - ximc.h, [211](#)
- set\_engine\_settings
  - ximc.h, [212](#)
- set\_engine\_settings\_calb
  - ximc.h, [212](#)
- set\_entype\_settings
  - ximc.h, [213](#)
- set\_extended\_settings
  - ximc.h, [213](#)
- set\_extio\_settings
  - ximc.h, [213](#)
- set\_feedback\_settings
  - ximc.h, [214](#)
- set\_gear\_information
  - ximc.h, [214](#)
- set\_gear\_settings
  - ximc.h, [215](#)
- set\_hallsensor\_information
  - ximc.h, [215](#)
- set\_hallsensor\_settings
  - ximc.h, [215](#)
- set\_home\_settings
  - ximc.h, [217](#)
- set\_home\_settings\_calb
  - ximc.h, [217](#)
- set\_joystick\_settings
  - ximc.h, [218](#)
- set\_logging\_callback
  - ximc.h, [218](#)
- set\_motor\_information
  - ximc.h, [218](#)
- set\_motor\_settings
  - ximc.h, [219](#)
- set\_move\_settings
  - ximc.h, [219](#)
- set\_move\_settings\_calb
  - ximc.h, [219](#)
- set\_network\_settings
  - ximc.h, [220](#)
- set\_nonvolatile\_memory
  - ximc.h, [220](#)
- set\_password\_settings
  - ximc.h, [220](#)
- set\_pid\_settings
  - ximc.h, [221](#)
- set\_position
  - ximc.h, [221](#)
- set\_position\_calb
  - ximc.h, [222](#)
- set\_position\_calb\_t, [85](#)
  - EncPosition, [86](#)
  - PosFlags, [86](#)
  - Position, [86](#)
- set\_position\_t, [87](#)
  - EncPosition, [87](#)
  - PosFlags, [87](#)
  - uPosition, [87](#)
- set\_power\_settings
  - ximc.h, [222](#)
- set\_secure\_settings
  - ximc.h, [222](#)
- set\_serial\_number
  - ximc.h, [223](#)
- set\_stage\_information
  - ximc.h, [223](#)
- set\_stage\_name
  - ximc.h, [224](#)
- set\_stage\_settings
  - ximc.h, [224](#)
- set\_sync\_in\_settings
  - ximc.h, [224](#)
- set\_sync\_in\_settings\_calb
  - ximc.h, [225](#)
- set\_sync\_out\_settings
  - ximc.h, [225](#)
- set\_sync\_out\_settings\_calb
  - ximc.h, [226](#)
- set\_uart\_settings
  - ximc.h, [226](#)
- SlowHome
  - home\_settings\_calb\_t, [61](#)
  - home\_settings\_t, [62](#)
- SN
  - serial\_number\_t, [85](#)
- Speed
  - move\_settings\_calb\_t, [74](#)
  - move\_settings\_t, [76](#)
  - sync\_in\_settings\_calb\_t, [101](#)
  - sync\_in\_settings\_t, [102](#)
- SpeedConstant
  - motor\_settings\_t, [72](#)
- SpeedTorqueGradient
  - motor\_settings\_t, [72](#)
- stage\_information\_t, [88](#)
  - Manufacturer, [88](#)
  - PartNumber, [88](#)
- stage\_name\_t, [89](#)

- PositionerName, [89](#)
- stage\_settings\_t, [89](#)
  - HorizontalLoadCapacity, [90](#)
  - LeadScrewPitch, [90](#)
  - MaxCurrentConsumption, [91](#)
  - MaxSpeed, [91](#)
  - SupplyVoltageMax, [91](#)
  - SupplyVoltageMin, [91](#)
  - TravelRange, [91](#)
  - Units, [91](#)
  - VerticalLoadCapacity, [92](#)
- StallTorque
  - motor\_settings\_t, [72](#)
- status\_calb\_t, [92](#)
  - CmdBufFreeSpace, [93](#)
  - CurPosition, [93](#)
  - CurSpeed, [93](#)
  - CurT, [93](#)
  - EncPosition, [93](#)
  - EncSts, [94](#)
  - Flags, [94](#)
  - GPIOFlags, [94](#)
  - Ipwr, [94](#)
  - Iusb, [94](#)
  - MoveSts, [94](#)
  - MvCmdSts, [95](#)
  - PWRSts, [95](#)
  - Upwr, [95](#)
  - Uusb, [95](#)
  - WindSts, [95](#)
- status\_t, [96](#)
  - CmdBufFreeSpace, [97](#)
  - CurPosition, [97](#)
  - CurSpeed, [97](#)
  - CurT, [97](#)
  - EncPosition, [97](#)
  - EncSts, [97](#)
  - Flags, [98](#)
  - GPIOFlags, [98](#)
  - Ipwr, [98](#)
  - Iusb, [98](#)
  - MoveSts, [98](#)
  - MvCmdSts, [98](#)
  - PWRSts, [99](#)
  - uCurPosition, [99](#)
  - uCurSpeed, [99](#)
  - Upwr, [99](#)
  - Uusb, [99](#)
  - WindSts, [99](#)
- stepcloseloop\_Kp\_high
  - engine\_advanced\_setup\_t, [42](#)
- stepcloseloop\_Kp\_low
  - engine\_advanced\_setup\_t, [42](#)
- stepcloseloop\_Kw
  - engine\_advanced\_setup\_t, [43](#)
- StepsPerRev
  - engine\_settings\_calb\_t, [45](#)
  - engine\_settings\_t, [47](#)
- SubnetMask
  - network\_settings\_t, [78](#)
- SupVoltage
  - analog\_data\_t, [18](#)
- SupVoltage\_ADC
  - analog\_data\_t, [18](#)
- SupplyVoltageMax
  - encoder\_settings\_t, [41](#)
  - hallsensor\_settings\_t, [59](#)
  - stage\_settings\_t, [91](#)
- SupplyVoltageMin
  - encoder\_settings\_t, [41](#)
  - hallsensor\_settings\_t, [60](#)
  - stage\_settings\_t, [91](#)
- sync\_in\_settings\_calb\_t, [100](#)
  - ClutterTime, [100](#)
  - Position, [100](#)
  - Speed, [101](#)
  - SyncInFlags, [101](#)
- sync\_in\_settings\_t, [101](#)
  - ClutterTime, [102](#)
  - Speed, [102](#)
  - SyncInFlags, [102](#)
  - uPosition, [102](#)
  - uSpeed, [102](#)
- sync\_out\_settings\_calb\_t, [103](#)
  - Accuracy, [103](#)
  - SyncOutFlags, [104](#)
  - SyncOutPeriod, [104](#)
  - SyncOutPulseSteps, [104](#)
- sync\_out\_settings\_t, [104](#)
  - Accuracy, [105](#)
  - SyncOutFlags, [105](#)
  - SyncOutPeriod, [105](#)
  - SyncOutPulseSteps, [105](#)
  - uAccuracy, [105](#)
- SyncInFlags
  - sync\_in\_settings\_calb\_t, [101](#)
  - sync\_in\_settings\_t, [102](#)
- SyncOutFlags
  - sync\_out\_settings\_calb\_t, [104](#)
  - sync\_out\_settings\_t, [105](#)
- SyncOutPeriod
  - sync\_out\_settings\_calb\_t, [104](#)
  - sync\_out\_settings\_t, [105](#)
- SyncOutPulseSteps
  - sync\_out\_settings\_calb\_t, [104](#)
  - sync\_out\_settings\_t, [105](#)
- t1
  - brake\_settings\_t, [19](#)
- t2
  - brake\_settings\_t, [19](#)
- t3

- brake\_settings\_t, [20](#)
- t4
  - brake\_settings\_t, [20](#)
- TSGrad
  - accessories\_settings\_t, [12](#)
- TSMaX
  - accessories\_settings\_t, [12](#)
- TSMIn
  - accessories\_settings\_t, [12](#)
- TSSettings
  - accessories\_settings\_t, [12](#)
- Temp
  - analog\_data\_t, [18](#)
- Temp\_ADC
  - analog\_data\_t, [18](#)
- TemperatureSensorInfo
  - accessories\_settings\_t, [12](#)
- Timeout
  - control\_settings\_calb\_t, [27](#)
  - control\_settings\_t, [29](#)
- TorqueConstant
  - motor\_settings\_t, [72](#)
- TravelRange
  - stage\_settings\_t, [91](#)
- UART\_PARITY\_BITS
  - ximc.h, [161](#)
- UARTSetupFlags
  - uart\_settings\_t, [106](#)
- uAccuracy
  - sync\_out\_settings\_t, [105](#)
- uAntiplaySpeed
  - move\_settings\_t, [76](#)
- uCurPosition
  - status\_t, [99](#)
- uCurSpeed
  - status\_t, [99](#)
- uDeltaPosition
  - control\_settings\_t, [29](#)
- uFastHome
  - home\_settings\_t, [62](#)
- uHomeDelta
  - home\_settings\_t, [63](#)
- uLeftBorder
  - edges\_settings\_t, [37](#)
- uMaxSpeed
  - control\_settings\_t, [29](#)
- uNomSpeed
  - engine\_settings\_t, [48](#)
- uPosition
  - get\_position\_t, [56](#)
  - set\_position\_t, [87](#)
  - sync\_in\_settings\_t, [102](#)
- uRightBorder
  - edges\_settings\_t, [37](#)
- uSlowHome
  - home\_settings\_t, [63](#)
- uSpeed
  - move\_settings\_t, [76](#)
  - sync\_in\_settings\_t, [102](#)
- uart\_settings\_t, [106](#)
  - UARTSetupFlags, [106](#)
- UniquelD0
  - globally\_unique\_identifier\_t, [57](#)
- UniquelD1
  - globally\_unique\_identifier\_t, [57](#)
- UniquelD2
  - globally\_unique\_identifier\_t, [57](#)
- UniquelD3
  - globally\_unique\_identifier\_t, [57](#)
- Units
  - stage\_settings\_t, [91](#)
- Upwr
  - status\_calb\_t, [95](#)
  - status\_t, [99](#)
- UserData
  - nonvolatile\_memory\_t, [78](#)
- UserPassword
  - password\_settings\_t, [79](#)
- Uusb
  - status\_calb\_t, [95](#)
  - status\_t, [99](#)
- VerticalLoadCapacity
  - stage\_settings\_t, [92](#)
- WIND\_A\_STATE\_ABSENT
  - ximc.h, [161](#)
- WIND\_A\_STATE\_MALFUNC
  - ximc.h, [162](#)
- WIND\_A\_STATE\_OK
  - ximc.h, [162](#)
- WIND\_A\_STATE\_UNKNOWN
  - ximc.h, [162](#)
- WIND\_B\_STATE\_ABSENT
  - ximc.h, [162](#)
- WIND\_B\_STATE\_MALFUNC
  - ximc.h, [162](#)
- WIND\_B\_STATE\_OK
  - ximc.h, [162](#)
- WIND\_B\_STATE\_UNKNOWN
  - ximc.h, [163](#)
- WindSts
  - status\_calb\_t, [95](#)
  - status\_t, [99](#)
- WindingCurrentA
  - chart\_data\_t, [25](#)
- WindingCurrentB
  - chart\_data\_t, [25](#)
- WindingCurrentC
  - chart\_data\_t, [25](#)
- WindingInductance
  - motor\_settings\_t, [72](#)

- WindingResistance
  - motor\_settings.t, 73
- WindingVoltageA
  - chart\_data.t, 25
- WindingVoltageB
  - chart\_data.t, 25
- WindingVoltageC
  - chart\_data.t, 26
- write\_key
  - ximc.h, 227
- XIMC\_API
  - ximc.h, 163
- ximc.h, 107
  - ALARM\_FLAGS\_STICKING, 132
  - ALARM\_ON\_BORDERS\_SWAP\_MISSET, 132
  - ALARM\_ON\_DRIVER\_OVERHEATING, 132
  - BACK\_EMF\_INDUCTANCE\_AUTO, 133
  - BACK\_EMF\_KM\_AUTO, 133
  - BACK\_EMF\_RESISTANCE\_AUTO, 133
  - BORDER\_IS\_ENCODER, 133
  - BORDER\_STOP\_LEFT, 133
  - BORDER\_STOP\_RIGHT, 133
  - BORDERS\_SWAP\_MISSET\_DETECTION, 134
  - BRAKE\_ENABLED, 134
  - BRAKE\_ENG\_PWROFF, 134
  - BRAKING\_OVERVOLTAGE\_PROTECTION, 134
  - CONTROL\_BTN\_LEFT\_PUSHED\_OPEN, 134
  - CONTROL\_BTN\_RIGHT\_PUSHED\_OPEN, 134
  - CONTROL\_MODE\_BITS, 135
  - CONTROL\_MODE\_JOY, 135
  - CONTROL\_MODE\_LR, 135
  - CONTROL\_MODE\_OFF, 135
  - CTP\_ALARM\_ON\_ERROR, 135
  - CTP\_BASE, 135
  - CTP\_ENABLED, 136
  - CTP\_ERROR\_CORRECTION, 136
  - calibration.t, 163
  - close\_device, 164
  - command\_clear\_fram, 165
  - command\_eeread\_settings, 165
  - command\_eesave\_settings, 165
  - command\_home, 166
  - command\_homezero, 166
  - command\_left, 167
  - command\_loft, 167
  - command\_move, 167
  - command\_move\_calb, 168
  - command\_movr, 168
  - command\_movr\_calb, 168
  - command\_power\_off, 169
  - command\_read\_robust\_settings, 169
  - command\_read\_settings, 170
  - command\_reset, 170
  - command\_right, 170
  - command\_save\_robust\_settings, 171
  - command\_save\_settings, 171
  - command\_sstp, 171
  - command\_start\_measurements, 172
  - command\_stop, 172
  - command\_update\_firmware, 172
  - command\_wait\_for\_stop, 173
  - command\_zero, 173
  - DRIVER\_TYPE\_EXTERNAL, 136
  - DRIVER\_TYPE\_INTEGRATE, 136
  - EEPROM\_PRECEDENCE, 136
  - ENC\_STATE\_ABSENT, 136
  - ENC\_STATE\_MALFUNC, 137
  - ENC\_STATE\_OK, 137
  - ENC\_STATE\_REVERS, 137
  - ENC\_STATE\_UNKNOWN, 137
  - ENDER\_SW1\_ACTIVE\_LOW, 137
  - ENDER\_SW2\_ACTIVE\_LOW, 137
  - ENDER\_SWAP, 138
  - ENGINE\_ACCEL\_ON, 138
  - ENGINE\_ANTIPLAY, 138
  - ENGINE\_CURRENT\_AS\_RMS, 138
  - ENGINE\_LIMIT\_CURR, 138
  - ENGINE\_LIMIT\_RPM, 138
  - ENGINE\_LIMIT\_VOLT, 139
  - ENGINE\_MAX\_SPEED, 139
  - ENGINE\_REVERSE, 139
  - ENGINE\_TYPE\_2DC, 139
  - ENGINE\_TYPE\_BRUSHLESS, 139
  - ENGINE\_TYPE\_DC, 139
  - ENGINE\_TYPE\_NONE, 140
  - ENGINE\_TYPE\_STEP, 140
  - ENGINE\_TYPE\_TEST, 140
  - ENGINE\_USE\_PEAK\_CURRENT, 140
  - ENUMERATE\_PROBE, 140
  - EXTIO\_SETUP\_INVERT, 140
  - EXTIO\_SETUP\_MODE\_IN\_ALARM, 141
  - EXTIO\_SETUP\_MODE\_IN\_BITS, 141
  - EXTIO\_SETUP\_MODE\_IN\_HOME, 141
  - EXTIO\_SETUP\_MODE\_IN\_MOVR, 141
  - EXTIO\_SETUP\_MODE\_IN\_NOP, 141
  - EXTIO\_SETUP\_MODE\_IN\_PWOF, 141
  - EXTIO\_SETUP\_MODE\_IN\_STOP, 142
  - EXTIO\_SETUP\_MODE\_OUT\_ALARM, 142
  - EXTIO\_SETUP\_MODE\_OUT\_BITS, 142
  - EXTIO\_SETUP\_MODE\_OUT\_MOTOR\_ON, 142
  - EXTIO\_SETUP\_MODE\_OUT\_MOVING, 142
  - EXTIO\_SETUP\_MODE\_OUT\_OFF, 142
  - EXTIO\_SETUP\_MODE\_OUT\_ON, 143
  - EXTIO\_SETUP\_OUTPUT, 143
  - enumerate\_devices, 173
  - FEEDBACK\_EMF, 143
  - FEEDBACK\_ENC\_ADAPTIVE\_HOLDING, 143

- FEEDBACK\_ENC\_FILTER\_BITS, 143
- FEEDBACK\_ENC\_FILTER\_MEDIUM, 143
- FEEDBACK\_ENC\_FILTER\_NONE, 144
- FEEDBACK\_ENC\_FILTER\_STRONG, 144
- FEEDBACK\_ENC\_FILTER\_WEAK, 144
- FEEDBACK\_ENC\_REVERSE, 144
- FEEDBACK\_ENC\_TYPE\_AUTO, 144
- FEEDBACK\_ENC\_TYPE\_BITS, 144
- FEEDBACK\_ENC\_TYPE\_DIFFERENTIAL, 145
- FEEDBACK\_ENC\_TYPE\_SINGLE\_ENDED, 145
- FEEDBACK\_ENCODER\_MEDIATED, 145
- FEEDBACK\_ENCODER, 145
- FEEDBACK\_NONE, 145
- free\_enumerate\_devices, 175
- get\_accessories\_settings, 177
- get\_analog\_data, 177
- get\_bootloader\_version, 177
- get\_brake\_settings, 178
- get\_calibration\_settings, 178
- get\_chart\_data, 178
- get\_control\_settings, 179
- get\_control\_settings\_calb, 179
- get\_controller\_name, 180
- get\_ctp\_settings, 180
- get\_debug\_read, 181
- get\_device\_count, 181
- get\_device\_information, 181
- get\_device\_name, 182
- get\_edges\_settings, 182
- get\_edges\_settings\_calb, 183
- get\_emf\_settings, 183
- get\_encoder\_information, 184
- get\_encoder\_settings, 184
- get\_engine\_advanced\_setup, 184
- get\_engine\_settings, 185
- get\_engine\_settings\_calb, 185
- get\_entype\_settings, 186
- get\_enumerate\_device\_controller\_name, 186
- get\_enumerate\_device\_information, 186
- get\_enumerate\_device\_network\_information, 187
- get\_enumerate\_device\_serial, 187
- get\_enumerate\_device\_stage\_name, 188
- get\_extended\_settings, 188
- get\_extio\_settings, 188
- get\_feedback\_settings, 189
- get\_firmware\_version, 189
- get\_gear\_information, 190
- get\_gear\_settings, 190
- get\_globally\_unique\_identifier, 190
- get\_hallsensor\_information, 191
- get\_hallsensor\_settings, 191
- get\_home\_settings, 191
- get\_home\_settings\_calb, 192
- get\_init\_random, 192
- get\_joystick\_settings, 193
- get\_measurements, 193
- get\_motor\_information, 194
- get\_motor\_settings, 194
- get\_move\_settings, 194
- get\_move\_settings\_calb, 195
- get\_network\_settings, 195
- get\_nonvolatile\_memory, 195
- get\_password\_settings, 196
- get\_pid\_settings, 196
- get\_position, 197
- get\_position\_calb, 197
- get\_power\_settings, 197
- get\_secure\_settings, 198
- get\_serial\_number, 198
- get\_stage\_information, 198
- get\_stage\_name, 199
- get\_stage\_settings, 199
- get\_status, 199
- get\_status\_calb, 200
- get\_sync\_in\_settings, 200
- get\_sync\_in\_settings\_calb, 201
- get\_sync\_out\_settings, 201
- get\_sync\_out\_settings\_calb, 202
- get\_uart\_settings, 202
- goto\_firmware, 203
- H\_BRIDGE\_ALERT, 145
- HOME\_DIR\_FIRST, 146
- HOME\_DIR\_SECOND, 146
- HOME\_HALF\_MV, 146
- HOME\_MV\_SEC\_EN, 146
- HOME\_STOP\_FIRST\_BITS, 146
- HOME\_STOP\_FIRST\_LIM, 146
- HOME\_STOP\_FIRST\_REV, 147
- HOME\_STOP\_FIRST\_SYN, 147
- HOME\_STOP\_SECOND\_BITS, 147
- HOME\_STOP\_SECOND\_LIM, 147
- HOME\_STOP\_SECOND\_REV, 147
- HOME\_STOP\_SECOND\_SYN, 147
- HOME\_USE\_FAST, 148
- has\_firmware, 203
- JOY\_REVERSE, 148
- LOW\_UPWR\_PROTECTION, 148
- logging\_callback\_stderr\_narrow, 203
- logging\_callback\_stderr\_wide, 204
- logging\_callback\_t, 164
- MICROSTEP\_MODE\_FRAC\_128, 148
- MICROSTEP\_MODE\_FRAC\_16, 148
- MICROSTEP\_MODE\_FRAC\_2, 148
- MICROSTEP\_MODE\_FRAC\_256, 149
- MICROSTEP\_MODE\_FRAC\_32, 149
- MICROSTEP\_MODE\_FRAC\_4, 149
- MICROSTEP\_MODE\_FRAC\_64, 149
- MICROSTEP\_MODE\_FRAC\_8, 149
- MICROSTEP\_MODE\_FULL, 149
- MOVE\_STATE\_ANTIPLAY, 150

MOVE\_STATE\_MOVING, [150](#)  
MOVE\_STATE\_TARGET\_SPEED, [150](#)  
MVCMD\_ERROR, [150](#)  
MVCMD\_HOME, [150](#)  
MVCMD\_LEFT, [150](#)  
MVCMD\_LOFT, [151](#)  
MVCMD\_MOVE, [151](#)  
MVCMD\_MOVR, [151](#)  
MVCMD\_NAME\_BITS, [151](#)  
MVCMD\_RIGHT, [151](#)  
MVCMD\_RUNNING, [151](#)  
MVCMD\_SSTP, [152](#)  
MVCMD\_STOP, [152](#)  
MVCMD\_UKNWN, [152](#)  
msec.sleep, [204](#)  
open\_device, [204](#)  
POWER\_OFF\_ENABLED, [152](#)  
POWER\_REDUCE\_ENABLED, [152](#)  
POWER\_SMOOTH\_CURRENT, [152](#)  
PWR\_STATE\_MAX, [153](#)  
PWR\_STATE\_NORM, [153](#)  
PWR\_STATE\_OFF, [153](#)  
PWR\_STATE\_REDUCE, [153](#)  
PWR\_STATE\_UNKNOWN, [153](#)  
probe\_device, [205](#)  
REV\_SENS\_INV, [153](#)  
RPM\_DIV\_1000, [154](#)  
SETPOS\_IGNORE\_ENCODER, [154](#)  
SETPOS\_IGNORE\_POSITION, [154](#)  
STATE\_ALARM, [154](#)  
STATE\_BORDERS\_SWAP\_MISSET, [154](#)  
STATE\_BRAKE, [154](#)  
STATE\_BUTTON\_LEFT, [155](#)  
STATE\_BUTTON\_RIGHT, [155](#)  
STATE\_CONTROLLER\_OVERHEAT, [155](#)  
STATE\_CONTR, [155](#)  
STATE\_CTP\_ERROR, [155](#)  
STATE\_DIG\_SIGNAL, [155](#)  
STATE\_EEPROM\_CONNECTED, [156](#)  
STATE\_ENC\_A, [156](#)  
STATE\_ENC\_B, [156](#)  
STATE\_ENGINE\_RESPONSE\_ERROR, [156](#)  
STATE\_ERRC, [156](#)  
STATE\_ERRD, [156](#)  
STATE\_ERRV, [157](#)  
STATE\_EXTIO\_ALARM, [157](#)  
STATE\_GPIO\_LEVEL, [157](#)  
STATE\_GPIO\_PINOUT, [157](#)  
STATE\_IS\_HOMED, [157](#)  
STATE\_LEFT\_EDGE, [157](#)  
STATE\_LOW\_USB\_VOLTAGE, [158](#)  
STATE\_OVERLOAD\_POWER\_CURRENT, [158](#)  
STATE\_OVERLOAD\_POWER\_VOLTAGE, [158](#)  
STATE\_OVERLOAD\_USB\_CURRENT, [158](#)  
STATE\_OVERLOAD\_USB\_VOLTAGE, [158](#)  
STATE\_POWER\_OVERHEAT, [158](#)  
STATE\_REV\_SENSOR, [159](#)  
STATE\_RIGHT\_EDGE, [159](#)  
STATE\_SECUR, [159](#)  
STATE\_SYNC\_INPUT, [159](#)  
STATE\_SYNC\_OUTPUT, [159](#)  
STATE\_WINDING\_RES\_MISMATCH, [159](#)  
SYNCIN\_ENABLED, [160](#)  
SYNCIN\_GOTOPOSITION, [160](#)  
SYNCIN\_INVERT, [160](#)  
SYNCOUT\_ENABLED, [160](#)  
SYNCOUT\_IN\_STEPS, [160](#)  
SYNCOUT\_INVERT, [160](#)  
SYNCOUT\_ONPERIOD, [161](#)  
SYNCOUT\_ONSTART, [161](#)  
SYNCOUT\_ONSTOP, [161](#)  
SYNCOUT\_STATE, [161](#)  
service\_command\_updf, [205](#)  
set\_accessories\_settings, [205](#)  
set\_brake\_settings, [206](#)  
set\_calibration\_settings, [206](#)  
set\_control\_settings, [206](#)  
set\_control\_settings\_calb, [207](#)  
set\_controller\_name, [207](#)  
set\_correction\_table, [208](#)  
set\_ctp\_settings, [208](#)  
set\_debug\_write, [209](#)  
set\_edges\_settings, [209](#)  
set\_edges\_settings\_calb, [210](#)  
set\_emf\_settings, [210](#)  
set\_encoder\_information, [211](#)  
set\_encoder\_settings, [211](#)  
set\_engine\_advanced\_setup, [211](#)  
set\_engine\_settings, [212](#)  
set\_engine\_settings\_calb, [212](#)  
set\_entype\_settings, [213](#)  
set\_extended\_settings, [213](#)  
set\_extio\_settings, [213](#)  
set\_feedback\_settings, [214](#)  
set\_gear\_information, [214](#)  
set\_gear\_settings, [215](#)  
set\_hallsensor\_information, [215](#)  
set\_hallsensor\_settings, [215](#)  
set\_home\_settings, [217](#)  
set\_home\_settings\_calb, [217](#)  
set\_joystick\_settings, [218](#)  
set\_logging\_callback, [218](#)  
set\_motor\_information, [218](#)  
set\_motor\_settings, [219](#)  
set\_move\_settings, [219](#)  
set\_move\_settings\_calb, [219](#)  
set\_network\_settings, [220](#)  
set\_nonvolatile\_memory, [220](#)  
set\_password\_settings, [220](#)  
set\_pid\_settings, [221](#)

- set\_position, [221](#)
- set\_position\_calb, [222](#)
- set\_power\_settings, [222](#)
- set\_secure\_settings, [222](#)
- set\_serial\_number, [223](#)
- set\_stage\_information, [223](#)
- set\_stage\_name, [224](#)
- set\_stage\_settings, [224](#)
- set\_sync\_in\_settings, [224](#)
- set\_sync\_in\_settings\_calb, [225](#)
- set\_sync\_out\_settings, [225](#)
- set\_sync\_out\_settings\_calb, [226](#)
- set\_uart\_settings, [226](#)
- UART\_PARITY\_BITS, [161](#)
- WIND\_A\_STATE\_ABSENT, [161](#)
- WIND\_A\_STATE\_MALFUNC, [162](#)
- WIND\_A\_STATE\_OK, [162](#)
- WIND\_A\_STATE\_UNKNOWN, [162](#)
- WIND\_B\_STATE\_ABSENT, [162](#)
- WIND\_B\_STATE\_MALFUNC, [162](#)
- WIND\_B\_STATE\_OK, [162](#)
- WIND\_B\_STATE\_UNKNOWN, [163](#)
- write\_key, [227](#)
- XIMC\_API, [163](#)
- ximc\_version, [227](#)
- ximc\_version
  - ximc.h, [227](#)