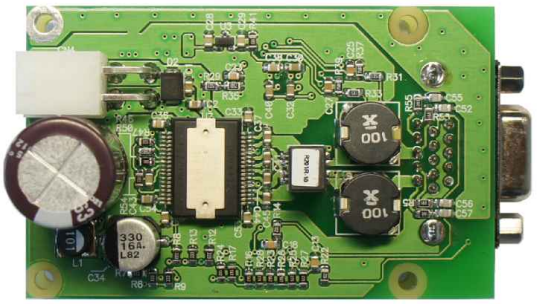
8SMC4-USB User Manual

1. About

1.1. General information

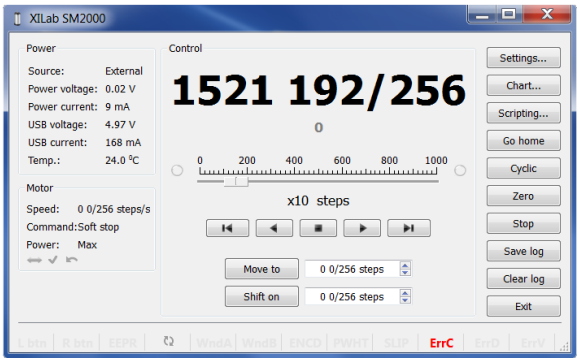


Controller board.

We offer an inexpensive and ultra-compact servo-drive with USB interface for stepper motors with external power supply.

Forget about cumbersome and expensive servo-drives! All you need is a stepper motor, a controller, an USB cable and any stabilized external power supply. That is all you need! Forget about active coolers as well. Controller's board is about the same size as a notepad or a mobile phone, therefore, you may just put it down on the worktable without any assembly procedures. The controller works with any type of compact stepper motors with the rated winding current of up to 3A. Controller works with stepper motors equipped with encoders in feedback, including linear encoders at the positioners as well as with no feedback motors. The motor connector at the controller board is the same as one used by Standa company and it is fits to all the Standa positioners. Usual USB connector provides easy communication and work with computer. Several controllers can be connected to one computer either via USB ports or through a special backboard supplied with multiaxis systems. The controller is fully compatible with almost all operating systems, e.g., Windows, Mac OS X, Linux, etc.

Note: The device is under development stage. All the features and specifications are preliminary.



XILab main window.

All the necessary software including all configuration files are supplied with the controller. It makes the start quickly, according to "plug-and-play" principle. We continuously develop new configuration files for more supported motors. Therefore, all you need is to download the file for your positioner from Configuration files chapter, open it with XILab software and Apply. It is enough, all specific settings will be done! Just use "Move to" and "Shift" commands and controller will do whatever you want.

1.2. Benefits

Main benefits

- **Compact and powerful!** The controller's dimensions are 47x37x80mm including all connectors. The device is adapted to all stepper motors with rated winding current of up to 3A.
- **Compatible with 8SMC1-USBh** After updating the MicroSMC driver, all the software for 8SMC1-USBh will be working with 8SMC4-USB.
- **Compatible with all the Standa positioners!** Have you got a Standa positioner? Just plug and play, we have already done the rest for you!
- **It does remember all!** Feel free from saving the current position on the computer: the controller does it itself using its own nonvolatile memory that works even after a sudden power cut.
- **It works with peripherals!** It supports a quadrature encoder, magnetic brake, a joystick, limit switches, a zero position sensor – it all included, just plug and work!
- **Built-in zero calibration!** Using the limit switches, the revolution sensor, the external signal or their combination, the zero calibration is performed by a single command!
- **It works with all computers!** All the supplied software is compatible with Windows 8, Windows 7, Windows Vista, Windows XP, Linux, Mac OS X, including 64-bit versions.

All benefits

- **The most compact!** The controller's dimensions are 47x37x80mm including all connectors.
- **Really powerful!** It is adapted to all the stepper motors with rated winding current of up to 3A.
- **Compatible with 8SMC1-USBh** After updating the MicroSMC driver, all the software for 8SMC1-USBh will be working with 8SMC4-USB.
- **Compatible with all the Standa positioners!** Have you got a Standa positioner? Just plug and play, we have already done the rest for you!
- **It knows its own set!** A built-in feature for downloading the configuration file right from the positioner is available! The positioner's support of such memory is required.
- **Choose your interface!** Both USB and serial port are built-in and ready to use.
- **Really fast!** Up to 35,000 steps per second for any step grade!
- **Precise!** The step grade modes from full step to 1/256 of the step on all the speeds.
- **It does remember all!** Feel free from saving the current position on the computer: the controller does it itself using its own nonvolatile memory that works even after a sudden power cut.
- **It works with peripherals!** It supports a quadrature encoder, magnetic brake, a joystick, limit switches, a zero position sensor – it all included, just plug and work! Additional stabilized output for peripherals (5V, 100mA) is available.
- **Built-in zero calibration!** Using the limit switches, the revolution sensor, the external signal or their combination, the zero calibration is performed by a single command!
- **Stand-alone!** Would you like to work in the stand-alone mode? Just go ahead! An external joystick, a keypad or their combination is supported.
- **Energy conserving!** Programmable current reduction in the motor winding in the hold mode within 1% accuracy.
- **Silent!** Smooth movement at lower speeds and no extra noise at higher speeds.
- **Protected!** An ESD protection on all pins of external connectors and additional short circuit protection for the motor windings.
- **Attentive!** It controls the temperature of the processor and the power driver as well as both currents and voltages for the power supply and USB.
- **Modern!** The micro software in the nonvolatile memory of the controller is updating via USB interface.
- **Controlling and controllable!** The built-in synch input and output allow to start the rotation to desired position by the incoming external signal and/or to transmit the outgoing signal after the desired position is reached. The analogue common input and the digital common input/output are built in as well.
- **Comprehensible!** The status LED displays the power supply and the controller's state. For convenience of use both signals doubled at the external LEDs as well as the state of the limit switches.
- **Multiaxis machine!** The multiaxis systems are designed by using standard USB hubs, either external or mounted at a special backplane. There may be up to 32 axis in the system.
- **It works with all computers!** All the supplied software is compatible with Windows 8, Windows 7, Windows Vista, Windows XP, Linux, Mac OS X, including 64-bit versions.
- **Examples for all programming languages!** A cross-platform library with examples allows to start quickly programming using C++, C#, .NET, Delphi, Visual Basic, gcc, Xcode, Matlab, Java and LabVIEW is supplied with the controllers.
- **Full-featured interface!** The XILab user interface is supplied with the controller. It allows to easily control all the functions and features of the device without any programming.
- **Unique scripting language!** A scripting language is integrated into XILab software. It allows easy setting the sequence of actions, including cycles and branches, without compilation or learning any programming language.

1.3. Table of specifications

Power supply	external 12V - 36V DC
Power consumption	up to 5A (depending on the voltage) from external power supply, up to 400mA from USB
Current in the motor winding	up to 3A
Protection types	current overload protection, voltage overload protection, short circuit protection, motor hot plug/unplug protection
Motion	left/right move, move to point, shift on delta, continuous speed, acceleration and deceleration ramps, backlash compensation mode, automatic home position calibration mode
Step modes	full-step, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256
Rotation speed	up to 35 000 full steps per second
Speed profile	trapezoidal
Position meter	40 bit
Feedback	quadrature encoder (optional)
Limit switches	2
Feedback range width	200 kHz

Note: According to this manual, the controller's working voltage range is 12V to 36V DC. The voltage limits are 2V and 48V DC. Once the voltage exceeds 48V, the controller is assured to fall out. While the voltage is less than 2V, the controller is unable to provide the motor rotation.

1.4. Specifications

Motor requirements

- Motor type: bipolar stepper motor.
- Rated winding current: minimum 100mA.
- Rated winding voltage: minimum 2V DC.

Electric specifications of the controller

- Power supply modes: external and USB.
- Current in each motor winding: up to 3A.
- Maximum encoder pulses frequency: 200 kHz.
- Stabilized 5V DC output (the power supply for encoder and other peripherals): the maximum output current is 100mA, the minimum output voltage stability is 5%.
- ESD-protection on all pins of the output connectors (e.g., D-Sub 15 pin, mini-USB or power jack).
- Winding-to-ground short circuit protection.
- Winding-to-winding short circuit protection.
- Motor hot-swapping protection.
- Wrong power polarity protection (not more than 1s).
- Voltage overload protection (not more than 1s).
- USB-supplied current limitation.
- External power supply current limitation.
- Motor rotation speed limitation.
- Programmable full winding current with 10mA precision.
- Programmable winding current decrease with 1% precision for the hold mode.

Rotation control features

- Step grade modes: full-step, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256.
- Noiseless at the lower speed.
- Minimum speed is 1/256 of the full step per second.
- Maximum speed is up to 35 000 full steps per second for all the step grade modes.
- Minimum displacement is 1/256 of the step.
- Maximum displacement is 2,147,483,647 full steps for all the step grade modes.
- Smooth start/stop mode.
- 40-bit position counter (32 bit for full step and 8 bit for microstep).
- Motion: left/right move, move to point, shift on delta, continuous speed, acceleration and deceleration ramps, backlash compensation mode, automatic home position calibration mode.

Firmware additional features

- Automatic HOME calibration at firmware level.
- The nonvolatile memory used for saving/downloading the controller settings.
- Software update via USB interface.
- Automatic position saving according to step counter and encoder with power-off protection.

Additional features available via motor connector

- Processing the signals of one or two limit switches adjustable by the software.
- The Standa positioners recognition and automatic downloading the configuration file right from the positioner if the last one supports this feature. **Will be in firmware 3.9**.
- The "steps loss" detection using either a revolution sensor or a quadrature encoder (if positioner supports this feature).
- The position detection using a quadrature encoder. The x4 mode.
- The stepper motor control using master quadrature encoder mode, providing the maximum speed without any step loss. **Will be in firmware 3.9**.

Additional features available via backplane connector

- USB connector on backplane that duplicates USB input on controller board.
- A serial (RS-232) port. TX and RX lines only. Specifications: speed 9600 - 921600 baud, TTL 3.3V. The Ethernet configurations based on the serial port are available by request.
- Synchronization input: once the pulse is transmitted via this pin, the controller starts rotating the motor to predetermined position or by predetermined displacement value. The triggering mode, the polarity and duration are adjustable by user. Specifications: TTL 3.3V.
- Synchronization output: the pulse rise on this pin if rotation is started or finished, or predetermined user-defined displacement value is reached. The triggering mode, the polarity and duration are adjustable by user. Specifications: TTL 3.3V.

- Left or right buttons. Once the button is pressed, the rotation for corresponding direction starts and the speed increases gradually according to acceleration and other settings. Specifications: TTL 3.3V.
- Joystick pin allowing operations with various joysticks with the voltage range not wider than 0–3V.
- Magnetic brake control pin providing to control magnetic brake mounted at the motor shaft. Specifications: TTL 3.3V, 5mA.
- Common analogue input pin allowing operations with signals within 0–3V range. Reading frequency is 1kHz. The configuration is programmable.
- Common digital input/output pin. Updating frequency is 1kHz. The configuration is programmable. Specifications: TTL 3.3V, 5mA.
- Limit switches indication pins designed for direct connection of LEDs. Specifications: TTL 3.3V, 2mA.
- Digital "Power" and "Status" pins duplicate the status LED and designed for direct connection of LEDs. Specifications: TTL 3.3V, 2mA.
- External driver control interface allowing to control any type of external driver using three signals: enable, direction, clock. **Will be in firmware 3.9**.
- Multiaxis systems development. The multiaxis systems are designed by using standard USB hubs, either external or mounted at a special backplane. Normally multiaxis system looks at the PC side as a list of virtual serial ports, according to the quantity of axis being connected.

Programming the controller

- All the software supplied with controller is compatible with Windows 8, Windows 7, Windows Vista, Windows XP, Linux, Mac OS X, including their 64-bit versions.
- A cross-platform library with examples allow to quickly start programming using C++, C#, .NET, Delphi, Visual Basic, gcc, Xcode, Matlab, Java and LabVIEW is supplied with the controller.
- The XILab user interface is supplied with the controller. It allows to easily control all the functions and features of the device without any programming.
- A scripting language, an EcmaScript language dialect, is integrated into XILab software. It allows easy setting the sequence of actions, including cycles and branches, without compilation or learning any programming language.

Note: the basic software version doesn't include some extra features; nevertheless, they are available by request.

2. Safety instructions while operating the controllers

IMPORTANT. While the power supply unit is on, while the controller board is connected to PC if power supply unit and PC are grounded, and while the common electrode of power supply is disconnected from controller board, **the plus power electrode connection to controller or power cord hot-swapping is strictly forbidden, otherwise PC can get damaged!** This is a common requirement for all the self-powered electronic devices connected to PC via USB interface.

IMPORTANT. While operating the controller, **the power overload over the maximum voltage of 48V is forbidden**. Exceeding this value will immediately damage the controller.

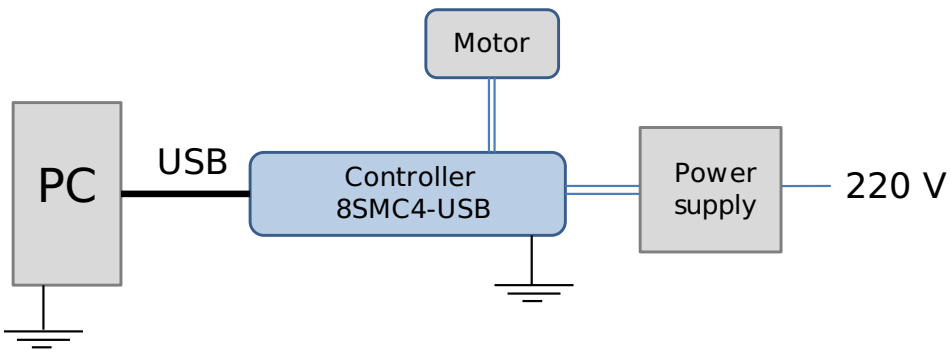
Warning. Power supply unit is designed for providing the current required for motor rotation. Absolutely minimum current value is calculated using the following formula: $I_{power.min} = \frac{2 * I_{motor} * U_{motor}}{U_{power}}$ where $I_{power.min}$ is the minimum working current of power supply unit, I_{motor} is the operating current in the winding, U_{power} is stabilized power supply unit voltage, and U_{motor} is rated operating voltage of the motor. The power supply unit with operating current equal to $I_{power} \geq 2 * I_{power.min}$ is recommended to use. The U_{power} voltage is to be more than U_{motor}. The higher the voltage, the faster rotation speed is reached.

Power consumption of power supply unit may be used instead of working current. Absolute minimum of power consumption is $W_{power.min} = I_{power.min} * U_{power} = 2 * I_{motor} * U_{motor}$. For example, for motor with operating winding current of 1A and operating voltage of 5V (with 5W rated power consumption), the operating voltage of power supply unit may be defined as 20V with the output power of at least 10W (the maximum operating current of power supply unit is at least 0.5A).

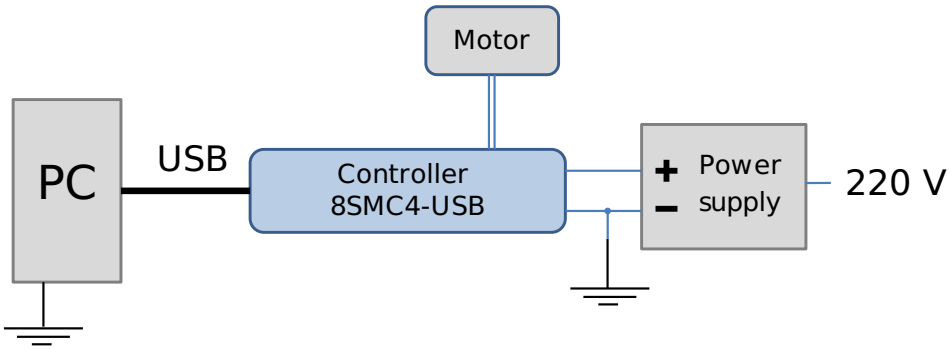
Warning. Before connecting the controller to the motor or to the PC via USB interface, it is recommended to connect controller to power supply with proper grounding or to use separate grounding for controller with the specially marked ground terminal.

Warning. While operating the controller, the following is **not recommended**:

- Connection/disconnection the USB cable while the motor is powered on
- Connection/disconnection the cable between the motor and controller while the motor is powered on



General controller connection diagram.

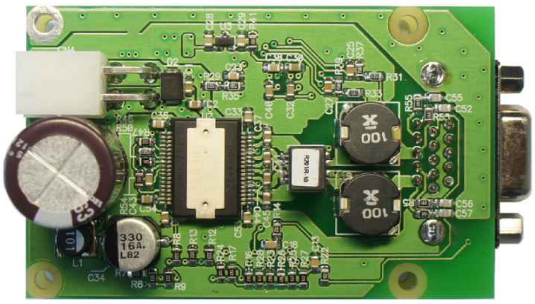


Controller grounded via minus electrode of power cable connection diagram.

IMPORTANT. The power supply unit is to be plugged to grounded 220V AC socket (a three-electrode connection scheme) or the controller has to be grounded separately using the specially marked terminal. Non-compliance with these rules may make controller malfunction and decrease its noise immunity.

Warning. If the controller is equipped with ground terminal, you must plug it to the grounding circuit in your lab. The grounding via the power supply unit is also permitted. In this case the minus power electrode is used as the grounding electrode. Make sure that the minus electrode of your power supply unit is grounded. Earth the controller via grounding terminal ONLY if the power supply unit with minus electrode disconnected from the grounding circuit is used.

3. Overview and getting started



Controller board.

Introduction

We thank you for purchasing our motor controller and congratulate you as its fortunate owner!

This manual describes the controller installation procedures and getting use of XILab software for Windows 7 started. The detailed controller specifications are described in Specifications chapter. For developing your own applications, please read the Programming guide chapter and download the programming software pack from the software chapter.

System requirements

Note. There are only brief requirements in this chapter. For detailed information please read the Specifications chapter.

For successful installation you will need:

A standard PC with USB port.

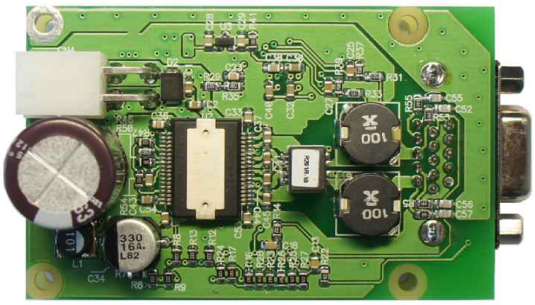
Make sure that all the actual Windows updates are installed on your PC. It will help you to maintain the higher level of software compatibility. If necessary, please download the latest updates from www.microsoft.com/updates.

The software.

Plug the USB drive with XILab installation pack to USB port of your PC or download the latest software updates from the folder with your operating system name.

USB-A to mini-USB-B cable.

For detailed information about the USB cable please read the Mini USB connector chapter.



The motor controller board.

The controller appearance may differ from the one shown on the above figure depending on its configuration and version. For detailed information about versions please read the Specifications chapter.

The stepper motor-based positioner.

The stepper motor-based positioner used in the operations is shown at the figure. The detailed motor requirements are described in Specifications chapter. If you use your own cables for connecting the positioner to the controller, please check the motor connection scheme according to positioner connection scheme and considering the controller's output connectors scheme described in Technical specification chapter. For positioners with limited displacement range, two limit switches must be used: SW1 and SW2. These pins are used to determine the displacement limits.

Stabilized power supply unit.

Note. Please use the 12–36V DC stabilized power supply. Too high voltage may damage the controller. For more information please read the Safety instructions while operating the controllers chapter. The power supply unit must provide the current enough for sustainable rotation of the motor. The details are described in the Safety instructions while operating the controllers chapter.

Note. Please pay attention on the manual supplied with your controller. The more strict power voltage limitation is possible depending on the controller model. Please check the connection of external power supply unit to the controller carefully.

Note. If controller is supplied inside the metal case, the case must be earthed. If controller is supplied without any body, the grounding circuit of power supply unit is used. For more information please read the Safety instructions while operating the controllers chapter.

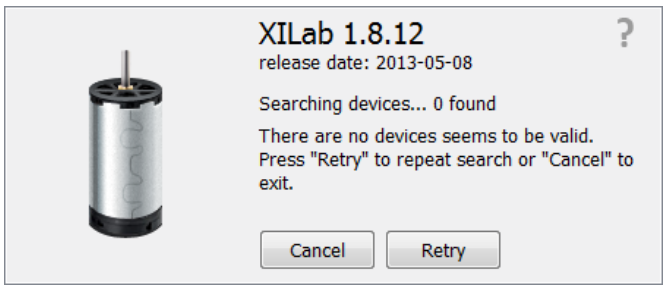
Note. Make sure that if the board working without case is laying on the insulating surface and there is no any inconsequential object or particle on the board itself and around it.

Software installation and startup procedures

Make sure that all controllers are disconnected from your PC.
The software installation manual is here. Select XILab-X.X.X-win32.exe file for 32-bit version of Windows or XILab-X.X.X-win64.exe for 64-bit version and launch the installation program. The installation window will appear. (The software versions may slightly differ from each other).

XILab main installation window.

Press "Next>" button and follow the instructions on the monitor. All the necessary software, including all drivers, packs and programs, will install automatically. After installation is finished, the XiLab software starts by default and the following window will open:



XILab "No devices found" dialogue window.

Don't press any button.
Earth the controller or power supply unit. Plug the positioner to the controller. Then connect the stabilized power supply unit. Plug the controller to your PC using the USB-A to mini-USB-B cable.

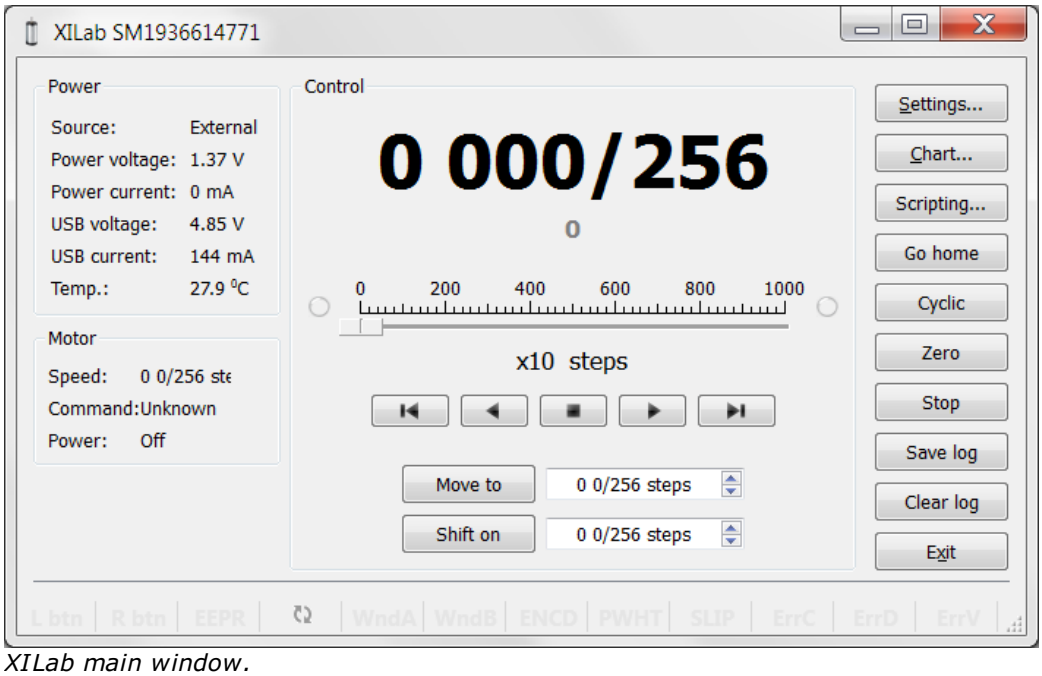
The LED indicator at the controller board will start flashing. The New Hardware Wizard starts working after the first connection of the controller to PC. Please wait until Windows detects a new device and installs all necessary drivers for it.

If the automatic driver installation has failed, please select ”No, not this time” in the window being opened and press “Next>” button. Select “Install from a list or specific location (Advanced)” in the next window and press “Next>” again. Browse the software disk supplied with controller and find the *.inf file there or in the XiLab-install-path\driver\ folder and wait until installation will complete.

Go back to XILab "No devices found" dialogue window and press "Retry" button.
If this window is closed, please go to “Start” menu, select Programs->XILab X.X.X->XILab and launch it. The dialogue window will open again.

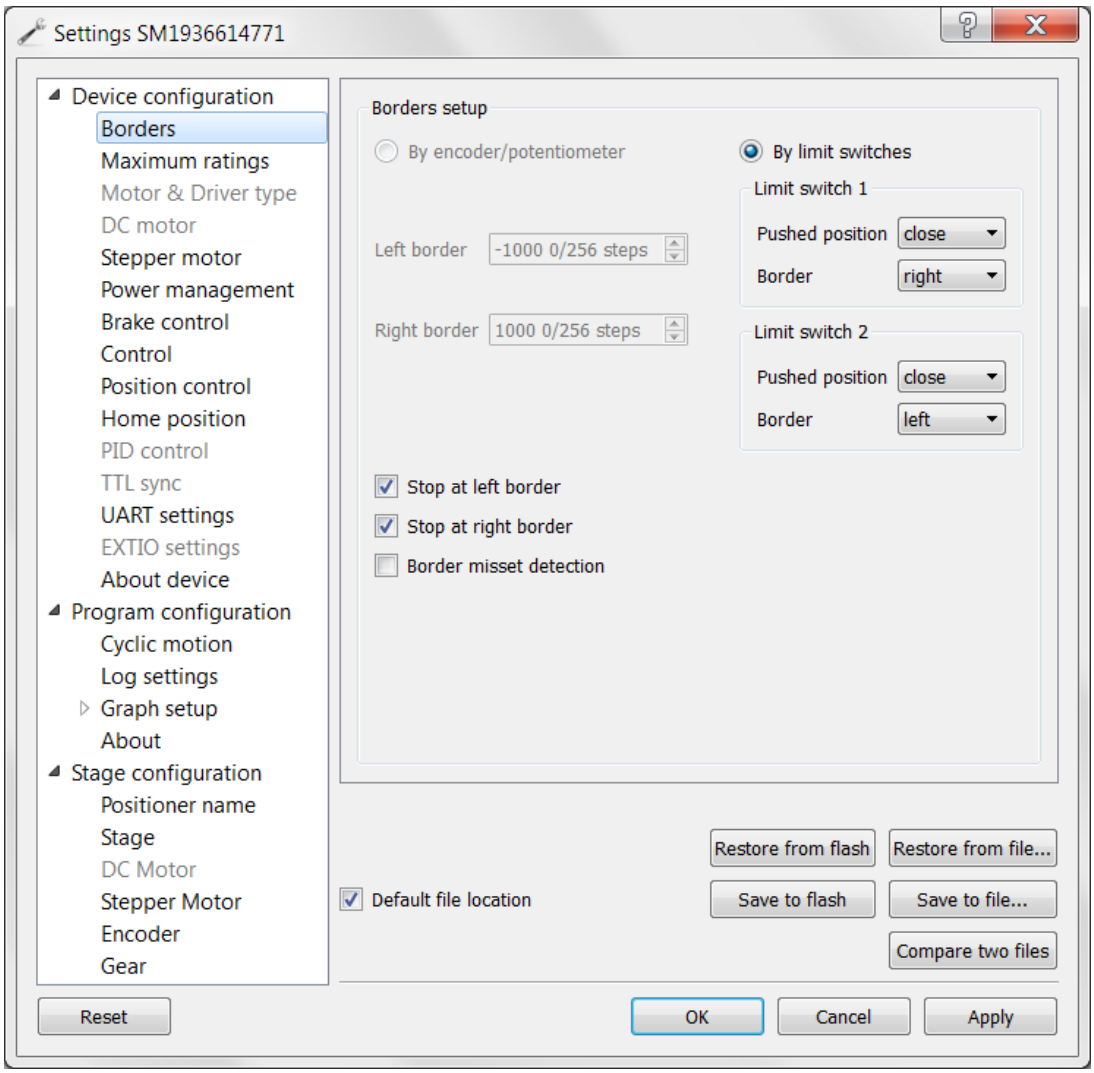
Getting started with XILab software

XILab is a user-friendly graphic interface designed for control, diagnostics and adjustment of motors. It also provides an easy installation and saving/downloading the parameters for any type of motors. This chapter describes the startup procedures with XILab software. For complete information please refer to XILab application User's guide chapter.



XILab main window.

Open “Settings...”, then press “Restore from file...” and select the configuration file for your positioner from the opened XiLab-install-path\profiles\ folder. The values applicable for your positioner will automatically fill all the fields of “Settings...” menu. If the necessary file isn't found, please refer to the Configuration files chapter. If there is still no solution, please leave your request at our customer service website.



XILab, the Settings menu window.

Warning. For successful operation of your controller with stepper motors, you must configure properly the following:

- working current,
- displacement limits and limit switches,
- critical parameters,
- limiters,
- power supply mode.

If you decided to configure your controller by yourself, please check these parameters carefully!

Congratulations, your controller is ready for operation!

Functional test

Check the proper controller configuration by pressing left or right button in the central row of XILab main window control buttons. The positioner has to start moving. The central smooth halt button is used in order to stop the rotation. Please pay attention on power supply parameters of the controller in the *Power* section. The power voltage, the consumption current and the temperature of the controller are shown there.

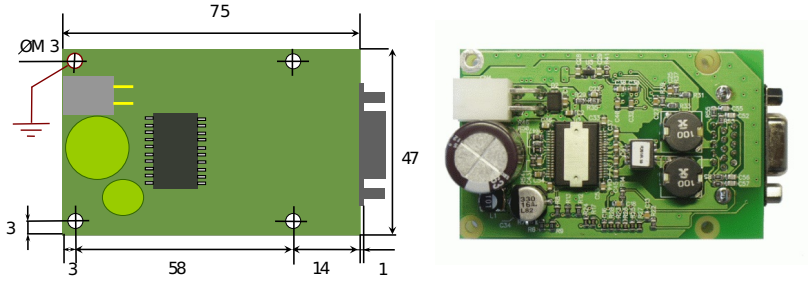
The XiLab main window turned to red while starting the rotation indicates that some protection and Alarm mode are activated. Either improper configuration or incorrect connection of the positioner or controller fault may be the reason. For detailed information please read the Critical parameters chapter.

4. Technical specification

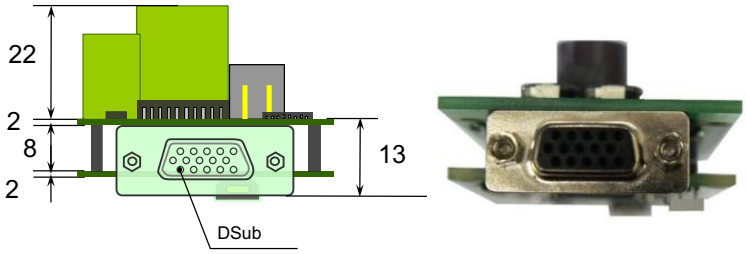
4.1. Appearance and connectors

4.1.1. Dimensions and arrangement

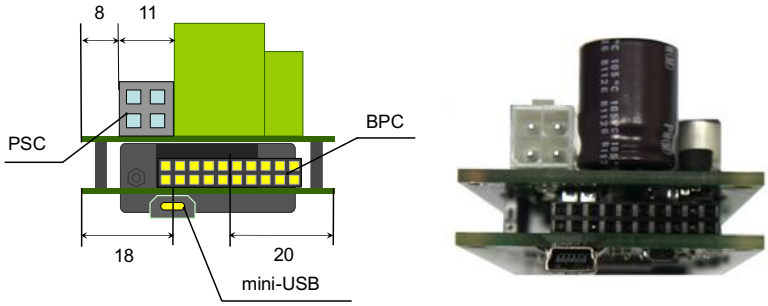
Structurally the controlled is designed as two boards, 47x75mm each, rigidly connected to each other. A logic controller and control systems are mounted at the bottom board, a power part is at the top board. A radiator at the power part is available.



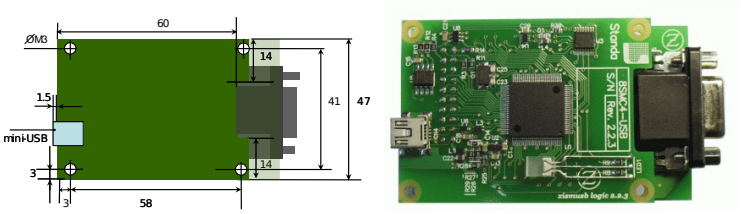
Top view on the controller. The view from power part and radiator side.



Front view on the controller. The view from positioner cable side.



Rear view on the controller. The view from backplane connector side.



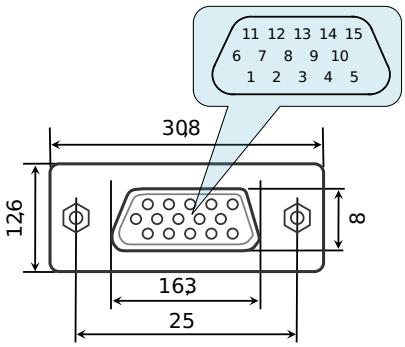
Bottom view on the controller. The view from micro USB port side.

Note. Controller's appearance may differ depending on items supplied with it. For example, the controller is supplied either in plastic or in metal body, there may be one or several axis in the body, etc. In that case the dimensions will differ from that are given in specifications.

IMPORTANT. Mounting the radiator to the power part by yourself, please make sure that there is no contact between heat-conducting surfaces and conductive elements of the unit. Such contact may damage the power circuit! This warning is applicable only to controllers supplied without body.

4.1.2. Positioner connector

A female DSub 15-pin connector for positioner is mounted on the controller board.



Dimensions and numbers of the pins in DSub connector (front view).

Pins functionality:
1 - Not phase B of SM or - DC of the motor
2 - Phase B of SM or + DC of the motor
3 - Not phase A of SM or - DC of the motor
4 - Phase A of SM or + DC of the motor
5 - 5V / up to 100mA stabilized output for encoder power supply
6 - One-wire interface for positioner identification (for Standa hardware only)
7 - Logic ground for limit switches, encoder, etc.
8 - 2nd limit switch
9 - 1st limit switch
10 - Encoder channel A
11 - Encoder channel B
12 - Revolution sensor input
13 - Hall effect sensor channel A
14 - Hall effect sensor channel B
15 - Hall effect sensor channel C

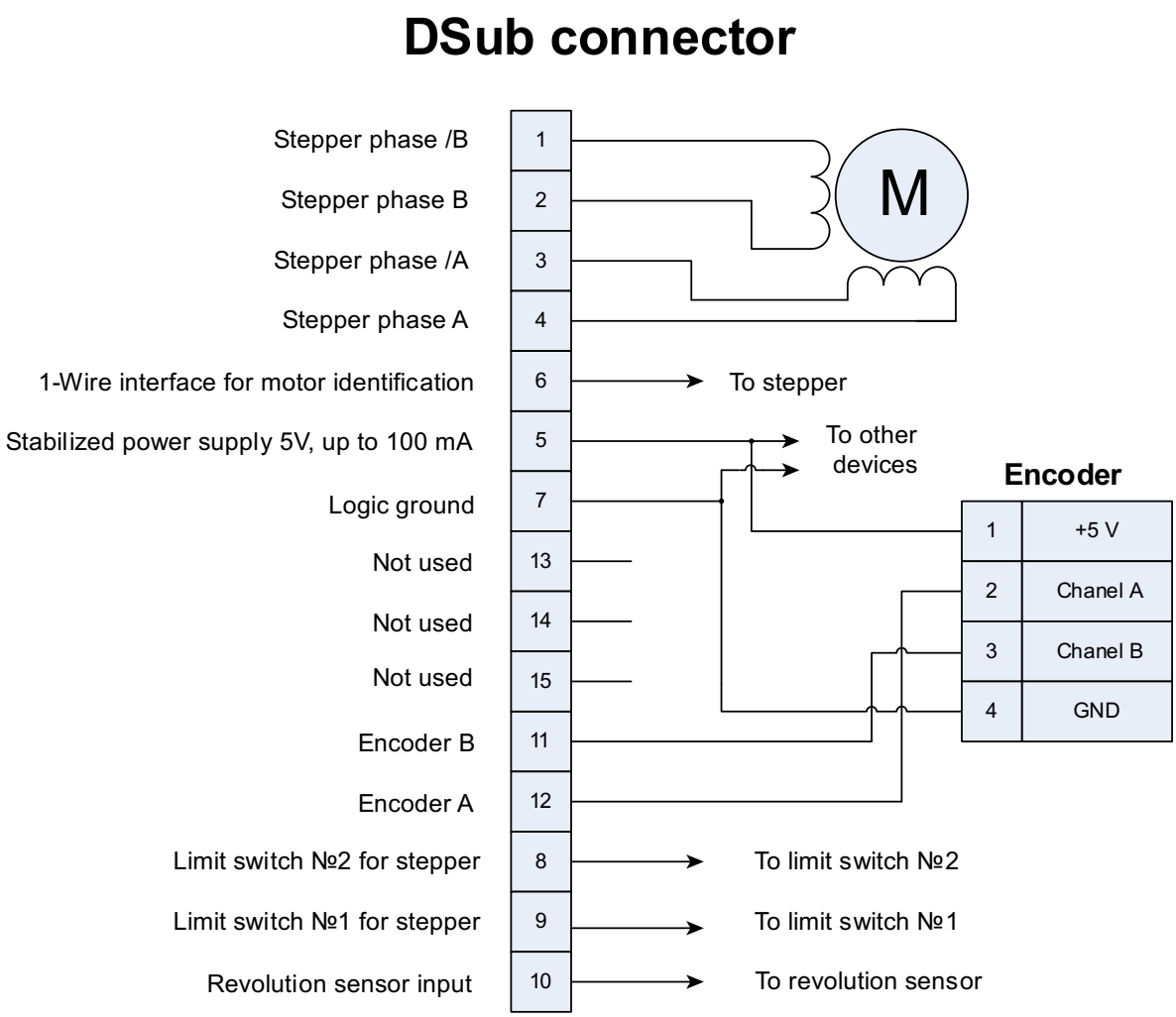
Note. Operating modes for DC motors and brushless motors are actually inactive and will be available in further controller revisions.

Note. Outputs 1 & 3 and 2 & 4 must be connected together for proper DC motor function.

Warning. Plugging in/out the motor to the controller is not recommended while motor windings are under voltage.

4.1.3. Positioner standard connection diagrams

Electric diagram of positioner and encoder connection

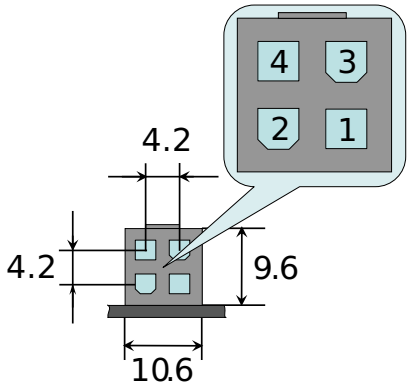


General diagram of positioner and encoder connection using Dsub connector.

Note. If A and B encoder channels work in open collector mode, some extra shifting of encoder outputs to 5V power voltage using the resistors may be required at high rotation speeds in order to provide the maximum signal transmission speed (see Operation with encoders).

4.1.4. Power supply connector at the board

A male 4-pin Mini-Fit connector with 4.2mm interval (type MF-4MRA) is mounted at the controller board for plugging in to power supply. Its comparable benefits are as following: high 8A current per pin, a fixation available, a possible coupling with both cable-mounted (type MF-4F, PN 39-01-2040 according to Molex catalogue) and board-mounted counterparts, including vertical (PN 15-24-7041 according to Molex catalogue). All Mini-Fit connectors are available in Molex catalogue at www.molex.com.



Dimensions and numbers of the pins in PSC (Power Supply Connector), front view.

Pins functionality:
1 - "-" power electrode.
2 - 12–36V "+" power electrode.
3 - "-" power electrode..
4 - 12–36V "+" power electrode.

Note. For controllers in the body (e.g., multiaxis systems), a different connector will probably be used in order to supply the power to controller and all the axis and peripherals simultaneously. Please check the pin-out of such connector with manufacturer.

IMPORTANT. Never supply the power to the controller and don't plug it to power connector if you are not confident that your power supply parameters comply the required ones. Never attempt plugging the power supply with controller if you are not confident about compatibility of power supply unit and controller connectors! The acceptable connection parameters are described in Safety instructions while operating the controllers.

IMPORTANT. Hot-swapping the power supply connector may damage PC or the controller. For more details please refer to Safety instructions while operating the controllers.

Note. Please read carefully the user manual for your controller: there may be a stricter limitation for power voltage, depending on controller model. Please carefully check up the reliability of connection between controller and external power supply unit.

4.1.5. Mini USB connector

A mini USB port is mounted at the controller board. It is intended for operating the controller using PC and for power supply in single-axis configuration. While developing the multiaxis systems, a backplane connector is used instead of mini USB port. The controller port supports a full-speed mode and USB 2.0 specification. The driver installation iFor more details please refer to Overview and getting started and Software.

USB-to-mini-USB cable.

Type B mini-USB connector.

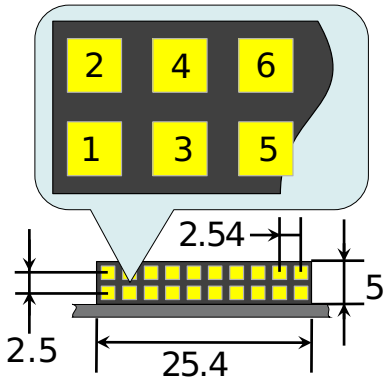
Table of mini USB output pins

Pin #	Name	Wire colour	Description
1	VCC	Red	+5V DC
2	D-	White	Data -
3	D+	Green	Data +
X	ID		N/C
4	GND	Black	Ground

Warning. Use verified and knowingly usable USB cables only! Damaged or dreck USB cable may cause improper controller operation, including motor rotation errors and errors of device recognition by PC operating system. Short cables with thick wires and screening are ideal for sustainable connection.)

4.1.6. Backplane connector

A female 20-electrode double-row pin connector (PBD-20R) with 2.54mm interval is mounted at the controller board for backplane connection.



Dimensions and numbers of output pins in BPC (BackPlane Connector), connector-side view.

Pins description:
1 - GND, duplicates the corresponding pin of mini USB connector.
2 - VBUS, duplicates the corresponding pin of mini USB connector.
3 - D-, duplicates the corresponding pin of mini USB connector..
4 - D+, duplicates the corresponding pin of mini USB connector..
5 - Joy, an analogue 0-3V input used for external joystick connection.
6 - Pot, a common analogue 0-3V input.
7 - RX, a serial port input, 3.3V logic.
8 - TX, a serial port output, 3.3V logic.
9 - A Clock signal for external driver control, 3.3V logic.
10 - A Direction signal for external driver control, 3.3V logic.
11 - BUT_R, an external "Right" button input.
12 - BUT_L, an external "Left" button input.
13 - A common input/output (or an Enable signal if an external driver is used), 3.3V logic.
14 - An output for magnetic brake control, 3.3V logic.
15 - Synchronization output, 3.3V logic.
16 - Synchronization input, 3.3V logic.
17 - A LED Power output displaying 3.3V power supply for controller.
18 - A LED Status output displaying controller's operating mode.
19 - A LED output for left limit switch.
20 - A LED output for right limit switch.

Note. No additional connection or shifting to ground/power supply is required for idle pins. Just don't use them.

IMPORTANT. A maximum designed operating voltage for Joy and Pot analogue inputs is UP TO 3V. Never supply these inputs with higher voltage, including 3.3V value, otherwise the proper operation of all the analogue channels of the controller may get corrupted and the controller itself or the motor may get damaged.

IMPORTANT. The outputs available at the internal connector are not protected. Mounting of any additional buffers or filtering chains is a total responsibility of user or developer who designs a backplane intended to use these lines.

IMPORTANT. While the controller is de-energized, no voltage over 0.3V relative to the DGND pin is allowed for the internal connector. Provision of any additional protection preventing those accidents (if possible) is a total responsibility of user or developer who designs a backplane intended to use these lines.

4.2. Kinematics and rotation modes

4.2.1. Predefined speed rotation mode

Predefined speed rotation mode is the main operating mode of the controller for all the motor types. It provides supporting the predefined rotation speed at a distance from the destination point and is usually used simultaneously with Rotation for predefined point or Predefined displacement modes. This mode may also get called by left and right motion commands.

Motor steps or encoder count-offs (if the encoder is available) are used as speed measurement units. Encoders work with all the types of motors. For DC motors the fixed speed is maintained even at the variable load.

Turning on the Acceleration mode temporarily deactivates the predefined speed rotation mode.

After the controller receives the command to start the motion, it rotates the motor with user-defined speed. **The speed adjustment is available** at the appropriate section of the "Settings..." menu of XILab software or using the set_move_settings() function (refer to Programming guide section). For stepper motors the **speed values** may be defined either in full steps per second or in microsteps per second, for DC motors the speed is defined in revolutions per minute (RPM).

The speed for special motion modes, e.g., for backlash compensation or automatic zero position calibration, is different from the general rotation speed and is adjusted separately.

The controller allows limitation of the maximum speed if appropriate parameter is defined by user. In that case any rotation with the speed over the maximum is performed with the maximum speed. A separate adjustment is available, providing use of the maximum speed for all the ordinary motion modes, except for special ones, e.g., backlash compensation or automatic zero position calibration. The maximum speed adjustment as well as the adjustment for modes using this speed are available at the appropriate section of the "Settings..." menu of XILab software.

The **actual speed** is displayed in XILab main window, at the Speed field, or at main operating parameters charts.

Note. If the **stability of the speed maintenance** seems to be insufficient while the encoder is used, please refer to recommendations for accurate rotation.

4.2.2. Rotation for predefined point

The rotation for predefined point mode is the main operating mode for all the types of motors and is usually used simultaneously with predefined speed rotation mode. It provides moving the positioner for the defined position with **absolute value** for destination point coordinates which is different from predefined displacement mode. An additional reciprocal motion close to predefined point may be performed at Backlash compensation mode.

While using the encoder, few barely noticeable "vibrations" may require in order to detect the actual position before the motor stops in predefined point.

After the starting command is received by the controller, it either switches on the acceleration mode (if the appropriate option is on) or immediately starts rotating the shaft of the motor with user-defined speed. After the predefined point is reached, the rotation stops whereas the deceleration activates (if the appropriate option is on). The **destination point** is set in XILab main window. The **destination point** may be defined either in full steps or in microsteps for stepper motors or in encoder count-offs for all types of motors.

The **actual position** is displayed in XILab main window, at the Control section, or at main operating parameters charts.

Note. If the **positioning accuracy** seems to be insufficient while the encoder is used, please refer to recommendations for accurate rotation.

4.2.3. Predefined displacement mode

The predefined displacement mode provides displacement of the positioner for predefined value relative to zero position (if this is a first command since the controller started) or relative to position reached by the motor after the previous commands are completed, i.e., the destination point coordinate is a **relative value**. This mode is useful when the absolute position is unknown or doesn't matter.

The predefined displacement mode is exactly analogous to rotation for predefined point mode. The differences concern the destination position computing rules only. While there is no motion actually, calling the predefined displacement command rotates the motor relatively to current position. If this command is sent during the rotation to the position (*MOVE*, *MOVR*, *SSTP*, *STOP*, *LOFT*), then the displacement interval adds to destination position and the controller re-adjusts for the new destination point while the motor is rotating. If this command comes during the rotation for certain direction, then the displacement value adds to previous destination position and the motion automatically re-adjusts for the new position.

IMPORTANT. The displacement is always performed relative to position reached by positioner while performing one of the previous command, *MOVE*, *MOVR*, *SSTP*, *STOP*, *LOFT*, or while performing the displacement after the previous incoming synchronization pulse has been received, despite whether the motion has been completed or interrupted.

Note. The predefined displacement mode is activated either by corresponding command or by incoming synchronization pulse. For more information please refer to TTL synchronization chapter.

4.2.4. Acceleration mode

The acceleration function is active **by default**. The acceleration is used for smooth start or finish of the rotation without shocks that are inevitable when the predefined speed is reached instantly. Moreover, inertia of rotor and the other components of positioner usually doesn't allow instant gathering of high speed which results to loss of steps as well as to failure in rotation during the operation without feedback. While operating the motor with feedback via encoder, the speed will increase as quickly as rotation limiters allow. Quick acceleration makes the rotation unstable as well as it makes more noise and vibration. Tat's why the acceleration mode is recommended. The acceleration function provides reaching of both maximum speed and sustainable motion even for motors with intermediate torque value.

The acceleration/deceleration mode works in the following way: during the speeding-up, while the required speed value is higher than the actual one, a gradual acceleration is performed at the Acceleration value which is measured in steps per squared second. After the required speed is reached, the controller switches to predefined speed rotation mode. Approaching the destination position, the controller begins to decrease the speed as he speeding-down would equal to Deceleration value and the motor would stop exactly at the destination point. Thus, this mode provides a trapezoidal speed profile. If the displacement distance is small, then the acceleration may change directly to deceleration; this will result to triangular speed profile. Turning the acceleration mode on and off, as well as acceleration or deceleration only, is possible using XiLab software (see the Settings of kinematics (Stepper motor) section) or by set_move_settings() command described in Programming guide.

The Acceleration value is adjusted independently from the Deceleration value – and there is a reason for it. Usually, due to friction effect resisting acceleration but contributing deceleration, the maximum acceleration value is less than deceleration one. Therefore, for the fastest response of the positioner either preset profiles should be used, or the acceleration/deceleration values should be established experimentally, according to what does your positioner provide. For stepper motors working without feedback there are the values that eliminate the steps loss. For motors with feedback the trapezoidal speed profile should be controlled using XiLab charts. The acceleration/deceleration values should be taken 1.5–2 times less than those resulting to speed profile distortion or to step loss.

Note. Turning the acceleration/deceleration mode off is sometimes useful for multiaxis systems control where, during the motion on multi-dimensional paths, a continuous speed projection on each of the axis is required.

Note. The acceleration value is not displayed in XILab main window.

4.2.5. Backlash compensation

Backlash occurs in any mechanical device, e.g., in reducer or in worm-gear. Backlash results to differences in actual positioner locations while approaching the same point from different directions, whereas the motor shaft is exactly in the required position.

Backlash compensation mode is used in order to eliminate such ambiquity. Its activation allows user to determinate the direction from where the positioner would approach the destination point. Further on, despite the motion, the positioner will approach the stop point from the defined direction only, eliminating the mechanical backlash. If the selected approaching direction doesn't match to intrinsic one, then the controller drives the motor for some user-defined distance beyond the destination point and after that turns the motor around and completes the approach from the required direction.

While a laden mechanical system is moving, its dynamic characteristics in the backlash zone do differ from the regular motion mode. Therefore, the rotation in the backlash zone should be performed with user-defined speed which should be normally set less than the rated speed.

The following parameters of backlash-compensating system are available for adjustment by user:

- Backlash compensation on/off flag.
- Rotation speed while performing the compensating motion.
- Distance ample for backlash compensation. The plus or minus sign for that parameter is used to determine the approach direction. The plus means the approach from the left side whereas the minus means the approach from the right side.

The controllers indicates if the backlash compensation is active using MOVE_STATE_ANTIPLAY flag in the structure of the state which is also displayed in XILab main window.

A forced backlash compensation by using the *LOFT* command may be performed if there is no confidence that the actual position is backlash-free. While carrying out this command, a displacement for backlash compensation distance is performed with subsequent return. Calling this command while driving will lead to a smooth stop the engine. This command makes sense when backlash system on only.

Note. The backlash compensation mode presumes no axis position correction, providing the user with just the choice of the direction from where the positioner should approach the destination point, sticking to this selected direction.

The backlash compensation adjustment using XILab software is described in Settings of kinematics (Stepper motor) section. Switching on and backlash compensation parameters detection commands are described in Programming guide.

4.2.6. Reversal rotation

It is considered that the coordinate increase corresponds to movement to the left, whereas its decrease corresponds to movement to the right. The rotation is to be reversed either if this rule is not satisfied due to physical positioner location, or if the positioner is supplied with an anchor which increase direction doesn't match to coordinate increase.

The reversal rotation may be switched on in the Motor parameters block of XiLab menu. Switching this feature on will change the current coordinate sign; thus, "left" and "right" terms will get interchanged. For example, the first movement during the Home position calibration will perform physically to the opposite direction, the Left and Right commands in XILab main window will interchange, etc.

Warning. Reverse is a setting that affects the whole controller operation if changed. The previously used XiLab scripts or your own controlling programs will work differently.

Particularly, the limit switches are adjusted independently from the reverse. Thus, switching this mode on or off, one must re-adjust them.

4.2.7. Recommendations for accurate rotation

The controller can automatically adjust itself for the required mode, in order to maintain either the speed or the coordinate. However, both the speed and the adjustment property depend on the controller settings. The stepper motor working in steps or microsteps positioning mode can instantly reach the required operating conditions. If the stepper motor is physically unable to provide the required speed or acceleration, the rotation will most likely stop (fail) at all. The movement will not fail if a feedback sensor such as quadrature encoder is used as a reference; however, the controller won't probably provide the rotation with the requirement parameters.

The additional position sensor (encoder) is required for DC motors. The indirect connection of controlling scheme affection with DC motor positioner displacement results to slowing down of reaching the required coordinate or speed. The following recommendations will help you to accelerate this process and to make it more stable:

- The profile corresponding to the positioner being used is normally uploaded to controller and is used by it. Please upload the profile from the Configuration files chapter if you aren't confident that it is proper;
- The motor doesn't fit the limitation mode for one of the operating parameters (refer to Motor limiters and Power control chapters). Such limitations are displayed by the horizontal bar above the Current indicator in either Power, Voltage or Speed blocks in the Motor section of XILab Main window in single-axis control mode. For more information please refer to Motor limiters and Power control chapters.
- The mechanical impediment for rotation, the axis or positioner jamming are lacking;
- The power consumption of power supply unit being used is sufficient (see the Safety instructions while operating the controllers).

4.2.8. PID-algorithm for DC engine control

Algorithm description

DC engine is controlled by the PID regulator, with the coordinate as the controlled parameter. The controlled coordinate changes according to motion settings and incoming commands to provide motion capability. We will call controller coordinate the running position. DC engine winding PWM signal fill factor is the control signal of the regulator.
The control action is calculated according to the following formula:

$U(t) = I + P + D = K_P \cdot E(t) + K_I \int E(t)dt + K_D \frac{dE(t)}{dt}$, where:

$U(t)$ - is the control action

$E(t)$ - is difference between the running coordinate and the current motor coordinate

K_P, K_I, K_D - are proportional, integral and differential coefficients of the regulator. Regulator coefficients are set on PID settings page of the XILab program or programmatically by calling set_pid_settings() function of the libximc library (see Programming guide).

The resulting value is normalized according to the following formula to make PID regulator action independent of motor type, feedback sensor and working voltage:

$DC(t) = \frac{U(t) \cdot U_{nom}}{U_{supp}(t) \cdot IPS}$, where:

$DC(t)$ - is the PWM fill factor

U_{nom} - is the nominal (maximum) motor voltage, (see Motor limiters).

$U_{supp}(t)$ - is the current supply voltage

IPS - is the feedback encoder resolution in counts per revolution

This approach allows to change motor type, feedback sensor and voltage supply unit without reconfiguring PID regulator.

Warning. Do not forget to change PID regulator coefficents accoring to abovementioned formula if you are going to change maximum motor voltage.

Particular properties of the algorithm

PID regulator coefficients

User set settings are normalized to keep optimal PID regulator coefficients in [0..65535] range.
Let's consider the effects different components have for better understanding.
We will assume the supply voltage $U_{supp}(t)$ is constant and equal to the motor nominal voltage U_{nom}. With this assumption PWM fill factor will be equal to 1 in the following cases:

1. $K_P = 1, K_I = 0, K_D = 0$ - if target position is ahead of real position by 256 motor shaft revolutions
2. $K_P = 0, K_I = 1, K_D = 0$ - if integral in the formula above is equal to 52,5 revolutions · second
3. $K_P = 0, K_I = 0, K_D = 1$ - if real motor speed is higher than the required speed by 96000 rpm.

Reaching target position

Target position is considered to be reached when motor shaft reaches the target position. Some oscillations around target position are possible. Motor will need some time to stop and return to correct position if smooth deceleration is not used and an immediate stop command is received or an emergency stop by limit switch has happened.

Warning. Long time oscillations around the target position while the motion is considered finished are possible if the PID regulator is set up incorrectly.

PID regulator tuning recommendations

1. It is recommended to disable all DC motor limits when you start tuning PID regulator
2. It is recommended to tune PID regulator to maintain constant speed first.
3. It is recommended to start tuning PID regulator with integral and differential coefficients set to zero.
4. Set proportional coefficient $K_P \leq 10$
5. Set required movement speed and start the movement.
6. Gradually increase K_P and observe current speed graph. Find optimal K_P which makes the controller maintain current speed the best.
7. It is recommended to tune K_I next. Integral PID regulator coefficient has more influence on position reaching behavior.
8. Set integral coefficient $K_I \leq 10$ and start movement to position. It is convenient to use "shift on offset" command for this. It is advisable to use sufficiently long shifts so that motor reaches at least 20% of required speed.
9. Increase of K_I makes the controller reach target position faster.
10. Achieve fast reaching of target position by gradually increasing K_I. You should observe the process of reaching the target so that no oscillations happen. Is is possible you will need to decrease K_P.
11. It is recommended to check how obtained coefficients work in speed maintenance mode. In case there are significant speed oscillations around the target speed it is recommended to decrease both coefficients until speed is maintained satisfactorily. This can slow the process of reaching the target position but can help eliminate vibration nad noise during movement.
12. After setting K_I and K_P you can set up K_D.
13. It is recommended to do K_D tuning in speed maintenance mode.
14. Gradual increase of this coefficient should lead to decrease in speed oscillations around target speed. Increase K_D to achieve optimal result.
15. Check if target position is still reached in a satisfactory way. If there are oscillations then it is recommended to decrease K_D.

After these steps you can adjust coefficients in a loop $K_P \Rightarrow K_I \Rightarrow K_D$ without returning to their initial values. Several iterations will lead to optimal results of speed maintenance and time to reach target position.

Note. It is not recommended to adjust more than one coefficient at a time while tuning the PID regulator.

IMPORTANT. It is extremely not recommended to start with large values of PID regulator coefficients or to rapidly adjust them. This can lead to self-excitation of speed oscillations and motor failure.

4.3. Main features

1. Supported motor types
2. Motor limiters
3. Limit switches
4. Automatic Home position calibration
5. Operation with encoders
6. Revolution sensor
7. Steps loss detection
8. Power control
9. Critical parameters
10. Saving the parameters in the controller flash memory
11. User defined position units

4.3.1. Supported motor types

Currently the controller supports stepper motors only. The parameters of supported motors are described in Specifications chapter.

Rated current is the main parameter of the stepper motor. The rated current is adjustable at the Settings of kinematics (Stepper motor) section.

IMPORTANT. The motor will gradually get overheated and physically damaged if the overstated current value is set. Make sure that the rated current value is set according to the used pull-up/pull-down. All the settings are proper in default pull-up/pull-down profiles.

Step division mode is another important parameter. The following step division options are available to user:

- 1 (full) step
- 1/2 of the step
- 1/4 of the step
- 1/8 of the step
- 1/16 of the step
- 1/32 of the step
- 1/64 of the step
- 1/128 of the step
- 1/256 of the step

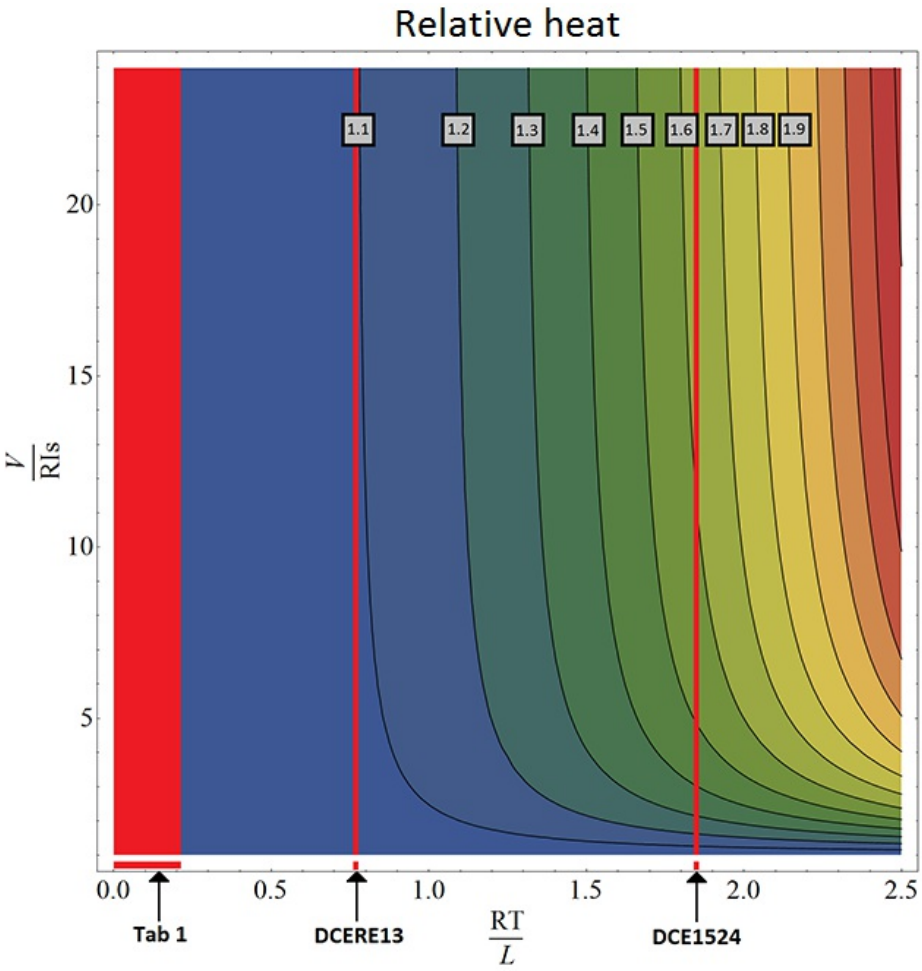
The microstep mode is set either in Settings of kinematics (Stepper motor) folder or by motor adjustment commands. See the Communication protocol specification and the description of the related functions in the Programming guide chapter.

Note. The controller always uses the internal step division value equal to 1/256. If the user selects a coarser step division, the software will display only the multiple of the coarser step division positions and both adjustment and transmission are possible only in that coarser step division mode. This is done for compatibility with both obsolete and actual software operating with small multiples of the step division values. On the other hand, operations with the largest step division value provides the most smooth and silent rotation at the smaller speed values.

The number of steps per revolution is the another direct parameter of the stepper motor. This setting affects the rotation but is used either in slipping control block or while operating with the encoder feedback.

Select engine criteria

Pulse-width modulation (PWM) is widely used way to control winding's current in different motor types. It leads to current oscillations at PWM switching frequency (so-called "current ripple"). Current ripple's amplitude depends on motor characteristics like its winding inductance and resistance. Motor can be warm stronger than it is expected with nominal current due to current ripple, i.e. $\frac{P_{real}}{RI_s^2}>1$. There is RI_s^2 - power dissipated by I_s(stabilization current), P_{real} - real power, dissipated in motor. For overheating estimation we recommend to use this graph:



Tab 1

Motop	RT/L
20	0.19576
28	0.07253
28s	0.07168
4118L1804R	0.02715
4118S1404R	0.02844
4247	0.0273
D42.3	0.0223
5618	0.0146
5618R	0.0146
5918	0.0116
5918B	0.012
VSS42	0.029
VSS43	0.0256
ZSS	0.04248
DCERE25	0.2106

The motor's overheat is determined by this sequence:

- $\frac{RT}{L}$ calculation. There is R,L - resistance and inductance of motor winding (refer to the motor documentation). T - PWM period time. It's value should be 51.2 us for stepper motors and 25.6 us for DC motors.
- $\frac{V}{RI_s}$ calculation. This parameter shows power voltage excess under nominal voltage. There is V - power voltage, R - winding resistance, I_s - stabilization current.
- Overheat definition. After first two steps point can be plotted at the graph. Now we should define the region,

which corresponds to overheat degree. For example, the region between 1.1 and 1.2 corresponds overheat value between $1.1*RI_s^2$ and $1.2*RI_s^2$

There is DCE1524 overheat calculation example:

T = 25.6 us R = 5.1 Ohm L = 70 uH

$\frac{RT}{L}$ = 1.86

Now we can draw vertical line corresponds this value (look at graph) and find out overheat with different power voltages. Let's assume I_s = 500 mA. Nominal voltage in this case is $R*I_s$ = 2.55 B. If power voltage exceeds nominal more than 5 times but less than 10 times DCE1524's overheat will be between 1.5 and 1.6. Motor overheats about 1.65 times with 30 V of power voltage.

All the major engines and their parameters have been marked in Tab. 1.

4.3.2. Motor limiters

Motor limiters of the windings current and voltage as well as limiters of the maximum rotation speed of the motor shaft are designed for safe operations of the motors. If these limitations are activated, they lead to graduate decrease of both the power and the rotation speed to the values that result to decrease of the parameters being limited to predefined values. This function operates with the current and voltage values directly at the motor which is different from critical parameters operating with the current and voltage values at the motor input. The limiters don't result to motor shutdown with switching on the **Alarm** mode but bound the current or voltage increase at the motor windings, which is their another difference from critical parameters.

For DC motors:

- Max voltage defines the maximum power voltage at the motor windings. It is normally used in order to bound the voltage increase during either jam or abnormal displacement operations.
- Max current defines the maximum current value at the motor windings. It is normally used in order to bound the current increase during either jam or abnormal displacement operations.
- Max RPM is the maximum shaft rotation speed. It is normally used in order to bound the rotation speed while operating with reducers or other mechanisms having the strict limitations for the maximum rotation speed.

For stepper motors:

- Max(nominal) Speed is the maximum shaft rotation speed in steps pre second. The actual rotation speed is defined by the Speed parameter (refer to Predefined speed rotation mode).
- Nominal current defines the rated current value at the motor windings. This value cannot be exceeded due to the stepper motor control characteristics.

The limiters settings are described in Settings of kinematics (DC motor) and Settings of kinematics (Stepper motor) sections of XILab software

4.3.3. Limit switches

Limit switches designation

Limit switches are designed in order either to prevent the positioner movement out of permissible physical range of its displacement or to limit its displacement range according to user-defined requirements. Incorrect settings of the limit switches may result to positioner jam if the controller goes beyond the permissible range.

General settings

If the limit switch is active, a corresponding flag is placed in the state structure and the appropriate icon (left or right) is displayed in XILab Main window. The controller can either stop any movement in the direction of any active limit switch (left or right) or stop the movement to the single limit witch (left or right) or not to limit the movement. Limit switches settings are performed in XILab software (see the Motion range and limit switches section).

Programmable motion range limitation

If there is no hardware limit switches for the motion range but the positioner requires such limitation, the programmable limiters are used. For doing that, the limiters should be switched to limitation mode according to position reading (see the Motion range and limit switches section). The left and right margin fields are used (the right margin value should be higher than the left one). In this mode, the left limit switch is active if the actual position is less than the left margin value and the right one is active if the actual position is greater than the right margin value. The operation time is about one millisecond.

Warning. The programmable motion range limitation is reliable only if there is no direct setting of the new position by ZERO or SPOS commands, or if there is no steps loss ore encoder malfunction if it is used for positioning, or if there is no frequent power-cut during the rotation. If any of these problems appears, the programmable range should be re-adjusted. The appropriate reference sensor allows the automatic re-adjustment using the automatic Home position calibration feature.

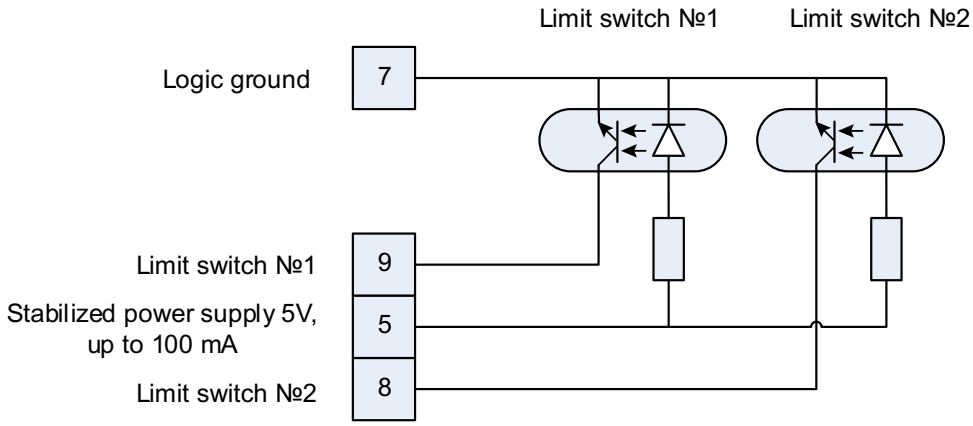
Hardware limit switches

The controller may operate with limit switches based either on dry contacts, or on optocouplers, or on reed switches, or on any other sensor types generating a 5V TTL-standard "logic one" electric signal in one state and a "logic zero" in the other. Each limit witch may be configured independently. There is also possible to change the position of limit switches or their polarity by using the software.

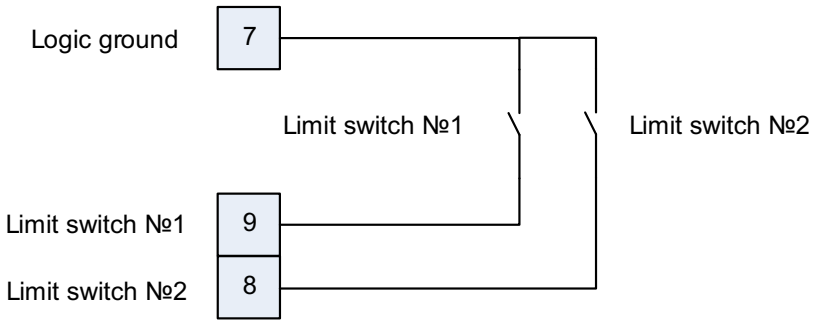
Note. Limit switches are also useful for automatic Zero position calibration.

Limit switches connecting instructions

Limit switches should be connected to DSub connector pins (see the Positioner connector chapter) as it is shown at the diagrams:



The "optocoupler" limit switches connection diagram.



The "dry contact" limit switches connection diagram.

The settings of which limit witch is left or right is required by the controller. Sometimes it is unknown a priori, just it is clear that both limit switches are connected and fire if the corresponding limit of the motion range is reached. The positioner jam is possible if the limit switches are configured improperly. Therefore, the controller supports just a simple detection of incorrectly configured limit switches, shutting down the movement on both of them. Please make sure that their polarity is configured correctly and the shutdown mode is activated on both of limit switches. The flag of improper limit switch connection detection should be turned on in corresponding XiLab software menu. Start the movement to any direction until the limit switch shuts the movement down. If there was right-side movement but the left limit switch became active, or vice versa, the limit switches should be interchanged (see the Motion range and limit switches chapter). If the improper actuation of the limit switch is detected and if the corresponding feature is set in the Critical parameters menu, the controller can turn the Alarm mode on.

Warning. The protection against mistaken limit switches connection doesn't guarantee the complete solution of the problem, it only makes the initial configuration procedure easier. Particularly, don't start the movement if any of the limit switches is active, even if the protection is on.

4.3.4. Automatic Home position calibration

This feature is used for detection and placing the movement to the starting position. This option of the controller is designed for simple search of "Home", or "Zero" position by user himself, with no need of any programming skills. The Home position is set relative to one or two external sensors and/or to an external signal.

Automatic Home position calibration feature is configured by user in *Device configuration->Home position* folder of XILab software (see the Home position settings chapter) and is activated by the button in XiLab main window.

After this feature is activated, the controller rotates the motor to the defined direction with the defined speed. The calibration speed is usually set lower than the normal rotation speed in order to improve the calibration accuracy. There are three ways to determine the completion of the motion, according to user-defined preferences:

- **rotation until the limit switch is reached**, the actual configuration of limit switches is used (e.g., polarity, location). For more details please refer to Motion range and limit switches chapter.
- **rotation until a signal from the synchronization input is received**, the actual configuration of synchronization input is used. For more details please refer to Synchronization settings chapter. If the synchronization input is switched off by program, the signal from it will never be processed.
- **rotation until a signal from the revolution sensor is received**, the actual configuration of revolution sensor is used. For more details please refer to Position control chapter.

Note. If the completion of the motion should happen at the limit switch, the calibration program is not crashed.

Fine calibration

After the first displacement the controller position is defined. However, some additional rotation may probably get turned on before the movement to Home position is performed. It provides fine Home position calibration that may reach 1/256 of the step for some positioners. If the corresponding flag is switched on, the controller rotates the motor to user-defined direction with defined rotation speed until a signal either from the synchronization input or from the revolution sensor is received or until the limit switch is reached, according to user-defined preferences. Using the signal from the sensor on the shaft from motor to reducer and performing the motion with the small speed does make sense. It provides the maximum accuracy. In case if the signals about the completion of the first and second motion contemporize, a flag of the start of tracking of the signal about the completion of the second motion is provided. This allows avoiding the signals of the completion of the first and second motions to be in ambiguous sequence. As a result of optional second motion, the calibrating position becomes more accurate.

Note. If the second phase of the motion is used, the first motion usually can be performed at the high speed since it provides just a coarse calibration and the accuracy is not required there. The accuracy will not improve if the second limit switch is used for the second phase of the motion since its physical characteristics are the same as for the first limit switch.

Finally the controller moves the motor to the defined direction with the defined speed at the defined Standoff distance. This is what required to reach the Home position.

Note. The position reached as a result of calibration will slightly depend on the speed of the last motion until the selected sensor responsed. Therefore, don't change the speed parameters for further successful reaching the same position.

The automatic Home position calibration **features are described** in the Programming guide chapter.
The parameters configuration commands are described in the Communication protocol specification chapter
The position calibration command is described in the Communication protocol specification chapter.

A set_zero script is supplied with XiLab software pack, providing the automatic Home position configuration. This script changes the Standoff setting in Home position folder, making the actual position as the Home one.

How to use the script:

- place the movement to the desired position

- launch the script and wait until it's finished.

As a result, the movement will be in the same position and all the following calls of homing function will move it there. Make sure to save the settings to controller's nonvolatile memory.

4.3.5. Operation with encoders

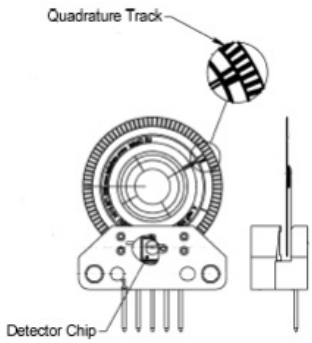
Application of encoders

Encoders are designed for creation of accurate and fast feedback according to the coordinate for all the electric motor types.The feedback is performed by the motor shaft position, by positioner's linear position, by the motorized table rotation angle or by any other parameter related to the shaft position and measured by using the two-channel quadrature encoder complying the requirements described in Specifications chapter for the appropriate controller type.

What is quadrature encoder?

Encoder is a mechanical motion sensor. The quadrature encoder is designed for direct detection of the shaft position. The sensor transmits the relative shaft position by using two electric signals at CH A and CH B channels shifted relative to each other at 1/4 of period.

The signals at CH A and CH B outputs of quadrature encoder.



An optical quadrature encoder mechanics.

An optical quadrature encoder mechanics is shown at the figure above. There are two optocouplers used. The operational principle of an optocoupler is as following: a LED and a detector are arranged opposite to each other from different sides of a disc. The optocoupler opens when disc's "window" coincides with the detector (the outgoing signal is logic zero). The outgoing signal is logic one if the detector is closed by opaque part of the disc.

Number of steps per revolution (CPR) is the main parameter of the quadrature encoder. The standard resolution values for encoder are from 24 to 1024 CPR. Each period of signal alteration is interpreted by 1, 2 or 4 codes which is corresponding to X1, X2 and X4 operating modes. This controller uses the most accurate X4 mode. The maximum frequency of each encoder's signal depends on the applied encoder itself, since for 200 kHz in X4 mode the controller can read up to 800,000 encoder counts per second.

Controller's features

There are two operating modes with encoder available for the controller:

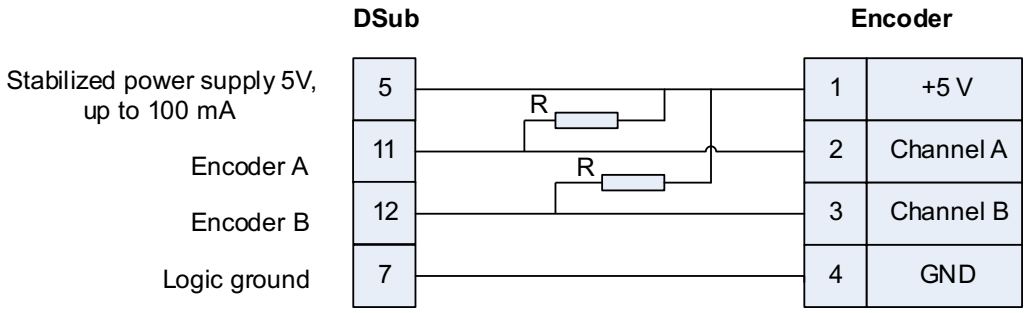
- the encoder is used as the main position sensor (this is the main operating mode for DC motors and the optional one for stepper motors).
- slippage, backlash or steps loss detection (the most often used mode for joint operations with stepper motors, see Steps loss detection).

Driving encoder mode

This is the mode when all the coordinates and positions of the motor are measured directly by the encoder and are denominated in encoder counts. Speeds are denominated in RPM (revolutions per minute) for DC motors and in count/s for stepper motors. The DC motor speed is calculated by the controller basing on the speed alteration data as well as on the number of encoder pulses per one complete revolution of the motor shaft that are displayed in feedback configuration block at the Settings of kinematics (DC motor) folder. Note that in this case the speed maintaining mode, themode of movement to the predefined point as well as all their derivations use PID control algorithms and the appropriate settings are required. The driving encoder mode provides the fastest movement available for stepper motors with no risk of steps loss when the coordinate flounders and the recurrent calibration is required.

Encoder connection

The encoder is connected to the controller via DSub connector (see the Appearance and connectors chapter).



The diagram of encoder connection using DSub connector.

See also the Positioner standard connection diagrams chapter.

Warning. Encoder inputs of the controller internally pulled up to logic one by using the 5.1kΩ resistors. Frequently encoder outputs are of "open collector" type equipped with internal pull-up resistor. During the data transmission they provide good characteristics while passing from higher logic level to lower. However, the pass from logic 0 to logic 1 is more graduate. It passes through the RC circuit formed by pull-up resistor and cable capacitance. This is the most important thing if the cable is long. If the internal pull-up is not sufficient, the pull-up resistor with r=1.5kΩ may be added for every +5V to each output in order to improve the transmission speed parameters; before doing that please check if the open collector of the encoder can transmit 5mA current. The resistors insertion diagram is shown above. The maximum operating speed for quadrature encoder may be increased by adding a push-pull driver with the outgoing current over 10mA to its output, providing quick 0 - 1 and 1 - 0 transmission edges.

4.3.6. Revolution sensor

Revolution sensor is designed for stepper motor shutdown (failure) detection and for better accuracy of Home position calibration procedure (see Automatic Home position calibration).

The controller may receive the actual position data from the external revolution sensor mounted on the stepper motor shaft. The sensor transmits its signals to the controller once or many times per one revolution of the motor.

Usually the revolution sensor is a small disc with precise graduation scale mounted on the motor shaft. A light source (LED) and a sensor of the optocoupler are placed at the opposite sides of the disc. The sensor is open if there is no interrupter between the LED and the sensor (the logic zero is transmitted to optocoupler's output), whereas the logic one is transmitted if the light source is closed by the interrupter.

By default, the lower logic level is interpreted by the controller as the active mode of the revolution sensor. The controller's input is pulled to logic one level, thus, the disconnected revolution sensor means its inactive mode. The controller's input can be inverted if necessary, in that case the logic one level will mean the active mode.

The revolution sensor should be connected to the controller via 15pin DSub connector (see the Positioner connector chapter).

4.3.7. Steps loss detection

This mode is generally used while operating the stepper motor at full speed or limit loads when the shaft jam resulting to the steps loss is possible. In this case an additional position sensor (revolution sensor) or encoder allows tracking this moment, informing the user about it. This feature should be applied **with stepper motors only** and it allows detection of the steps loss. Steps and microsteps are the measurement units for all coordinates and shaft positions.

When the encoder is used, the controller stores both number of steps and number of encoder's counts per revolution (see the Settings of kinematics folder of XiLab program). When the feature is activated, the controller saves the current position in steps of the motor and the current position according to the encoder data. Then, during the motion, the position data according to the encoder converts to steps and if the difference exceeds the predefined value then the slippage is indicated and the Alarm mode turns on (if the related option is active). For more information regarding use of encoder as the steps loss detector please refer to Operation with encoders chapter.

If the revolution sensor is used, the position is controlled according to it. The controller stores the current position in steps according to active and inactive edges at the sensor;s input. Then, at every revolution (number of steps per one full revolution is set by *Steps per turn* parameter, see the Settings of kinematics (Stepper motor) chapter) the controller checks if the shaft has been displaced and how many steps for. If the mismatch exceeds the predefined *Minimal error* value (which is defined in position control settings, see the Position control chapter), the slippage is indicated by the state structure flag. If the appropriate flag is set and if the error is detected, the controller turns the Alarm mode on and the motor shuts down, otherwise the motion is continued. If the slippage indication flag is active, the controller turns the Alarm mode on when the appropriate parameter in the settings is active.

Note. A coarse STOP launches the the re-calibration process of the revolution sensor position, and the calibration starts after the revolution sensor activates during the motion controlled by the motor. It means that the slippage won't be detected if the shaft has been rotated manually right after the coarse stop since the calibration hasn't been performed yet.

Note. If the motor revolution sensor is bouncing (mechanically), the misoperations of the revolution sensor are possible at the very low speeds.

Note. The position control of the revolution sensor can't detect the shaft rotation at the zero speed, i.e., if the motor is shut down and the shaft is rotated manually, it won't be detected.

4.3.8. Power control

Reduction of current consumption

In standby mode the controller provides to specify the level of current consumption less than the rated value in order to reduce the power consumption by the stepper motor. This mode is active by default. It is applied everywhere in order to reduce the heating of stepper motors in "hold" mode with the stability of the precision being in the predefined position. A hold current level is specified in percents from the rated current value in the windings. The time in milliseconds after which the current should be reduced is also specified. The current reduction option can get switched off using the special flag. For setting the current reduction function refer either to set_power_settings function (see the Programming guide chapter) or to XiLab settings folder (see the Power consumption settings chapter). The rated current of the stepper motor is set with set_engine_settings command (see the Programming guide or Settings of kinematics (Stepper motor) chapters).

A sensible level of the hold current level is 40% to 70%. Such level reduces the power consumption from half to quarter, whereas the holding force usually remains sufficient. The sensible time interval after which the current is reduced is 50 to 500 ms. This is the time when the low-frequency mechanical vibrations of the system that sometimes can bring the hold position down are normally over.

The motor power shutdown

For power consumption reduction there is also a timed power shutdown mode. Basically this mode is used in order to prevent the power consumption for holding the position when the operations with the unit are finished an there is no motion within the long time. It is activated by default but users can switch it off. The sensible time interval is 3.600 s (1 hour). For setting this function please refer to set_power_settings function (see the Programming guide chapter) or to XiLab settings folder (see the Power consumption settings chapter).

Time delay calculation specifics

All time delays work as follows: the time rounded to a millisecond is stored at every motor shutdown. Then, after the user-defined timeouts are elapsed and if the PowerOff/CurrentReduce functions are on, either power shutdown or windings current reduction occurs. All settings are adjustable online. For example, if the PowerOff timeout is increased after it has already happened, the windings will get powered again and the PowerOff function will get activated after the timeout since the motor has been shut down is reached. It also concerns switching on and off the flags of PowerOff/CurrentReduce modes. The timeout countdown stops and PowerOff/CurrentReduce functions are cancelled if any movement starts.

Jerk free function

Sometimes the windings current should be gradually changed every time in order to eliminate the mechanical vibrations of the system. A Jerk free option of the controller is available for it, providing to specify the rate of increase of the windings current to the nominal value, accurate within a millisecond. An appropriate flag is used to turn this option on. After that, all the changes of stabilization current will be performed with graduate increase or decrease of the hold current. For example, if the rate of the current increase is set equal to 100ms and the hold rate decrease down to 50% occurs, the current will get decreased gradually within 50ms, not within 100ms, since 100 ms is the time required in order to decrease the current completely down to zero. Also, after the new movement tarts, the current will increase gradually up to the rated value within 50ms. For the configuration of the Jerk free option please refer to set_power_settings function (see the Programming guide chapter) or to XiLab settings folder (see the Power consumption settings chapter). The stepper motor rated current configuration is performed by set_engine_settings command (see the Programming guide or Settings of kinematics (Stepper motor) chapters).

The graduate current increase function works at any change of the current in the windings, i.e., if the nominal hold current is changing. The increase/decrease rate of the current is calculated basing on the maximum of the hold currents being specified, either the old or the new one. The current is decreasing down to zero if the windings should be unpowered and only after that the output power circuits of the controller are getting de-energized. If the windings should be powered, the zero current is used for it and then it increases up to the rated value.

There are some exceptions when the current is dropped down to zero instantly and the windings get unpowered even if the Jerk free function is on. It happens in case of emergency or if the Alarm mode is activated (see the Critical parameters chapter), as well as during the controller's reboot while the software is updated. All these events are pretty rare and should not occur during the operations with positioner.

The 500-200ms time interval is sensible for Jerk free function, since it will result to low-energy vibrations at 3 to 10Hz frequency only which are much less than the vibrations caused by the common sources (e.g., steps, draughts, etc.). Setting the higher Jerk free time value with switching on the current decrease function in order to avoid the motor overheating and for power saving will result to continued delays for the current decrease and increase, therefore the Jerk free time value should not be too high.

4.3.9. Critical parameters

The minimum values of currents, voltages and temperatures are to be specified for controller's and motor's safe operations. Running out of the valid range for any of these parameters results to the motion shutdown when the motor windings get unpowered and the controller turns the Alarm mode on. Switching the Alarm mode off is possible only if the reason of running out of bounds for the critical parameter is eliminated and the "STOP" command is sent. This configuration is used for all types of motors.

The following parameters are available:

- Low voltage off defines the minimum voltage value of the controller power supply (measured in tens of mVs). The Low voltage protection flag is used to turn this option on, otherwise the minimum unpowering threshold doesn't work. The 6000mV to 8000mV, range is sensible for operating power range of 12V to 36V. This type of protection helps to determine the power-cut moment due to activation of any sort of power supply unit protection. This may occur if the operating power consumption of the stabilized power supply unit is exceeded.
- Max current (power) defines the maximum current of the controller power supply (measured in mAs). The sensible value is twice as higher as the maximum operating consumed current registered during the tests. Use the XiLab charts for registration of the consumed current.
- Max voltage (power) defines the maximum voltage value of the controller power supply (measured in tens of mVs). The sensible value is 20% over the limitation equal to the operating power supply unit voltage.
- Max current (usb) defines the maximum current of the controller power supply via the USB cable (measured in mAs). The controller demands the current received via USB interface for its own needs (150mA) and for supplying the power to the devices connected to the motor's connector (200mA). This limitation is sensible within the range of 250mA to 450mA, depending on external devices powered by the controller. The USB interface is not designed for the current of over 500mA, although it is often able to stand the 1000mA to 1500mA current.
- Max voltage (usb) defines the maximum voltage value of the controller power supply via the USB cable (measured in tens of mVs). The controller may get damaged if the USB power supply voltage exceeds 5.5V value. Thus, 5.2V is the sensible limitation.
- Min voltage (usb) defines the minimum voltage value of the controller power supply via the USB cable (measured in tens of mVs). The controller doesn't work if the power voltage supplied via the USB interface is less than 3.6V. The voltage should be measured at the controller;s input. The input and output voltages while there is the consumed current may defective. At least 4.0V voltage is required if the consumed current is heightened. The 3.8V limitation of the minimum voltage is sensible.
- Temperature defines the maximum temperature of the microprocessor (measured in tenth parts of a Celsius degree). The microprocessor can operate at the working temperature of up to 75ºC and doesn't overheat itself. Rise of its temperature indirectly indicates the overheating of the power part of the board. The overheating threshold range from 40ºC to 75ºC is sensible.

Flags:

- ALARM_ON_DRIVER_OVERHEATING means turning the Alarm mode on if the driver's critical temperature (over 125ºC) is exceeded. The power driver indicates if its temperature is approaching the critical value. If the driver is still working then the further heating will automatically shut it down. Forced shutdown isn't recommended whereas the unconstrained shutdown flag is recommended to be set on.
- H_BRIDGE_ALERT means turning the Alarm mode on if any fault of the power driver due to board overheating or damage is detected. This flag should be set on.
- ALARM_ON_BORDERS_SWAP_MISSET means turning the Alarm mode on if the triggering of the wrong limit switch, not corresponding to direction, is detected (see the Limit switches chapter). This flag is intended for clear indication of the response of the subsystem for muddled limit switches detection. The flag is recommended to be set on.
- ALARM_FLAGS_STICKING flag activates the sticking of the error indicators in the status structure of the controller, otherwise indicators are active only during the accident that caused the error. If there was a short-time error and its cause had been solved independently, then sometimes the accident resulted to Alarm mode remains uncertain. In that case the sticking is useful and the accident cause can get diagnosed in XiLab main window.
- USB_BREAK_RECONNECT - This flag configures the operation of an USB break reconnect block. When set, this unit starts to operate and monitor the loss of communication over the USB bus (for example, in case of a shock with static electricity).

Configuration of parameters is described in Critical board ratings menu of XiLab software. The maximum available values configuration commands are described in Programming guide.

4.3.10. Saving the parameters in the controller flash memory

The controller provides saving all its parameters in the non-volatile memory. The configuration is restored when the controller is powered on, after that the controller itself is instantly ready for operation. The positioner requires no new adjustment every time the power is on. The controller stores its user-defined name which is useful for its further identification.

The non-volatile memory stores all the actual operating parameters of the controller related to Device configuration folder of XiLab settings menu. Either **Save to flash** button of XiLab program or command_save_settings function are used for it (see the Programming guide chapter).

All the configuration parameters can get restored to controller's RAM from the non-volatile memory, not only when the power is turned on but also by clicking the **Restore from flash** button of XiLab program which provides the access to the data saved in the flash memory. The command_read_settings function can be used for the data uploading, see the Programming guide chapter. The restored settings become active instantly and all the modules of the controller will get re-initialized.

4.3.11. User defined position units

Controller position is set and read in stepper motor steps or encoder counts, if encoder is available and enabled. Is it convenient to set position in mm (in case of translation stages), in degrees (in case of rotator stages) or any other natural units. Controller software can translate coordinates to user-defined units: user can set a ratio, where a certain amount of controller steps is equal to the certain amount of user-defined units. This enables one to issue movement commands and read controller position in these user units. Speed and acceleration are also set in units derived from user-defined ones (for example mm/s).

You can enable user-defined units in XiLab on page User units settings.

Libximc library calibrated functions have a _calb suffix, for more information see Programming guide.

4.4. Safe operation

Several controller settings are directly connected with safe operation. Settings misadjustment may lead to controller or positioning element damage. Positioning element can be damaged by exceeded capacity, rotation speed, or in case of propel range bounds violation. In most cases it is enough to upload a made-up positioning element profile for safe operation, where all necessary settings are already made.

Propel range bounds and pin switches

inear positioning element has limited propel range bounds as compared to circular rotator. Violation of physical propel range bounds of positioning element is the main reason for positioning element seizure and damage. To prevent such kinds of breakages the propel range of positioning element is limited by user requirements. For this reason Limit switches are used, but in some cases when for example positioning element is not equipped by pin switches or has only one pin switch propel range bounds can be defined with a help of a program. (see Limit switches). Commonly, Limit switches are reversed. In this case use the mechanism of reversed limit switches detection which is described in Limit switches section because otherwise the first motion to the bound will lead to positioning element seizure. Motion range and limit switches is described in corresponding section. Settings commands are described in Programming guide.

Propel range limiters

Repeater motor has basic safety setting- nominal current in coils. This is basic parameter which defines capacity delivered to the engine. The nominal current should be set on a level which doesn't exceed the limits of given engine. For more detailed information see chapter Limiters on Engines. For DC engines nominal current is limited and should be set according to the maximum permissible current through DC engine. If maximum current is not known, then maximal voltage delivered to the engine may be limited. This also will prevent engine overheat although voltage limiting is more coarse way than current limiting. For more detailed information see chapter Motor limiters.
Position element can be damaged or its wear-out can be increased by drive-up. It is necessary to fix speed limitation flag no more than maximum speed and to set right maximum speed for the given positioning element. For more detailed information see chapter Motor limiters.

Critical Parameters

Controller tracks voltages and currents which appear in its circuits and can react on their suspicious values. This reaction blocks the engine and prevents any further movement until the source of the problem will be eliminated. Due to this it is possible to track engine fault to engine or to earth which may happen because of positioning element cable damage or damage of positioning element itself. This reaction also has informational character because it allows to track incorrect values of source voltage or oncoming overheating. That's why you should read Critical Parameters chapter and set necessary protection. In case of dangerous situation controller will go to Alarm mode and the main window of Xilab program will be colored in red. If this happened track and eliminate the source of danger before shutting of Alarm mode. If you are using your own application for engine control you should pay close attention to Alarm status flag. (See Controller status)

Operation with Encoder

If during encoder connection sensor channels are mixed up, then during engine motion in reversed direction encoder will show decreasing functions. During operation with DC engine this situation will appear if engine control channels (DC+ и DC-)are mixed up. To fix these errors just fix Encoder Reverse flag which is shown in reverse connection section in Settings of kinematics (stepper motor) appendix for Stepper Motor and Settings of kinematics (DC motor) appendix for DC engines.

It is also possible that there is no contact with one of encoders. In this case values of sensor will be changing in [-1..1] range from the starting position during engine motions.

During operation with DC engine both these errors will lead to misfunction in control algorithm, which is described in PID-algorithm for DC engine control. If you connected new DC engine for the first time it is strongly recommended to check encoder connection before starting the operation. To do this you should set corresponding regulation factor values $K_P = 1, K_I = 0, K_D = 0$ and try to make motion to the right or to the left at a low speed. After the motion please check if encoder values are changing in correspondence with chosen directions. Fix Encoder Reverse flag if it is needed.

4.5. Additional features

1. Operating modes indication
2. Operations with magnetic brake
3. Joystick control
4. Left-Right buttons control
5. TTL synchronization
6. Design of multi-axis system
7. General purpose digital input-output
8. Operations with potentiometer feedback
9. External driver control interface
10. Serial port
11. Saving the position in controller's FRAM memory
12. The Standa positioners detection

4.5.1. Operating modes indication

Mode annunciation

Controller status

Mode annunciation is provided in controller. For this purpose one bi-color LED is located on the plate.
Green Power indicator shows presence of 3.3 V power supply of controller.
Red Status indicator represents controller operating mode.
Simultaneous glowing of both lights looks like **yellow glowing**

Flicker frequency Hz	Description		
Lights don't glow	the device is shut down, there is no power supply	LED is green	the device is broken or the microprogram is not downloaded
LED is yellow	the device is in Alarm		
0,25	the device is operating but there is no connection with USB from PC		
1	the device is operating, the movement is stopped		
4	the device is operating, with movement		
8	the device is in re-flashing mode		
10	the device is in USB bus reconnecting mode		

_Indicator operation modes Power/Status

Indicator is duplicated by outputs on multi functional 20 pin BPC connector

Indication of outputs in BPC connector (see Backplane connector)

Note In case of additional LEDs usage on BPC connector they should be designed for 4 mA operational current. There is no need in additional resistors which are limiting the current. Operational current for typical LEDs is about 2 mA. It is not recommended to use blue and violet LEDs because of their high cut off voltage and as a consequence low brightness.

Indication of pin switches

Controller on multi functional 20 pin BPC connector is provided with pin switches operation indication. High logic level appears on the corresponding output in the moment of pin switch activity. Active condition is indicated relying on pin switches settings (see Motion range and limit switches)

_Indication of outputs in BPC connector (see Backplane connector)

Note In case of additional LEDs usage on BPC connector they should be designed for 4 mA operational current. There is no need in additional resistors which are limiting current. Operational current for typical LEDs is about 2 mA. It is not recommended to use blue and violet LEDs because of their high cut off voltage and as a consequence low brightness.

4.5.2. Operations with magnetic brake

4.5.2. Operation with magnetic brake

There is an output on BPC connector for magnetic brake control, magnetic brake is located on repeated motor action. Magnetic brake is used hold down of engine position in case of power fail.

Description of operation

Magnetic brake consists of a magnet and a coil spring, which performs stop engine axis shut down. In case of power fail in a magnet coil spring clamps axis in current place which allows to preserve necessary engine position. After magnet voltage supply the coil spring release the axis.

Controller operating sequence during adjustment movement shutdown.

Engine shutdown (the time of shutdown is recorded in controller) -> magnet power supply cut off -> axle fixation -> plate power supply cut off
During adjustment movement actuation operating sequence is reversed.

As any movement is inert for magnetic brake and position fixation control the following parameters are used:

- Time between engine power supply actuation and brake shutdown (ms)
- Time between brake shutdown and readiness for movement (ms)
- Time between engine shutdown and brake actuation (ms)
- Time between brake actuation and power supply cut off (ms)

During magnetic brake function shutdown controller is constantly giving a signal of engine release. This allows to move engine, equipped by magnetic brake without usage of rotor fixation during pauses. During deactivation of coils deenergizing function controller performs only holdbacks between brake switching and movement actuation\ shutdown.

All magnetic brake settings can be changed on line and brake will be switched to the mode which would be active in case if the setting would have a new value. For example critical holdbacks of braking action increase when brake is already activated will lead to brake's intermission and as soon as it will get to the new hold back from the moment of intermission it will snap into action again. It is also possible to actuate or to shutdown magnetic brake itself or coils power function.

Identification of outputs in BPC connector (see Backplane connector)

Type	TTL
Active condition (brake is released)	0.3 V
Passive condition (brake is not powered)	0 V
Operational current	no more than 4 mA

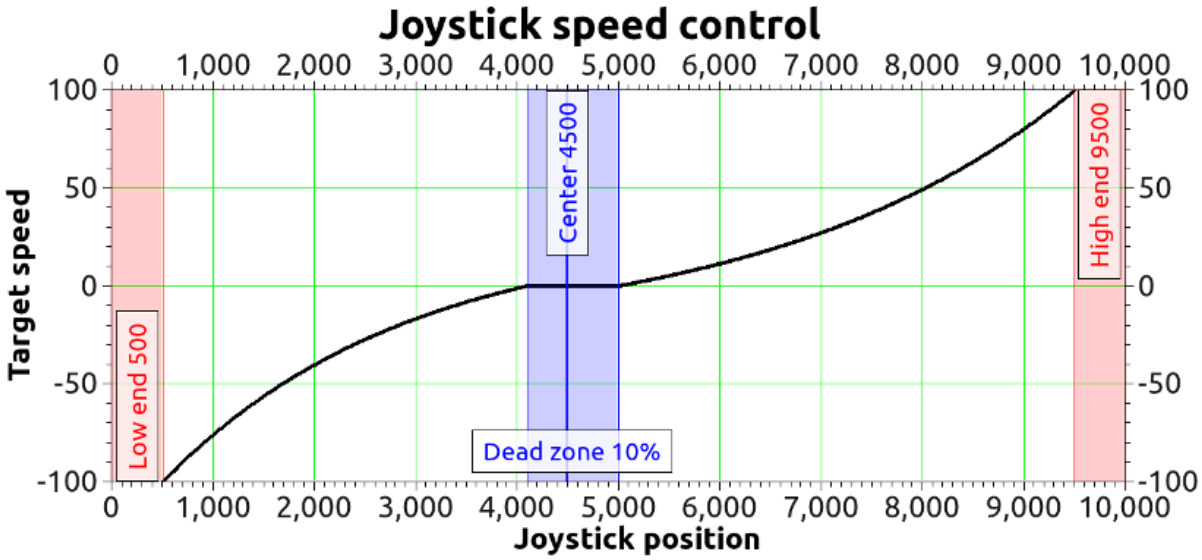
Output electric parameters
Magnetic brake setting in XIlab program is described "Brake settings" section.

4.5.3. Joystick control

Controller allows operating with joystick which has analog voltage in 0-3 V range. In this case voltage in fixed (central) position and also maximal and minimal variations voltage can be set in any operational voltage range in case of observing one condition. Minimum variation < central position < maximum variation. Controller is using voltage numeric representation on joystick output: 0 V is considered zero in controller and 3 V voltage – 10 000.

In order to make it possible for movement to stop in central position *Dead zone* is provided. It is counter able in percents starting from central position. Controller is making stop of motion within *Dead zone*. In case of joystick deviation from central position which is bring its out from *Dead zone* motion with speed which is limited by joystick deviation from *Dead zone* border begins up to the maximum deviation. Connection of joystick movement directions and its deviation can be controlled with a help of reverse flag which can be convenient for the consequence. "Right deviation" – "right motion" without taking into account physical orientation of joystick and rotating part. Speed of motion exponentially depends of joystick deviation. This allows to achieve high accuracy of position setting by small deviations. And achieve quick movements by strong joystick deviations. Parameter of nonlinearity can be changed. During nonlinearity parameter setting at zero motor motion speed starts to depend from joystick deviation linearly.
An example of motion speed dependence from joystick deviation for the following settings can be seen on graph:

Central deviation	4500
Minimal deviation	500
Maximal deviation	9500
Dead zone *Dead zone*	10%
Maximal deviation speed	100



An example of motion speed dependence from joystick deviation.

Sometimes exponential joystick response which consists of high accuracy and speed can be not enough. That's why controller supports maximal seeds table which can be switched by "right"/"left" control keys. In this case during right key pressing the speed is changed at 100% of corresponding deviation. From *Max speed*[i] to *Max speed*[i+1]. In case of "left" control key pressing *Maximum speed* is changed from *Max speed*[i+1] to *Max speed*[i]. During controller start I = 0
The quantity of speeds in table – 10. *Max speed*[x] is equal to zero of integer and fraction parts you can't switch to *Max speed*[x-1]. This is made to limit the table by smaller quantity of speeds. An attempt to get from table index bounds (0-9) will also lead to nothing.

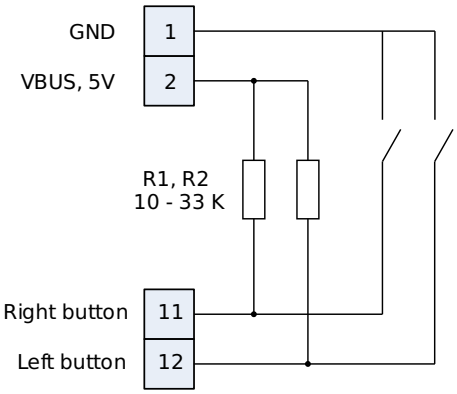
Controller reduces contact bounce on control buttons. For keys operation pressing duration should be more than 3 ms.

If joystick is within dead zone for more than 5 seconds it wont get out of *Dead zone* until it will be out of it more than 100 ms.
This allows user to release joystick and to be confident that even occasional noise on joystick output won't lead to unnecessary motor motions. While joystick is within *Dead zone* it can receive any commands from computer including motions commands and home position calibration commands, etc. If during command performance joystick is brought of *Dead zone* the motion command is canceled and motor is switched to joystick control.
This allows to start motor control mode with a help of joystick and not to use it without necessity at the same time. In case of contact with joystick it intercepts control.

Anything which is considered to be influence of control commands can be used in joystick mode: speed up congruence, maximal speed limitation, coils cut off modes, operation with magnet brake, backlash removal. For example if you move joystick handle within *Dead zone* roughly then controller will slow down motor smoothly during start of corresponding modes it will move to the side for backlash removal, it will stop motor and fix motor shaft by magnetic brake, it will reduce the current smoothly and cut off coils power.
"Left"\"right" key connection is described in Left-right buttons control.
Max speed [i], *Dead Zone* parameter changes is described in Settings of external control devices.

4.5.4. Left-Right buttons control

"Right" and "Left" control buttons can be connected to controller via BPC connector for motor motion control (see Backplane connector). Active button state is set with a help of a program and can be logical nil or one. There are no integrated pull-down resistors for identification of input potential by default, that's why it is necessary to use (pull up) and (pull down) resistors scheme.



Scheme of buttons connection to BPC connector

Controller supports a 10-item speed array MaxSpeed [0-9], which is used both for joystick and button control. If a left or right button is clicked then motor does a shift on an offset, specified by DeltaPosition and uDeltaPosition. If a left or right button is pressed and held for longer than MaxClickTime, then motor starts moving with MaxSpeed [0] and counting down to Timeout [0]. After Timeout [i] microseconds have elapsed speed is changed from MaxSpeed [i] to MaxSpeed [i+1] for any i between 0 and 9 (inclusive).

Note: You can fill only the upper part of the 10-item speed array if you don't need all of them. Controller changes its speed to the next one only if the target speed is not zero and the timeout is not zero. For example, if MaxSpeed [0] and MaxSpeed [1] are nonzero and MaxSpeed [2] is zero (both step and microstep part), then the controller will start moving with MaxSpeed [0], then change its speed to MaxSpeed [1] after Timeout [0] and will keep moving with MaxSpeed [1] until the button is released. You can also set Timeout [1] to zero and leave MaxSpeed [2] set to any value to achieve the same result.
Controller uses movement settings (with the exception of target speed). For example, when changing its speed from MaxSpeed[i] to MaxSpeed[i+i] controller will either accelerate with set acceleration value or change its speed instantly if acceleration is disabled.

Controller uses button contact debouncing. The button is considered pressed if active state lasts for longer than 3 ms.

Type	TTL
Logic zero level	0 V
Logic one level	3.3 V

Output parameters

4.5.5. TTL synchronization

Principles of operation

Synchronization of motions performed by controller with external devices and\or events TTL synchronization is provided. For example controller can produce synchronization impulse which starts some kind of measurement during movement for the given distance anytime. And vice versa receiving synchronization impulse from external device, which for example means that experimental set up is ready for movement to the next measurement position, controller can make a displacement for the previously given distance.

The protection for contact bouncing is provided for operation with input synchronization signals which are generated with a help of mechanical contacts. It is possible to set minimum input impulse time after which synchronization signal is considered to be received. The logical one state is considered to be active and triggering edge is positive-going. If this input synchronization operational logic is not effective it can be inverted.

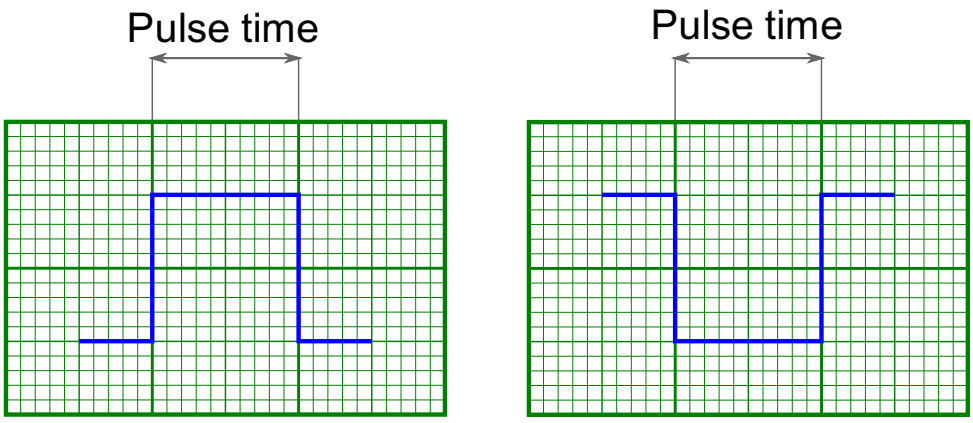


Illustration of output\input impulse inverting

Note.
For simultaneity start of multi-axis systems minimum duration of input signal should be the same for all controllers. Mechanical contacts debouncing in system should not be used in places were such effect is not present, but there are short inductions on synchronization line input. It enough to add RC circuit which filters such phantom input impulses instead of this.

Synchronization is important during multi-axis systems creation because it allows several axes to start motions at the same time. All axes are prepared to motion start in all drive axes start motion mode is set according to input synchronization impulse, one lead axes is set for synchronization impulse sending during start of motion. Lead axes synchronization output is connected with lead axes inputs. Motion of leading axes induce immediate response and start all drive axes motion after such kind of setting and connection.

Note. It is necessary to set minimum duration of input calls synchronization to zero in case of such connection. This shuts down input signal debouncing protection but there are no mechanical contacts in describe configuration and no contact bouncing correspondingly. If minimum input signal duration is zero then to avoid nonsimultaneous motion start of leading and drive axes it is necessary to set similar duration for all controllers and to connect synchronization output not only with drive controllers inputs but also with lead controller input and after this send initiating impulse by manual switching of synchronization output state.

Synchronization input and output are completely independent from each other and other methods of motion control. So motion control via XILab (see XILab Main window in single-axis control mode) or joystick control via user program (see Joystick control), via manual control buttons (see Left-right buttons control, is made independently from input|output synchronization state. The principle "later command has a priority" is always valid this means that for example motion command which is given via XILab will cancel anything which is done via impulse of input synchronization motion, but it own influence on input synchronization performance, and coming of synchronization impulse in case of corresponding setting will cancel the current motion initiated by user program, it will change it to the motion defined by synchronization input performance settings.

Note. Synchronization settings may be saved by non-volatile controller memory in this case everything which is connecting with synchronization performance will be part of controller autonomous performance. I.e you can for example set a displacement on the given distance in case of synchronization impulse coming with synchronization impulse outputting as a displacement result. To connect controller to measuring unit which starts measurement by input synchronization signal and which gives synchronization impulse at the end of measurement and to initialize such measurement system without PC. After first impulse receiving all displacements and measurements will be made automatically without PC.

Connection

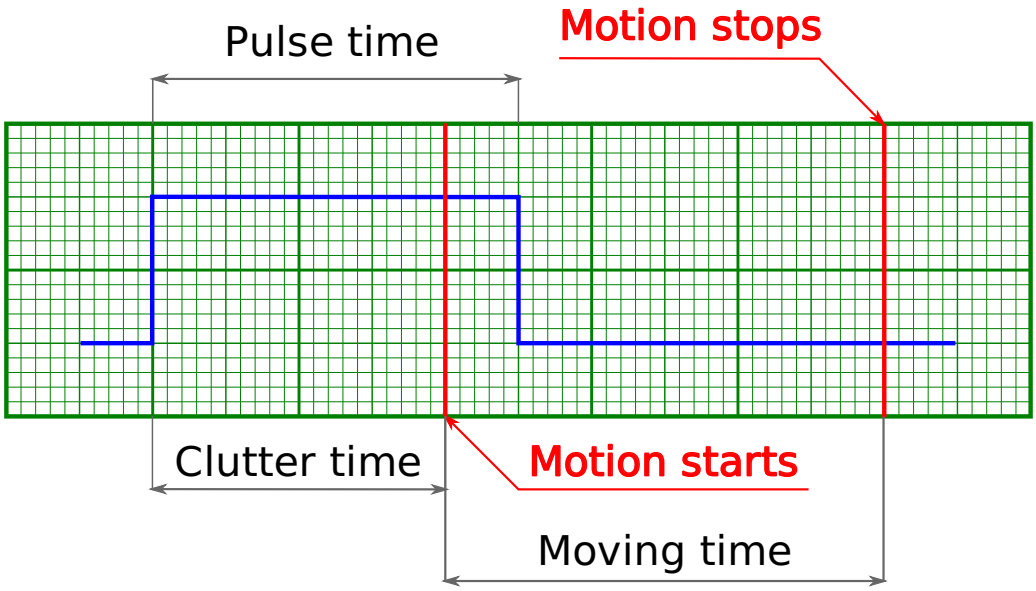
Two TTL synchronization channel provided in controller on BPC connector (see Backplane connector)

Synchronization input

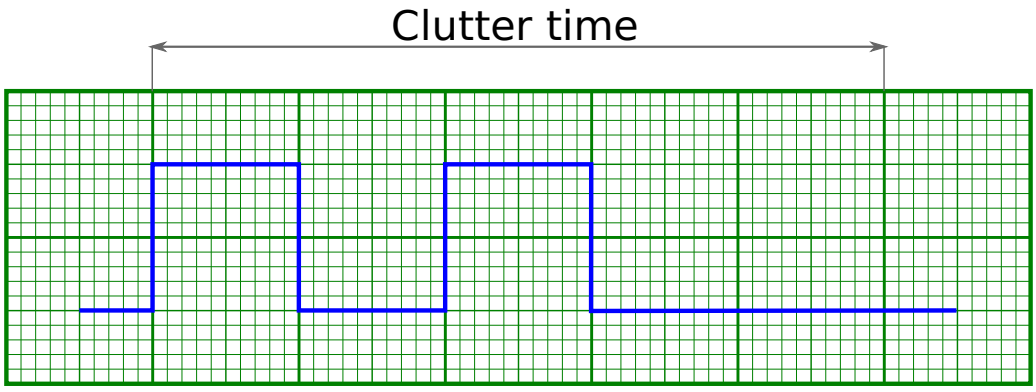
Minimum duration of input synchronization impulse setting which may be registered is provided for synchronization input. This duration is set in microseconds and rounded to milliseconds. Lapse of impulse duration identification is one millisecond. Use this setting for controller noise resistance increasing. Synchronization input can be “on” or “off” . If it is “on” then impulse sending on synchronization input will lead to the motion which is the same as command execution Predefined displacement mode, in which displacement value set by sign Position with microstep part and speed Speed with micro step part. Synchronization input settings change during performance of motion will not lead to motion parameter change (speed, target coordinate). These parameters will be used during coming of the next initialization front to synchronization input. This is dedicated for confirmed multi-coordinate motion performance which can be programmed by the next displacement for each axes during performance of the current displacement multi-coordinate motion. In this case it is necessary to stop axes motion during each displacement.

Note. Position and Speed are independent variables which can be saved in controller non-volatile memory and are used during synchronization input performance.

Note. Motion according to input synchronization impulse is subjected to speed-up settings, maximum speed and other subsystem connected with motion. Its incorrect setting may interrupt coordinated synchronize motion in multi axes system.

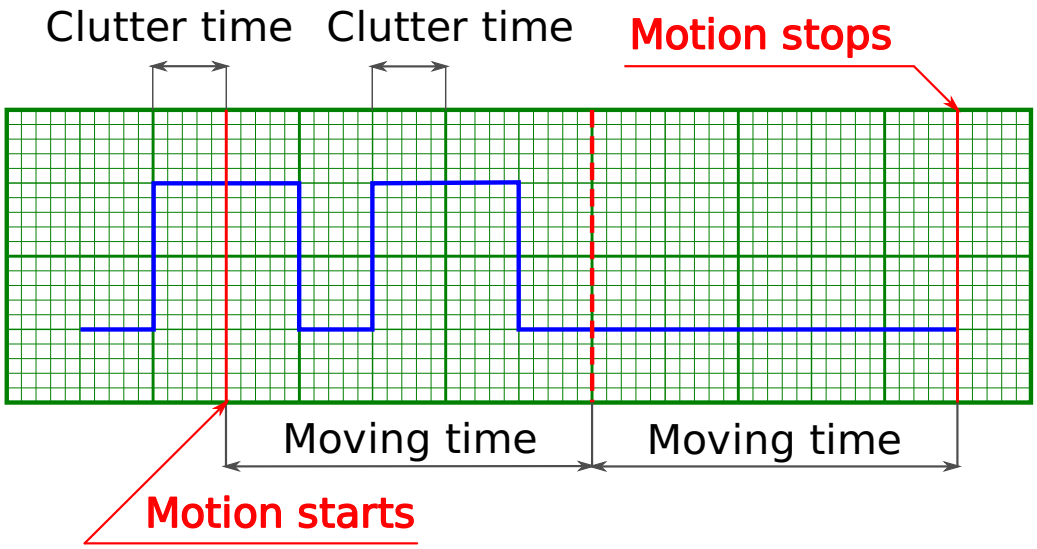


The motion will start if input impulse is longer than time of debouncing.

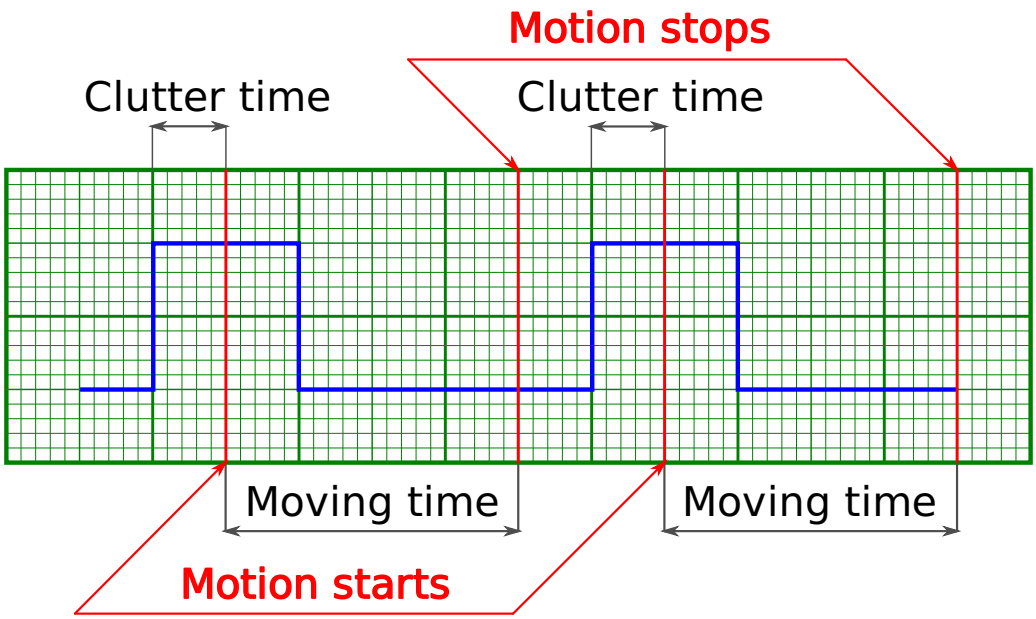


The motion will not start because input impulses are shorter than time of debouncing.

Warning. If during displacement performance another input synchronization impulse is coming then displacement will be made but value will be doubled. If two displacements were made then it will be tripled.



The motion will be performed once but on the double distance, as the second synchronization impulse was made before the end of the first motion.

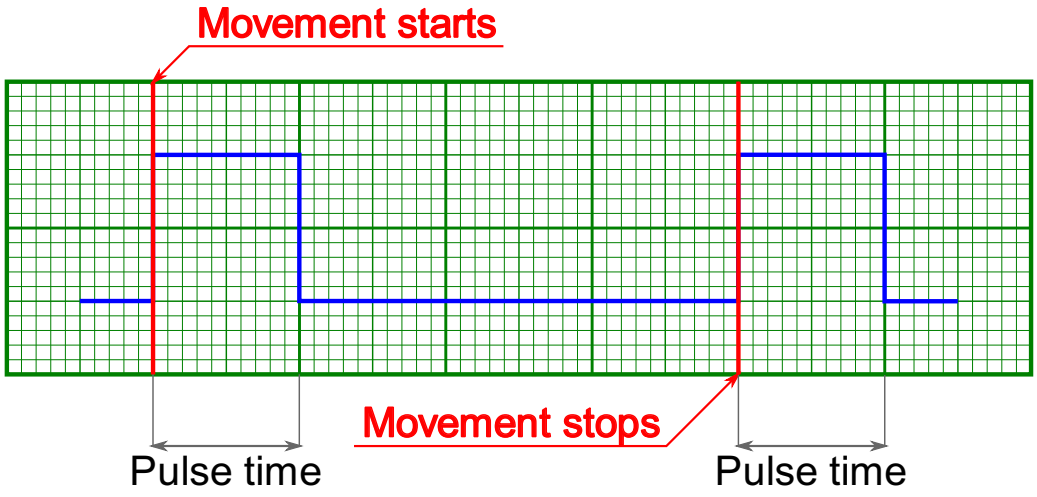


The motion will be performed with two starts and two stops

One state is considered to be active state by default and motion start signal is positive-going slope. Synchronization input can be inverted. zero state is considered to be active in this case and motion start signal is negative-going slope.

Note. Input synchronization inverting is leading to change of active and non-active states definition which begin to appear in[Controller status| Status of controller] However, program inverting itself is not a motion start signal, even if during this a transfer to active state happened.

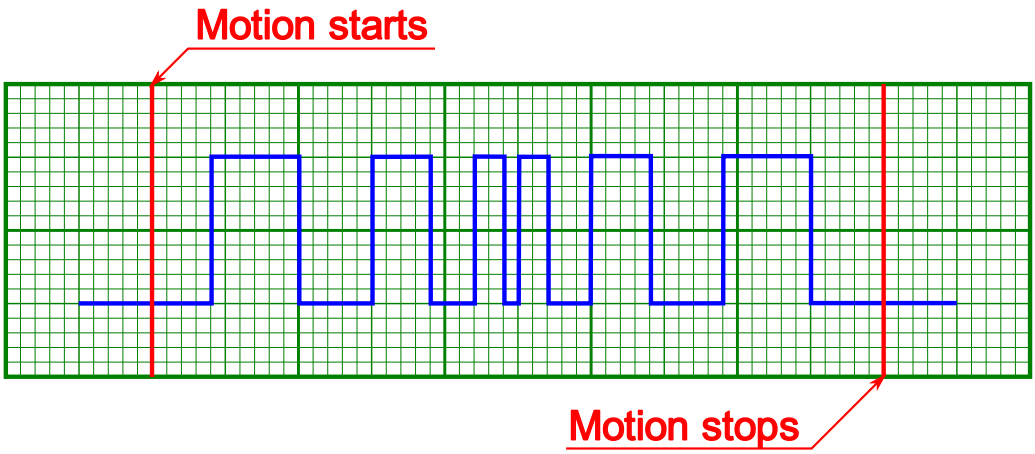
h2. Synchronization output
Output synchronization is used for control of external devices connected to specific motion events. Output sync impulse can be sent during start of motion and\ or during end of motion and\or during positioner shift to the given distance. impulseTime Setting defines synchronization impulse duration (can be defined in microseconds or shift). If duration is defined in microseconds, then it is rounded to milliseconds and is set with accuracy up to one millisecond) Synchronization input can be transferred to controlled digital input mode. In this mode one or zero logic level can be set on output.



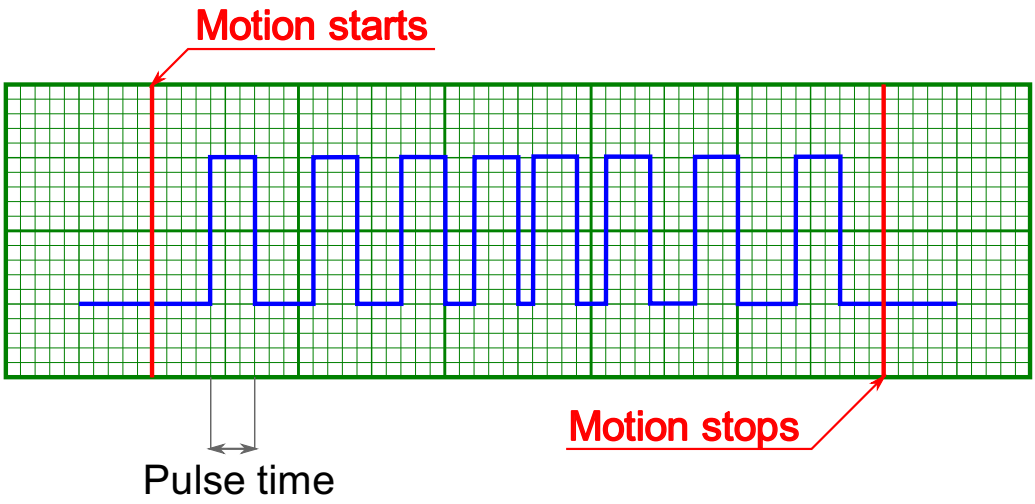
Input impulses of synchronization during their generation at starts and stops of motion (impulse of fixed activity).

If sync impulse duration is expressed in shift units for example 10 steps of stepped motor and is set to the mode " to give sync impulse after stop of motion" then logic level on synchronization output will be sent at the end of motion, but it will be canceled only after 10 steps of the next motion.

Note. If you want to reconfigure synchronization output and you are not sure about its current state transfer it into general-purpose output and set desirable logic level.



Output synchronization impulses during motion with speed-up and impulses generation at every shift to the given distance (impulse is measured in shift units)



Output synchronization impulses during motion with speed-up and impulses generation at every shift to the given distance (impulse is measured in microseconds with rounding to milliseconds)

Note. Periodic impulse generation is operating like complete turn sensor simulator with reduction ratio. Coordinates in which impulse generation is taking place are counted from zero coordinate, not from the coordinate at the start of motion. If for example impulse generation is enabled in settings for every one hundred steps then impulses will be made during passing 0, 1000, 2000, 3000, etc. points. impulses generation is made during motion in both directions. impulse is generating in the moment when fraction from division of current coordinate to the period is changed to one. i.e impulses generation during achievement of 1000 coordinate at motion from smaller coordinate to the bigger and during passing 1000 coordinate at motion from bigger coordinate to the smaller. Also impulse is always generated while passing zero point from any other coordinate (including coordinate zeroization by ZERO button)

Note. In case when impulses on synchronization output are superimpose on one another they are merge into one impulse.

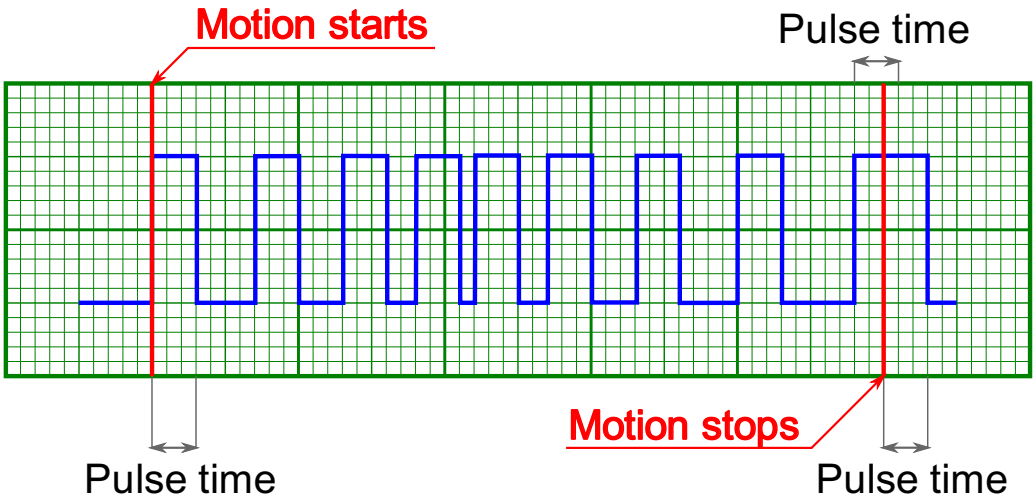


Illustration of impulses synchronization superimpose during motion with speed-up and impulses generation each shift to the given distance at start and after stop of motion (impulse is measured in microseconds).

Synchronization parameters setting in **XILab** are described in section "Synchronization settings".

4.5.6. Design of multi-axis system

Identification of axes within multi-axes systems is performed by controller's serial number. Each controller has its own unique serial number which is reflected in XILab software tab.
About controller.
Getting controller's serial number is possible with a help of GET serial number function. Programming manual:
Multi axes systems on the base of given controller are built with a help of active backplane on the USB hub base or on base of external USB hub.

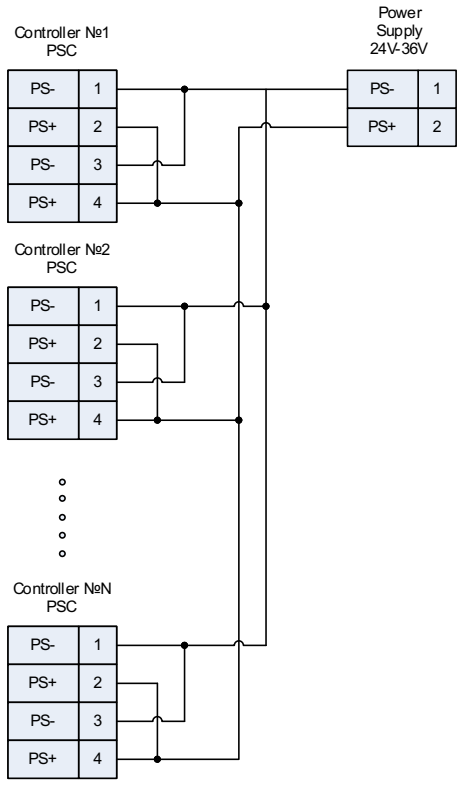
IMPORTANT. We recommend to build backplane on the USB hub base with galvanic isolation from PC and additional power source or converter 36 В -> 5 В because such scheme provides additional noise-immunity and guaranties satisfactory power supply by 5В.

For correct functioning of multi axis board configuration it is necessary to connect all controllers of multi axes configuration to each other by power buses and USB signals (controller mounting on backplane).
Then the following actions should be done in any order:

- Power supply connection to backplane
- Connection of external devices
- Connection of leading controller via USB

Note. Request to additional power source 5V: output current not less than 250 mA for each axis and not less than 400 mA for each axis for full functionality.

Scheme of board's connection to multi board configuration consists of power wiring to the 1 and 2 pins or BPC connector. Supply of independent data channels D-/D+ from USB hub to BPC connector 3 and 4 pins correspondingly. Then power supply connection.



Scheme of controller's power supply connection to multi axes configuration.
PSC - Power Supply Connector, connector for controller power supply.
BPC - Back Panel Connector, connector for additional devices connected to controller.

4.5.7. General purpose digital input-output

Output is located on BPC connector (see. Backplane connector). It allows user to configure it like input or output logical one level is considered to be active.
However it can be inverted so that zero logical level will be considered active.
In input mode you can get information about logical level on output (see. Controller status), or initiate the following actions during transfer to active state (or during transfer to non active state during enabled inverting):
To perform *STOP* command (quick stop)
To perform *PWOF* command (windings power supply switch off).
To perform *MOVR* command (shift to the given distance with the usage of the later settings).
To perform *HOME* command (automatic calibration of position).

Note. Digital input has weak pull down to the ground. The input remains pulled down to the ground during usage Invert flag (see General purpose input-output settings section), but now transfer from logical zero state to logical one state will not lead to the action performance it will be performed only during back transfer from one to zero.

Active on none active logical level or any other state at choice can be set in output mode:

- EXTIO_SETUP_MODE_OUT_MOVING – Active state during motor movement.
- EXTIO_SETUP_MODE_OUT_ALARM – Active state during which the motor is in the Alarm state.
- EXTIO_SETUP_MODE_OUT_MOTOR_ON – Active state while power is not supplied to the motor winding.
- EXTIO_SETUP_MODE_OUT_MOTOR_FOUND – Active state while motor is connected.

Logic type	TTL 3.3 V
Update frequency	1 khz
Nominal current	5 mA

Output parameters

4.5.8. Operations with potentiometer feedback

Potentiometer can be used for cheap and fast but not very accurate position sensor creation for motors of any type. For this controller potentiometer range is presented in 10 000 nominal units. Potentiometer output is located on *BPC* connector (see Backplane connector section)

IMPORTANT. Voltage taking from potentiometer should not exceed 0.0 – 3 V limits. In case of potentiometer power supply excess the errors in its performance and in performance in other controller systems are possible. This may lead to both controller breakage and breakage the motor connected to this controller.

Feedback via potentiometer.

Performance with feedback via potentiometer is used mostly if stepped motor is used on maximum speeds and loads where axis seizing which leads to steps losses is possible. In this case the loss of position will not happen and the motion at available speed will continue.
Also this mode is useful when backlashes do not allow to get one-to-one association between motor axes position at positioner table. In this case potentiometer which is mounted directly on the table allows mechanical system backlashes compensation in real time mode providing accurate positioning.
h2. Potentiometer like an analog input of general purpose.
If potentiometer is used for feedback then free analog input can be used for personal needs. For example, for measurement of some externals signals. The value received from potentiometer can be considered *GETC* command or in XiLab graphs.

Signals voltage	0-3V
Scanning frequency	1kHz

Input parameters

4.5.9. External driver control interface

Interface allows to control any external driver with a help of 3 standard signals enable, direction, clock. This mode is convenient when controller power capability is not enough but there is a desire to use its capacity such as switch limits, joystick control sensor or buttons sensor, magnetic brake, etc. For example, for creation of multi-axes system with one powerful lifting axis which is controlled by external controller with two less powerful horizontal axes, you can use XiLab with 3 axes interface scripts and also synchronize the motion of all three axes. I.e usage of external driver interchanges only power section of controller.
Clock signal defines the quantity of signals in the given Direction (logical one to the right, logical zero to the left). Displacement is a minimum step in the current settings of step division. For step division 1/32 there will be 32 impulses per one step. Don't forget to set external driver in such a way that it would use the same step division.

WARNING. Clock signal frequency in the given controller is limited by 78 kHz. That's why for necessary speed achievement you should minimize step division. Foe example, if rotation speed of 4000 steps per seconds is needed it is necessary to use step division with the accuracy no more than 1/8.

For external driver connection three outputs in BPC connector are used (see Backplane connector).

BBC output connector number	Output function
9	Direction
10	Clock
13	Enable

WARNING. 13 output is general purpose input\output (see General purpose digital input-output) ,but it looses its functionality when external driver control is enabled.

WARNING. Outputs for external driver control are under-protected and can be damaged in case of incorrect usage. Providing of correct connection and necessary electric protections is a duty of an engineer who is constructing drivers' connection system.

Type	TTL
Logical zero level	0 V
Logical one level	3.3 V

Output parameters of external driver control

4.5.10. Serial port

Controller allows UART serial port control with TTL 3.3 V logic. UART outputs are located on BPC connector (see Backplane connector). Due to high occurrence of UART and its adapters to USB, Bluetooth, Ethernet and other standard interfaces wireless control of controller (Bluetooth) or via Internet (Ethernet) is possible. UART data protocol is the same as USB data protocol. I.e it is enough only to enable nonstandard serial port polling on XiLab or in libximc and the device will be found if response delay doesn't exceed two seconds. It is also to control controller with a help of other independent programmed micro-controller, however in this case support is needed Communication protocol.
UART supports the following settings.

Transfer rate	9600-921600 bit/sec
Quantity of bits in transfer	8
Evenness	enabled or disabled
Type of evenness	even, odd, one, zero – at users choice
Quantity of stop bits	1 or 2

WARNING. High speed of data transfer via long cable in electromagnetic blast condition is not possible. Use RC circuit as described on the picture of recommended connection and low down speed, if during data transfer error happens, and circuit characteristic time should be minimum 4 times less than time of one bit transfer. RC circuit characteristic time is set individually.

Note. To make a contact with a controller via UARTit is necessary to contact with it via USB beforehand and to set necessary speed settings: evenness, stop bits quantity and then to save them in non-volatile controller memory. If connection is not made then try to use them.

!UART connection with RC!
Recommended scheme of connection

Logic type	TTL 3.3V
Maximum speed of data transfer	921 kbit
Nominal output current	5 mA

Output and input parameters

4.5.11. Saving the position in controller's FRAM memory

Controller has function of automatic position saving which allows just to cut-off its power supply after the stop and during the next power on controller will be holding on the same motor position, position value and incremental encoder counter position. This function is performed if at the moment when controller was cut-off there were no motor access rotation.

Note. For the performance of this function it is necessary to wait a least 0.5 second after the stop of rotation till controller power supply cut off via USB. Controller cut off during rotation will also lead to position saving, but it will be only approximate so the new calibration is needed.

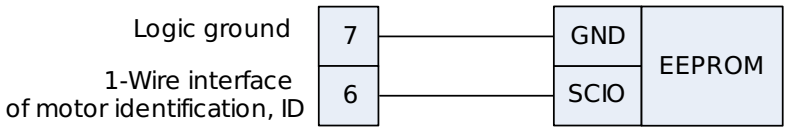
4.5.12. The Standa positioners detection

In the newest positioner made by Standa company (the list of specific devices can be taken from manufacturer) saving of settings and informational parameters of positioner, in memory which is integrated, into movement is provided. This memory is flashed beforehand during movement manufacturing; this allows not to care about optimal settings for every specific positioner and to start operation immediately after system building. There is also a User field of movement name in the memory which user can set himself (see XiLab tab Positioner name).
Download of movement informational parameter in controller memory is done automatically during connection of such positioner to controller (more detailed information about connection can be found in Positioner connector section). If EEPROM_PRECEDENCE flag was fixed (settings of movement memory are in priority Controller status), then all system settings except UART settings and controller name are considered and set additionally.
You don't have to check and/or to setup settings anymore. For example polarity and limit switches location, operational current, parameters of encoder and magnetic brake. All this will be done automatically during positioner connection, if it is equipped by such integrated memory. However if EEPROM_PRECEDENCE flag is fixed then download of this settings will be performed any time of movement connection if movement is equipped by memory, or during enabling of controller to which such movement is connected. That's why if it is necessary to change any kind of settings you need to remove this flag, change necessary settings and save them in the inner controller memory.

Note. There will be no changes in the settings during shutdown of such positioner.

For developers.

Current movements are saved in DS28EC20 chip which is connected via 1-wire interface.
Detection reset signal is sent to EEPROM chip on a periodic basis during movement. In case of receiving backward signal from the chip controller it reads all movement data in its own RAM, automatically sets to the movement and sets out STATE_EEPROM_CONNECTED in status structure. In XiLab this is reflected by EEPR indicator in the main window. the check of memory presence is performed regularly after that. In case of loss of connection with EEPROM (absence of request on reset signal) EEPR indicator in XiLab blinks off.



Scheme of connection for external memory testing

4.6. Secondary features

1. Zero position adjustment
2. User-defined position adjustment
3. Controller status
4. USB connection autorecovery

4.6.1. Zero position adjustment

Controller supports zero position adjustment. This function should be used for anchor marked position, to merge position according to anchor with programmed position. This function is very convenient if one choosing physical position in movement range exists.
For zero position adjustment Special command is used. In this case all steps counters, micro steps counters and encoder are zeroed. Zero position adjustment is made simultaneously for all position counters and can't lead to detuning. There is no influence on current movement command. If controller was performing movement to some physical position at the moment when current position zeroed then movement will be completed in previous physical position. For example during movement to 1000 physical position at the movement of 200 passing the command about position zeroing was sent. Then position counter will minimize by 200 and the movement will be completed for 800 coordinate.

Note. Zero position adjustment during operation in displacement mode (see Predefined displacement mode) won't change physical position to which movement was performed the next displacement will be performed from the same physical position without usage of zero position adjustment.

4.6.2. User-defined position adjustment

SPOS command exists when it is necessary to adjust a position or encoder counter not to zero but to some user defined position. In this command new position counter values, micro steps for stepped motors values and encoder counter values are transferred, if it is integrated position sensor. The flags for ignorance of command field part should be used it is necessary to adjust only one position.
The difference of this command from position zeroing is that there is no zeroing of the last position in which displacement was performed. There are no differences in command performance at the moment of movement or during stop. If command was applied at the moment of movement to the position them movement will be completed in the same physical point at which it will be performed without changing of position by *SPOS* command.

4.6.3. Controller status

Controller tracks its own status and can transfer it to the command status structure GETS. Controller status contains information about performed movement, its result, state of power supply, state of encoder, state of motor windings, digital output-input states, numeric information about position and powering voltage and currents and also error flags.

Movement status

MoveSts contains:

- movement flag which is fixed when controller changes motor position.
- flag of requested speed achievement is fixed if the speed is equal to the speed at which controller should perform current movement.
- Anti-backlashes flag is fixed in case of backlashes suppression during movement finishing stage (see Backlash compensation).

MvCmdSts contains information about executed command. All motor movements are called by movement commands to the *MOVE* aimed position, *MOVR* displacement to the last aimed position, *RIGHT* movement to the right, *LEFT* movement to the left, smooth *SSTP* of rough *STOP* stop, *HOME* home position calibration and *LOFT* despotic backlashes compensation. Buttons, joystick, synchronization impulse control etc. is also performed by these commands. Joystick for example calls right and left movement commands during deflection or smooth stop command in central position (see Joystick control). In MvCmdSts variable is current movement command or the last performed command and also command status: performing or performed. If the command is performed then one more bit shows the result of its performance (successful or not successful). Unsuccessful performance means that we didn't get the position to which we were trying to move or we were unable to perform backlash. Sudden stop by limit switches can also be the reason of getting into Alarm state. Primary state of this field shows unknown command and successful performance status.

Motor power supply status

PWRSts Contains information about powering voltage. windings can be:

- Disabled (in this case no voltage is applied to them.
- Powered by reduced current as to nominal current (for example during usage reduced current function in windings during stop).
- Powered by nominal current.
- Powered by under-voltage, to provide set nominal current.
  The last status frequently appeared during high rotation speeds, the higher the speed of step switching the higher is the powering voltage to provide current rise in motor windings induction coefficient. Underfeeding doesn't mean that motor won't be rotating it means that motor will make more noise and torque moment reducing (see Power control).

Encoder status

EncSts contains information about connected encoder if control mode without feedback is enabled (for stepped motors for example). There are several states of encoder:

- Not connected
- Unknown state, when there is enough data to define encoder's state.
- Connected and operable
- Connected and reversed, in this case it is necessary to enable reverse in encoder settings.
- Connected and inoperable
  The last state is realized when switch signals come to encoder inputs but they don't correspond to the motor rotor movement. Change of states is made after sufficient statistic collection. That's why detection is not made immediately. It is also impossible to define encoder status without movement (see Operation with encoders).

Motor windings status

WindSts contains information about windings state. State of each from two windings is reflected separately. They can be:

- Disconnected from controller
- Connected
- Short-circuited
- Or their state can be unknown
  Too Small voltage and induction in windings is considered to be short-circuiting. Voltage with to high resistance is considered to be windings` disabling.

Position status.

All data about position and positioner speed is reflected in status structure. Fields of basic position (CurPosition,

uStep), secondary position (EncPosition), speeds (CurSpeed, uCurSpeed) are used for this. Basic position is counted in steps and microsteps of stepped motor if control without feedback is used. In case of leading encoder mode in CurPosition usage encoder counter position is saved and zero is writing to uStep. Secondary position field can operate in three modes, stepped motor control without usage of feedback is used – encoder field, – if stepped motor is enabled in leading encoder mode – steps counter, if DC motor is enabled – zero mode. The speed is always reflected for the basic position sensor and it is measured in the same units in which movement speed is set.
h3. Controller power supply status and temperature.
Status structure reflects:

- Power section useful current (in mA)
- Power section voltage (in dozens of mV)
- Useful current via USB (in mA)
- Voltage in USB (in dozens of mV)
- Microprocessor temperature (deciles of Celsius degrees)

Status flags

There are several types of flags – control command errors flags, critical parameters excess errors flags, general errors flags and state flags. Many flags cannot be removed themselves they can only be removed despotically by *STOP* command.
Protocol commands errors:

- errc – Unknown protocol command. This error should not appear if the used software corresponds to the used controller protocol version. Flag can't be removed by itself.
- errd – Data integrity command check code is incorrect. This error appears in case of data transfer failure. The flag can't be removed by itself.
- errv – It is impossible to use variables distribute value in command. It appears when command was received and successfully recognized but transferred data was incorrect or was not fully encoded. This error can also mean that necessary operation is impossible because of hardware failure. For example, this error appear during step division mode setting which is not in the list of supported modes or during setting of zero steps quantity pre motor rotational rate. The flag can't be removed by itself.
  Critical parameters excess errors:
- Flag which means that controller is in Alarm mode
- Flag which means that power driver gives overheat signal. The flag is removed by itself depending on critical parameters settings
- Flag which means that microprocessor temperature is above acceptable range. The flag is removed by itself depending on critical parameters settings
- Flag which means that power supply exceeded acceptable value. The flag is removed by itself depending on critical parameters settings
- Flag which means that power supply voltage is lower than acceptable value. The flag is removed by itself depending on critical parameters settings
- Flag which means that useful current from power unit exceeded acceptable value. The flag is removed by itself depending on critical parameters settings
- Flag which means that USB voltage exceeded acceptable value. The flag is removed by itlsef depending on critical parameters settings
- Flag which means that USB voltage is under acceptable value. The flag is removed by itself depending on critical parameters settings
- Flag which means that useful current in USB power bus exceeded acceptable value. The flag is removed by itself depending on critical parameters settings
- Flag which means that limit switches are mixed up. The flag can't be removed by itself.
  Error general flag:
- Flag which means that position control system detected steps counter and position sensor desynchronization. The flag can't be removed by itself.
  State flags:
- Presence of connected movement equipped by EEPROM memory.
- Presence of external power supply. Otherwise power supply is internal. Always fixed.

Digital signals status.

Controller reflects input and output digital signal status as active state flags or as current logical level. Active state corresponds to one or to zero depending on specific block settings, for example on inverting settings. Flags can be:

- Right limit switch state (one if limit switch is active)
- Left limit switch state (one if limit switch is active)
- Right button state (one if button is pressed)
- Left button state (one if button is pressed)
- 1 if EXTIO pin operates as output. Otherwise - as input.
- EXTIO pin state (1 if state is active on input or on output)
- Hall A sensor state (1 if logical one is on input)
- Hall B sensor state (1 if logical one is on input)
- Hall C sensor state (1 if logical one is on input)

- Magnetic brake state (1 if power supply is applied to brake)
- Complete revolution sensor state (1 if sensor is active)
- Input synchronization pin state (1 if synchronization pin is in active state).
- Output synchronization pin state (1 if synchronization pin is in active state).
- Input encoder A channel state (1 if logical one is on input)
- Input encoder B channel state (1 if logical one is on input)

4.6.4. USB connection autorecovery

This unit is designed to reboot the USB bus in the event of loss of communication (for example, this may occur in the event of electrostatic discharge or when the USB bus is disconnected without powering down the controller). The on/off state of the block is determined by the flag USB_BREAK_RECONNECT (see Critical parameters). If the unit is turned on, it monitors the connection loss on the USB bus. In the case of communication lost on the USB bus after 500 ms the firmware restart the USB device on the bus, and then checks the state of the USB bus. If for a certain time there is no recovery of connection (i.e. data communication), then block reconnect the USB bus again. Thus, in case of no restoration of USB connection, the controller will continuously reconnected to the USB bus until recovery of USB connection or until the time between reconnection exceeds 1 minute. So, in the case of disconnect the USB bus without powering down the controller (for example, in the case of motor control with buttons or joystick) for about five minutes the controller will be in the reconnected mode.

Note. Reconnecting mode doesn`t affect other characteristics of the controller (for example, movement or withholding required current in the windings)

To avoid simultaneous reconnect to the USB bus from both the controller and the computer side, the time between the reconnections changes exponentially (see Table 1).

Number of restart	timeout, ms
0 (after loss commonication)	500
1	483
2	622
3	802
4	1034
5	1333
6	1718

The status of the unit can be determined by frequency of LED flashing. In the case of the transition to reconnect mode the LED will flash at a frequency of 10 Hz (see Operating_modes_indication).

Warning. Because of the structure of the program unit, as well as USB bus specification, unit doesn`t guarantee 100% recovery of the communication with the computer after a static discharge.

Xilab software also tries to reconnect to the controller when it is running. On connection loss, which is defined as "result_nodevice" libximc library call error, Xilab waits for 1000 milliseconds, then attempts to reopen device port. On Windows operating systems Xilab uses WINAPI functions to check if corresponding COM-port device is present. If it is, then after two unsuccessful attempts to reopen it calls libximc ximc_fix_usbser_sys function, which resets the usbser.sys driver to fix the driver error. On Linux or MacOS Xilab simply tries to reopen the device every 1000 ms. After the device is opened Xilab sends several commands to read serial number, firmware version and controller settings which are needed to set up user interface.
Libximc library considers device lost (return error code result_nodevice) on critical errors from system calls ReadFile/WriteFile (Windows OS) or read/write (Linux/Mac OS).

5. XILab application User's guide

5.1. About XILab

XILab features a user-friendly graphical interface, which is designed for positioners control, diagnostic and fine tuning of the motors driven by the controllers. XILab allows quick adjustment of connected positioner by loading of previously prepared configuration files. The control process can be automated with script language that can be used either directly or to speed up the process of customized control program development. XiLab supports multiaxial mode and multidimensional control scripts. It is possible to output the data about controller and motor status in form of charts and save them to a file. The software is compatible with Windows XP, Windows Vista, Windows 7, MacOS X and Linux operating systems. Depending on the OS of your computer, appearance of some windows may vary.

Here you can find the Quick Installation Guide for the application. This chapter provides a detailed manual for the XILab software.

5.2. Main windows of the XILab application

1. XILab Start window
2. XILab Main window in single-axis control mode
3. XILab Main window in multi-axis control mode
4. Application Settings
5. Charts
6. Scripts
7. XiLab Log

5.2.1. XILab Start window

When started, XILab opens a controllers detection window. By means of libximc library, XILab queries controllers connected to the system and displays a list of found and successfully identified controllers.

XILab Start Window, 0 controllers found

If no controllers were found, the information about it is displayed on the start screen. Clicking the Retry button will start the detection again, clicking Cancel button will close the window. Add virtual controller button adds a virtual XIMC controller (available only for Windows version), which supports the request-response protocol of a real controller. However it returns zeroes in all structures, which are read from it, with an exception of the status of the last movement command MvCmdSts, which is stored as an internal state. Virtual controllers may be useful for testing and getting used to the XILab interface, if no real hardware controllers are connected to the system.

XILab Start Window, 1 controller found

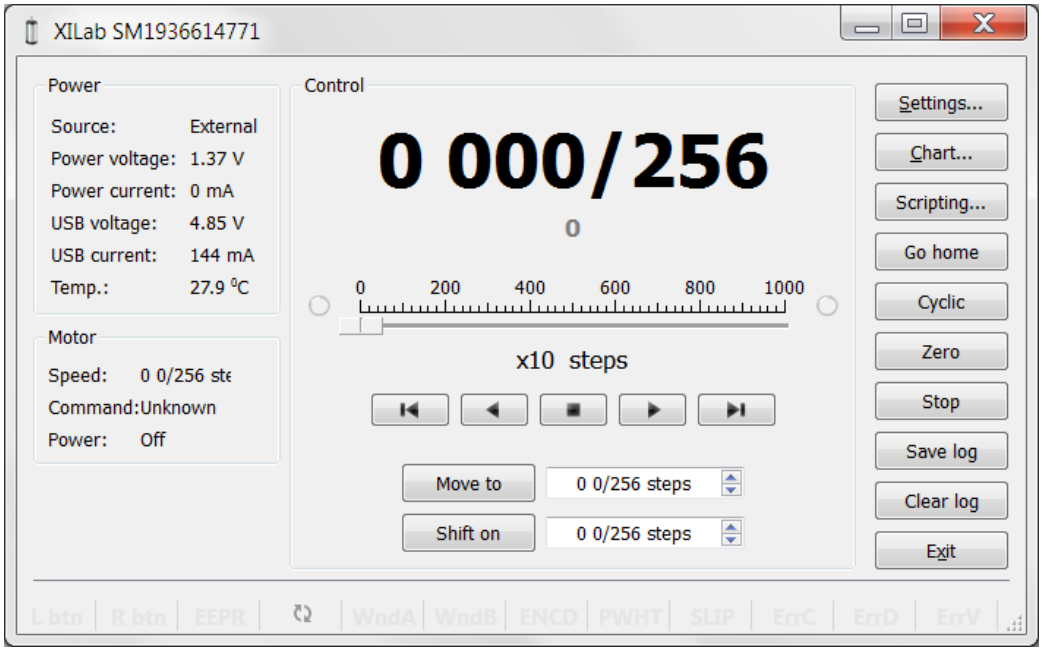
If only one controller is found, then XILab Main window in single-axis control mode is opened automatically.

XILab Start Window, 3 controllers found

If more than one controller is found, the list is displayed on the start screen. Here you can select one or more controllers and open them using the Select button. If one controller is chosen, then XILab Main window in single-axis control mode will be opened, if more than one controller is chosen the XILab Main window in multi-axis control mode window will be opened. You could repeat the search by clicking the Retry button or quit by clicking Cancel. If the Open last button is active it means that all the controllers that had been opened in the previous run of multi-axis XILab interface were found. Clicking the Open last button then will open the last saved multi-axis configuration.

Note: Since the libximc library opens XIMC devices in the exclusive access mode, when you start subsequent copies of XILab application, only free controllers will be found and available for selection.

5.2.2. XILab Main window in single-axis control mode



XILab Main Window

In the left part of the window in **Power** and **Motor** groups of parameters status of the controller and the motor is available. In the central part of the window there is the **Control** group, containing the elements of motor motion control. On the right there is a group of buttons to control the application as a whole. At the bottom there is a **log**, which is hidden as the window is resized to its minimum size. Below we consider these groups in more detail.

Motion Control Unit

Control Unit

In the central part of the block there is an indicator of the current position. Below it, in case the encoder is enabled, is located an encoder position indicator. In the master encoder mode (see Operation with encoders section) the main and the secondary indicators swap their places.

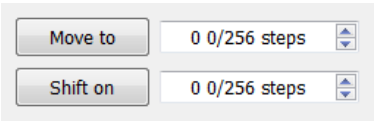
Below is the **Control** unit, containing the elements of motor motion control. Let us examine them in greater detail:

Movement without specifying the final position

Movement control buttons

- The buttons Left, Stop and Right trigger movement to the left without specifying the final position, stop any previously started movement and start the movement to the right without specifying the final position, respectively.
- Button Left to the border will make the motor rotate to the left border of the slider. Right to the border, respectively, will do it to the right edge of the slider.

Movement to the target point



Movement control to the given point

- Move to button starts the process of moving to the given position.
- Shift on button starts the process of shift to a given distance from the target position.

Target position for motion commands

Commands Move to and Shift on use the target position to calculate the movement. The target position is changed by the following commands:

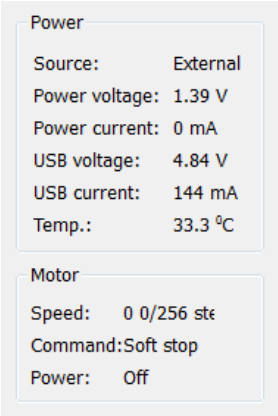
Move to <value>
Target position = <value>

Shift on <offset>
Target position = target position + <offset>

Zero (provided there is no movement at the moment of sending the command)
Target position = 0

Commands Stop, Left, Right, Left to the border and Right up to the border do not alter the target position.

Controller and motor status



Controller Power Supply

Power group of parameters contains the following indicators:

- Source - source of power supply for controller. The controller may be powered by USB or by External power source.
- Power voltage - voltage supplied to the power module.
- Power current - current consumption of the power module.
- USB voltage - voltage on USB connector.
- USB current - current consumed by the controller via USB
- Temp. - Temperature of the controller processor.

If the color of the indicator Power voltage changes to blue or red, it shows that voltage power supply exceeds the allowed value range over the acceptable value, which is defined in the Critical board ratings. In this case, the

controller switches to Alarm state. It is possible to quit the Alarm state after terminating of the events that caused Alarm, provided that the flag Sticky Alarm is not set. If the flag Sticky Alarm is on, use the Stop button to quit the Alarm state.

If color of the indicator Power current turns red, it shows that the current consumed by the controller from the power supply is over the acceptable value, which is defined in the Critical board ratings. In this case, the controller switches to Alarm state. It is possible to quit the Alarm state after terminating of the events that caused Alarm, provided that the flag Sticky Alarm is not set. If the flag Sticky Alarm is on, use the Stop button to quit the Alarm state.

If the color of the indicator USB voltage turns blue or red, it shows that the USB voltage goes out of the allowed value range towards lower and higher voltage respectively. In this case, the controller switches to Alarm state. It is possible to quit the Alarm state after terminating of the events that caused Alarm, provided that the flag Sticky Alarm is not set. If the flag Sticky Alarm is on, use the Stop button to quit the Alarm state.

If the color of the indicator USB current turns red, it shows that the USB current supply exceeds the acceptable value, which is defined in the Critical board ratings. In this case, the controller switches to Alarm state. It is possible to quit the Alarm state after terminating of the events that caused Alarm, provided that the flag Sticky Alarm is not set. If the flag Sticky Alarm is on, use the Stop button to quit the Alarm state.

If the color of the indicator Temp. turns red, it shows that the temperature of the controller board exceeds the acceptable value, which is defined in the Critical board ratings. In this case, the controller switches to Alarm state. It is possible to quit the Alarm state after terminating of the events that caused Alarm, provided that the flag Sticky Alarm is not set. If the flag Sticky Alarm is on, use the Stop button to quit the Alarm state.

Motor status

Motor group of parameters contains the following indicators:

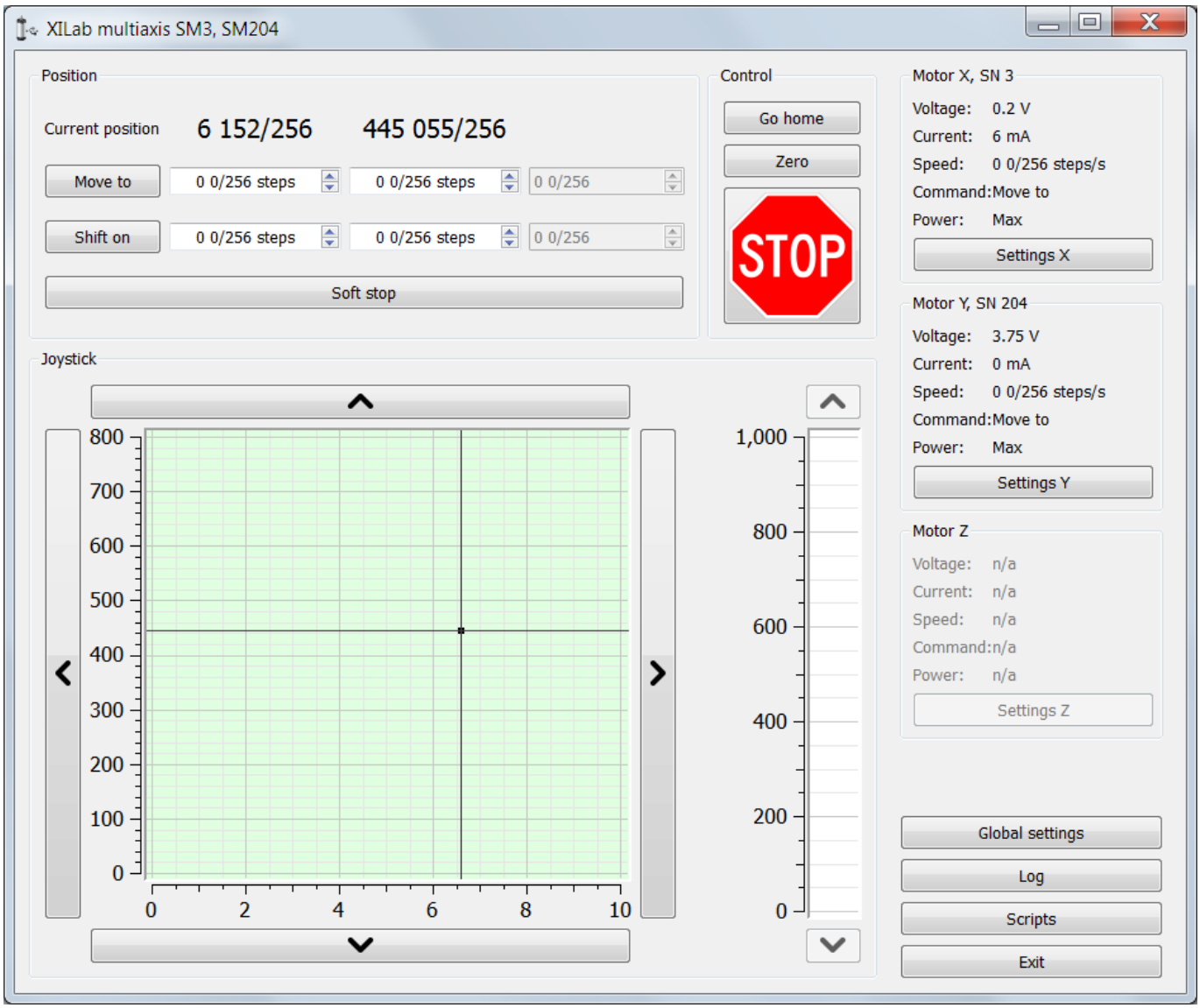
- Speed - rotation speed of the motor.
- Command - the last performing (bold font) or executed (plain font) controller command. Controller command appears in black if the flag of the motion error MVCMD_ERROR is not set, in red otherwise. Can be one of the following options:
  - Move to position - move to the set position
  - Shift on offset - offset for a predetermined distance
  - Move left - move left
  - Move right - move right
  - Stop - stop
  - Homing - find the home position
  - Loft - backlash compensation
  - Soft stop - smooth stop
  - Unknown - unknown command (it may appear immediately after the controller starts)
- Power - state of stepper motor power supply. Can be one of the following options:
  - Off - motor winding is disconnected and not controlled by the driver,
  - Short - winding is short-circuited through the driver,
  - Norm - winding is powered with nominal current,
  - Reduc - winding is deliberately powered with reduced current relatively to operational one to reduce the power consumption,
  - Max - winding is powered with maximum available current, which a scheme with a given voltage supply can output.

Group of application control buttons

- Button Settings ... opens controller settings, see Application Settings.
- Button Chart ... opens a window with charts, see Charts.
- Button Scripting ... opens window for scripting, see Scripts.
- Button Go home searches for the home position, see Home position settings.
- Cyclic button turns on the cyclic motion, see Cyclical motion settings.
- Zero button resets the current position of the motor and the encoder value.
- Stop button sends the command of immediate stop, and resets the Alarm state.

- Button Save log saves the contents of the log to a file in CSV format (opens a file selection dialog window).
- Button Clear log clears the contents of the log.
- Button Exit performs safe shutdown, see Correct shutdown.

5.2.3. XILab Main window in multi-axis control mode



XILab Main window

In the top left part of the screen in the **Position** and **Control** parameter groups there are elements of motion control and indicators of the current position. In the bottom left part of the window there is the **Joystick** block, which is a graphical control element for several axes. In the top right part, in the **Motor** blocks data on the current status of controllers and connected motors is located. In the bottom right part of the window there is a group of buttons for application control as a whole. Let us consider these groups in more detail.

Motion control block

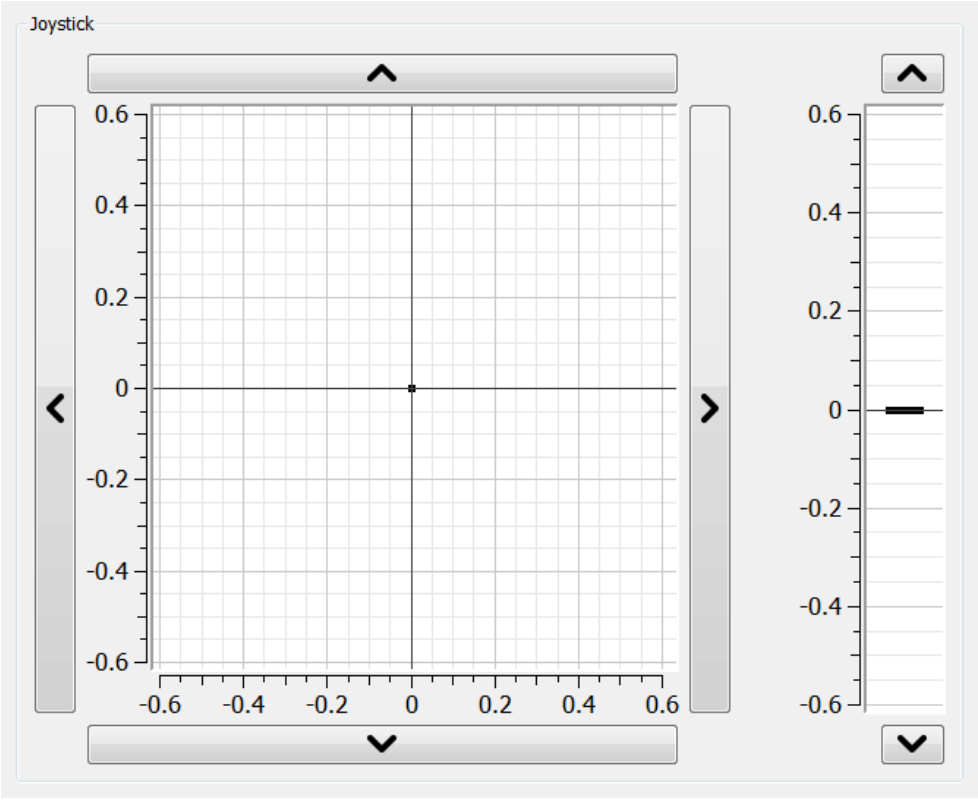
Motion control block

In the Current position line indicators of the current position in steps or calibrated units (see below) for the axes X, Y and Z (from left to right) are located. The Move button performs movement to the coordinate given by the controls of

this line, and the Shift button performs relative offset for a predefined distance. If one of the controllers is temporarily absent or disabled, the corresponding column becomes dimmed.
The Soft stop button executes a soft stop for each of the controllers.

The Go home button searches for the home position independently for each of the controllers; see Home position settings.
The Zero button resets the current position of the motor and value of the encoder for each controller.
The STOP button sends an immediate stop command to each controller and resets their Alarm statuses.

Virtual joystick block



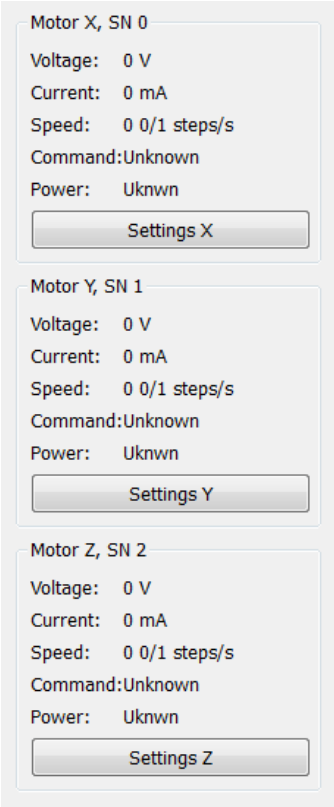
Virtual joystick block

In this block, the current coordinate of the controllers is visualized as a dot with two lines on the plane for X-Y axes and the line for the Z axis.
There are several possible ways to control movement of the controllers:

- When you click anywhere on the X-Y plane or in the Z column, the corresponding controller or controllers start to move to the selected coordinate in accordance to its own movement settings.
- When you press and hold the screen buttons with up, down, left and right arrows, the corresponding axis starts to move in that direction. The movement gradually stops when you release the button (soft stop command is sent).
- When you press and hold the keyboard buttons Right, Left, Up, Down, PageUp or PageDown when the joystick block has input focus, the axis X, Y or Z, respectively, starts to move in the direction of increasing or decreasing coordinate. The movement gradually stops when you release the button (soft stop command is sent). On receiving input focus the widget background color changes from white to light-green.

The scales of the axes are set in "Slider settings" on Program configuration page in Settings window separately for each controller. If user units are being used, then the corresponding axis scale is measured in these units. If the position read from any controller is out of its axis range, then the corresponding indicator will not appear on the screen.

Block of status indicators for controllers and motors



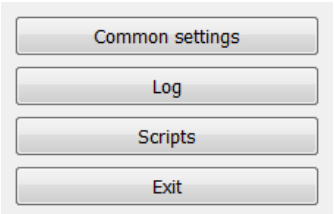
Block of status indicators for controllers and motors

Here are the three blocks of indicators of controllers and motors for axes X, Y and Z. In the title of the block the serial number of corresponding controller is displayed. Each block contains the following indicators:

- Voltage - the voltage at the power section of the motor.
- Current - current power consumption of the motor.
- Speed - current speed of the motor.
- Command - the last command of the controller that was executed, or is being executed.
- Power - the state of the motor power supply.

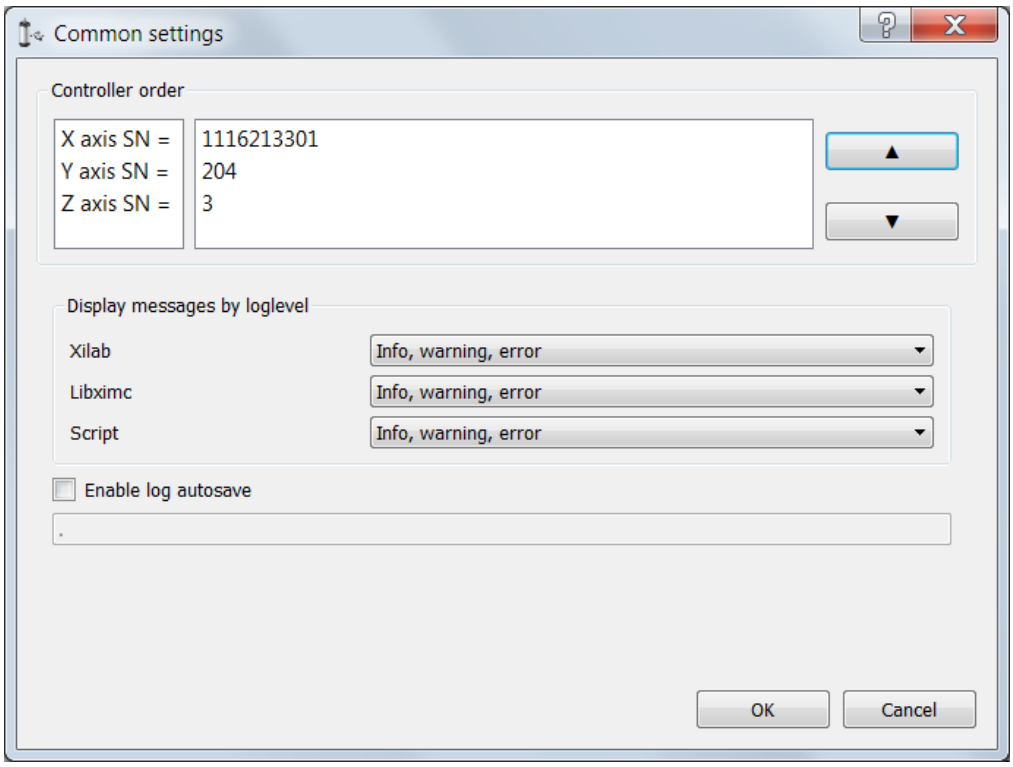
The buttons Settings X, Y, Z open the configurations of a controller, which corresponds to this axis, see Application Settings.

Block of buttons for application control



Block of buttons for application control

Common settings button opens a dialog window with general settings. Here you can choose which controller serial numbers are considered "X", "Y" and "Z" axes, also this window contains logging settings.
Log button opens a window with log information. Here you can find diagnostic information (errors, warnings, informational messages) from XiLab, libximc and script sources.
Scripts button opens a scripting window, see Scripts.
Exit button performs proper shutdown, see Correct shutdown.



Common settings dialog window

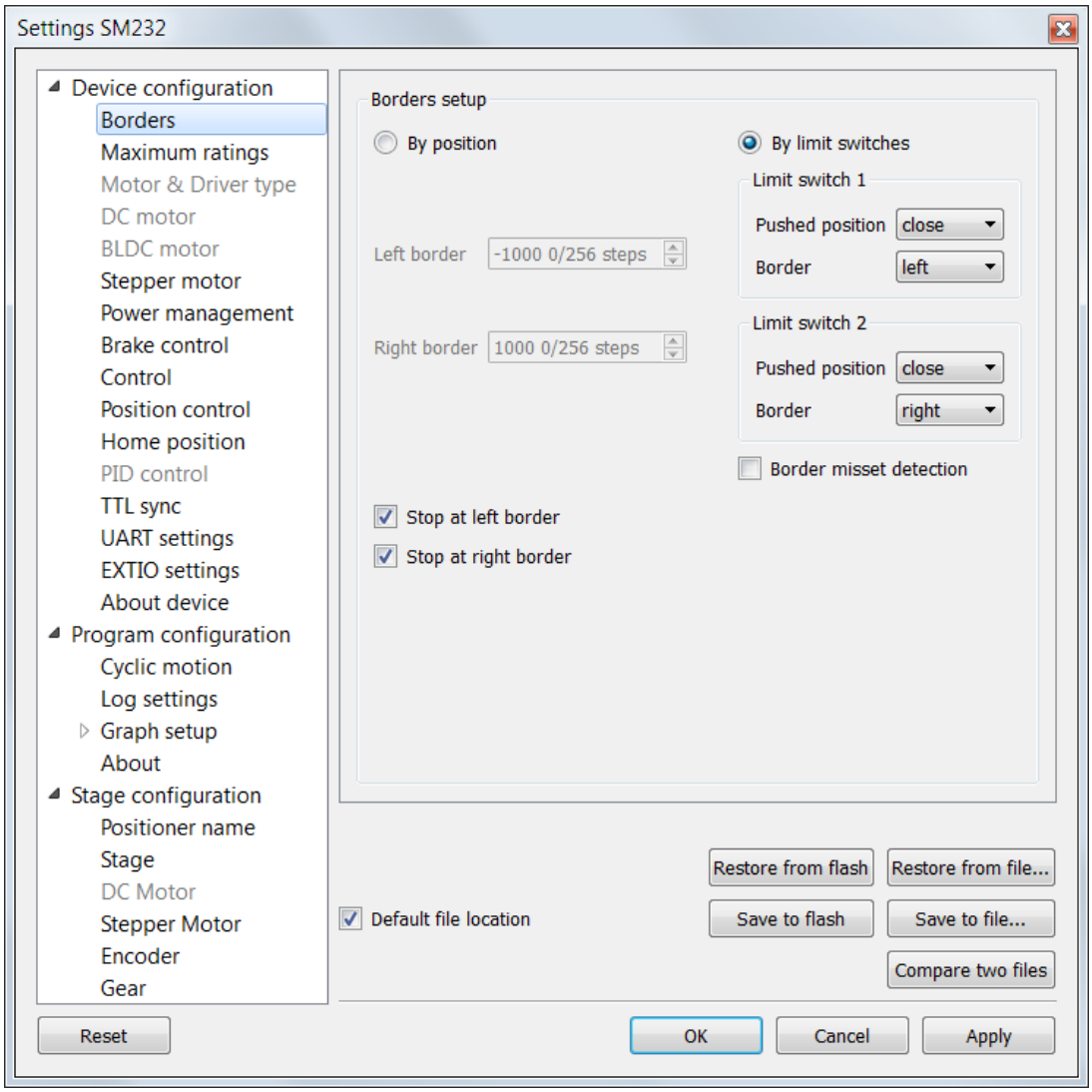
Common settings window contains axis order settings and logging settings.

You can reorder axes by choosing the axis serial, then pressing "up" or "down" buttons on the right.
First axis in the list to the right of "X axis SN = " label will be referred to as "X axis", the second one as "Y axis", the third one, if it is available, as "Z axis".

Common logging settings are completely identical to single-axis version logging settings.
Here you can configure logging detail ("None" for no logging, "Error" to only log errors, "Error, Warning" to log errors and warnings, "Error, Warning, Info" to log errors, warnings and informational messages) for each source: XiLab itself, libximc library and Scripts module.
If the autosave option is enabled the log is saved to the file, which is flushed every 5 seconds. It has a name of type "xilab_log_YYYY.MM.DD.csv", where YYYY, MM and DD are current year, month and day respectively. The log is saved in CSV format.

5.2.4. Application settings

Settings button from the main window opens the Settings window.



XILab Settings Main Window

Application settings are presented as a hierarchical tree and are divided into three groups: controller settings - "Device configuration", XILab application settings - "Application configuration", characteristics of positioner - "Stage confuguration".

The first group (***Device configuration***) contains the parameters that can be stored directly in the device (in the flash memory or in the RAM of controller).

The second group (***Application configuration***) contains the XILab application settings, which are not written into the controller, and are used to control the operation of the XILab interface, such as XILab general settings or Charts styles, XILab interface control of the controller, eg Cyclical motion settings.

The third group (***Stage confuguration***) contains information about the parameters of the positioner.

Description of **Restore from flash** and **Save to flash** buttons is located in the Saving the parameters in the controller flash memory.

All application settings can be saved to an external file when you click **Save to file**.
When you click the **Restore from file** button you can pick a file with application settings to be loaded into Settings window.

When you click the **Compare two files** button, a dialog window with file selection is opened. If you select two files all their settings are compared and a list of differences is displayed. Missing keys in one of the files are marked in the table as "<NO KEY>".

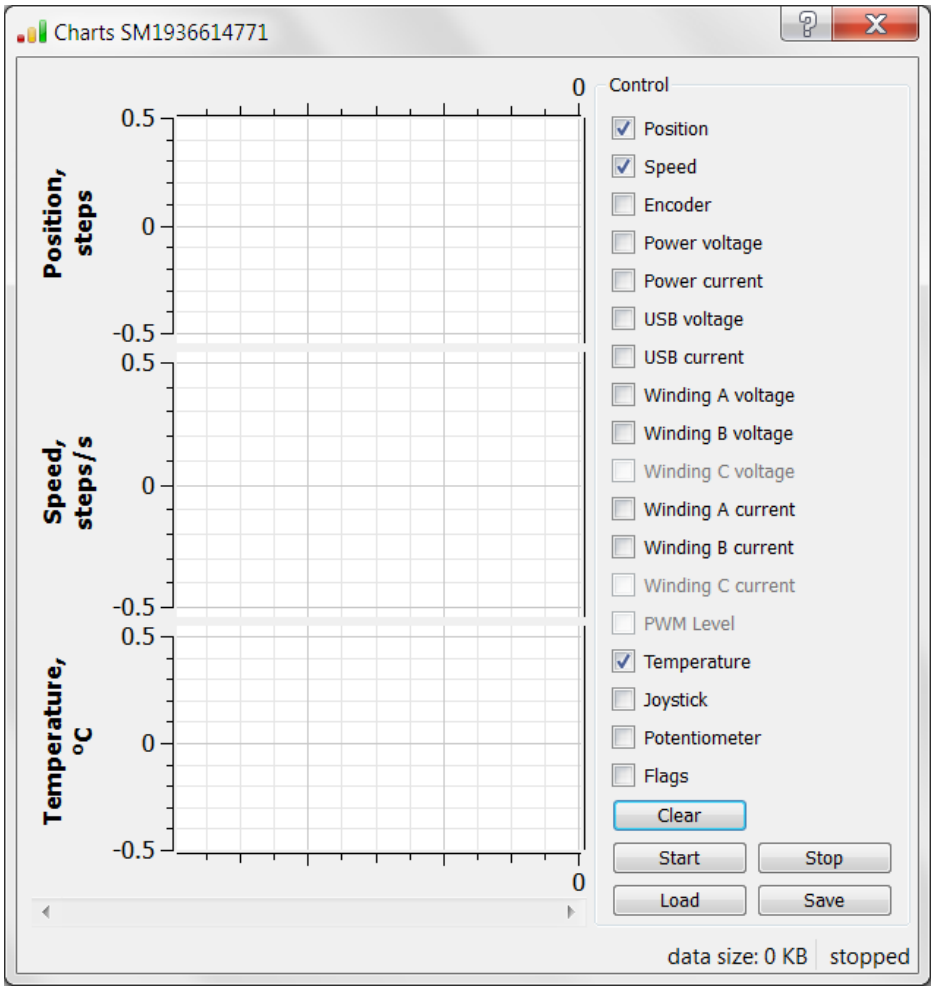
The **OK** Button closes the Settings window and saves all changes to the settings to the controller. The **Cancel** button closes the window without saving. The **Apply** button saves the settings without closing the window.

The **Reset** button resets all setting changes that were made after the last time the **Apply** button was pressed, or after opening the Settings, if the **Apply** button was not pressed.

Note. Unlike saving the settings to the flash memory of the controller, while saving to an external file, all of the Settings window parameters are saved.

5.2.5. Charts

Button **Chart** of the main window opens a window for working with charts.



XILab Charts window

In the left part of the window there are Charts of variables, in the right side of the window there is the **Control** block, which contains the charts control elements. Above the **Control** block there are flags which enable different charts, below the **Control** block there is a group of buttons to control the stored charts data.

Values displayed on the charts

- Position is the primary field in which the current position is stored, no matter how feedback is arranged. In the case of a DC-motor this field has the current position according to the encoder, in the case of a SM (stepping motor) this field contains the current position in steps;
- Speed is the current speed;
- Encoder is the position according to the secondary position sensor;
- Power voltage is voltage of the power section;
- Power current is current consumption of the power section;
- USB voltage is voltage on the USB;
- USB current is current consumption by USB;
- Winding A current - in the case of stepper motor, the current in winding A; in the case of a brushless motor, the current in the first winding and in the case of DC motor, the current in its only winding;
- Winding B current - in the case of stepper motor, the current in the winding B; in the case of brushless motor, the current in the second winding, unused in the case of DC motor;
- Winding C current - in the case of a brushless motor, the current in the third winding, unused in the case of DC or stepper motor;
- Winding A voltage - in the case of stepper motor, the voltage on winding A; in the case of a brushless motor, the voltage on the first winding and in the case of DC motor, the voltage on its the only winding;
- Winding B voltage - in the case of stepper motor, the voltage on the winding B; in the case of brushless motor, the voltage on the second winding, unused in the case of DC motor;
- Winding C voltage - in the case of a brushless motor, the voltage on the third winding, unused in the case of DC or stepper motor;
- PWM level the PWM duty factor (only for DC motors);

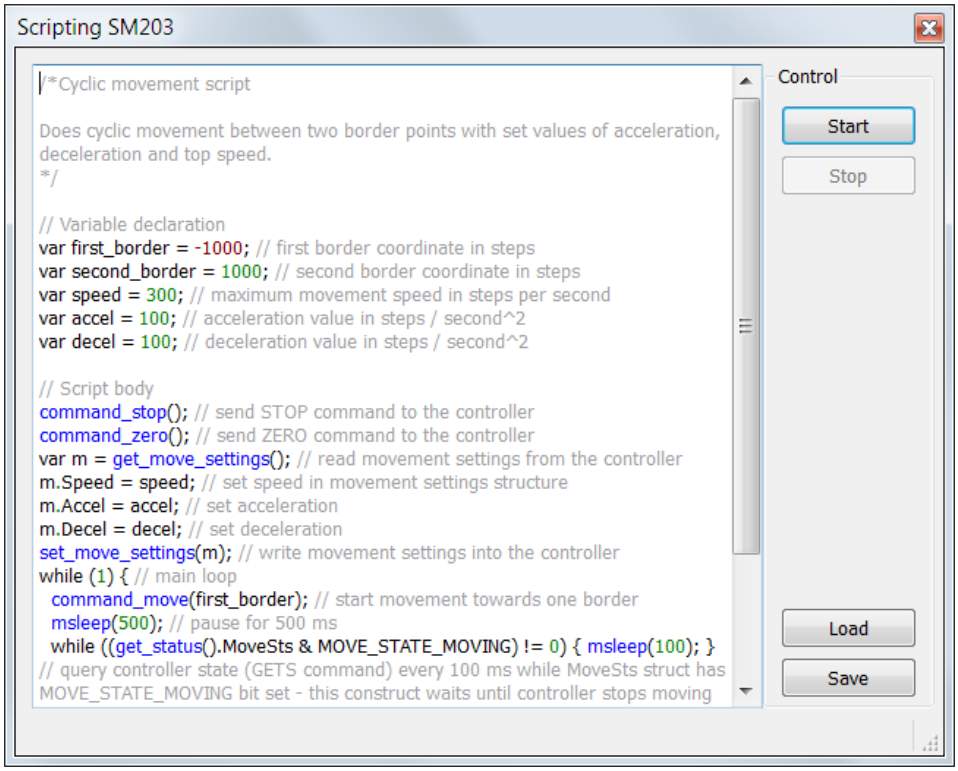
- Temperature is temperature of controller processor;
- Joystick is value of input signal from the joystick;
- Potentiometer is value of input signal from the potentiometer;
- Flags shows the state of the controller flags.

Button Functions

- Clear - clears the stored data and the Charts window;
- Start - starts recording the data and displaying charts. If the option "Break data update when motor stopped" in **Program configuration -> Graph setup -> Common** is turned on, no data recording and charts auto-scrolling will occur when the motor is stopped;
- Stop - stops data reading;
- Save - stores chart data to a file;
- Load - loads chart data from a previously saved file.

5.2.6. Scripts

The Scripts Button main window opens a window for working with scripts.



XILab Scripts Window

Xilab scripting language is implemented using QtScript, which in turn is based on ECMAScript.
ECMAScript is the scripting language standardized by Ecma International in the ECMA-262 specification and ISO/IEC 16262.

Brief description of the language

Data Types

ECMAScript supports five primitive data types:

- Number,
- String,
- Boolean,
- Null,
- Undefined.

Also, the language has "composite" data type

- Object.

Statements

ECMAScript has fifteen different types of statements, the details are given in the table below

Statement	**Brief information**
Block	{[<instruction>]}
VariableStatement	var <list of variable statements>
Empty Statement	;
ExpressionStatement	[string to ∉ {{, function}] statement
IfStatement	if (<user>) <expression> [else <expression>]

IterationStatement	do <expression> while (<statement>)
	while (<statement>) <expression>
	for ([<statement before start>] [<statement>] [<statement>]) <expression>
	for (<variable statement>, [<statement>] [<statement>]) <expression>
	for (<lvalue-expression> in <statement>) <expression>
	for (<variable statement> in <statement>) <expression>

ContinueStatement	continue [<identifier>]
BreakStatement	break [<identifier>]
ReturnStatement	return [<statement>]
WithStatement	with (<statement>) <expression>
LabelledStatement	<identifier> <expression>
SwitchStatement	switch (<statement>) case <statement> [<expression>] [case <statement> [<expression>] ...] [default: [<expression>]]
ThrowStatement	throw <statement>
TryStatement	try <block> catch (<identifier>) <block> try <block> finally <block> try <block> catch (<identifier>) <block> finally <block>
Debugger	debugger

Variable statements

The variables are defined through the keyword *var*. A declared variable is placed within visibility scope that corresponds to the function in which it is declared. If the variable is declared outside of functions, it is placed in the global visibility scope. Variable is created when the function within which it was declared, or, if the variable is global, at the start of the application. When a variable is created in ECMAScript, it takes the value *undefined*. If a variable is created with initialization, the initialization does not occur in the moment of variable creation, it happens when the string with the var statement executes.

Key and reserved words

The following words are the reserved words in the language and can not be used as identifiers:

```
break    do       instanceof typeof
case     else     new        var
catch    finally  return     void
continue for      switch     while
debugger function this       with
default  if       throw
delete   in       try
```

Functions

Functions are objects in ECMAScript. They are created with *Function ()* constructor. Functions like any other objects can be stored in variables, objects and arrays, can be passed as arguments to other functions and can be returned by functions. Functions, like any other objects may have properties. Essential specific feature of functions is that they can be invoked.

There are two types of functions in ECMAScript:

- Internal functions (eg, parseInt),
- Functions, defined in the application text.

Internal functions are built-in objects that are not necessarily implemented in ECMAScript.

In the application text, the denominated function in ECMAScript can be determined by one of the following ways:

```
// function statement
function sum(arg1, arg2) {
    return arg1 + arg2;
}

// defining of function via statement
var sum2 = function(arg1, arg2) {
    return arg1 + arg2;
};

// defining of function using the object form of record
var sum3 = new Function("arg1", "arg2", "return arg1 + arg2;");
```

Syntax Highlighting

Font in the Script window has syntax highlighting. Colours:

- Arbitrary functions - purple,
- XILab functions - blue,
- Positive numbers - green,
- Negative numbers - red,
- Comments - gray,
- All the rest - black.

During the script execution the background of line with the last executed command is changed to dark gray with update rate of 1 time in every 20 ms.

Additional XILab functions

- log(string text [, int loglevel]) – recording to the XILab log
- msleep(int ms) - script execution delay
- newaxis(int serial_number) - creation of new axis object
- new_calibration(int A, int Microstep) - creation of calibration structure to pass to calibrated functions
- get_next_serial(int serial) - get next serial out of an ordered list of opened controller serials
- wait_for_stop(int refresh_period) - waits until controller stops moving
- and all libximc library functions (см. Programming guide)

Record to XILab log

It is executed by log(string text [, int loglevel]) function .
Adds to the XiLab log the *text* line. If the second *loglevel* parameter is passed the message receives the appropriate logging level and is displayed in corresponding color.

Loglevel	Type
1	Error
2	Warning
3	Info

```
example: var x = 5; log("x = " + x);
```

Note: It is not recommended to invoke Xilab interface functions (log record) more than once in 20 ms.

Delay of the script execution

It is executed msleep(int ms) function.
The script makes a pause in the given part of execution for *ms* milliseconds.
example: `msleep(200);`

New axis creation

XILab multi-axis interface also provides the ability to manage controllers via scripts. The differences is that the command is sent to the exactly specified controller. For this, the axis object is introduced, which has methods that match the libximc library function names. Identification is implemented by the controller serial number.

```
var x = newaxis(123);
x.move(50);
```

Here, in the first script line an axis with variable name x corresponding to the controller with the serial number "123" was created. If this controller is not connected to the computer, the script will return an execution error and stop. The second line sends the command to the x-axis to move to the coordinate 50 [steps].

Creation of calibration structure

new_calibration(int A, int Microstep) function takes as a parameter a floating point number A, which sets the ratio of user units to motor steps, and microstep division mode, which was either read earlier from MicrostepMode field of get_engine_settings() return type, or set by a MICROSTEP_MODE_ constant. This function returns calibration_t structure, which should be passed to calibrated get_/set_ functions to get or set values in user units. The following two forms are functionally equivalent:

```
// create calibration: type 1
var calb = new_calibration(c1, c2);

// create calibration: type 2
var calb = new Object();
calb.A = c1;
calb.MicrostepMode = c2;
```

Get next serial

Function get_next_serial(int serial) takes as a parameter an integer number and returns the smallest serial from a sorted list of opened controller serials which is strictly greater than the parameter. If there are no such serials a zero is returned.
This function is a convenient shortcut for automatic creation of "axis" type objects without hardcoded serial numbers.

```
var first_serial = get_next_serial(0);
var x = newaxis(first_axis);
var y = newaxis(get_next_serial(first_serial));
```

In this example in the first line we obtain a serial, in the second line an axis-type object is created, in the third line we get the next serial and create an axis for it.

Wait for stop

Function wait_for_stop(int refresh period) takes as a paramater an integer denoting time delay in milliseconds between successive queries of controller state. This function waits until controller stops movement. It is a shorthand form of the following construct:

```
do { msleep(refresh_period); } while ((get_status().MvCmdSts & MVCMD_RUNNING) != 0);
```

This function is also present as a method of "axis"-type object.
Important: You should set refresh_period to a value greater than jerk free time, if you have "Jerk free" option enabled.

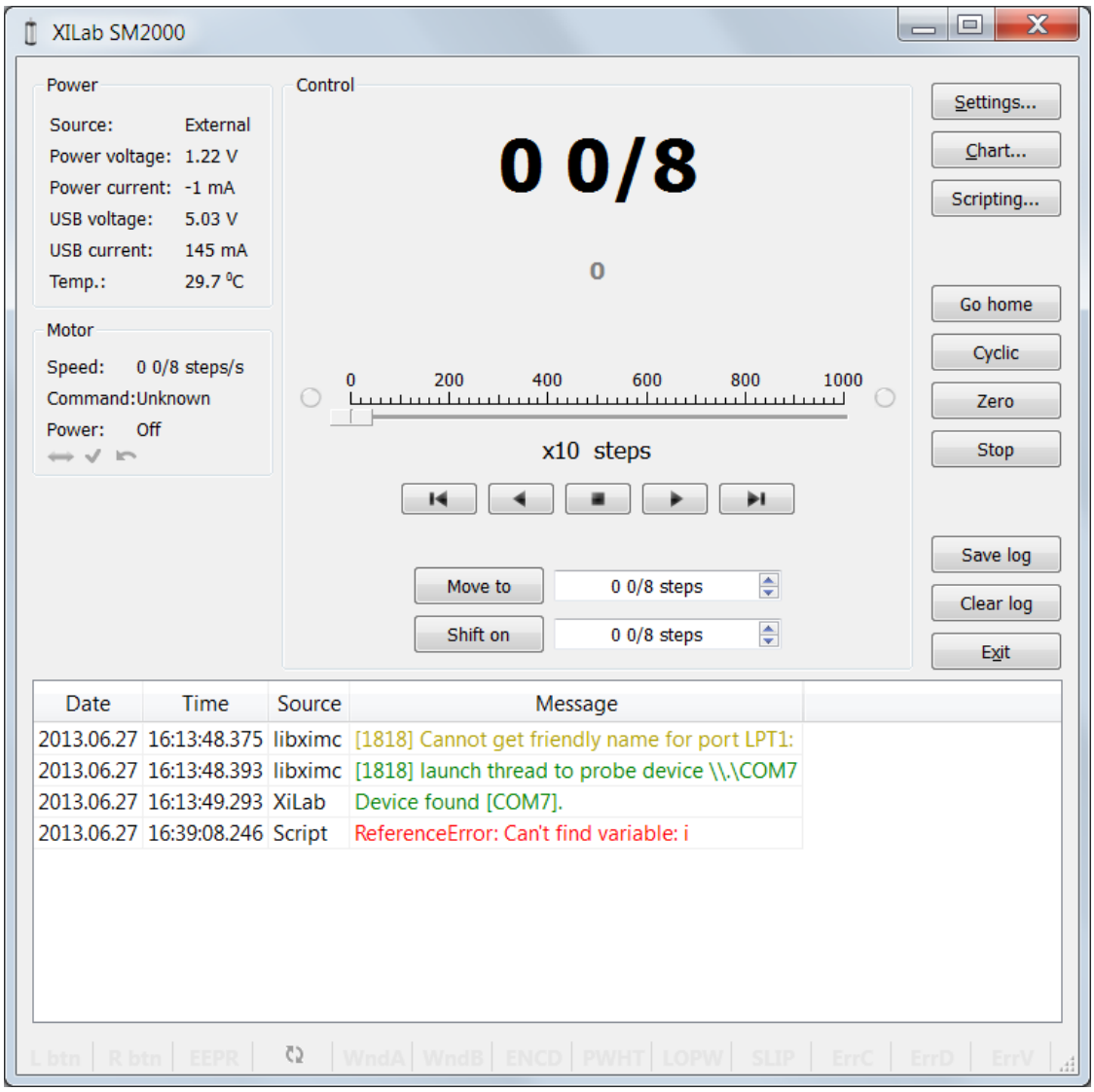
Sample cyclic motion script

```
var first_border = -1000;
var second_border = 1000;
var speed = 300;
var accel = 100;
var decel = 100;

command_stop();
command_zero();
var m = get_move_settings();
m.Speed = speed;
m.Accel = accel;
m.Decel = decel;
set_move_settings(m);
while(1) {
  command_move(first_border);
  msleep(500);
  while ((get_status().MoveSts & MOVE_STATE_MOVING) != 0) { msleep(100); }

  command_move(second_border);
  msleep(500);
  while ((get_status().MoveSts & MOVE_STATE_MOVING) != 0) { msleep(100); }
}
```

Sample script with calibrated functions

```
var sn1 = get_next_serial(0);
var sn2 = get_next_serial(sn1);
var x = newaxis(sn1);
var y = newaxis(sn2);
x.command_zero();
y.command_zero();
var A = 0.4;
var calb = new_calibration(A, x.get_engine_settings().MicrostepMode);
x.command_move_calb(100, calb);
x.wait_for_stop();
x.command_move(0);

calb.A = 0.0001;
calb.MicrostepMode = y.get_engine_settings().MicrostepMode;
var ymsc = y.get_move_settings_calb(calb);
ymsc.Speed = 0.02;
y.set_move_settings_calb(ymsc, calb);
y.command_move(300);
y.wait_for_stop();
y.command_move(0);
```

5.2.7. XILab log



XILab log window

XILab log at the bottom part of the main window shows libximc library messages. It also shows messages from XILab application and Scripts interpreter.

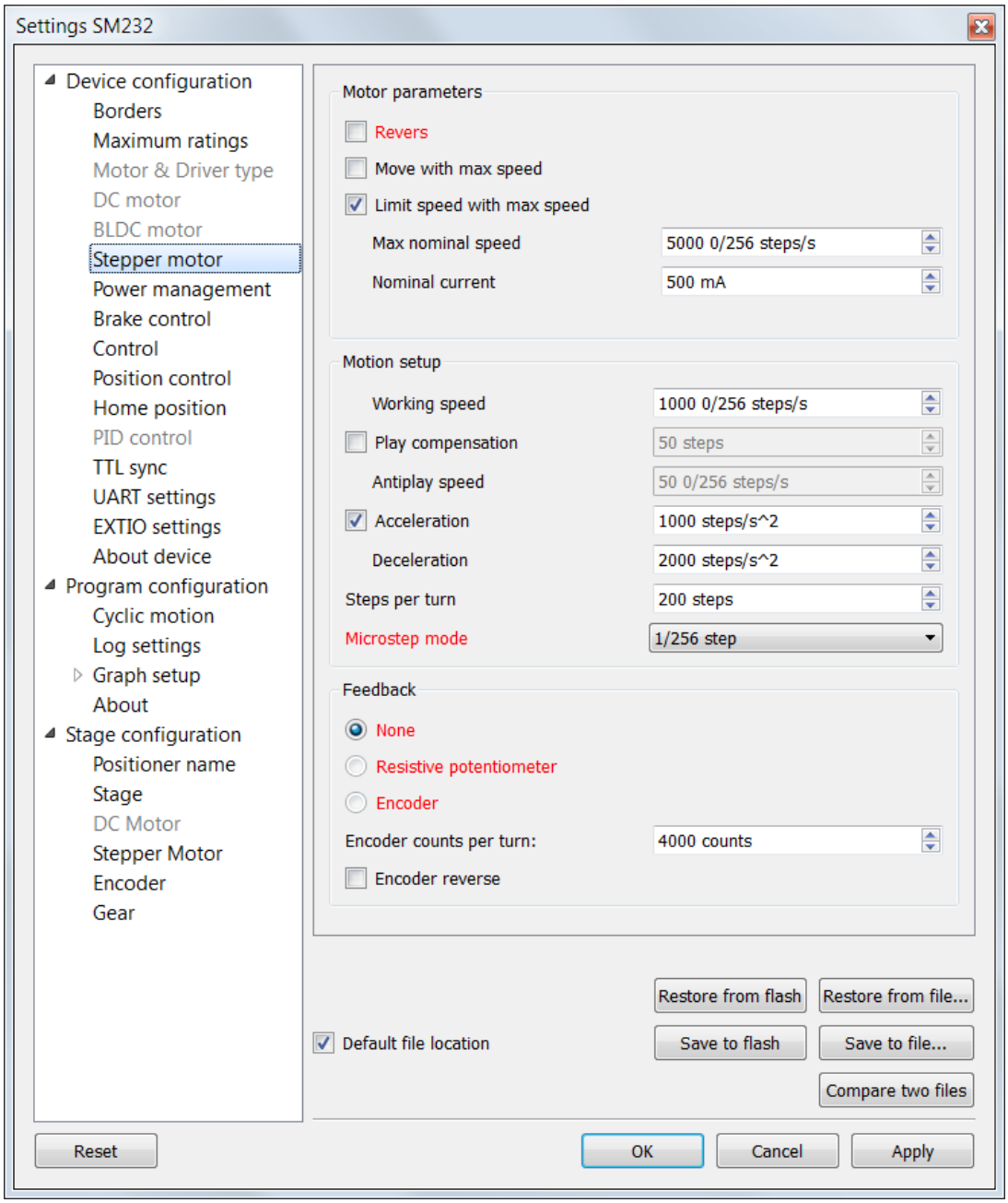
Log has 4 columns: date and time of record, the source and the message text.
Messages have a logging level indicating message importance: error, warning and informational message. Error messages are red, warnings are yellow and informational messages are green.
You can set a filter on displayed messages on the Log settings page in the Settings window.

5.3. Controller Settings

1. Settings of kinematics (Stepper motor)
2. Motion range and limit switches
3. Critical board ratings
4. Power consumption settings
5. Home position settings
6. Synchronization settings
7. Brake settings
8. Position control
9. Settings of external control devices
10. UART settings
11. General purpose input-output settings
12. Motor type settings
13. Settings of kinematics (DC motor)
14. Settings of PID control loops
15. About controller

5.3.1. Settings of kinematics (stepper motor)

In the Application Settings ***Device configuration -> Stepper motor***



Settings of stepper motor kinematics window

Motor parameters - directly related to the electric motor settings

Revers - checking this flag associate the motor rotation direction with the position counting direction. Change the flag if positive motor rotation decreases the value on the position counter. This flag effect is similar to the motor winding reverse polarity.

Move with max speed - if this flag is checked motor ignores the preset speed and rotates at the maximum speed limit.

Limit speed with max speed - if this flag is checked the controller limits maximum speed to the value specified in the Max nominal speed field. For example, if the speed exceeds the rated value, controller will reduce output action, until the speed come back to the normal range. However, the controller remains operational and will continue the current task.

Max nominal speed - motor rated speed.

Nominal current - motor rated current. The controller will limit the current with this value.

Motion setup - movement kinematics settings

Working speed - movement speed.

Play compensation - backlash compensation. Since the positioner mechanics are not ideal there is a difference between approaching a given point from the right and from the left. When the backlash compensation mode is on the positioner always approaches the point from one side. The preset value determines the number of steps which the positioner takes to pass a given point in order to come back to it from the same side. If the specified number is above zero the positioner always approaches the point from the right. If it is below zero the positioner always approaches the point from the left.

Antiplay speed - speed of backlash compensation. When the backlash compensation mode Play compensation is on the positioner approaches the point from the right or from the left with a preset speed determined in the number of steps per second.

Acceleration - enables the motion in acceleration mode, the numerical value of the field is the acceleration of movement.

Deceleration - movement deceleration.

Steps per turn - determines the number of steps for one complete motor revolution. The parameter is set by user.

Microstep mode - step division mode. 8 modes are available: from a whole step to 1/256 of step. Description of modes is in the Supported motor types.

Feedback settings

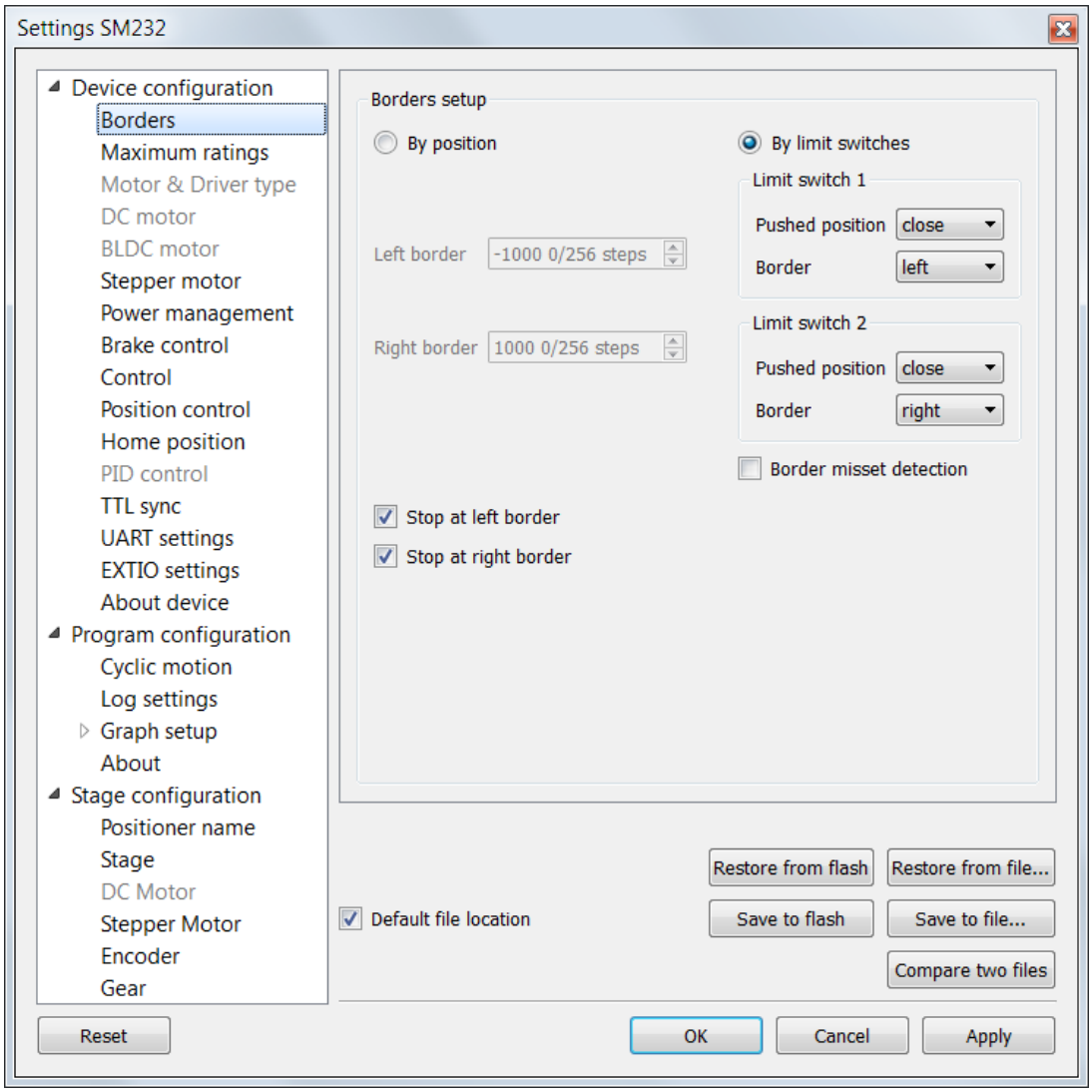
An encoder or potentiometer can be used as feedback sensor for stepper motors.

Counts per turn - this parameter defines the number of encoder impulses per one full motor axis revolution.

Encoder reverse - encoder reverse.

5.3.2. Motion range and limit switches

In the Application Settings ***Device configuration -> Borders***



Motion range and limit switches settings window

Borders setup parameter group contains borders and limit switches parameters. These parameters are used to keep the positioner in the permissible physical movement limits or motion range limit in accordance with the user requirements. Borders can be set either by feedback signal from (encoder or potentiometer) or by limit switches that setup in the positioner terminal points.

To set the borders by encoder (potentiometer) select the By encoder/potentiometer and specify the Left border and Right border values, which correspond to the left and right edge respectively.

To set the borders by the limit switches select By limit switches and set up both Limit switch 1 and Limit switch 2.
Pushed position - sets the limit switch condition when it is reached: open or closed.
Border - sets the limit switch position: on the left or on the right of the positioner working range.

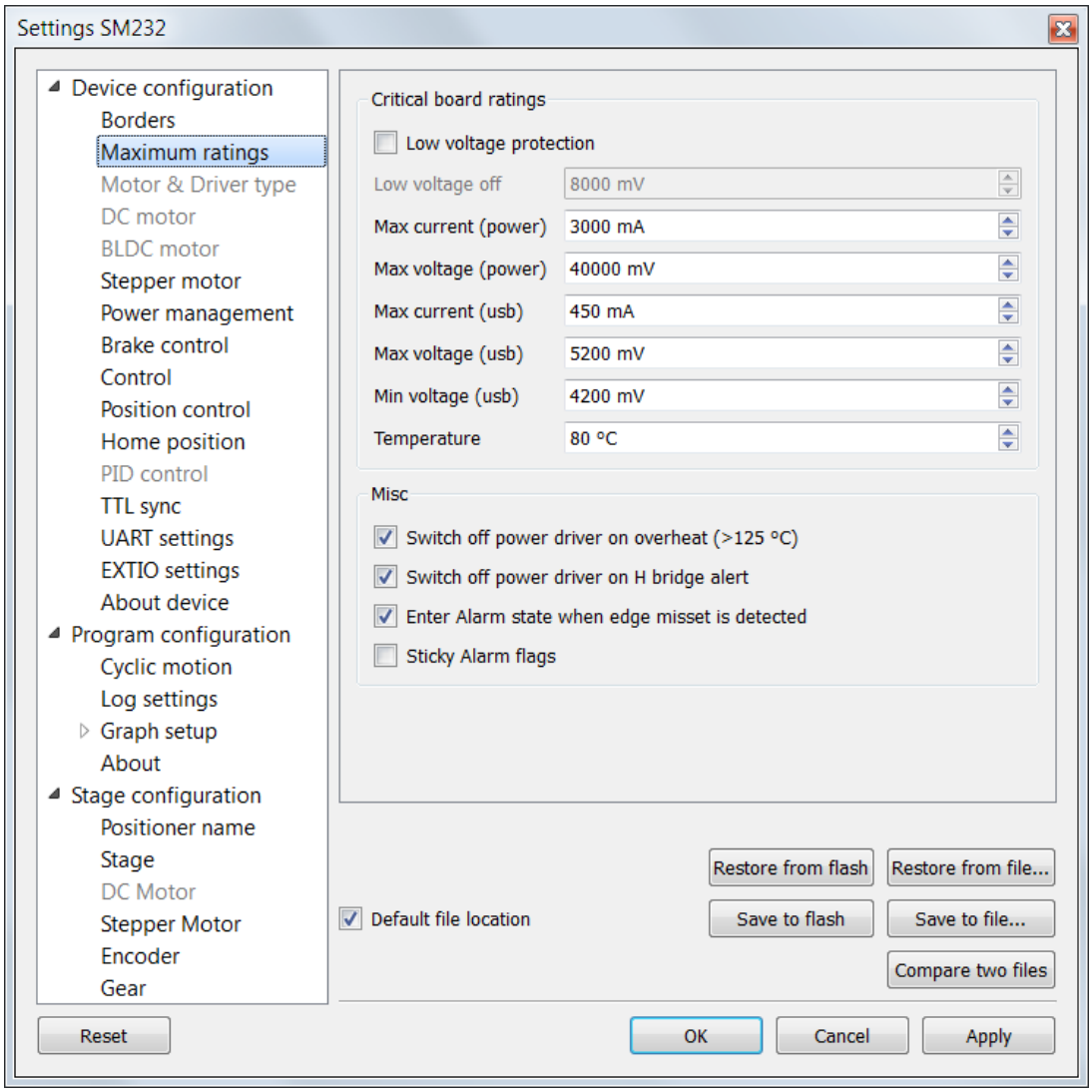
Check the Stop at left border and / or Stop at right border for a forced stop of motor when the border is reached. In this case the controller will ignore any commands of movement towards the limit switch if the corresponding limit switch has already been reached.

When the border position is reached the corresponding indicator flashes in the main application window.

If the Border misset detection flag is checked, the engine stops upon reaching of each border. This setting is required to prevent engine damage if limit switches appear to be potentially incorrectly configured. Read more about controller operation in this mode in the limit switches location on positioners.

5.3.3. Critical board ratings

In the Application Settings ***Device configuration -> Maximum ratings***



Controller critical parameters settings window

Critical board ratings - This group is responsible for the maximum values of input current Max current (power) and voltage Max voltage (power) on the controller, the maximum current Max current (usb) and voltage Max voltage (usb) for USB, the minimum voltage Min voltage (usb) for USB, and the board temperature Temperature (if the temperature is measured on the given controller version).

If the controller current consumption value, power supply voltage or the temperature exceeds the value defined here the controller turns off all power outlets and switches to Alarm state. At the same time the Alarm state information will appear in the main window (window background will change to red) and the exceeded parameter value will be displayed in blue or red (below or above the limit respectively).

If the Low voltage protection flag is checked the low voltage supply protection is active. Then Low voltage off and below values of input voltage turns the controller into Alarm state.

The group **Misc** includes all other critical parameters settings.

Switch off power driver on overheat (> 125 ° C) - this parameter enables the Alarm state in case of power driver overheat.

Switch off power driver on H bridge alert - this parameter enables the Alarm state in case of the power driver malfunction signal.

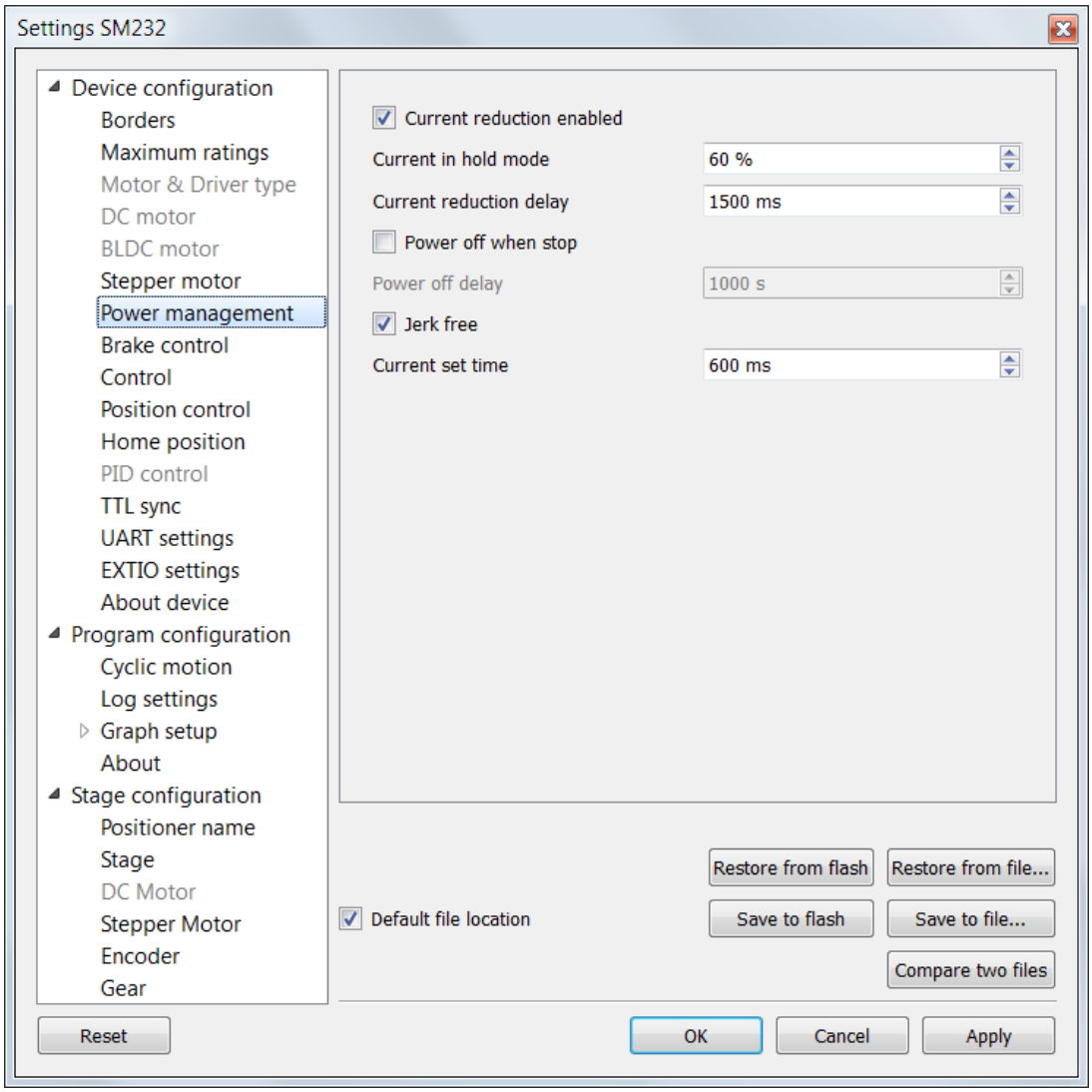
Enter Alarm state when edge misset is detected - this parameter enables the Alarm state in case of incorrect boundary detection (activation of right limit switch upon moving to the left, or vice versa).

Sticky Alarm flags - locking of Alarm condition. If the Sticky Alarm flags check-box is unchecked the controller removes the Alarm flag as soon as it its cause is removed (e.g. in case of over-current the windings are disconnected, which

results in the decreased current). If the Sticky Alarm flags is enabled, the Alarm mode cause and the Alarm mode are canceled by the Stop command only.

5.3.4. Power consumption settings

In the Application Settings ***Device configuration -> Power Management***



Power consumption settings window

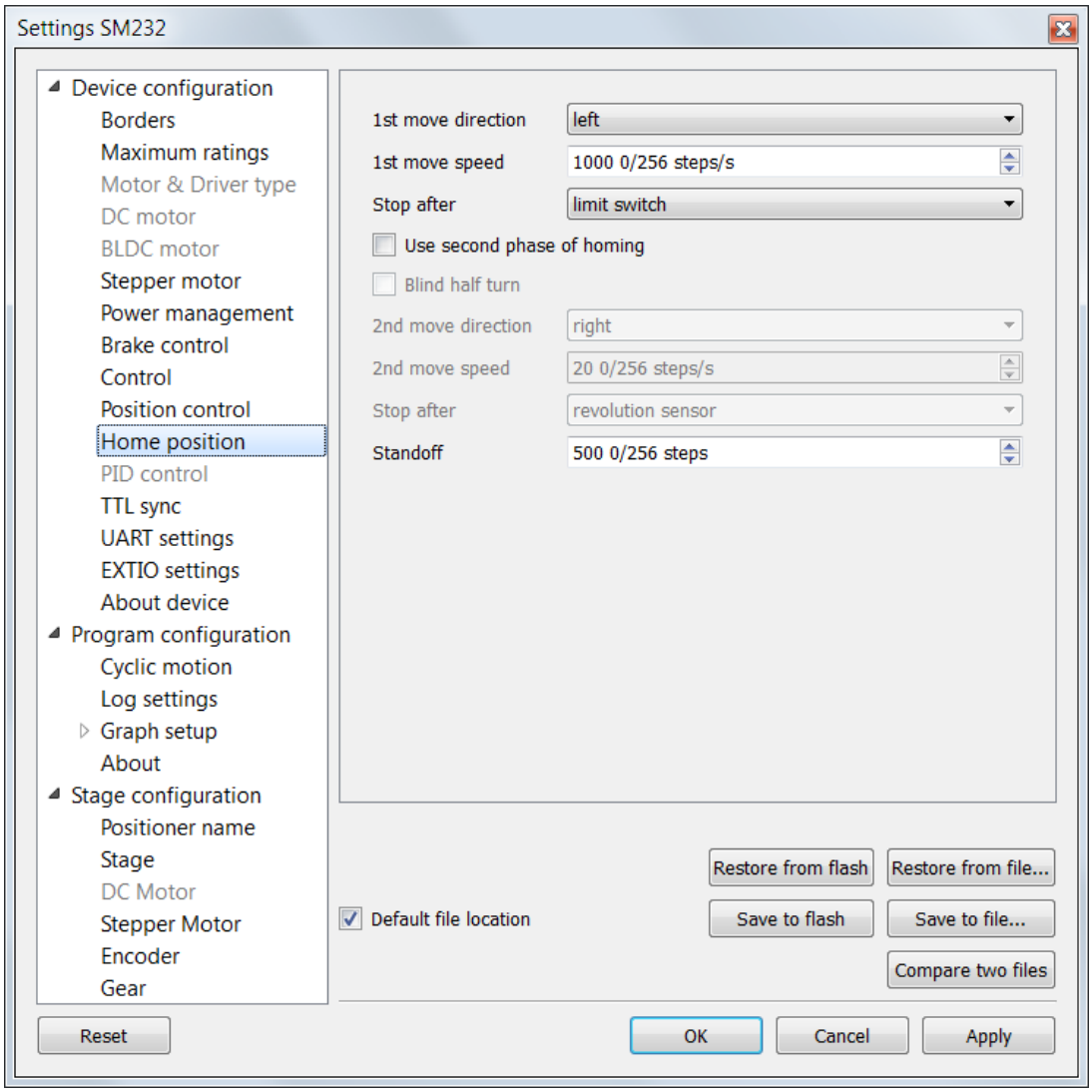
Current reduction enabled - activates the reduced energy consumption mode.

- Current in hold mode - it determines the current in the hold mode in % of the nominal value. Value range: 0 .. 100%.
- Current reduction delay - parameter determines the delay between switching to the STOP mode and power reduction activation. It is measured in milliseconds. Value range: 0 .. 65535 ms.
- Power off when stop - it activates the function that deenergized the motor windings after switching to the STOP state.
- Power off delay - parameter determines the delay in seconds between switching to the STOP mode and motor power-off. Value range: 0 .. 65535.
- Jerk free - activates the current smoothing function to eliminate the motor vibration.
- Current set time - parameter determines the time for jerk free current setting in milliseconds. Value range: 0 .. 65535 ms.

Detailed description of these parameters see in the Power control section.

5.3.5. Home position settings

In the Application Settings ***Device configuration -> Home position***



Home position settings window

Tab ***Home position*** sets the home position calibration parameters. Calibration comes to automatic accurate detection of the limit switch, the revolution sensor signal or moment of getting the external signal, which determines the zero position, and grading from it by a specified offset. It is helpful when the current location of the positioner is unknown but we know the base point position relative to limit switch or some other signal, which is called home position.

Homing comes to the motor movement towards the predetermined 1st move direction (left or right) at 1st move speed. The speed is usually set low enough to keep the next signal. Motor stops, depending on the stop after value, upon getting a sync input signal, signal from RPM sensor or reaching of the limit switch.

If the flag Use second phase of homing is checked the homing algorithm rotates the motor in a predetermined 2nd move direction (left or right) with the 2nd move speed.

If the flag Blind half turn is checked the motor ignores the end of the second phase signal during half a turn. It helps to set an unambiguous detection order for the sensors that have the first and the second movement phases signals close each other.

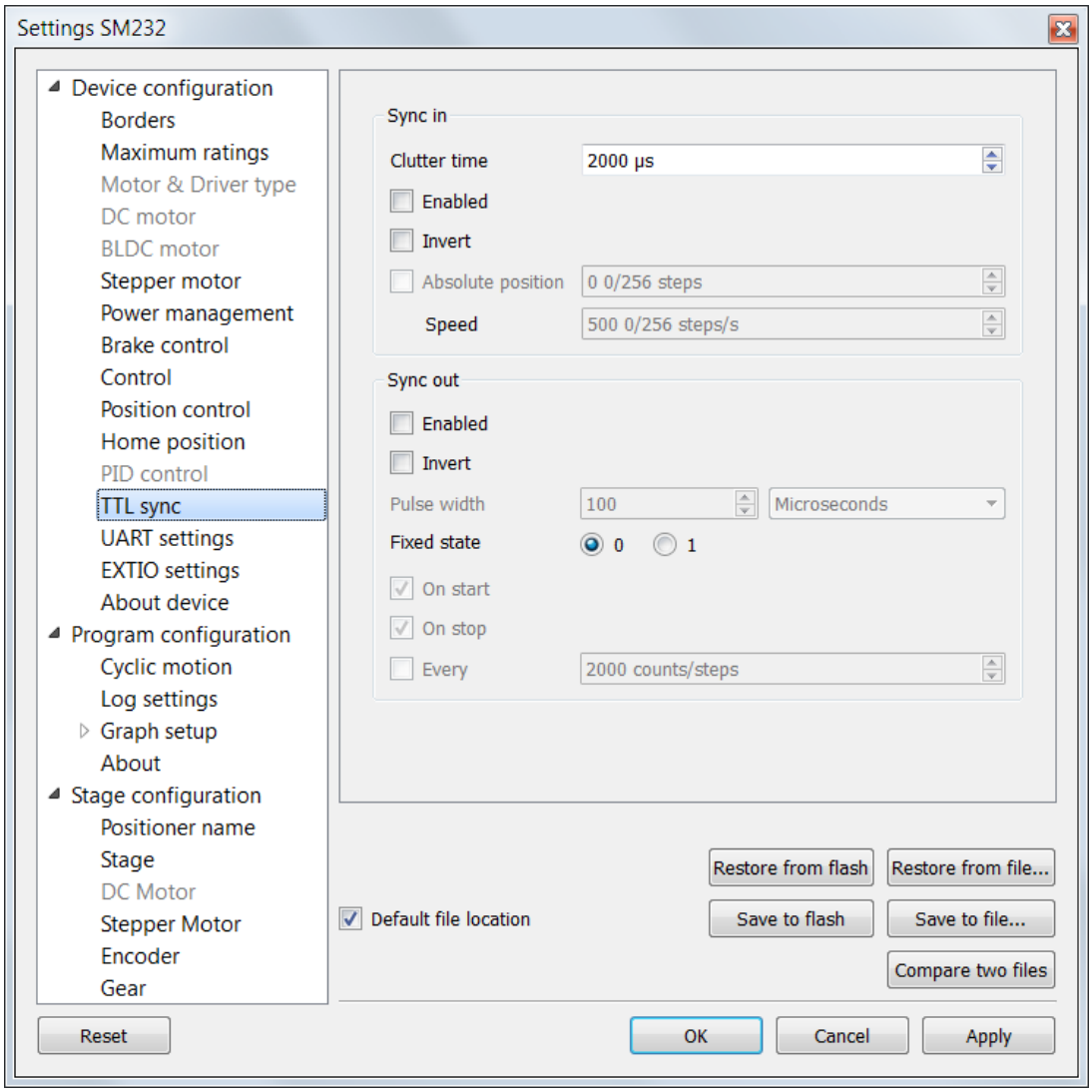
The third phase of homing is the absolute offset to the distance standoff.

The resulting point is called a home position. It is important that its location on the positioner does not depend on the calibration starting point.

Configuration commands are described in the Communication protocol specification section.

5.3.6. Synchronization settings

In the Application Settings ***Device configuration -> TTL sync***



Synchronization settings window

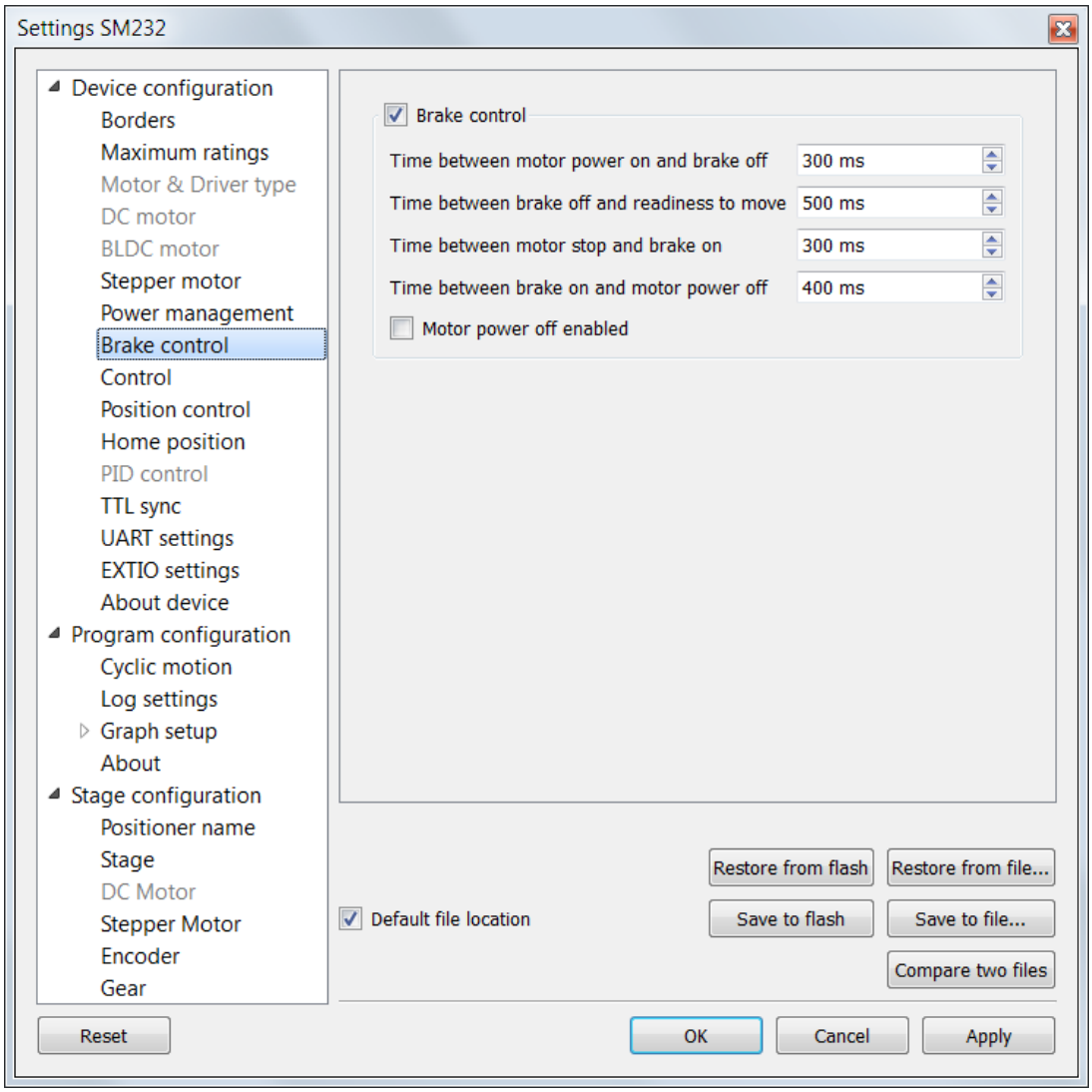
Synchronization is described in details in TTL synchronization section.

Sync in
Clutter time - setting minimum synchronization pulse duration (in microseconds). Defines the minimum duration, which can be detected (anti-chatter).
Enabled - check this box for the sync in mode enable.
Invert - checked flag shows that the operation is triggered by the falling sync pulse edge.
Absolute position - if the flag is checked, upon sync pulse the positioner moves into the absolute position specified in the field Step/Micro step. If the flag is unchecked, the shift is relative to the defined destination position.

Sync out
Enabled - if the flag is checked, the sync output functions according to the next settings. If the flag is unchecked, the output value is fixed and equal to the Fixed state.
Pulse width - specifies the duration of the output signal in milliseconds or steps/encoder pulses.
Fixed state - sets the logic level of output to 0 or 1, respectively.
Invert - if the flag is checked the zero logic level is set to active.
On start - synchronizing pulse is generated at the beginning of movement.
On stop - synchronizing pulse is generated at the end of movement.
Every - the pulse is generated every n encoder pulses.

5.3.7. Brake settings

In the Application Settings ***Device configuration -> Brake control***



Magnetic brake settings window

Check the Brake control flag to enable magnetic brake.

Parameters:

Time between motor power on and brake off - time between switching the motor on and switching off the brake (ms).

Time between brake off and readiness to move - time between switching off the brake and motion readiness (ms). All motion commands will be executed only after this time.

Time between motor stop and brake on - time between stopping the motor and turning on the brake(ms).

Time between brake on and motor power off - time between turning on the brake and motor power-off (ms).

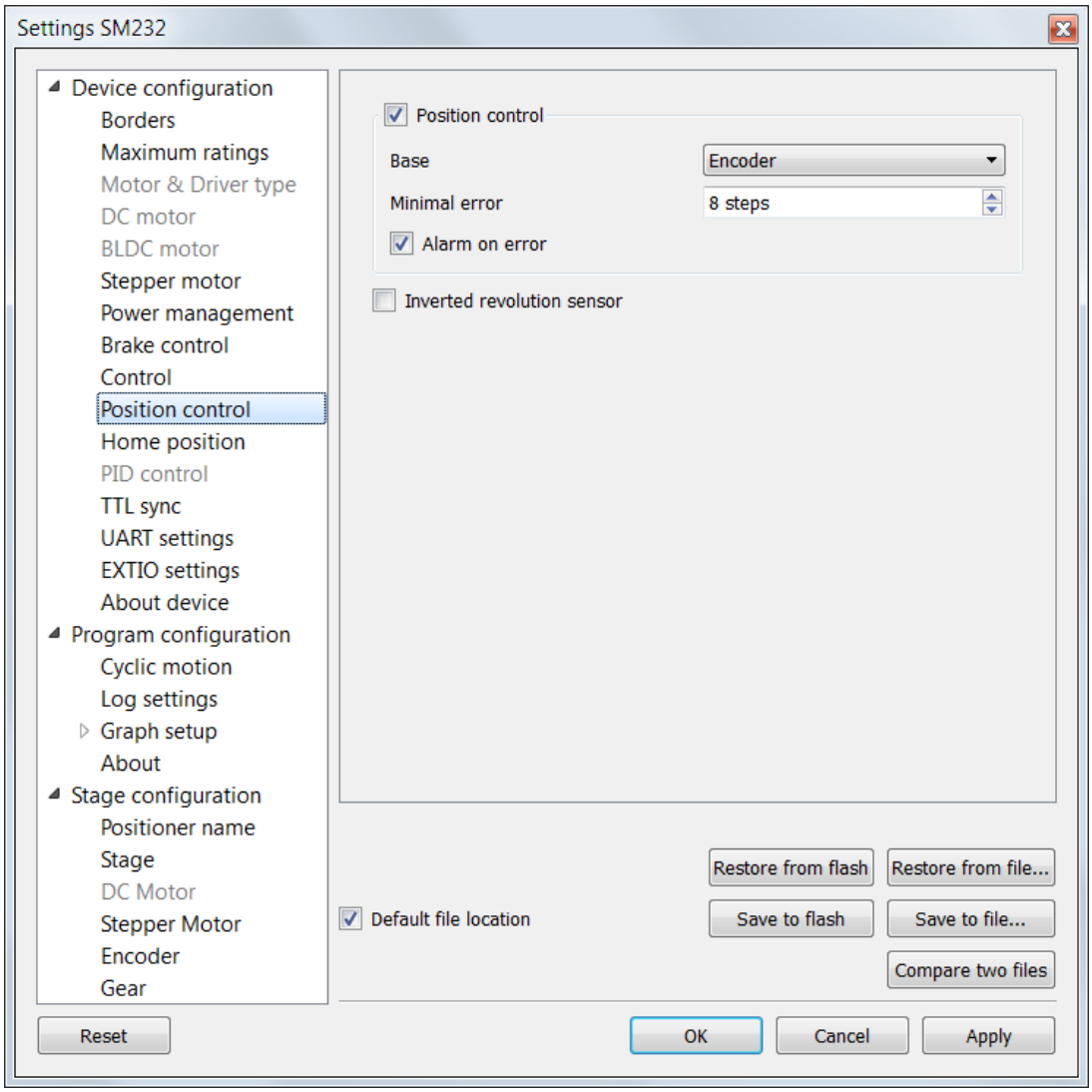
Value range is from 0 to 65535 ms.

Motor power off enabled flag means that when magnetic brake is powered off, the brake turns off motor power supply.

Configuration commands are described in Communication protocol specification.

5.3.8. Position control

In the Application Settings ***Device configuration -> Position control***



Window of Position control

Check the Position control parameter flag to activate the position control.

Base - selection of the position control device. Select in the drop-down list: encoder (*Encoder*) (see Operation with encoders) or positioner (*Revolution sensor*) (see Operations with potentiometer feedback).

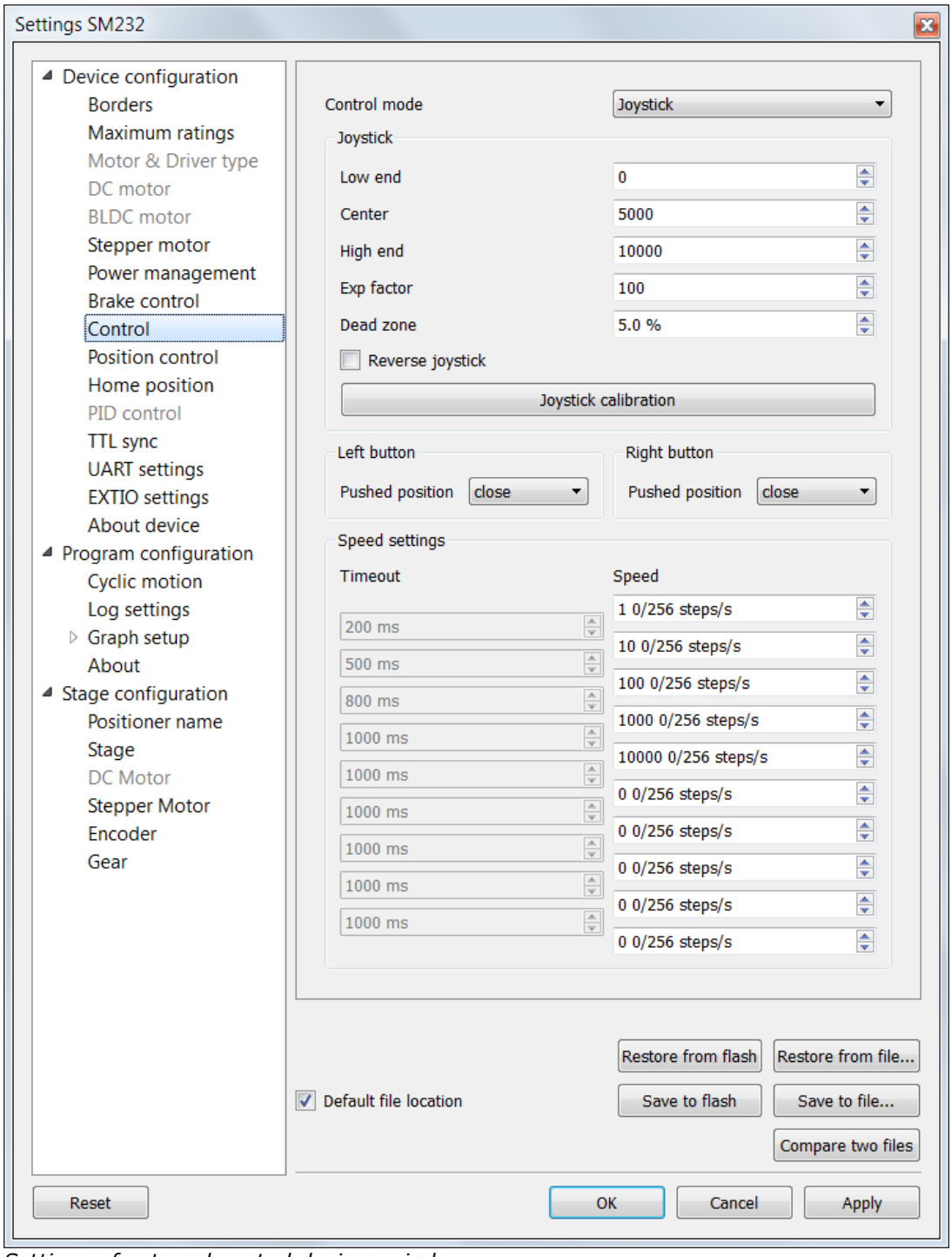
Minimal error - determines the number of lost steps (0 .. 255), which is considered to be erroneous. If the amount of the loss steps exceeds the specified number of steps the error flag appears and the movement can be stopped depending on the Stop on error flag.

Inverted revolution sensor - if the flag is checked the revolution sensor is triggered by the level 1. Unchecked flag means usual logic is valid - 0 is the trigger/activation/active state.

Configuration commands are described in the Communication protocol specification.

5.3.9. Settings of external control devices

In the Application Settings ***Device configuration -> Control***



Settings of external control devices window

Control mode - range of external motor control devices.

- Control disabled - external devices are not used
- Joystick - joystick is used
- Buttons - buttons are used

Low end, Center and High end determine the lower border, the middle and the upper border of joystick range respectively. Hence the joystick ADC normalized value equal to or less than Low end corresponds to the maximum joystick deflection towards lower values.

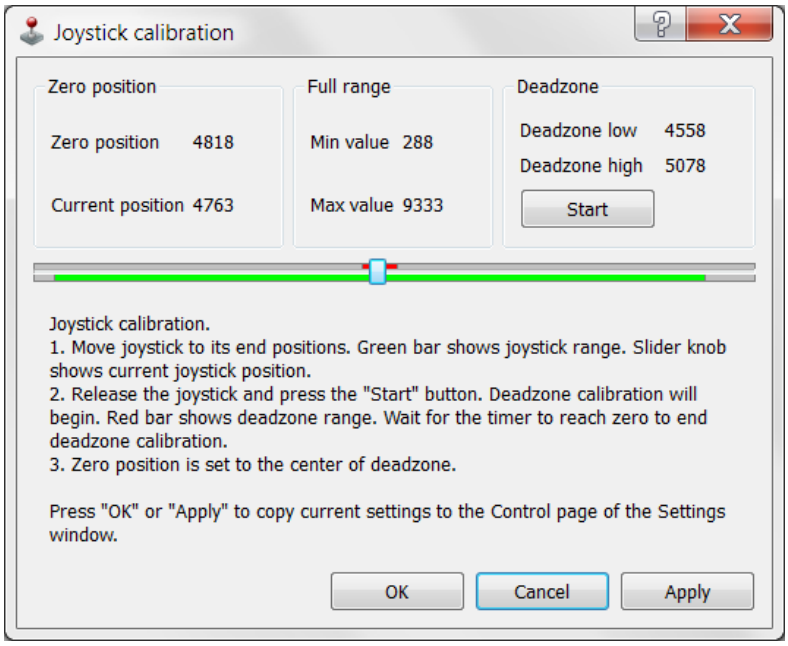
Exp factor - exponential nonlinearity parameter. See Joystick control.

Dead zone - dead zone of joystick deviation from the center position. Minimum step of variation: 0.1%, the maximum

value is 25.5%. The joystick deviation from Center position by less than Dead zone value corresponds to zero speed.

Reverse joystick - Reverse the joystick effects. Joystick deviation to large values results in negative speed and vice versa.

Button Joystick calibration opens calibration dialog box.



Dialog box of Joystick Calibration

Calibration is automatic border and the dead zone detection. Below is the process description:
Move the joystick to extremes to determine the borders. The range of all measured values is represented in a green line.

Release the joystick and press the Start to initiate detection of the dead zone. Within 5 seconds imitate accidental influences on the joystick, which should not be recognized as deviation from the joystick zero position. The dead zone range is represented in red.

Pressing the Apply button will send the computed values into the Settings window. Pressing OK button will send the values and close the calibration dialog box.

Pushed Position - determines the state (with pressed or released button) when signal will be sent to the motion controller.

- Open - released button is considered to be a motion command.
- Close - released button is considered to be a motion command.

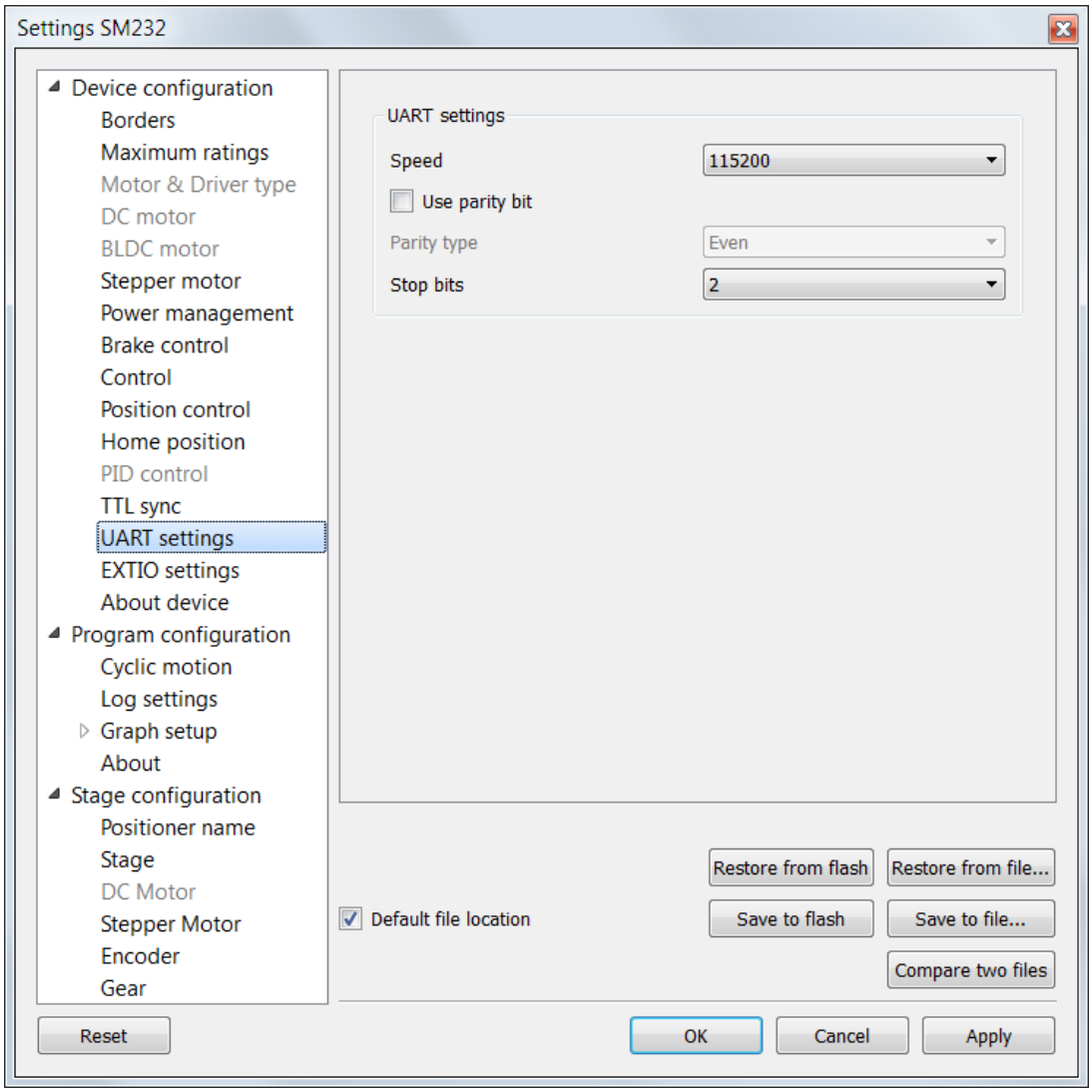
Timeout [i] - the time after which the speed switches from Speed[i] to Speed[i+1]. If any of the Timeout[i] is equal to zero, no switching to the next speeds will occur.

Speed[i] - speed of the motor after time equal to Timeout[i-1]. If any of the speeds is equal to zero, no switching to this and subsequent speeds will occur.

Configuration commands are described in Communication protocol specification.

5.3.10. UART Settings

In the Application Settings ***Device configuration -> UART***



UART settings tab

Speed - Select the UART speed from the preset values in the range from 9600 bit/s to 921600 kbit/s.

Use parity bit - use of parity bit.

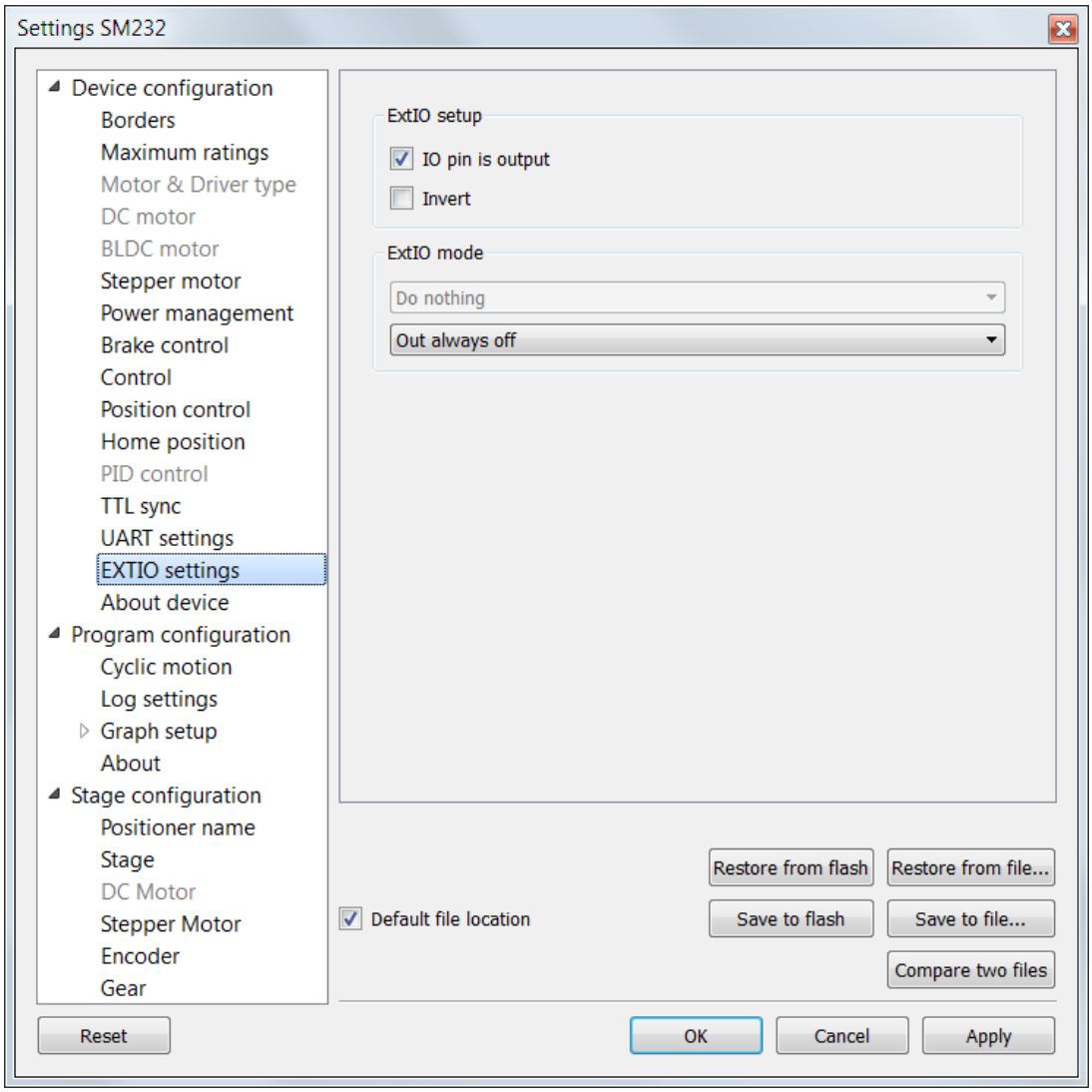
Parity type - Types of parity:

- Even - this bit type is set when total number of bits is odd
- Odd - this bit type is set when total number of bits is even
- Space - parity bit is always 0
- Mark - parity bit is always 1

Stop bits - number of stop bits (1 or 2).

5.3.11. General purpose input-output settings

In the Application Settings ***Device configuration -> EXTIO settings***



General purpose input-output settings tab

For detailed information, please see General purpose digital input-output.

ExtIO setup
IO pin is output - if the flag is checked the needle of ExtIO works in output mode, otherwise - in the input mode.
Invert - if the flag is checked the rising edge is ignored and the falling edge is active.

ExtIO mode - mode selection
If ExtIO configured for input mode the choice of controller action settings by the input pulse is active:

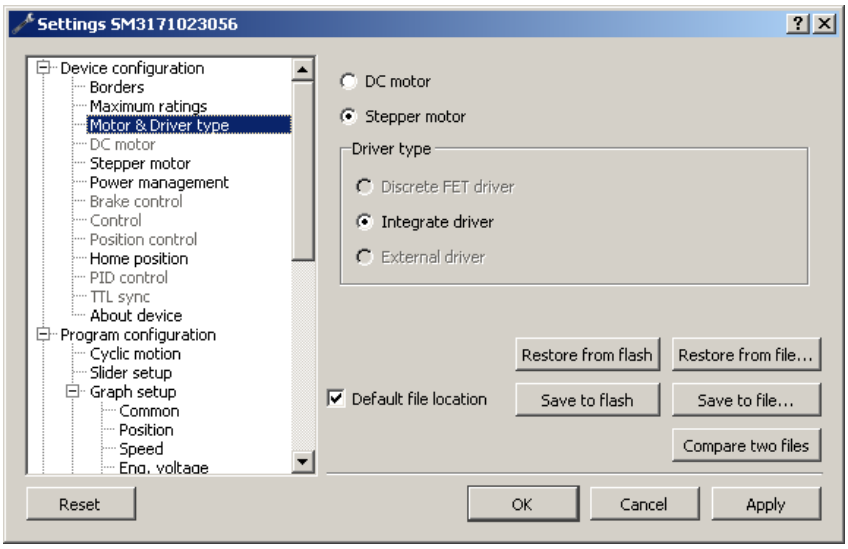
- Do nothing - do nothing.
- Stop on input - run STOP command.
- Power off on input - run PWOF.
- Movr on input - run MOVR.
- Home on input - run HOME.

If ExtIO configured for output mode the choice of the output state depending on the controller status is active:

- Out always off - always in inactive state.
- Out always on - always in active state.
- Out active when moving - in active state during motion.
- Out active in Alarm - in active state if the controller is in the Alarm state.
- Out active when motor is on - in active state if the motor windings are powered.
- Out active when motor is found - in active state if the motor is connected.

5.3.12. Motor type settings

In the Application Settings ***Device configuration -> Motor type***



Motor type settings window

Stepper motor or DC-motor - motor type indication. Control power driver should be selected as well:

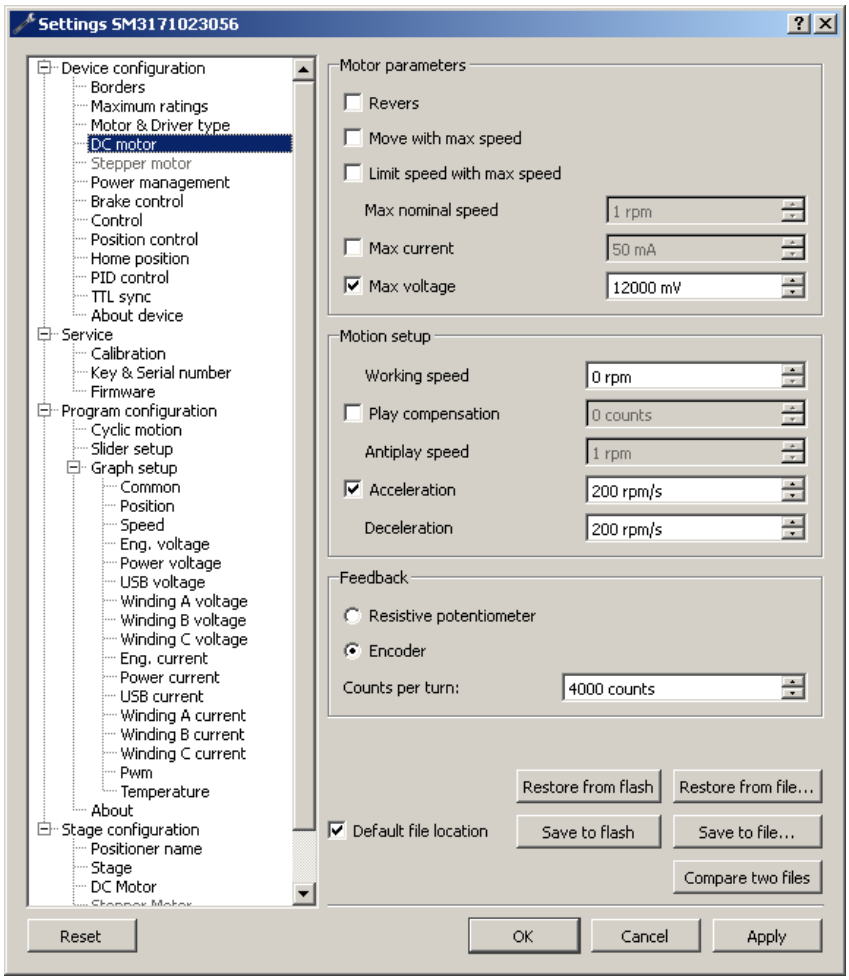
- Integrated. This type is used for this controller modification.
- On discrete keys. Will be used in future versions.
- External driver. Designed to control stepper motors using three standard signals (see External driver control interface)

Warning. Driver type or motor type changing is a critical operation that can not be performed while motor rotates. To implement the change correctly the motor winding should be de-energized and turned off, after that motor type can be changed and motor of another type can be connected. The same applies to changing of integrated driver to external one and vice versa.

Note. Available motor types are determined by your firmware upgrade. Available control drivers depend on the controller board type, except for the external driver.

5.3.13. Settings of kinematics (DC motor)

In the Application Settings ***Device configuration -> DC Motor***



Settings of kinematics (DC motor) window

Motor parameters - electric motor settings

Revers - checking this flag associate the motor rotation direction with the current position counting direction. Change the status of the flag if positive motor rotation decreases the value on the position counter register. This flag effect is similar to connecting the motor winding to reverse polarity.

Move with max speed - if this flag is checked motor ignores the preset speed and rotates at the maximum speed limit.

Limit speed with max speed - if this flag is checked the controller limits the maximum speed to the number of steps per second, specified in the Max nominal speed field.

Max nominal speed, Max voltage, Max current - are motor nominal parameters. If they are active and applicable for given type of motor, the controller limits these parameters within the specified values. For example, if the motor speed and voltage exceeds the nominal values, the controller will reduce output action until both values are within the normal range. However, the controller remains in operational condition, and will execute the current task.

Motion setup - settings related to the movement kinematics

Working speed - speed of motion.

Working speed - movement speed.

Play compensation - backlash compensation. Since the positioner mechanics are not ideal there is a difference between approaching a given point from the right and from the left. When the backlash compensation mode is on the positioner always approaches the point from one side. The preset value determines the number of steps which the positioner takes to pass a given point in order to come back to it from the same side. If the specified number is above zero the positioner always approaches the point from the right. If it is below zero the positioner always approaches the point from the left.

Antiplay speed - speed of backlash compensation. When the backlash compensation mode Play compensation is on the positioner approaches the point from the right or from the left with a preset speed determined in the number of steps per second.

Acceleration - enables the motion in acceleration mode, the numerical value of the field is the acceleration of movement.

Deceleration - movement deceleration.

Feedback settings

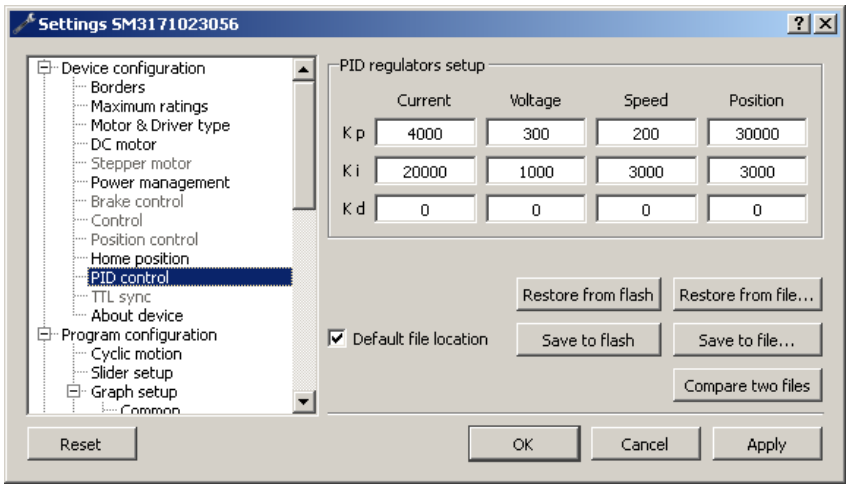
Resistive potentiometer - if potentiometer is used as a feedback sensor, the angle of rotation or offset is displayed in the current position, and the full range of offsets is represented by the dimensionless data from 0 to 1023.

Encoder - use of encoder as a feedback sensor.

Counts per turn - this parameter defines the number of encoder pulses per one motor axis full rotation.

5.3.14. Settings of PID control loops

In the Application settings ***Device configuration -> PID control***



Settings of PID control loops window

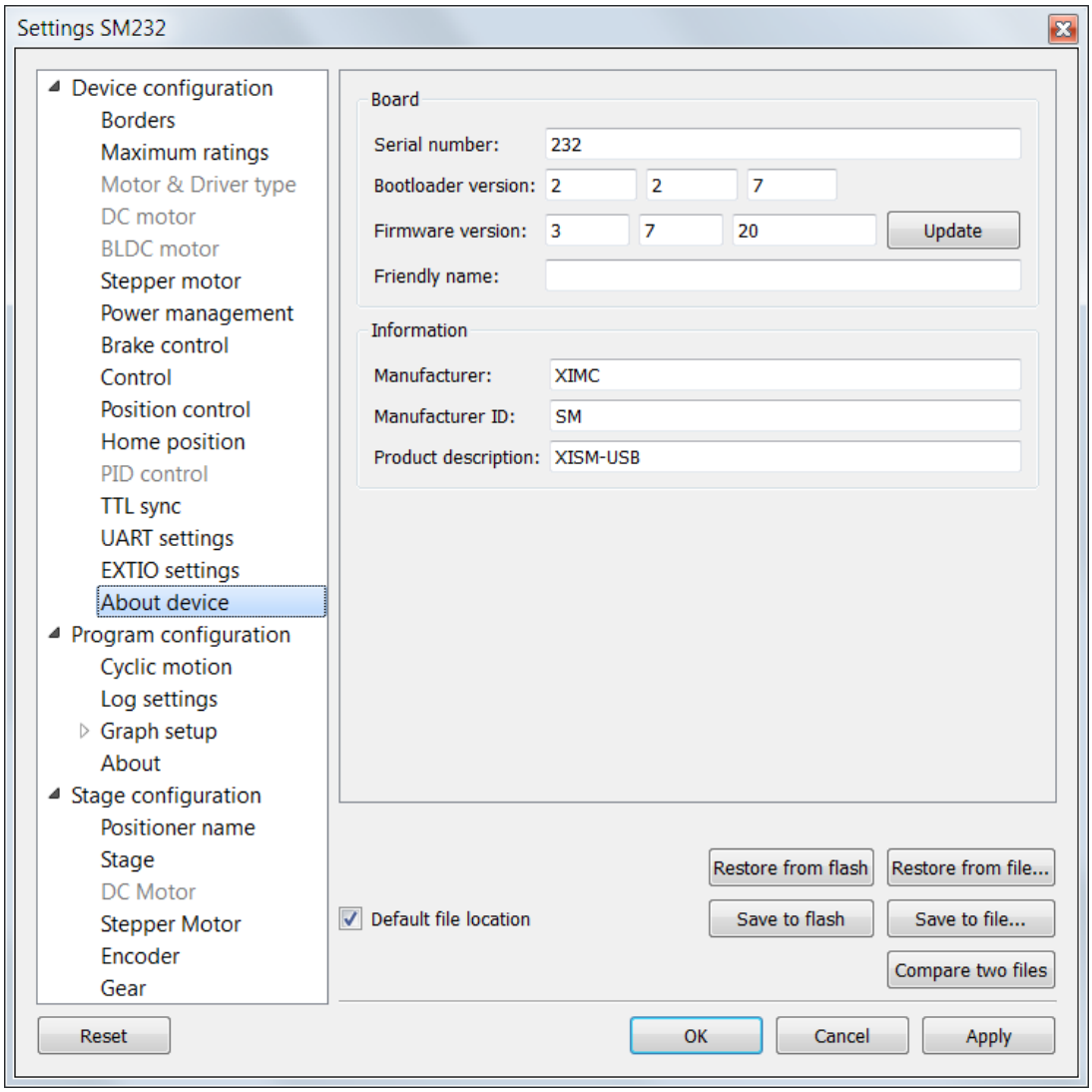
In this section, you can change the PID controllers coefficients.
A total of four controlers are in operation: for voltage, for current, for speed and for coordinate.
Each controller has 3 coefficients, which can vary in the range of 0 .. 65535.

Warning. Do not change the settings of PID controllers, unless you are sure you know what you are doing!

Configuration commands are described in the Communication protocol specification.

5.3.15. About controller

In the Application Settings ***Device configuration -> About device***



About device tab

The **Board** section displays information about the controller:

Serial number - device serial number.

Bootloader version - boot loader version.

Firmware version - firmware version.

Update button opens firmware update dialog box.
Select the firmware file with the extension .cod and click Open. XILab will start the firmware update and will display "Please wait while firmware is updating". Do not power off the controller during the upgrade. Upon completion of the update the "Firmware updated successfully" dialog will be displayed.

Information block contains information about the device: the manufacturer, device ID, device type. The data are read from the internal memory of the controller.

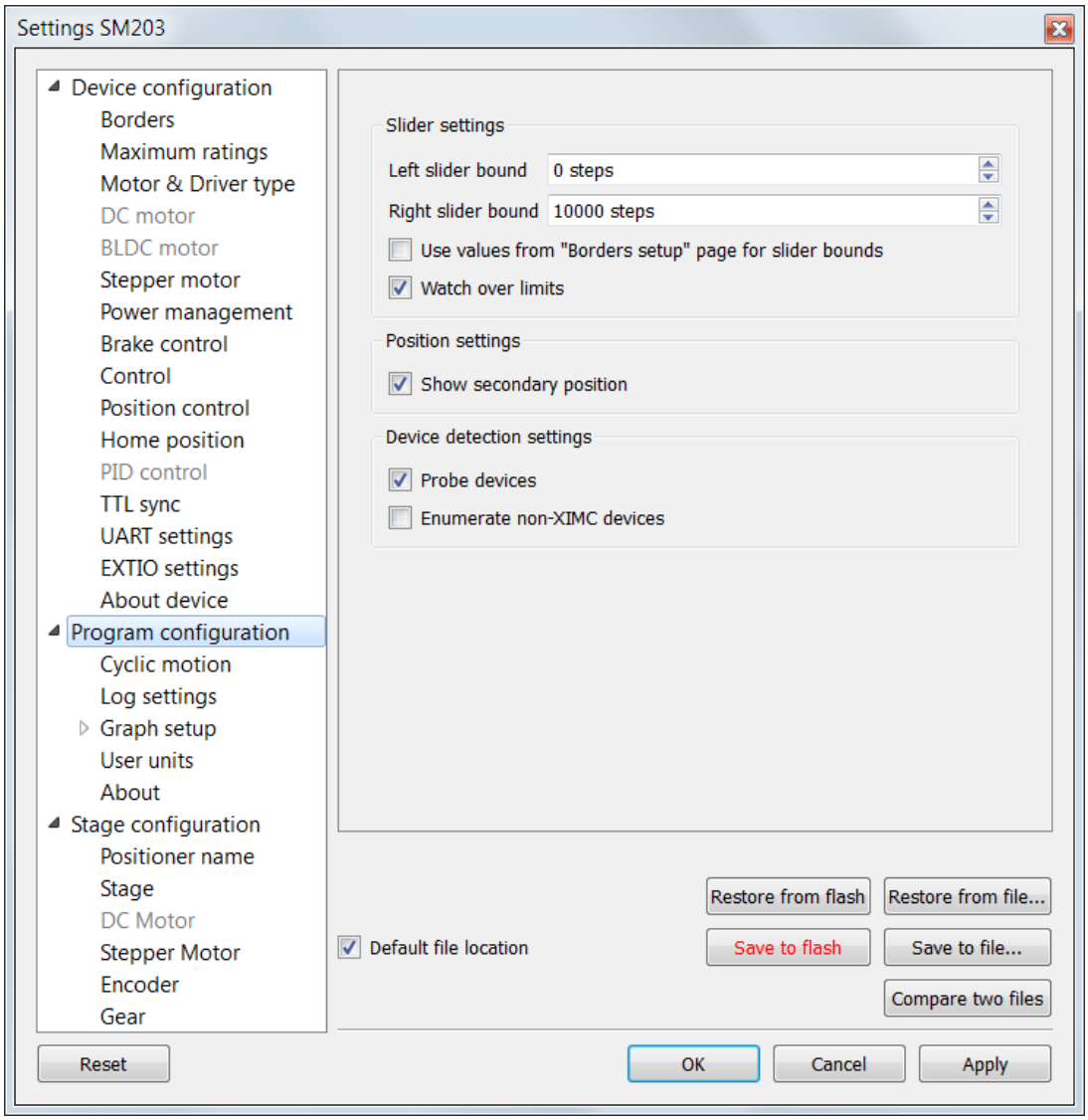
All of the data are reported to XILab application when the device is connected.

5.4. XILab application settings

1. XILab general settings
2. Cyclical motion settings
3. Log settings
4. Charts general settings
5. Charts customization
6. User units settings
7. About the application

5.4.1. XILab general settings

Program configuration in the Application Settings



XILab general settings tab

This tab configures the slider display settings, secondary position and XIMC devices detection. The position slider is in the main window and visually represents the positioner current position relative to the borders. Configuration comes to setting of displayable borders and defining the slider bounds behavior upon exit out of displayed range.

Fragment of Main application window containing slider

Group **Slider settings** contains the following slider settings:

Left slider bound and Right slider bound contain the left and right bounds of the slider respectively.

Checked Use values from "Borders setup" page for slider bounds loads the values of slider bounds equal to the values of borders set in the Motion range and limit switches into XILab, which are stored in the controller memory. The setting is used by default.

Checked Watch over ltimits defines such slider bounds behavior that upon moving out of the slider range, the scale shifts to display the current position. However, he total distance displayed on the slider remains unchanged. This option is not used by default. It is useful when you know the positioner motion range, but do not know the relation of that position to the values displayed in XILab, e.g. for the calibration purposes. The option is often used together with the settings of the tab Home position settings.

Position settings group contains the position display settings.

Checked Show secondary position displays the encoder position in the main application window.

Device detection settings group includes XIMC devices detection settings.

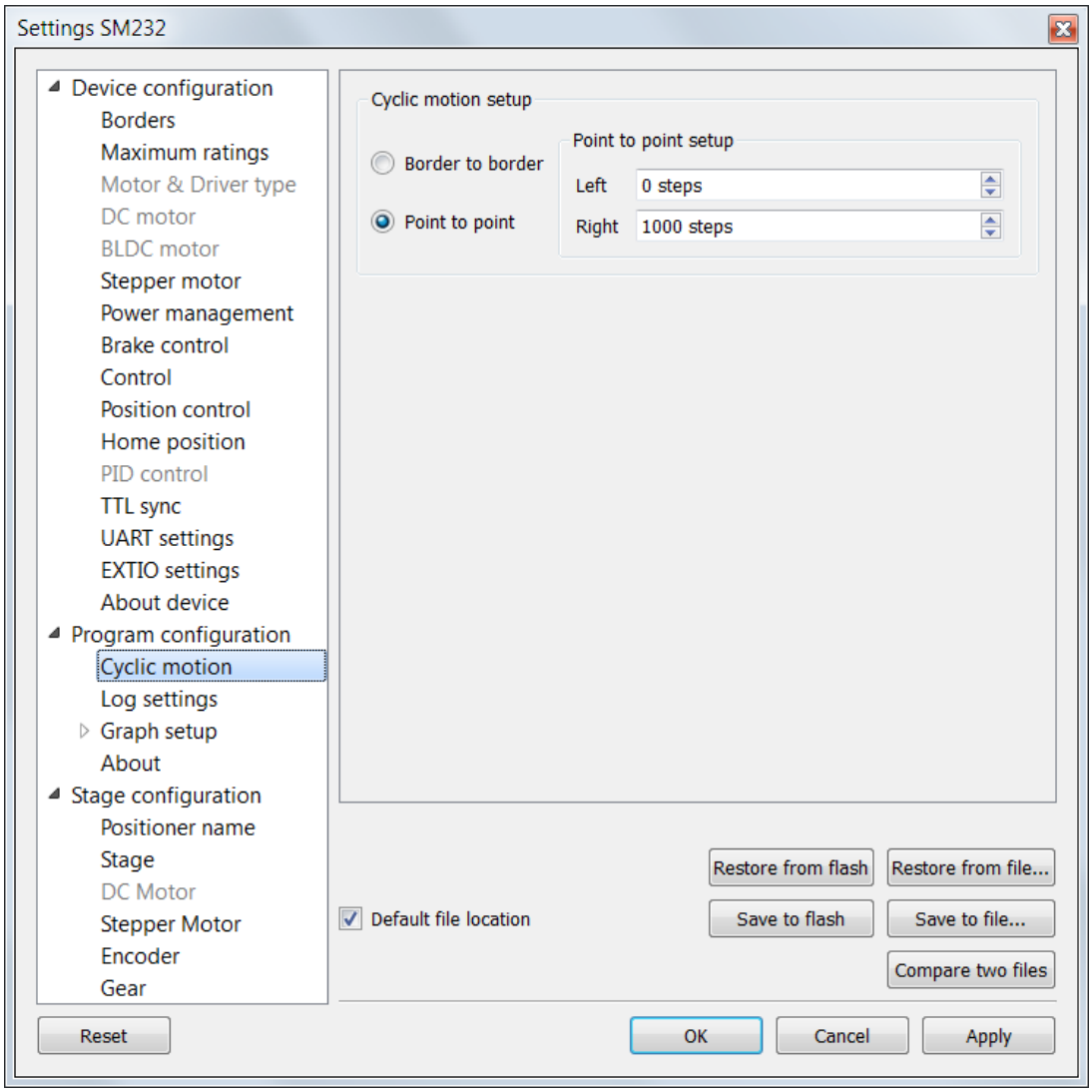
If Probe devices option is checked, at the start application tries to identify controllers by sending them commands GETI and GSER.

If Enumerate non-XIMC devices option is checked the application queries all COM-port type devices in the system. If the option is disabled, only devices with names matching the XIMC mask ("XIMC Motor Controller" in Windows; /dev/ximc/* and /dev/ttyACM* in Linux/Mac) are queried.

{{Warning
Warning. If both options Probe devices and Enumerate non-XIMC devices are enabled, while start XILab sends data to all COM-ports.
If the system has multiple Bluetooth COM-ports, due to the nature of Bluetooth operation, the interrogation will be conducted sequentially, and connection attempts may take from a few to tens of seconds each.

5.4.2. Cyclical motion settings

In the Application Settings ***Program configuration -> Cyclic motion***



Cyclic motion tab

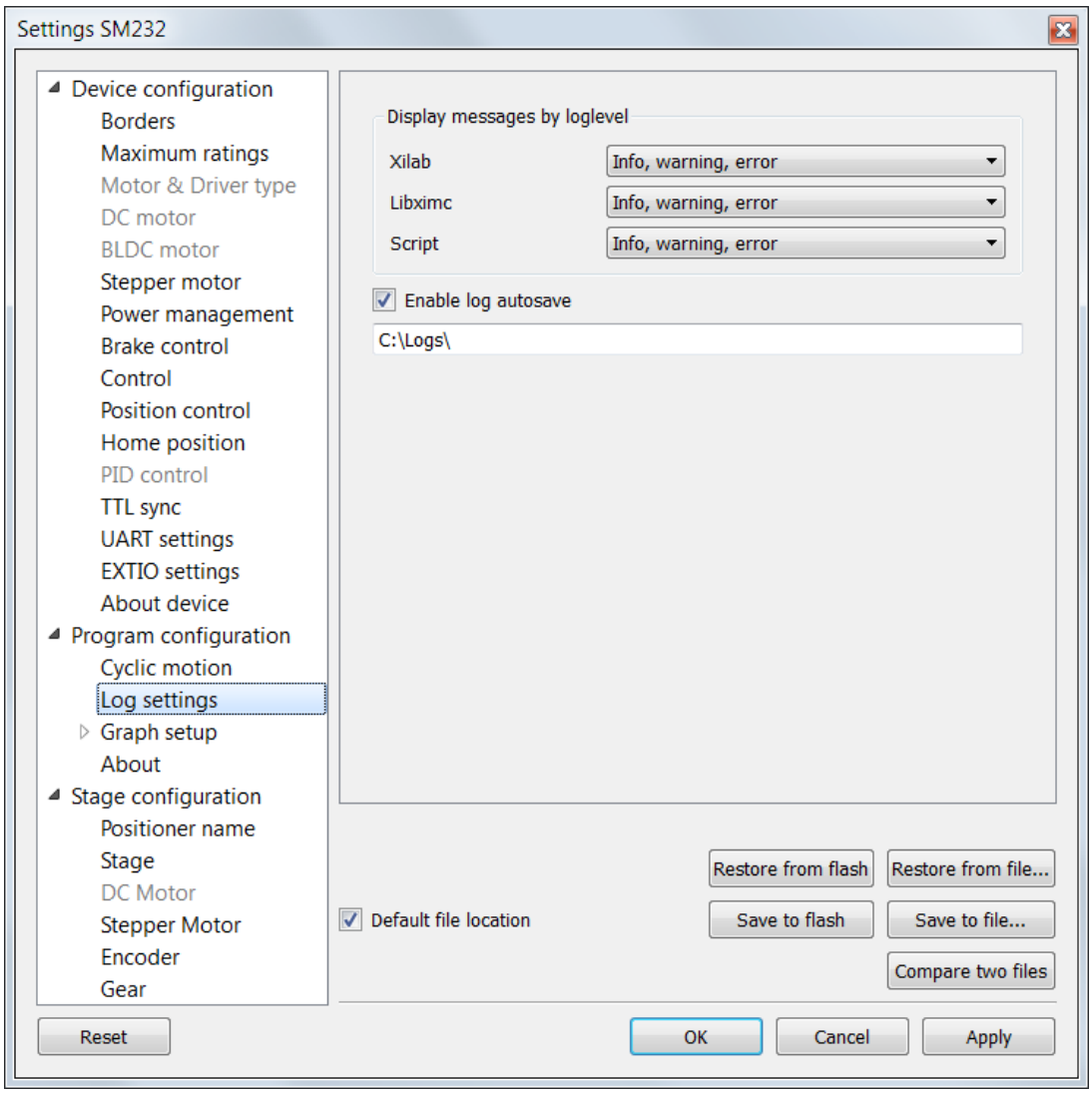
Use this tab to configure the cyclic motion between two preset positions. It is used mainly for demonstration purposes. This mode is activated by Cyclic button in the main window, and deactivated by Stop button in the main window.

Cyclic motion mode settings:

Border to border - cyclical motion between the borders configured in the Motion range and limit switches. The motion begins towards the left edge.

Point to point - cyclical motion between points specified in the Point to point setup group. The positioner moves to the left point, stops, then moves to the right point, stops, and then the cycle repeats.

5.4.3. Log settings



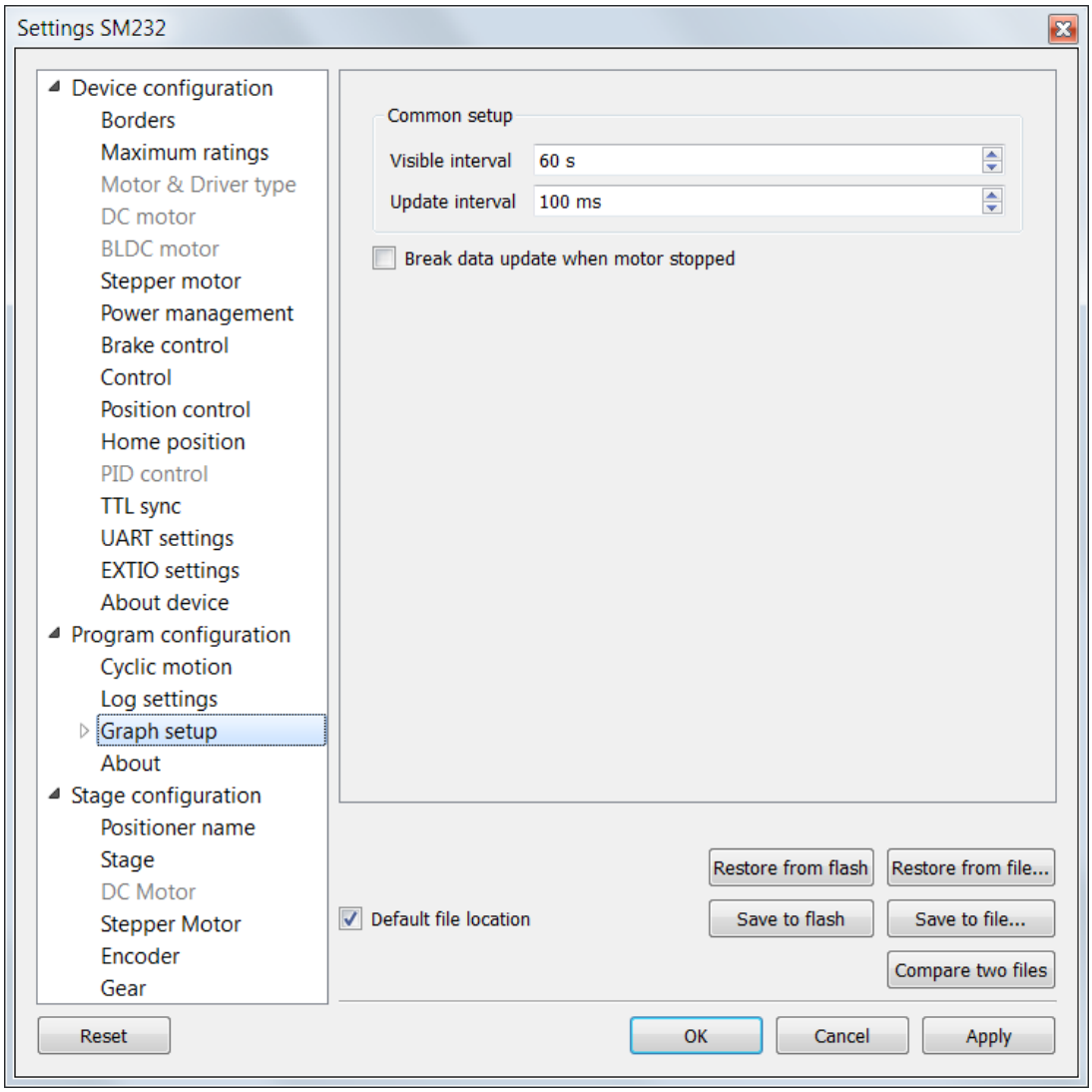
XILab log settings window

On this page (Program configuration -> Log settings) you can configure the logging detail level.
In Display messages by loglevel box you can choose an option to log nothing (None), log only errors (Error), errors and warning messages (Error, Warning), errors, warnings and information messages (Error, Warning, Info) for each source: XILab application, libximc library and Scripts module.

If the Enable log autosave checkbox is checked then the log is saved into file. Directory where the log file will be saved is set below. Log file is flushed to the disk every 5 seconds.
File has a name of type "xilab_log_YYYY.MM.DD.csv", where YYYY, MM and DD are current year, month and day, respectively. Data is stored in CSV format. Messages that are saved into the log file are not filtered by logging options.

5.4.4. Charts general settings

Program configuration -> Graph setup in the Application Settings



Charts general settings tab

Visible interval - the time interval displayed in charts on the horizontal axis.

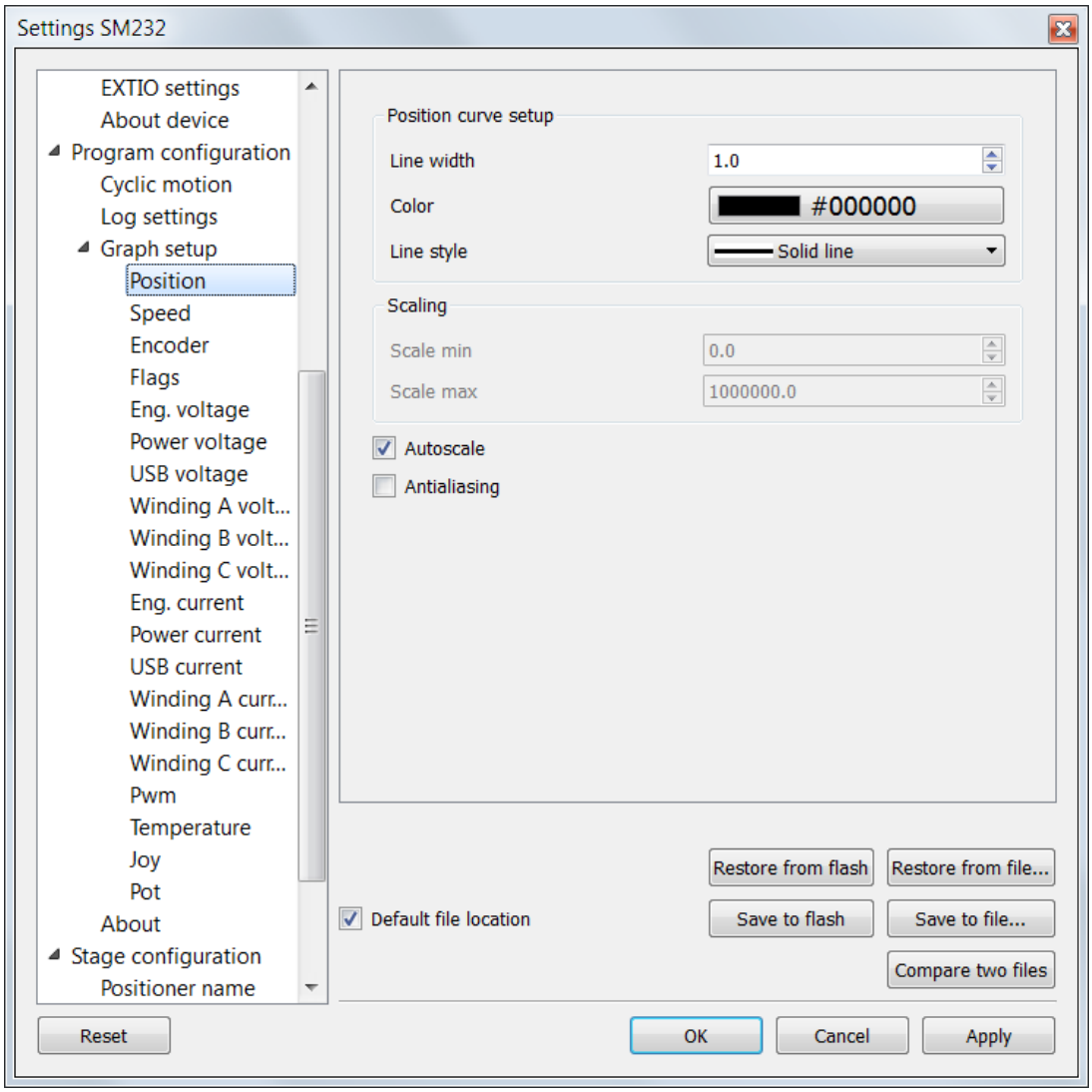
Update interval - chart data update interval.

Break data update when motor stopped - stops drawing charts when the motor stops. This option provides the possibility to use the chart space more rationally, removing the areas when there is no motor motion.

5.4.5. Charts customization

In the Application Settings ***Program configuration -> Graph setup -> ...***

This section equally applies to the individual appearance settings of speed, voltage, current, PWM duty factor, temperature, and other parameter charts, which can be displayed in the XILab application.



Charts customization on the example of the position chart tab

Charts display settings include line style and chart vertical axis scale adjustment.

Position curve setup group changes curve parameters. It includes the Line width, Color and Line style.

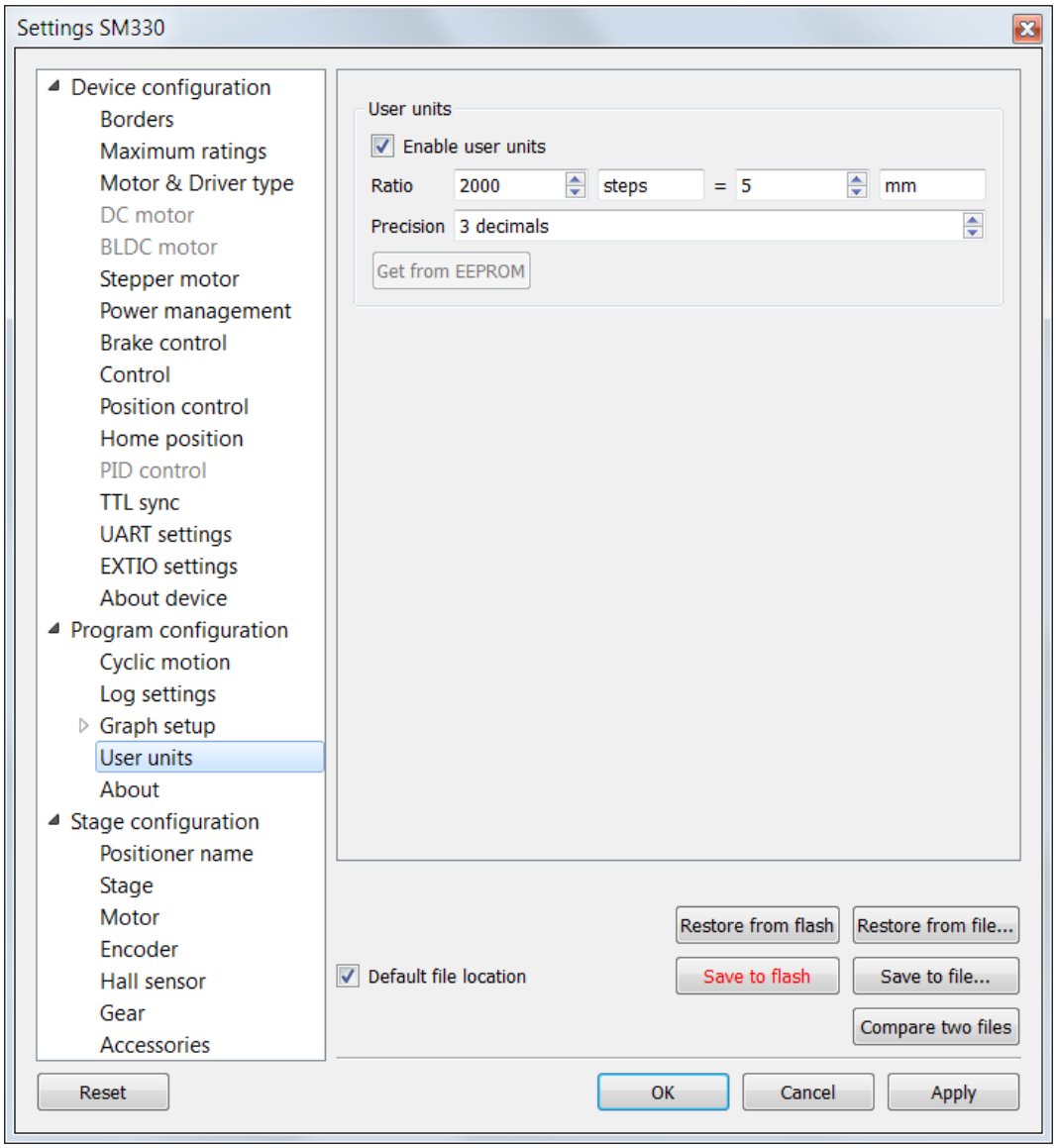
Scaling group changes curve display range on the vertical axis by setting values in Scale min and Scale max.

Checked Autoscale flag results in auto-scaling of the scale limits in accordance with the change limits of the variable on the axis Y. In this case, the parameters Scale min and Scale max are ignored.

Antialiasing flag enables chart lines smoothing, which provides the possibility to achieve a higher-quality display, but it slows a little the chart drawing process.

5.4.6. User units settings

In the Application Settings ***Program configuration -> User units***



User units tab

Use this tab to configure user units display. Used to replace internal controller coordinates with units familiar to the user.

User units settings:

Enable user units - enables user unit display instead of steps (in case of stepper motor) or encoder counts (in case of DC motor). User units replace steps(counts) only in the main XILab window and do not affect any of the Settings pages.

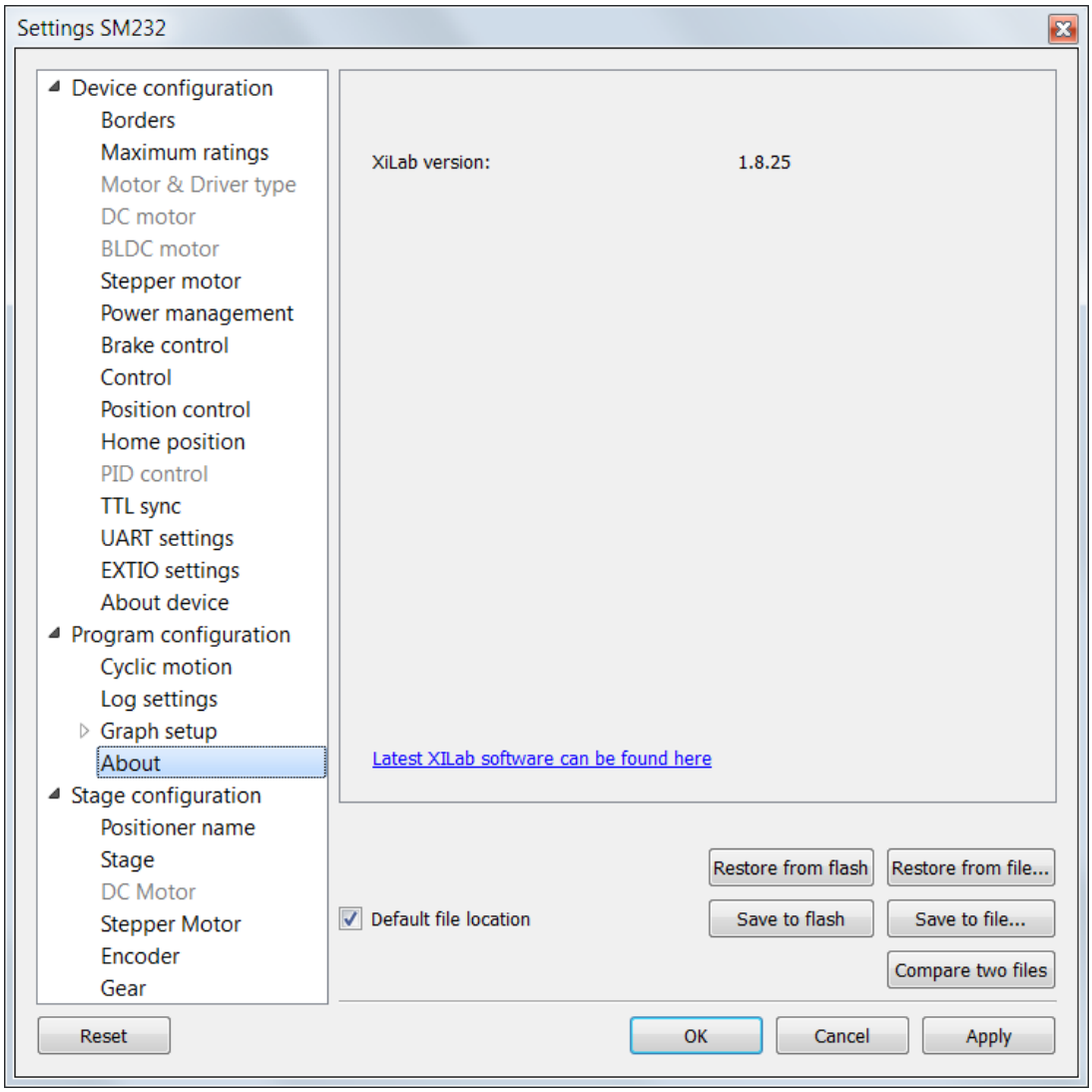
Ratio - conversion of controller steps to position units, set as a ratio of two integer values "x steps = y user units". Values "x", "y" and unit name string are set by user.

Precision - displayed precision.

Get from EEPROM button reads user unit settings from connected EEPROM.

5.4.7. About the application

Program configuration -> About in the Application Settings



About tab

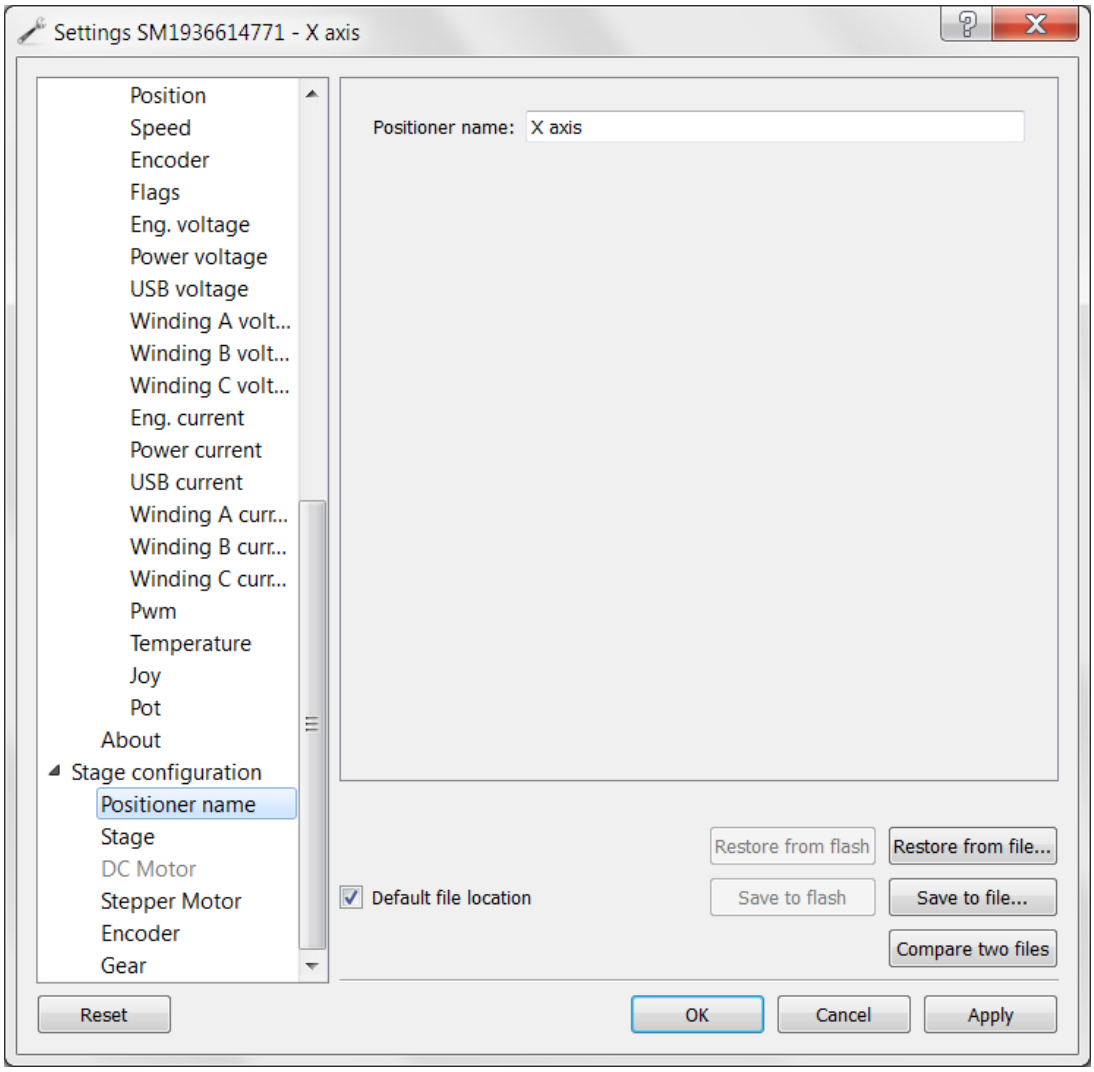
This section displays the XILab application version. It also contains a link to the page with the latest Software version.

5.5. Positioner specifications

1. Positioner name
2. Positioner general characteristics
3. DC motor characteristics
4. Stepper motor characteristics
5. Encoder specifications
6. Reducing gear specifications

5.5.1. Positioner name

Stage configuration -> Positioner name in the Application Settings

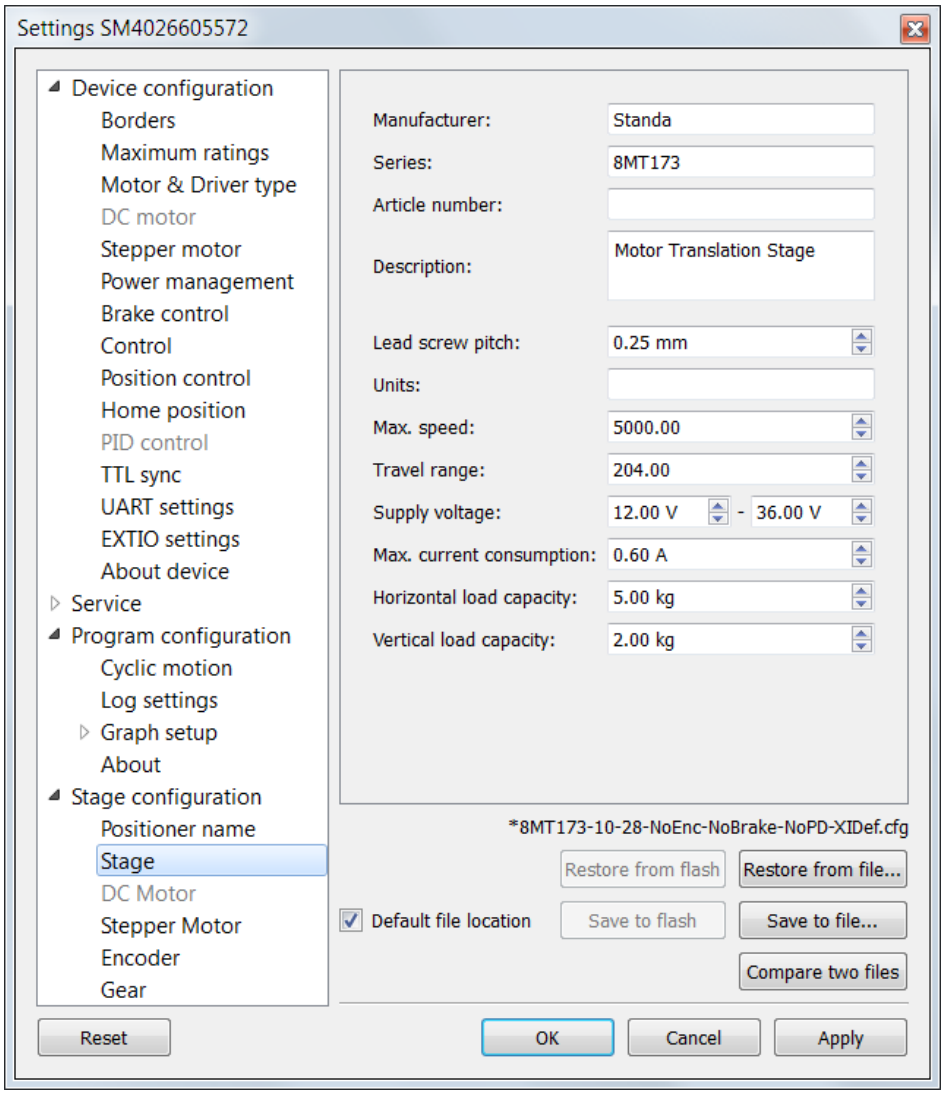


Positioner name window

This block contains the positioner name (defined by user).

5.5.2. Positioner general characteristics

Stage configuration -> Stage in the Application Settings



Positioner general characteristics window

Stage parameter prouop contains information about the positioner.

Manufacturer - the manufacturer name.

Series - positioner type.

Article number - the catalog number.

Description - brief description.

Lead screw pitch - lead screw pitch.

Units - measurement units of the given positioner motion (mm, degrees, steps).

Max speed - the maximum speed.

Travel range - range of motion.

Supply voltage - supply voltage.

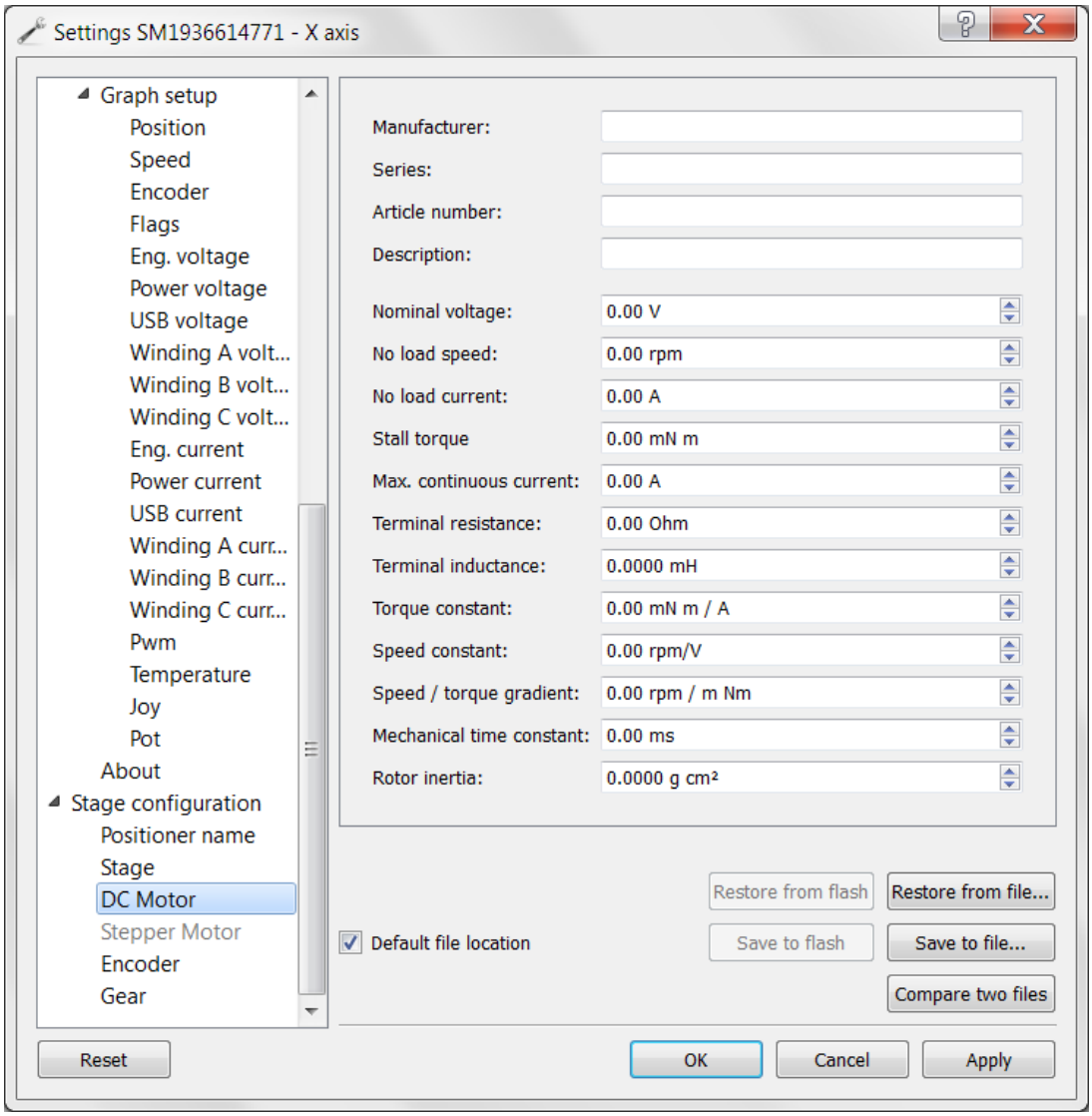
Max. current consumption - the maximum current consumption.

Horizontal load capacity - the maximum horizontal load on the positioner.

Vertical load capacity - the maximum vertical load on the positioner.

5.5.3. DC motor characteristics

Stage configuration -> DC-motor in the Application Settings



DC motor characteristics window

This section contains DC motor information.

Manufacturer - motor manufacturer.

Series - motor type.

Article number - catalog number.

Description - brief description of the motor.

Nominal voltage - nominal voltage on the winding.

No load speed - idle speed.

No load current - no-load current consumption.

Stall torque - torque at zero speed.

Max. continious current - maximum current consumption during a long period.

Terminal resistance - winding active resistance.

Terminal inductance - winding inductance.

Torque constant - torque constant.

Speed constant - speed constant.

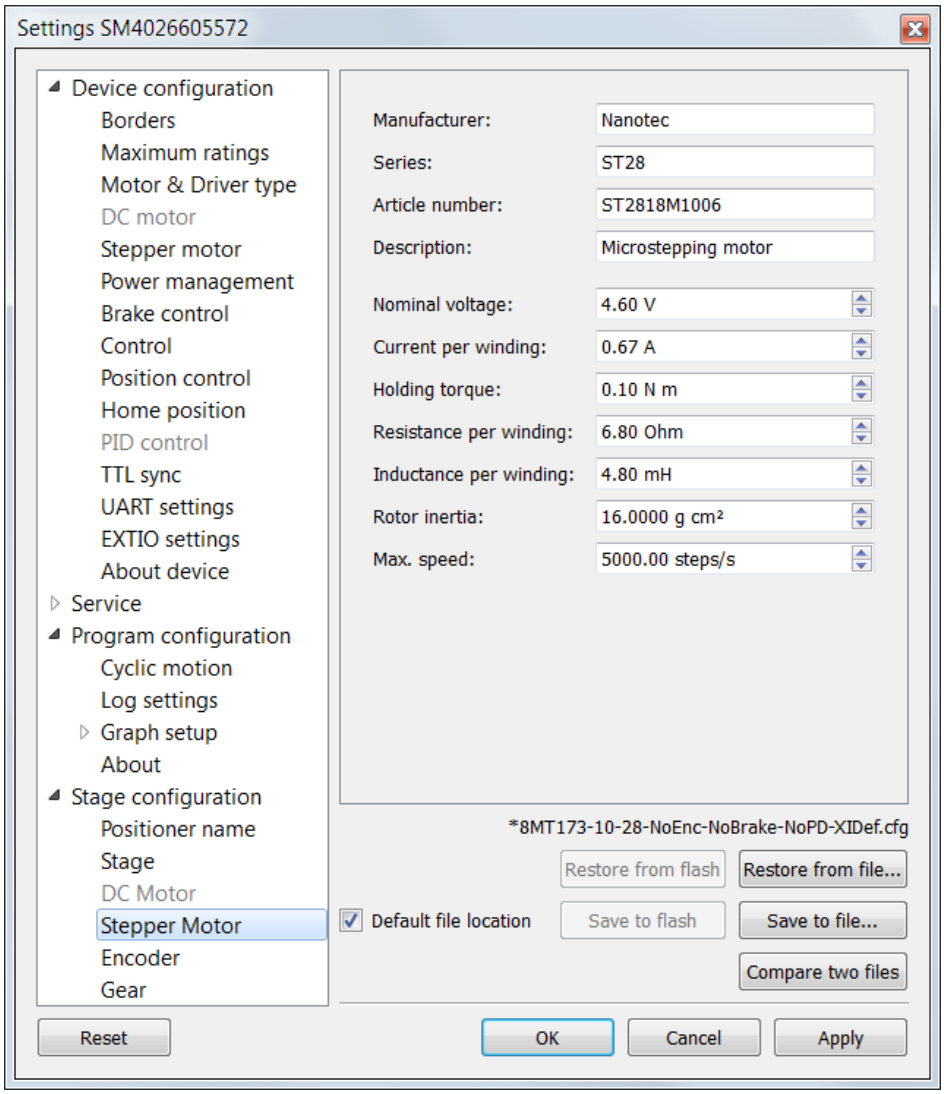
Speed torque gradient - speed/torque constant.

Mechanical time constant - motor time constant.

Rotor inertia - rotor inertia.

5.5.4. Stepper motor characteristics

Stage configuration -> Stepper motor in the Application Settings



Stepper motor characteristics window

This section contains stepper motor information.

Manufacturer - motor manufacturer.

Series - motor type.

Article number - catalog number.

Description - brief description of the motor.

Nominal voltage - nominal winding voltage.

Current per winding - winding nominal current.

Holding torque - nominal torque.

Resistance per winding - winding resistance.

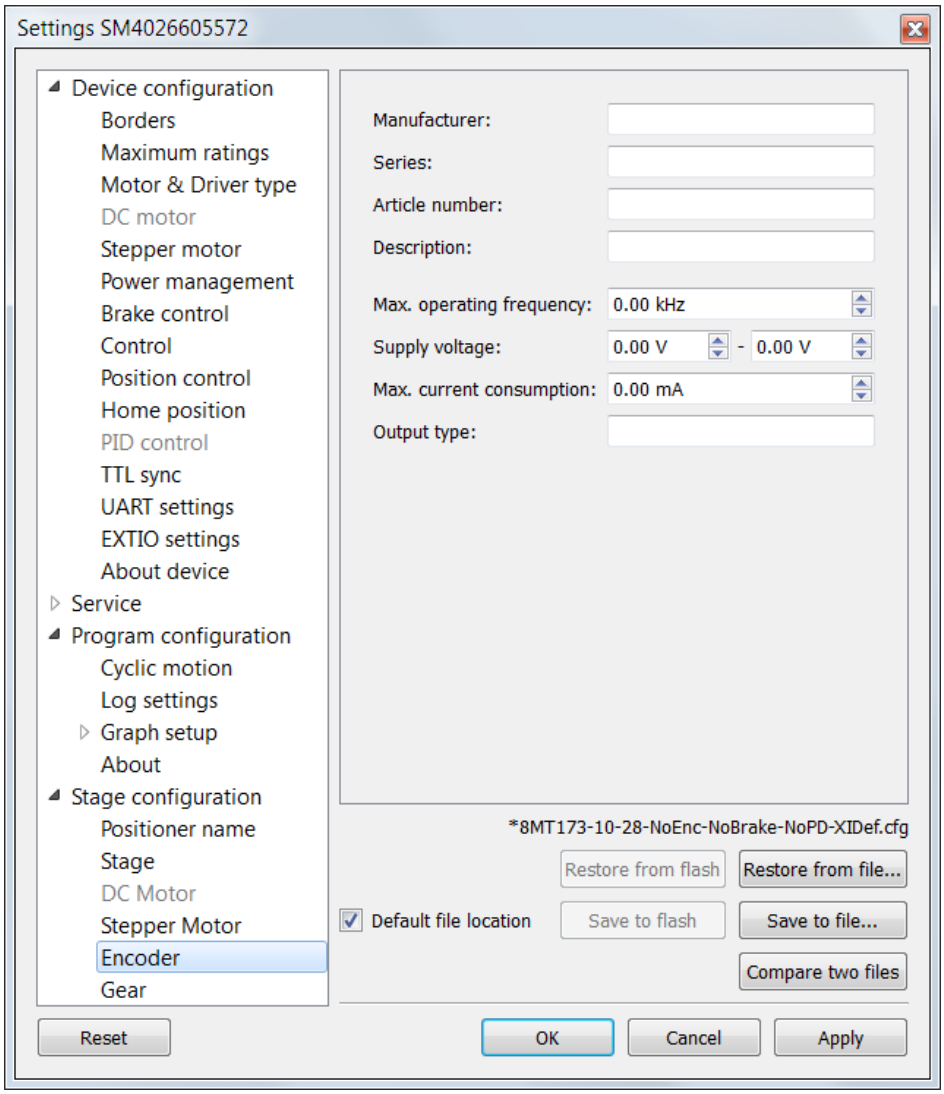
Inductance per winding - winding inductance.

Rotor inertia - inertia moment.

Max. speed - maximum speed.

5.5.5. Encoder specifications

Stage configuration -> Encoder in the Application Settings



Window Encoder specifications

This section contains information about encoder

Manufacturer - encoder manufacturer name.

Series - encoder type.

Article number - the catalog number.

Description - brief description of the encoder.

Max. operating frequency - the maximum operating frequency.

Max. speed - the maximum shaft speed.

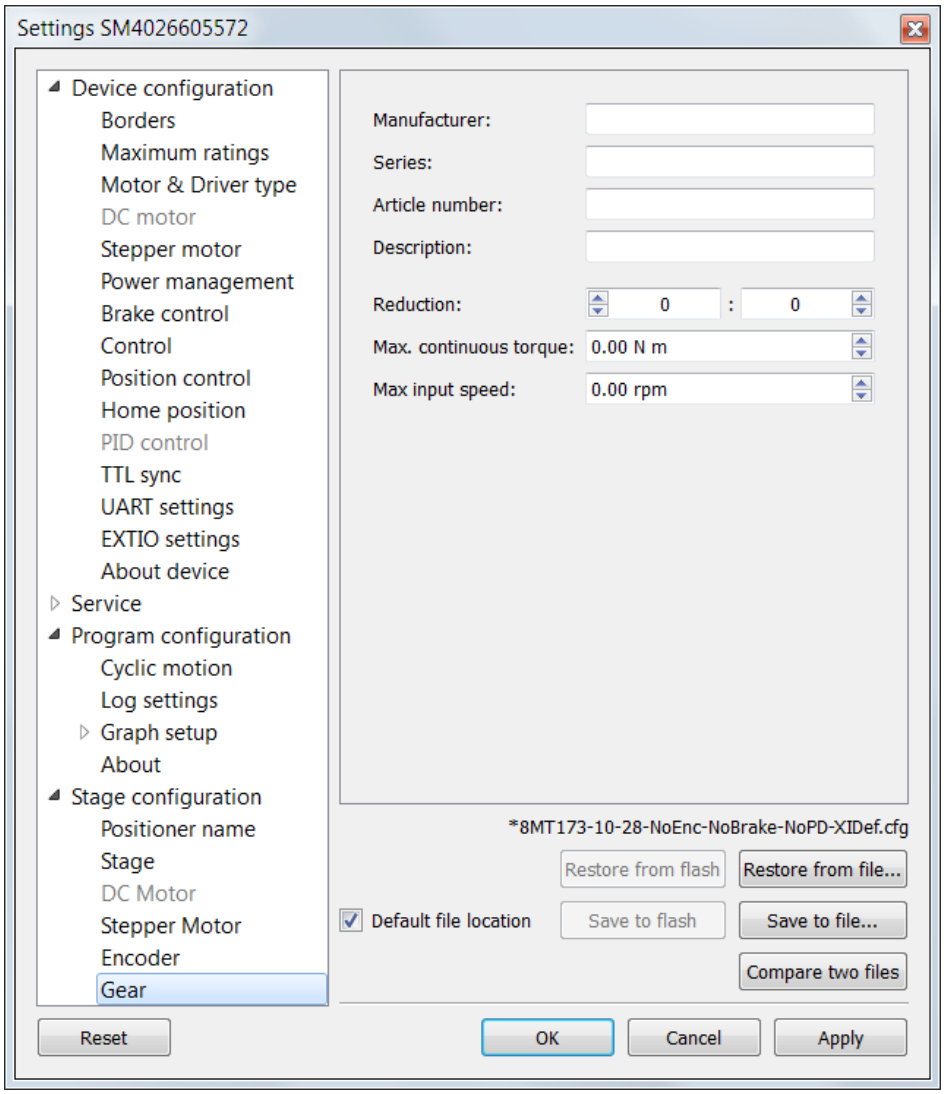
Supply voltage - supply voltage.

Max. current consumption - the maximum current consumption.

Output type - electric type of encoder output.

5.5.6. Reducing gear specifications

Stage configuration -> Gear in the Application Settings



Reducing gear specifications window

This section contains information about the reducing gear.

Manufacturer - the manufacturer name.

Series - gear type.

Article number - the catalog number.

Description - brief description of the gear.

Reduction - gear transmission ratio.

Max. continious torgue - maximum torque at the gear input.

Max. input speed - the maximum speed at the gear input.

5.6. Correct shutdown

Correct shutdown assumes shutdown of the motor and saving the current position by the controller. The current position is automatically saved, see Saving the position in controller's FRAM memory.

Exit button performs correct shutdown and exit. When you click it the application sends a soft stop command to the controller, and after the stop is complete, the application sends command of power-off. If execution of the soft stop command was interrupted by an event like a motion command from the joystick or signal of TTL synchronization, or if while sending of soft stop command or command of power off to the controller, the library returned an error, the exit will be canceled. In this case you need to check joystick settings and settings of "right" and "left" buttons and Synchronization settings.

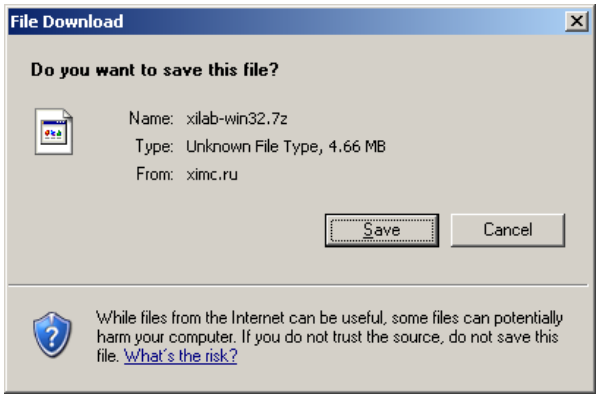
5.7. XILab installation

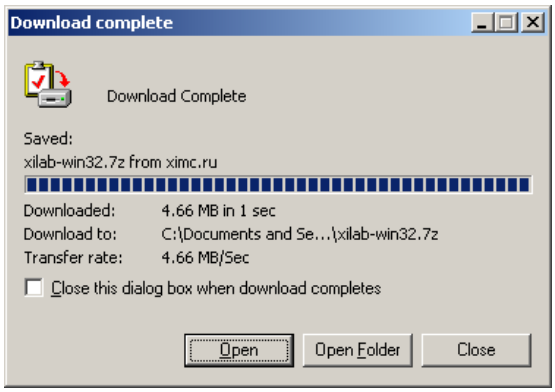
5.7.1. Installation on Windows

1. Installation on Windows XP
2. Installation on Windows 7
3. Installation on Windows 8

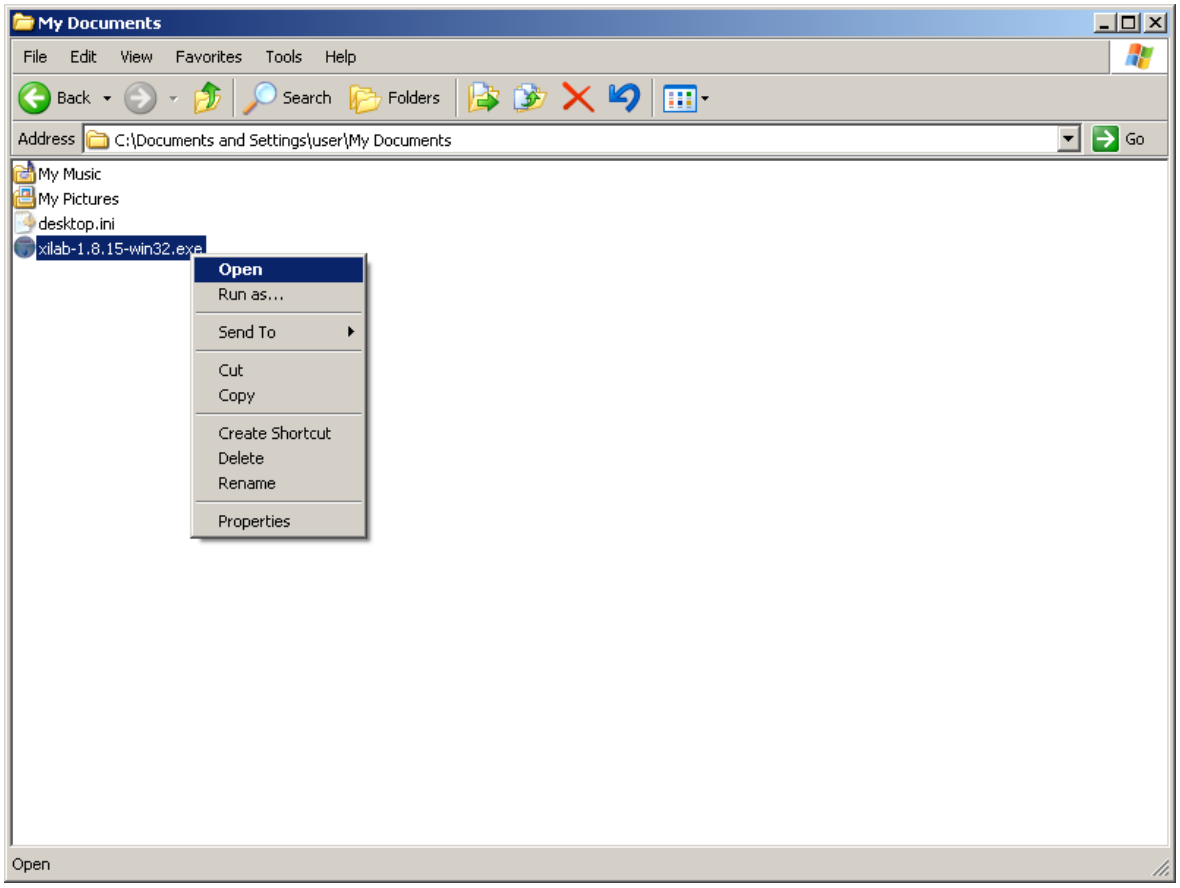
5.7.1.1. Installation on Windows XP

Copy the installer program file to your computer. The installer file name is "xilab-1.8.16-win32.exe" (for 32-bit versions of Windows) or "xilab-1.8.16-win64.exe" (for 64-bit versions of Windows).



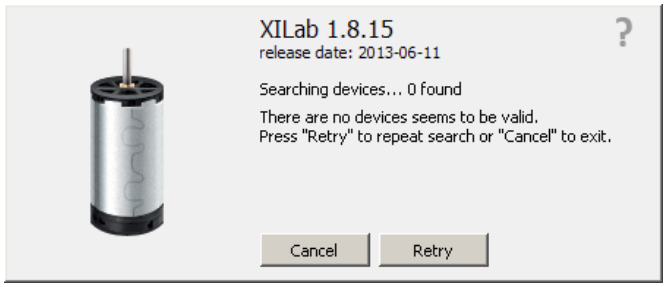


Run the installer and follow the on-screen instructions.



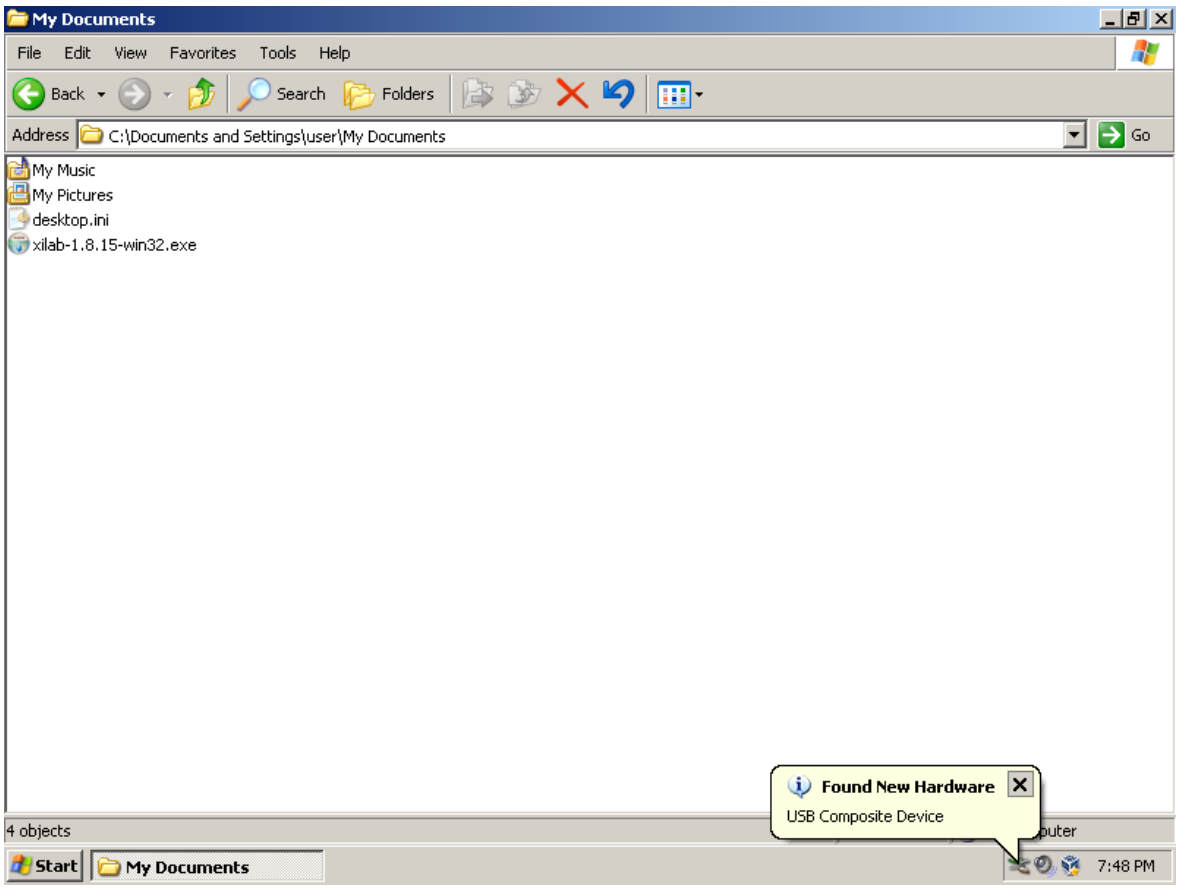
All the necessary software including drivers, packages and programs will be installed automatically.

After the installation is complete the XiLab application will be started by default.



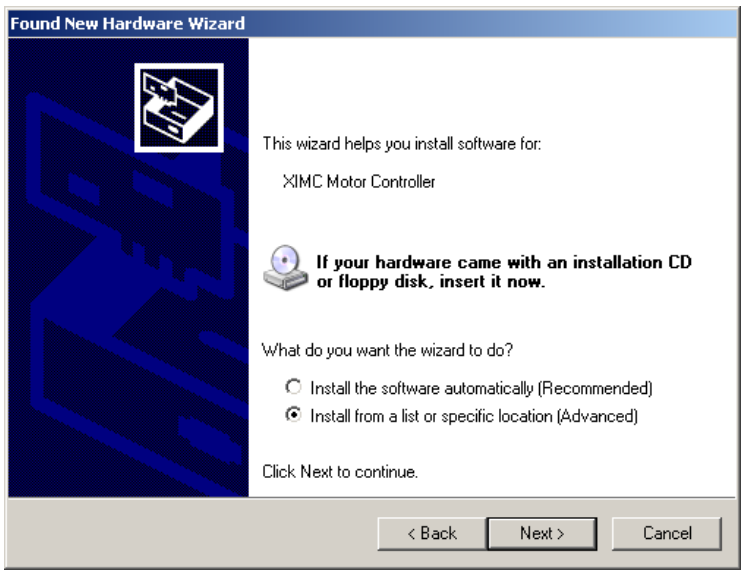
Connect the positioner to the controller. Connect regulated power supply to the controller. Ground the controller or the power supply unit. Connect the controller to the computer using a USB-A - mini-USB-B cable.

LED indicator on the controller board will start to flash. New Hardware Wizard will start after the first controller is connected to the computer. Wait until Windows detects the new device and installs the drivers for it.

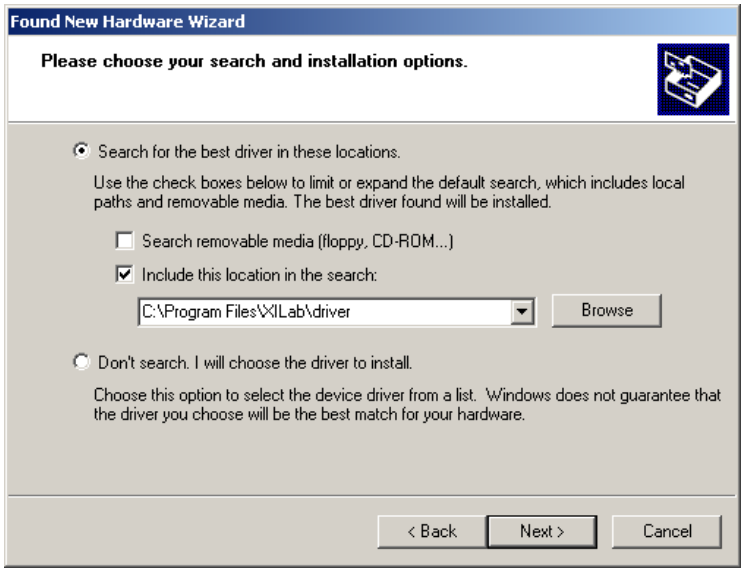


If the driver was not automatically installed select "No, not this time" and click "Next>" in the pop-up window.

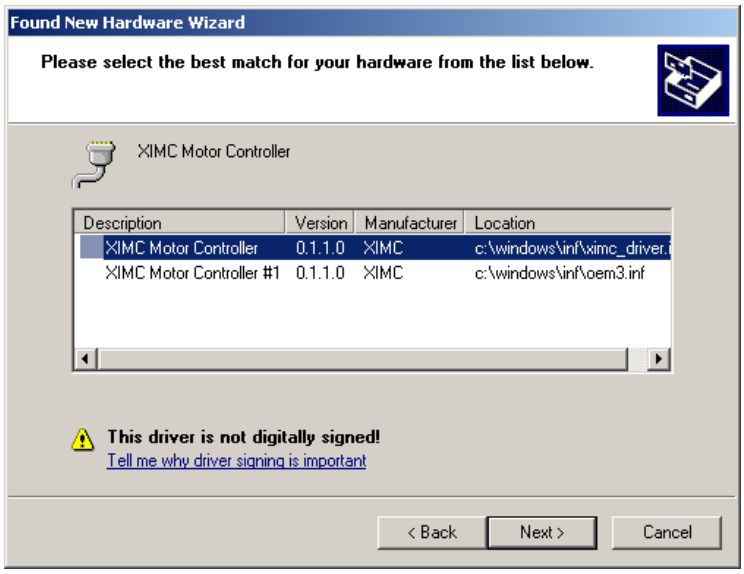
In the next window select "Install from a list or specific location (Advanced)" and click "Next>".



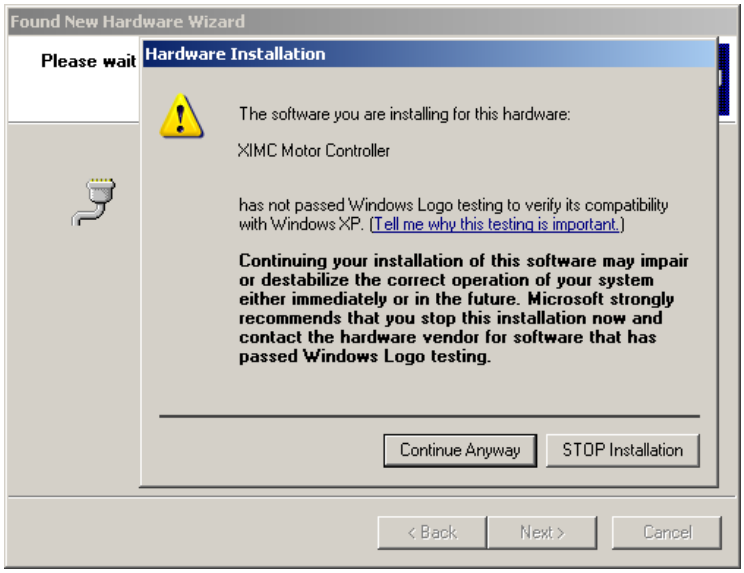
Select the *.inf file on the disk with the software supplied with the controller or in the program directory (by default the path is C:\Program Files\XiLab\driver\) and click "Next".



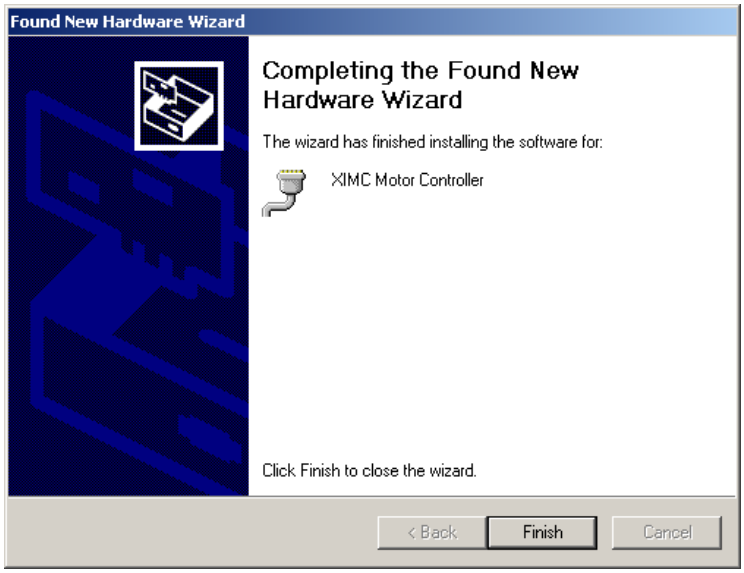
Click "Next".



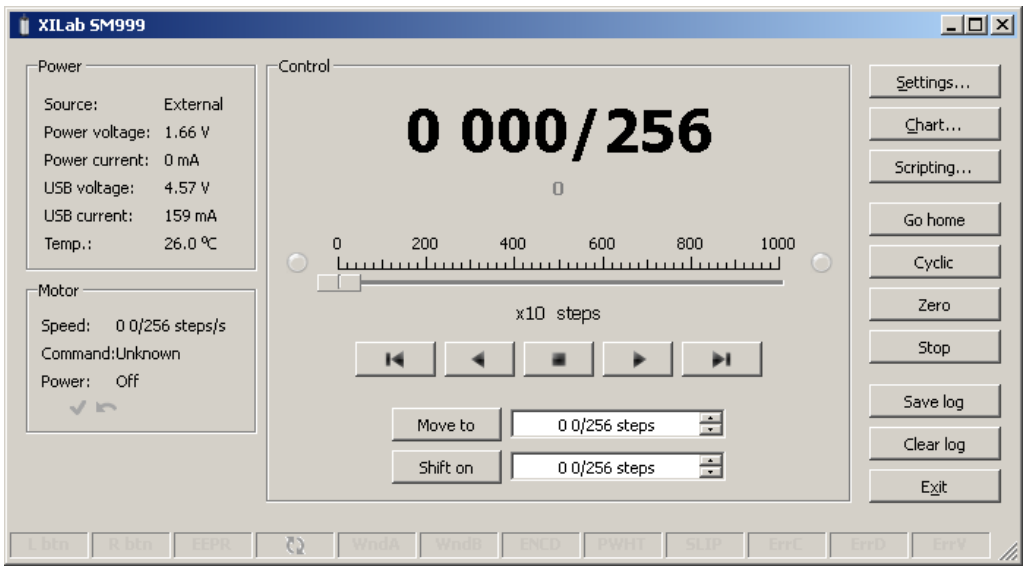
Click "Continue anyway".



Click "Finish". Driver installation is complete.

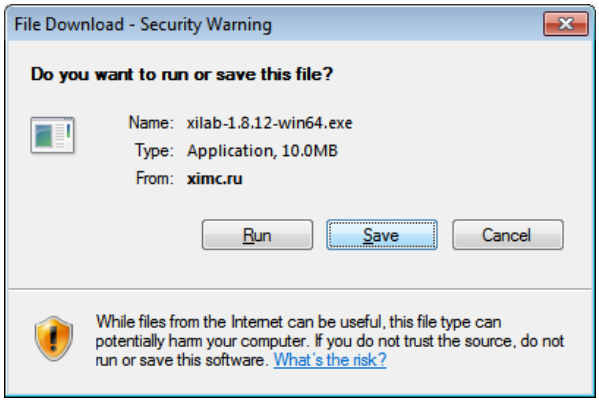


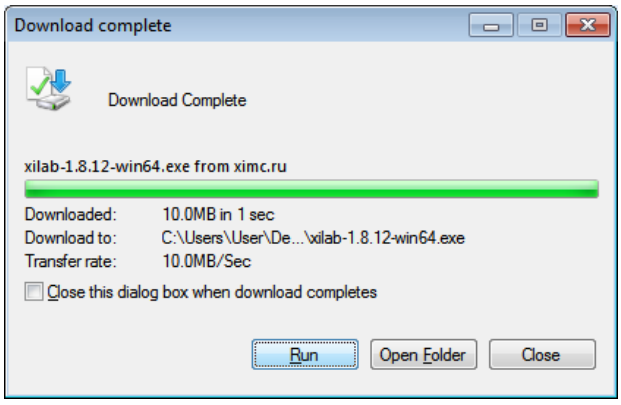
Click Retry or run the Xilab application again if it was closed. The system will detect the connected controller and open the main Xilab window.


XILab SM999
Power
Source: External
Power voltage: 1.66 V
Power current: 0 mA
USB voltage: 4.57 V
USB current: 159 mA
Temp.: 26.0 °C
Motor
Speed: 0 0/256 steps/s
Command:Unknown
Power: Off
Control
0 000/256
0
0 200 400 600 800 1000
x10 steps
Move to
0 0/256 steps
Shift on
0 0/256 steps
Settings...
Chart...
Scripting...
Go home
Cyclic
Zero
Stop
Save log
Clear log
Exit
L btn
R btn
EEPR
WndA
WndB
ENCD
PWHT
SLIP
ErrC
ErrD
ErrV

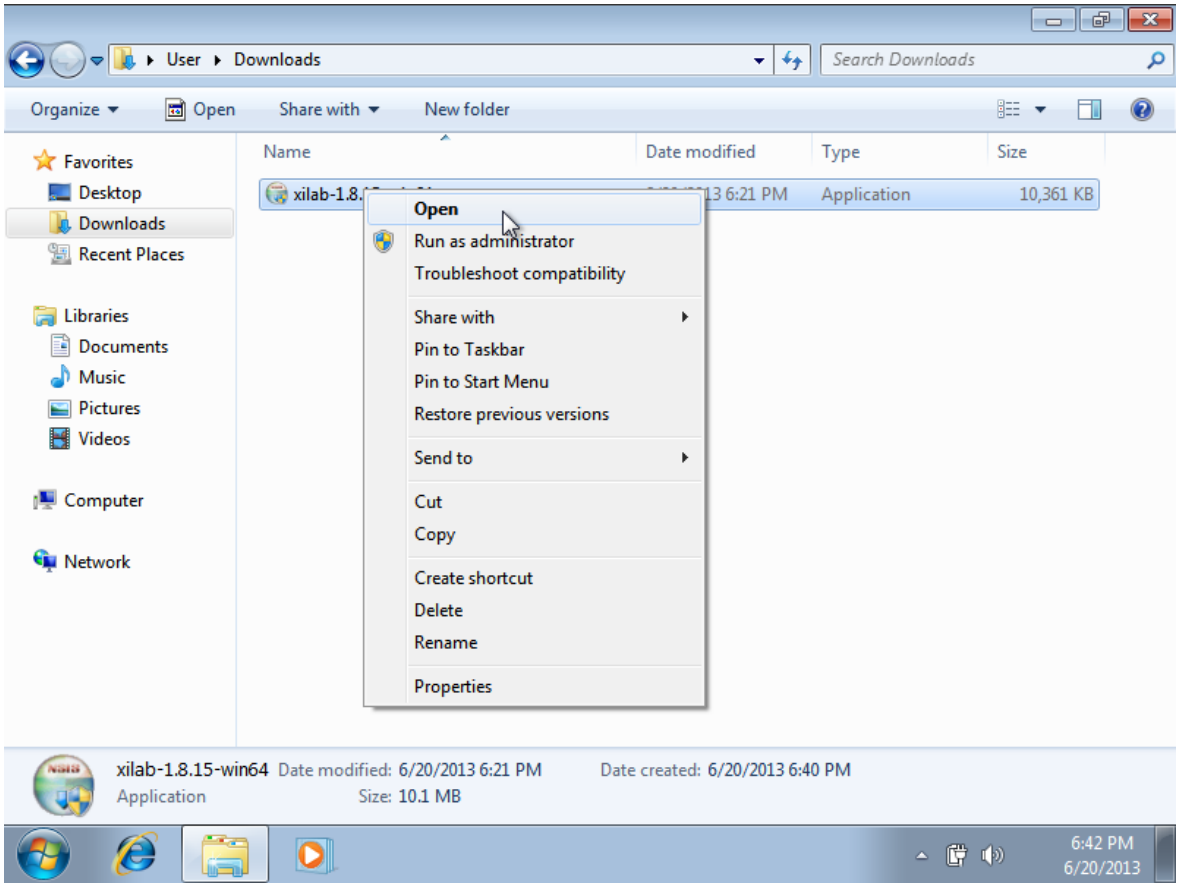
5.7.1.2. Installation on Windows 7

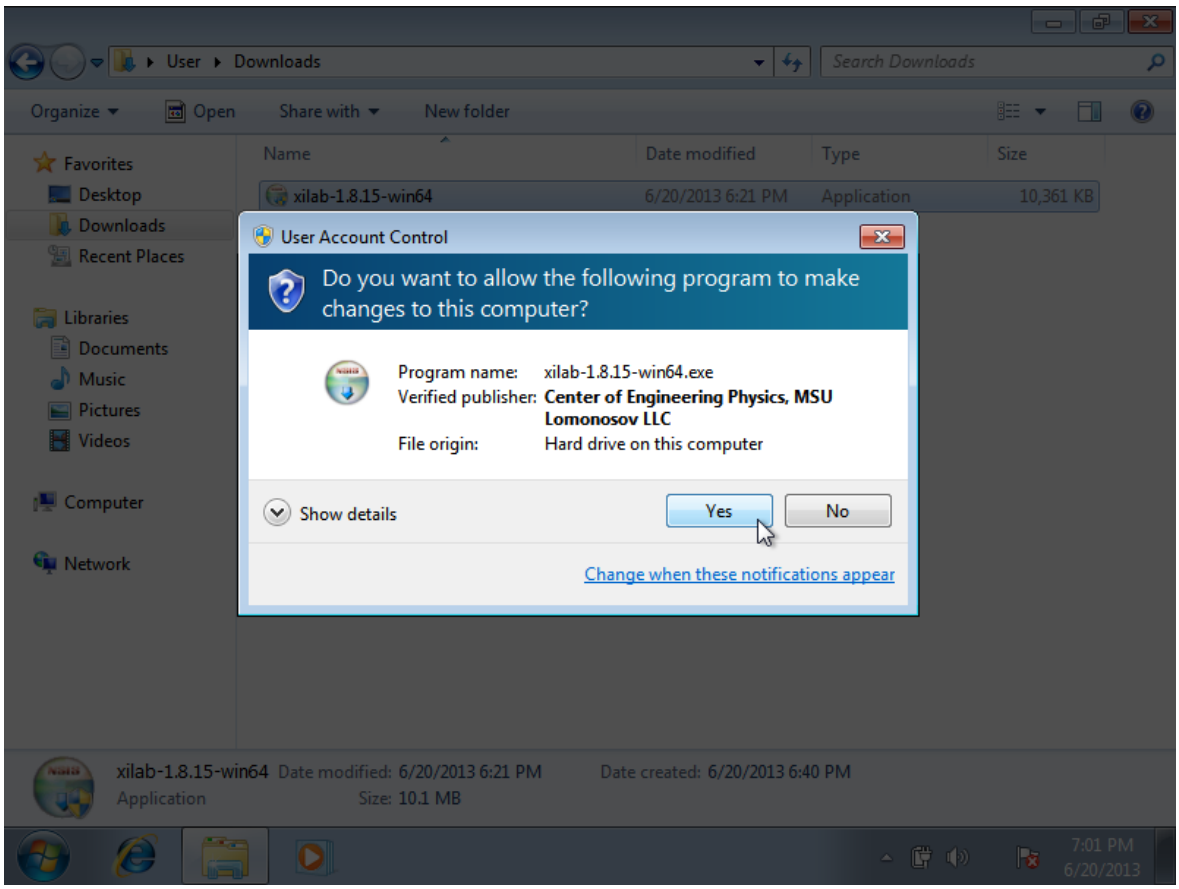
Copy installer program file to your computer. The installer file name is "xilab-1.8.16-win32.exe" (for 32-bit versions of Windows) or "xilab-1.8.16-win64.exe" (for 64-bit versions of Windows).





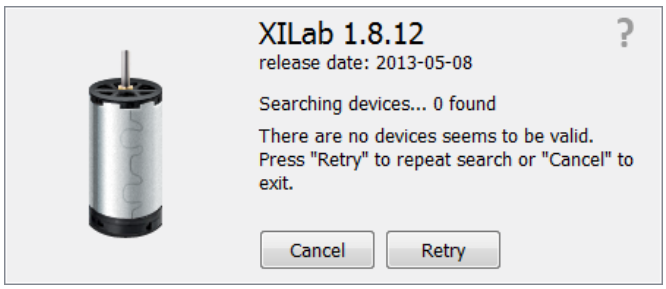
Run the installer and follow the on-screen instructions.





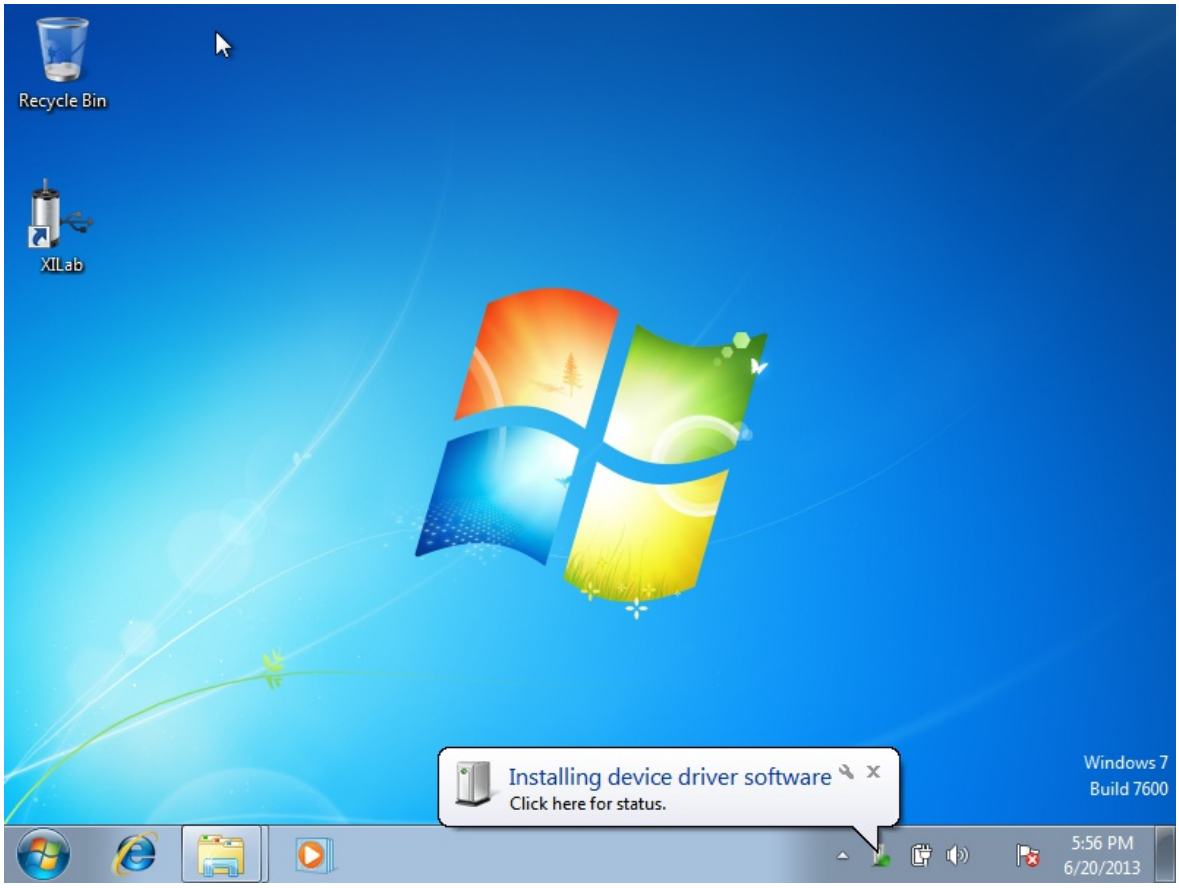
All the necessary software including drivers, packages and programs will be installed automatically.

After the installation is complete the XiLab application will be started by default.

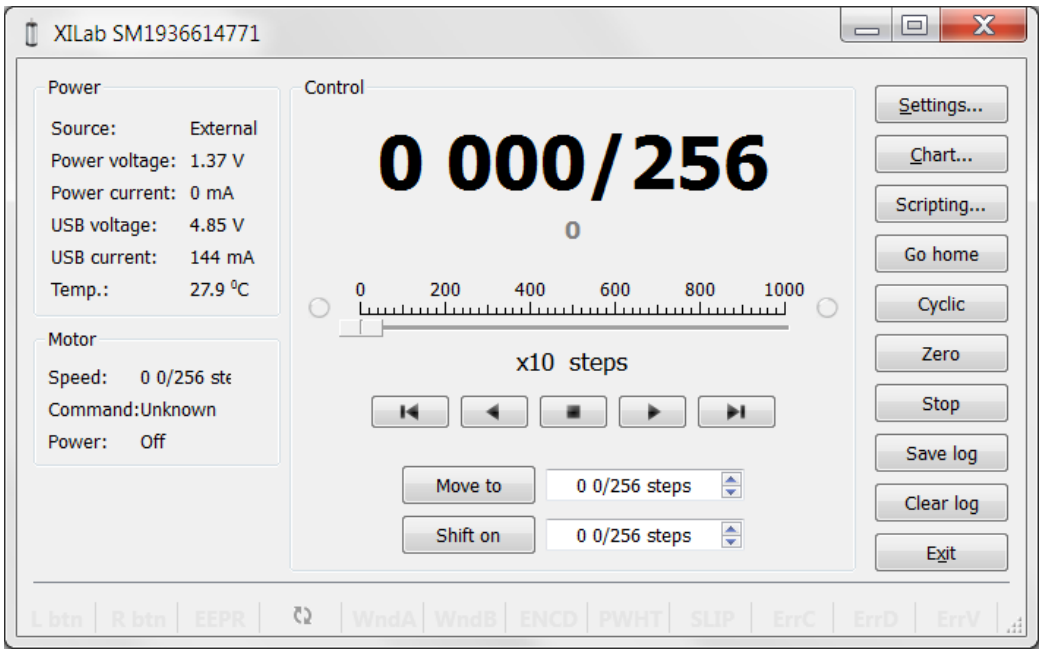


Connect the positioner to the controller. Connect regulated power supply to the controller. Ground the controller or the power supply unit. Connect the controller to the computer using a USB-A - mini-USB-B cable.

LED indicator on the controller board will start to flash. New Hardware Wizard will start after the first controller is connected to the computer.

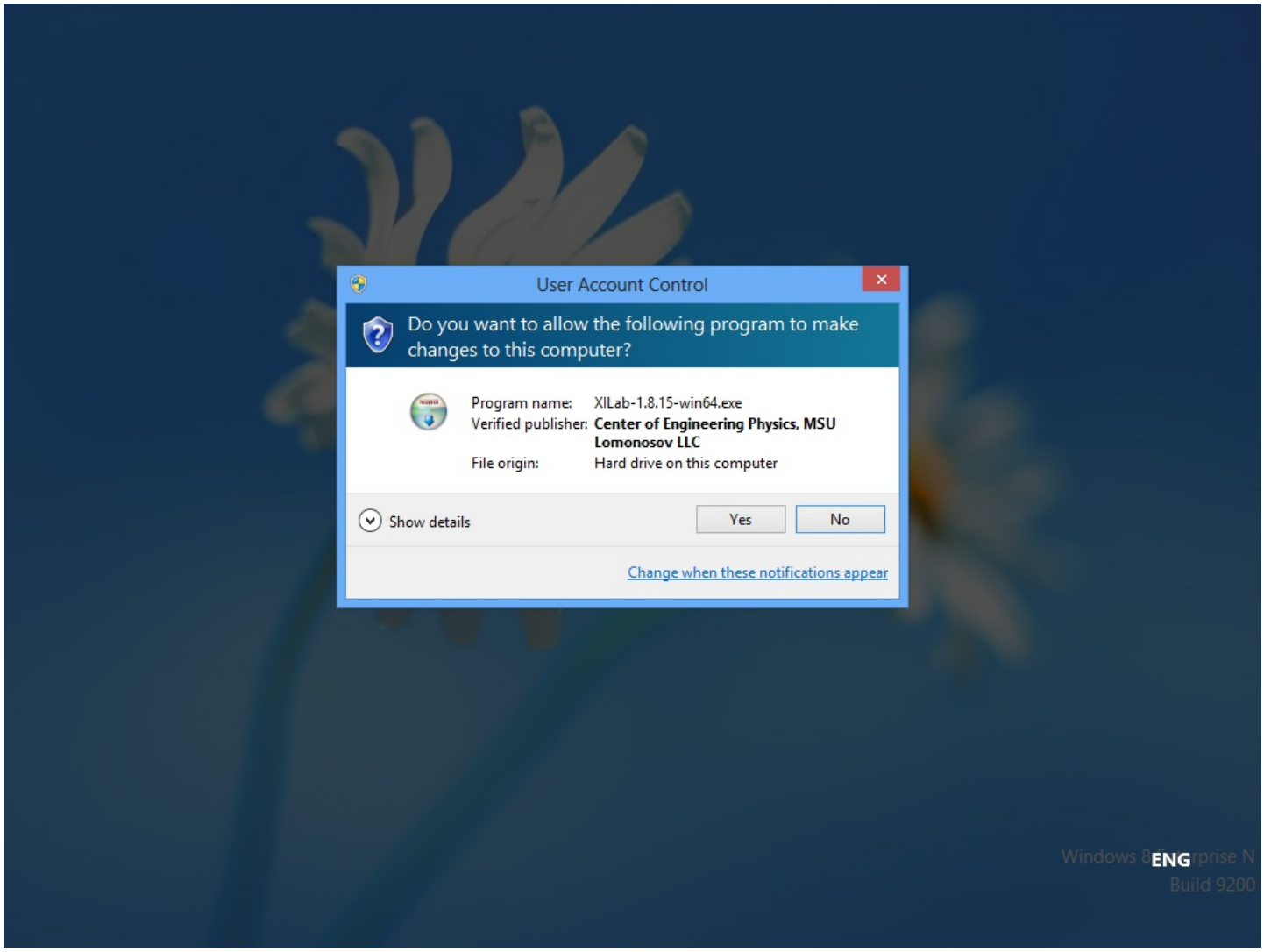


Wait until Windows detects the new device and installs the drivers for it. After the driver is successfully installed click Retry or run the Xilab application again if it was closed. The system will detect the connected controller and open the main Xilab window.


XILab SM1936614771
Power
Source: External
Power voltage: 1.37 V
Power current: 0 mA
USB voltage: 4.85 V
USB current: 144 mA
Temp.: 27.9 °C
Motor
Speed: 0 0/256 ste
Command: Unknown
Power: Off
Control
0 000/256
0
0 200 400 600 800 1000
x10 steps
Move to
0 0/256 steps
Shift on
0 0/256 steps
Settings...
Chart...
Scripting...
Go home
Cyclic
Zero
Stop
Save log
Clear log
Exit
L btn R btn EEPR WndA WndB ENCD PWHT SLIP ErrC ErrD ErrV

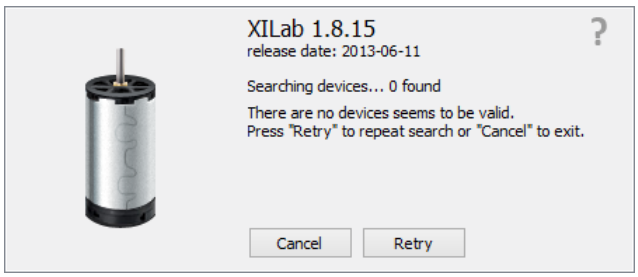
5.7.1.3. Installation on Windows 8

Copy the installer program file to your computer. The installer file name is "xilab-1.8.16-win32.exe" (for 32-bit versions of Windows) or "xilab-1.8.16-win64.exe" (for 64-bit versions of Windows). Run the installer and follow the on-screen instructions.



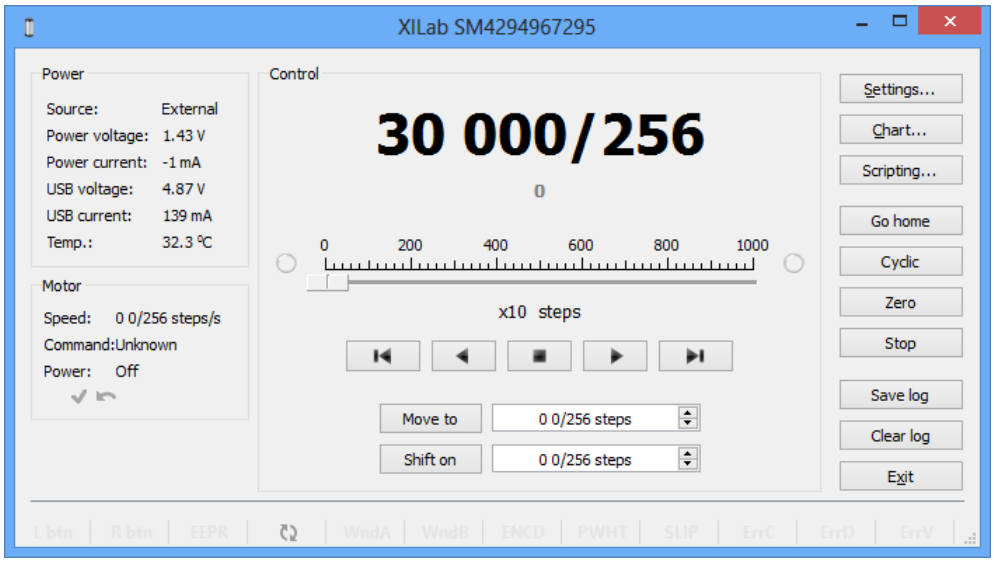
All the necessary software including drivers, packages and programs will be installed automatically.

After the installation is complete the XiLab application will be started by default.



Connect the positioner to the controller. Connect regulated power supply to the controller. Ground the controller or the power supply unit. Connect the controller to the computer using a USB-A - mini-USB-B cable.

LED indicator on the controller board will start to flash. New Hardware Wizard will start after the first controller is connected to the computer. Wait until Windows detects the new device and installs the drivers for it. After the driver is successfully installed, click Retry, or run the Xilab application again, if it was closed. The system will detect the connected controller and open the main Xilab window.



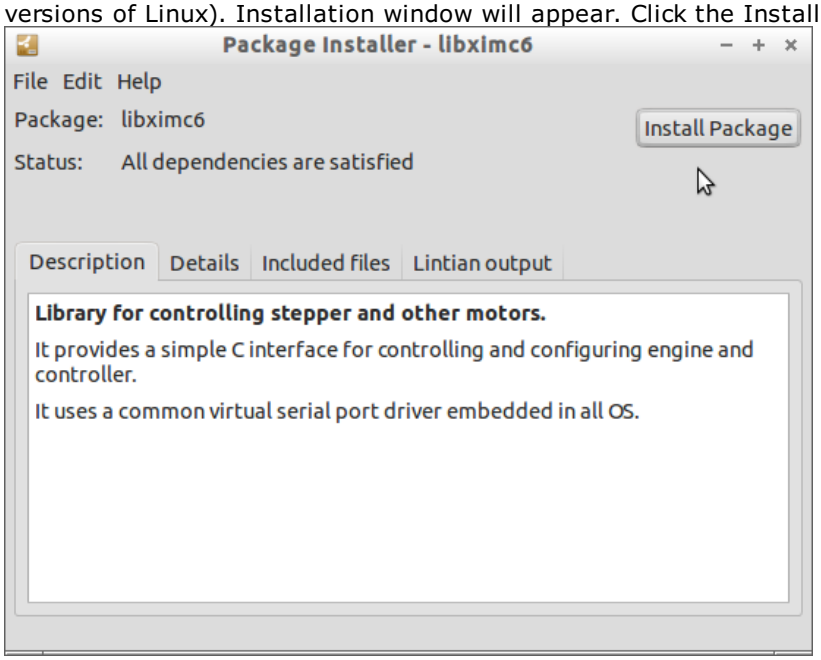
5.7.2. Installation on Linux

XILab package installs its files in the following directories:
/usr/bin/xilab - executable file
/usr/share/icons/xilab.png - Icon
/usr/share/xilab/scripts/ - directory scripts
/usr/share/xilab/profiles/ - directory with profiles
/usr/share/xilab/xilabdefault.cfg - file with the default settings

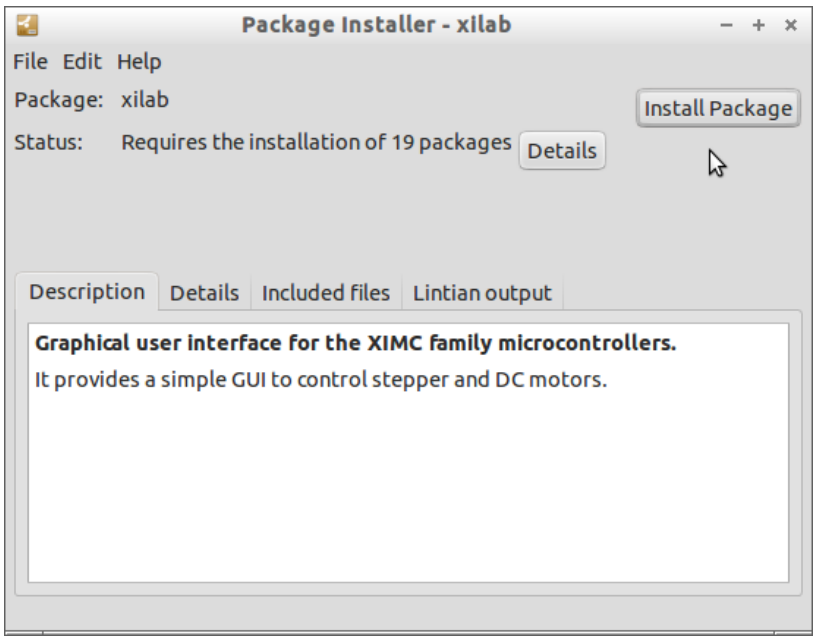
Debian / Ubuntu

Installing in graphical mode

Click on the file libximc6_n.nn-1_i386.deb (for 32-bit versions of Linux) or libximc6_n.nn-1_amd64.deb (for 64-bit versions of Linux). Installation window will appear. Click the Install package. This will install the libximc library.



Click on the file xilab_n.nn-1_i386.deb (for 32-bit versions of Linux) or xilab_n.nn-1_amd64.deb (for 64-bit versions of Linux). Installation window will appear. Click the Install package. Dependencies and Xilab application will be installed.



The text installer

Execute the following commands as super-user (root):

```
gdebi "<FILEPATH>/libximc6_<LIBVERSION>-1_<ARCH>.deb"
gdebi "<FILEPATH>/xilab-<VERSION>-1_<ARCH>.deb"
```

where <FILEPATH> is the path to package files (eg "/home/user/Downloads/"), <LIBVERSION> and <VERSION> are the version numbers of libximc library and xilab application respectively (for example, "2.0.2" and "1.8.12") and <ARCH> is the identifier of architecture ("i386" for 32-bit systems and "amd64" for 64-bit systems).
Example:

```
gdebi "/home/user/Downloads/libximc6-2.0.2-1_amd64.deb"
gdebi "/home/user/Downloads/xilab-1.8.12-1_amd64.deb"
```

Xilab application requires X-server (graphic mode) for operation.

RedHat / OpenSUSE

The text installer

Execute the following commands as super-user (root):

```
zypper install "<FILEPATH>/libximc6-<LIBVERSION>-1.<ARCH>.rpm"
zypper install "<FILEPATH>/xilab-<VERSION>-1.<ARCH>.rpm"
```

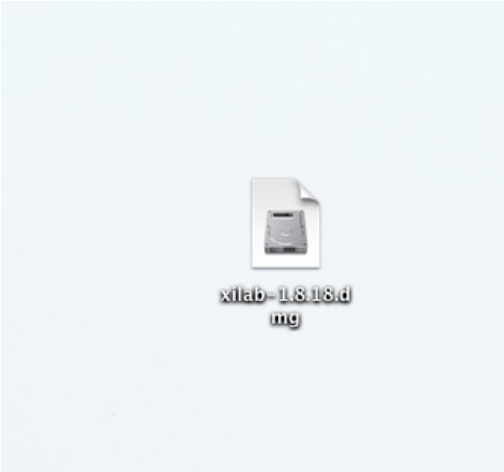
where <FILEPATH> is the path to package files (eg "/home/user/Downloads/"), <LIBVERSION> and <VERSION> are the version numbers of libximc library and Xilab application respectively (for example, "2.0.2" and "1.8.12") and <ARCH> is the identifier of architecture ("i386" for 32-bit systems and "amd64" for 64-bit systems).
Example:

```
zypper install "/home/user/Downloads/libximc6-2.0.2-1.x86_64.rpm"
zypper install "/home/user/Downloads/xilab-1.8.12-1.x86_64.rpm"
```

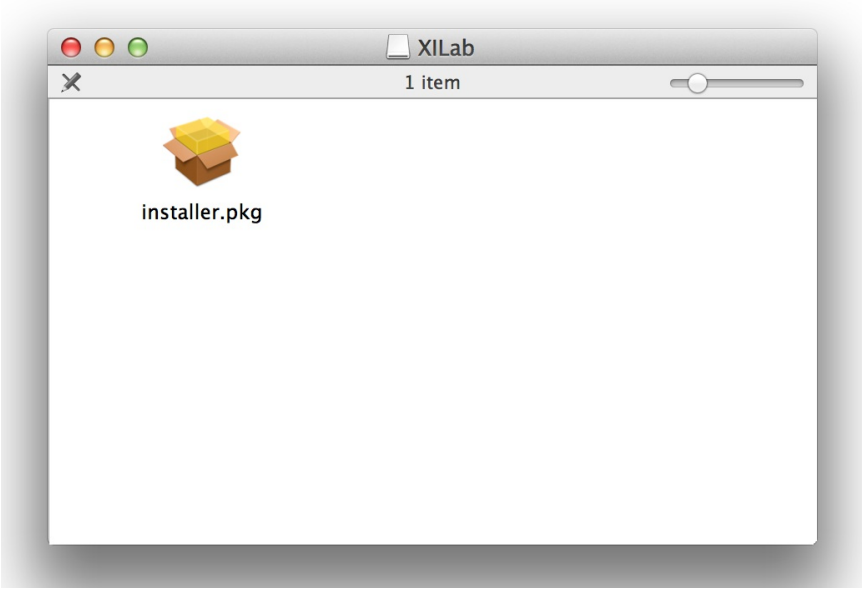
Xilab application requires X-server (graphic mode) for operation.

5.7.3. Installation on MacOS

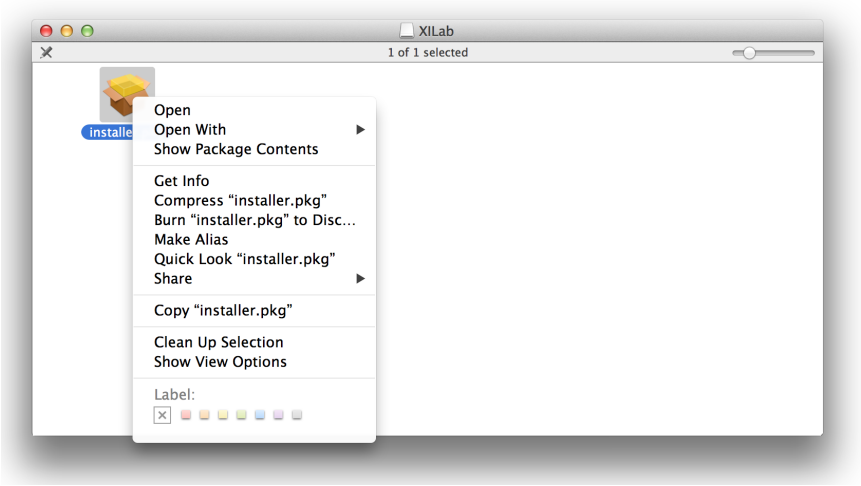
Copy the file with the installer archive to your computer. The installation program name is "xilab-vmxiosx64.tar.gz".



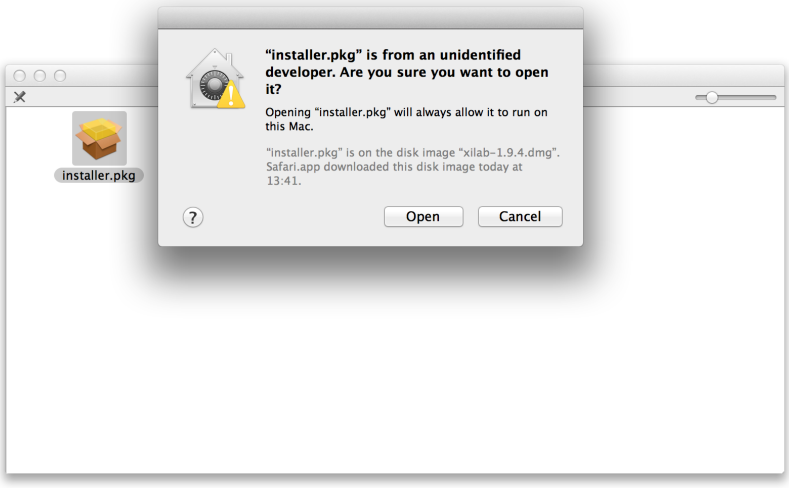
Unpack the archive by a mouse click.



Make right button click on installer.pkg.

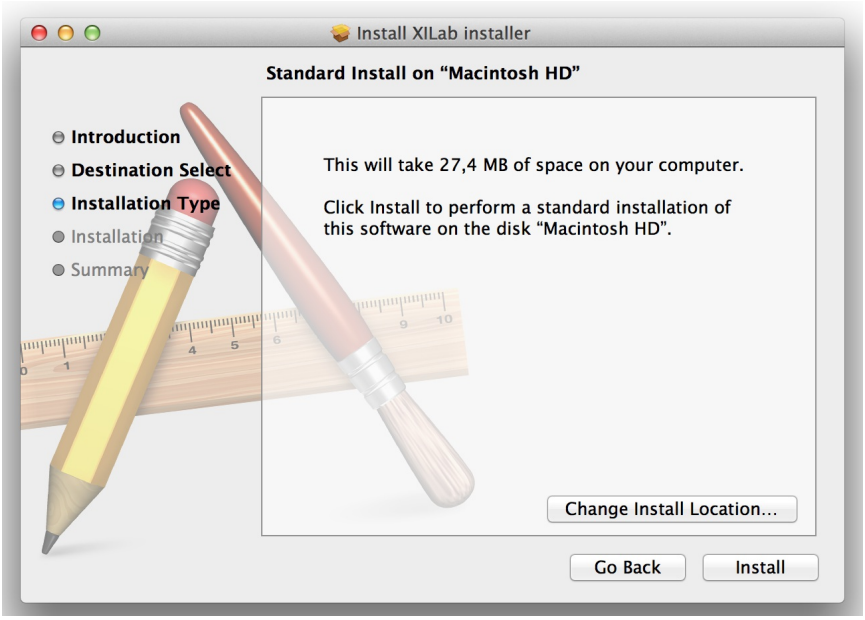


Choose "Open".

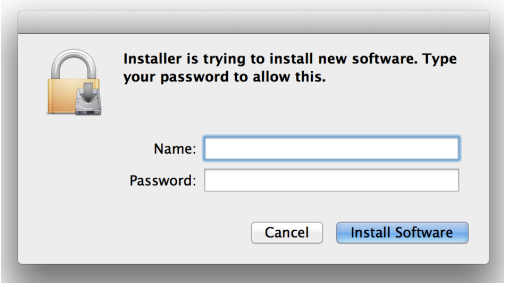


Choose "Open".

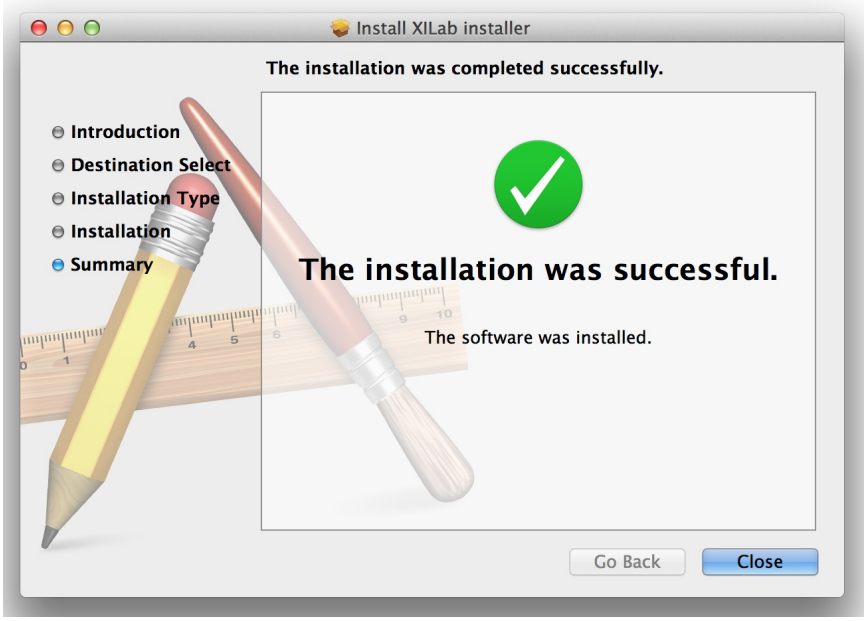
Select "Continue" in the main window of the installer.



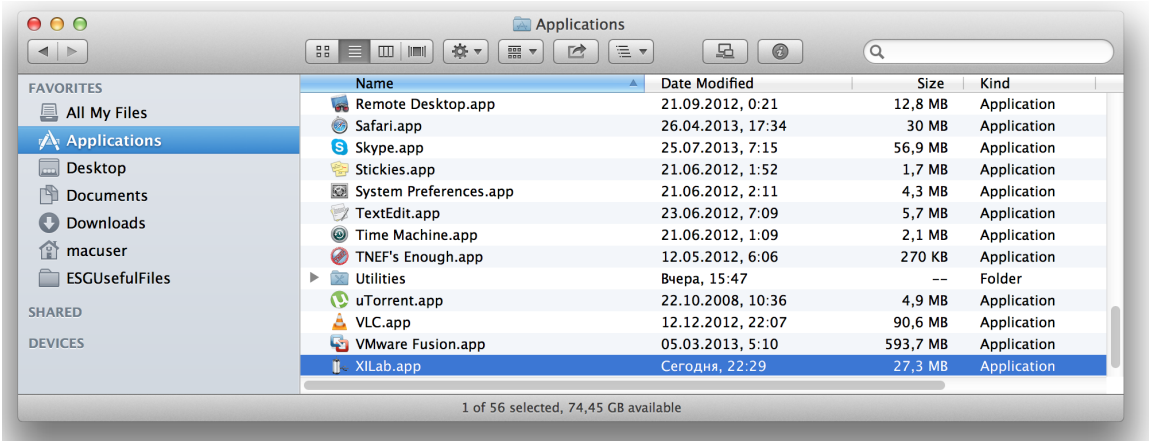
Now select "Install."



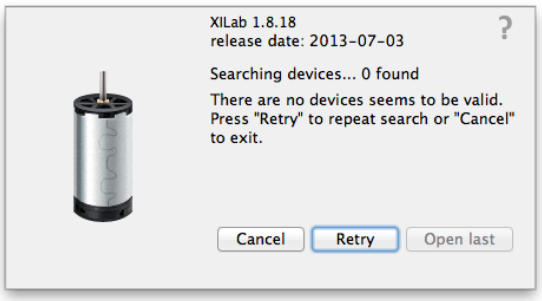
Enter the password.



Wait until the installation is complete.



Select the XILab application in the Programs block



Start it.

6. Programming

6.1. Programming guide

Programming guide is included in development kit libximc 2.X.X, where 2.X.X is the version number. It is located in /ximc-2.X.X/ximc/doc-en/libximc6-en.pdf. Development kit can be downloaded on Software page. Programming guide is Doxygen-based.

6.2. Communication protocol specification (v16.4)

Protocol description

Controller can be controlled from the PC using serial connection (COM-port). COM-port parameters are fixed controller-side:
Speed: 115200 baud;
Frame size: 8 bits;
Stop-bits: 2 bits;
Parity: none;
Flow control: none;
Byte receive timeout: 400 ms;
Bit order: little endian;

Byte order: little endian.

Command execution

All data transfers are initiated by the PC, meaning that the controller waits for incoming commands and replies accordingly. Each command is followed by the controller response, with rare exceptions of some service commands. One should not send another command without waiting for the previous command answer.
Commands are split into service, general control and general information types.
Commands are executed immediately. Parameters which are set by Sxxx commands are applied no later than 1ms after acknowledgement.
Command processing does not affect real-time engine control (PWM, encoder readout, etc).
Both controller and PC have an IO buffer. Received commands and command data are processed once and then removed from buffer.
Each command consists of 4-byte identifier and optionally a data section followed by its 2-byte CRC. Data can be transmitted in both directions, from PC to the controller and vice versa. Command is scheduled for execution if it is a legitimate command and (in case of data) if its CRC matches. After processing a correct command controller replies with 4 bytes - the name of processed command, followed by data and its 2-byte CRC, if the command is supposed to return data.

Controller-side error processing

Wrong command or data

If the controller receives a command that cannot be interpreted as a legitimate command, then controller ignores this command, replies with an "errc" string and sets "command error" flag in the current status data structure. If the unreconized command contained additional data, then it can be interpreted as new command(s). In this case resynchronization is required.
If the controller receives a valid command with data and its CRC doesn't match the CRC computed by the controller, then controller ignores this command, replies with an "errd" string and sets "data error" flag in the current status data structure. In this case synchronization is not needed.

CRC calculation

CRC is calculated for data only, 4-byte command identifier is not included. CRC algorithm in C is as follows:

```
unsigned short CRC16(INT8U *pbuf, unsigned short n)
{
  unsigned short crc, i, j, carry_flag, a;
  crc = 0xffff;
  for(i = 0; i < n; i++)
  {
    crc = crc ^ pbuf[i];
    for(j = 0; j < 8; j++)
    {
      a = crc;
      carry_flag = a & 0x0001;
      crc = crc >> 1;
      if ( carry_flag == 1 ) crc = crc ^ 0xa001;
    }
  }
  return crc;
}
```

This function receives a pointer to the data array, pbuf, and data length in bytes, n. It returns a two byte CRC code.

Transmission errors

Most probable transmission errors are missing, extra or altered byte. In usual settings transmission errors happen rarely, if at all.
Frequent errors are possible when using low-quality or broken USB-cable or board interconnection cable. Protocol is not designed for use in noisy environments and in rare cases an error may match a valid command code and get executed.

Missing byte, controller side
A missing byte on the controller side leads to a timeout on the PC side. Command is considered to be sent unsuccessfully by the PC. Synchronization is momentarily disrupted and restored after a timeout.

Missing byte, PC side
A missing byte on the PC side leads to a timeout on PC side. Synchronization is maintained.

Extra byte, controller side
An extra byte received by the controller leads to one or several "errc" or "errd" responses. Command is considered to be sent unsuccessfully by the PC. Receive buffer may also contain one or several "errc" or "errd" responses. Synchronization is disrupted.

Extra byte, PC side
An extra byte received by the PC leads to an incorrectly interpreted command or CRC and an extra byte in the receive buffer. Synchronization is disrupted.

Altered byte, controller side
An altered byte received by the controller leads to one or several "errc" or "errd" responses. Command is considered to be sent unsuccessfully by the PC. Receive buffer may also contain one or several "errc" or "errd" responses. Synchronization can rarely be disrupted, but is generally maintained.

Altered byte, PC side
An altered byte received by the PC leads to an incorrectly interpreted command or CRC. Synchronization is maintained.

Timeout resynchronization

If during packet reception next byte wait time exceeds timeout value, then partially received command is ignored and receive buffer is cleared. Controller timeout should be less than PC timeout, taking into account time it takes to transmit the data.

Zero byte resynchronization

There are no command codes that start with a zero byte ('\0'). This allows for a following synchronization procedure: controller always answers with a zero byte if the first command byte is zero, PC ignores first response byte if it is a zero byte. Then, if synchronization is disrupted on either side the following algorithm is used:
In case PC receives "errc", "errd" or a wrong command answer code, then PC sends 4 to 250 zeroes to the controller (250 byte limit is caused by input buffer length and usage of I2C protocol, less than 4 zeroes do not guarantee successful resynchronization). During this time PC continuously reads incoming bytes from the controller until the first zero is received and stops sending and receiving right after that.
Received zero byte is likely not a part of a response to a previous command because on error PC receives "errc"/"errd" response. It is possible in rare cases, then synchronization procedure will start again. Therefore first zero byte received by the PC means that controller input buffer is already empty and will remain so until any command is sent. Right after receiving first zero byte from the controller PC is ready to transmit next command code. The rest of zero bytes in transit will be ignored because they will be received before controller response.
This completes the zero byte synchronization procedure.

Library-side error processing

Nearly every library function has a return status of type `result_t`.
After sending command to the controller library reads incoming bytes until a non-zero byte is received. All zero bytes are ignored. Library reads first 4 bytes and compares them to the command code. It then waits for data section and CRC, if needed. If first 4 received bytes do not match the sent command identifier, then zero byte synchronization procedure is launched, command is considered to be sent unsuccessfully. If first 4 received bytes match the sent command identifier and command has data section, but the received CRC doesn't match CRC calculated from the received data, then zero byte synchronization procedure is launched, command is considered to be sent unsuccessfully. If a timeout is reached while the library is waiting for the controller response, then zero byte synchronization procedure is launched, command is considered to be sent unsuccessfully.
If no errors were detected, then command is considered to be successfully completed and `result_ok` is returned.

Library return codes

- `result_ok`. No errors detected.
- `result_error`. Generic error. Can happen because of hardware problems, empty port buffer, timeout or successfull synchronization after an error. Another common reason for this error is protocol version mismatch between controller firmware and PC library.
- `result_nodevice`. Error opening device, lost connection or failed synchronization. Device reopen and/or user action is required.

If a function returns an error values of all parameters it writes to are undefined. Error code may be accompanied by detailed error description output to system log (Unix-like OS) or standard error (Windows-like OS).

Zero byte synchronization procedure

Synchronization is performed by means of sending zero ('\0') bytes and reading bytes until a zero byte is received. Optionally one may clear port buffer at the end of synchronization procedure. Initially 64 zero bytes are sent. If there

were no zero bytes received during the timeout, then a string of 64 bytes is sent 3 more times. After 4 unsuccessful attempts and no zero bytes received device is considered lost. In this case library should return `result_nodevice` error code. In case of successful syncronization library returns `result_error`.

Controller settings setup

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

Command SFBS

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

Command code (CMD): "sfbs" or 0x73626673.

Request: (18 bytes)

INT32U	CMD	Command
INT16U	IPS	The number of measured counts per revolution encoder
INT8U	FeedbackType	Type of feedback
		0x00 - FEEDBACK_POTENTIOMETER (Feedback by potentiometer.)
		0x01 - FEEDBACK_ENCODER (Feedback by encoder.)
		0x02 - FEEDBACK_ENCODERDIFF (Feedback by encoder with differential input.)
		0x03 - FEEDBACK_ENCODERHALL (Feedback by Hall detector.)
		0x04 - FEEDBACK_EMF (Feedback by EMF.)
		0x05 - FEEDBACK_NONE (Feedback is absent.)
INT8U	FeedbackFlags	Flags
		0x01 - FEEDBACK_ENC_REVERSE (Reverse count of encoder. Used with motor type ENGINE_TYPE_STEP_ENC only.)
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set feedback settings.

Command GFBS

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

Command code (CMD): "gfbs" or 0x73626667.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (18 bytes)

INT32U	CMD	Command (answer)
INT16U	IPS	The number of measured counts per revolution encoder
INT8U	FeedbackType	Type of feedback
		0x00 - FEEDBACK_POTENTIOMETER (Feedback by potentiometer.)
		0x01 - FEEDBACK_ENCODER (Feedback by encoder.)
		0x02 - FEEDBACK_ENCODERDIFF (Feedback by encoder with differential input.)
		0x03 - FEEDBACK_ENCODERHALL (Feedback by Hall detector.)
		0x04 - FEEDBACK_EMF (Feedback by EMF.)

		0x05 - FEEDBACK_NONE (Feedback is absent.)
INT8U	FeedbackFlags	Flags
		0x01 - FEEDBACK_ENC_REVERSE (Reverse count of encoder. Used with motor type ENGINE_TYPE_STEP_ENC only.)
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Description:
Read feedback settings.

Command SHOM

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

Command code (CMD): "shom" or 0x6D6F6873.

Request: (33 bytes)

INT32U	CMD	Command
INT32U	FastHome	Speed used for first motion. Range: 0..1000000.
INT8U	uFastHome	Part of the speed for first motion, microsteps. Range: 0..255.
INT32U	SlowHome	Speed used for second motion. Range: 0..1000000.
INT8U	uSlowHome	Part of the speed for second motion, microsteps. Range: 0..255.
INT32S	HomeDelta	Distance from break point. Range: -2147483647..2147483647.
INT16S	uHomeDelta	Part of the delta distance, microsteps. Range: -255..255.
INT16U	HomeFlags	Set of flags specify direction and stopping conditions.
		0x01 - HOME_DIR_FIRST (Flag defines direction of 1st motion after execution of home command. Direction is right, if set; otherwise left.)
		0x02 - HOME_DIR_SECOND (Flag defines direction of 2nd motion. Direction is right, if set; otherwise left.)
		0x04 - HOME_MV_SEC_EN (Use the second phase of calibration to the home position, if set; otherwise the second phase is skipped.)
		0x08 - HOME_HALF_MV (If the flag is set, the stop signals are ignored in start of second movement the first half-turn.)
		0x30 - HOME_STOP_FIRST_BITS (Bits of the first stop selector.)
		0x10 - HOME_STOP_FIRST_REV (First motion stops by revolution sensor.)
		0x20 - HOME_STOP_FIRST_SYN (First motion stops by synchronization input.)
		0x30 - HOME_STOP_FIRST_LIM (First motion stops by limit switch.)
		0xC0 - HOME_STOP_SECOND_BITS (Bits of the second stop selector.)
		0x40 - HOME_STOP_SECOND_REV (Second motion stops by revolution sensor.)
		0x80 - HOME_STOP_SECOND_SYN (Second motion stops by synchronization input.)
		0xC0 - HOME_STOP_SECOND_LIM (Second motion stops by limit switch.)
INT8U	Reserved [9]	Reserved (9 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set home settings. This function send structure with calibrating position settings to controller's memory.

Command GHOM

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

Command code (CMD): "ghom" or 0x6D6F6867.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (33 bytes)

INT32U	CMD	Command (answer)
INT32U	FastHome	Speed used for first motion. Range: 0..1000000.
INT8U	uFastHome	Part of the speed for first motion, microsteps. Range: 0..255.
INT32U	SlowHome	Speed used for second motion. Range: 0..1000000.
INT8U	uSlowHome	Part of the speed for second motion, microsteps. Range: 0..255.
INT32S	HomeDelta	Distance from break point. Range: -2147483647..2147483647.
INT16S	uHomeDelta	Part of the delta distance, microsteps. Range: -255..255.
INT16U	HomeFlags	Set of flags specify direction and stopping conditions.
		0x01 - HOME_DIR_FIRST (Flag defines direction of 1st motion after execution of home command. Direction is right, if set; otherwise left.)
		0x02 - HOME_DIR_SECOND (Flag defines direction of 2nd motion. Direction is right, if set; otherwise left.)
		0x04 - HOME_MV_SEC_EN (Use the second phase of calibration to the home position, if set; otherwise the second phase is skipped.)
		0x08 - HOME_HALF_MV (If the flag is set, the stop signals are ignored in start of second movement the first half-turn.)
		0x30 - HOME_STOP_FIRST_BITS (Bits of the first stop selector.)
		0x10 - HOME_STOP_FIRST_REV (First motion stops by revolution sensor.)
		0x20 - HOME_STOP_FIRST_SYN (First motion stops by synchronization input.)
		0x30 - HOME_STOP_FIRST_LIM (First motion stops by limit switch.)
		0xC0 - HOME_STOP_SECOND_BITS (Bits of the second stop selector.)
		0x40 - HOME_STOP_SECOND_REV (Second motion stops by revolution sensor.)
		0x80 - HOME_STOP_SECOND_SYN (Second motion stops by synchronization input.)
		0xC0 - HOME_STOP_SECOND_LIM (Second motion stops by limit switch.)
INT8U	Reserved [9]	Reserved (9 bytes)
INT16U	CRC	Checksum

Description:
Read home settings. This function fill structure with settings of calibrating position.

Command SMOV

```
result_t set_move_settings (device_t id, const move_settings_t* move_settings)
```

Command code (CMD): "smov" or 0x766F6D73.

Request: (30 bytes)

INT32U	CMD	Command
INT32U	Speed	Target speed(for stepper motor: steps / c, for DC: rpm). Range: 0..1000000.
INT8U	uSpeed	Target speed in 1/256 microsteps/s. Using with stepper motor only. Range: 0..255.
INT16U	Accel	Motor shaft acceleration, steps/s^2(stepper motor) or RPM/s(DC). Range: 0..65535.
INT16U	Decel	Motor shaft deceleration, steps/s^2(stepper motor) or RPM/s(DC). Range: 0..65535.
INT32U	AntiplaySpeed	Speed in antiplay mode, full steps/s(stepper motor) or RPM. Range: 0..1000000.
INT8U	uAntiplaySpeed	Speed in antiplay mode, 1/256 microsteps/s. Used with stepper motor only. Range: 0..255.
INT8U	Reserved [10]	Reserved (10 bytes)

INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command GMOV

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

Command code (CMD): "gmov" or 0x766F6D67.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (30 bytes)

INT32U	CMD	Command (answer)
INT32U	Speed	Target speed(for stepper motor: steps / c, for DC: rpm). Range: 0..1000000.
INT8U	uSpeed	Target speed in 1/256 microsteps/s. Using with stepper motor only. Range: 0..255.
INT16U	Accel	Motor shaft acceleration, steps/s^2(stepper motor) or RPM/s(DC). Range: 0..65535.
INT16U	Decel	Motor shaft deceleration, steps/s^2(stepper motor) or RPM/s(DC). Range: 0..65535.
INT32U	AntiplaySpeed	Speed in antiplay mode, full steps/s(stepper motor) or RPM. Range: 0..1000000.
INT8U	uAntiplaySpeed	Speed in antiplay mode, 1/256 microsteps/s. Used with stepper motor only. Range: 0..255.
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

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

Command SENG

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

Command code (CMD): "seng" or 0x676E6573.

Request: (34 bytes)

INT32U	CMD	Command
INT16U	NomVoltage	Rated voltage. Controller will keep the voltage drop on motor below this value if ENGINE_LIMIT_VOLT flag is set(Used with DC only). Range: 1..65535
INT16U	NomCurrent	Rated current. Controller will keep current consumed by motor below this value if ENGINE_LIMIT_CURR flag is set. Range: 1..65535
INT32U	NomSpeed	Nominal speed (in whole steps / s or rpm for DC and stepper motor as a master encoder). Controller will keep motor shaft RPM below this value if ENGINE_LIMIT_RPM flag is set. Range: 1..1000000.
INT8U	uNomSpeed	The fractional part of a nominal speed in microsteps (is only used with stepper motor). Range: 0..255
INT16U	EngineFlags	Set of flags specify motor shaft movement algorithm and list of limitations
		0x01 - ENGINE_REVERSE (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.)
		0x04 - ENGINE_MAX_SPEED (Max speed flag. If it is set, engine uses maxumum speed achievable with the present engine settings as nominal speed.)
		0x08 - ENGINE_ANTIPLAY (Play compensation flag. If it set, engine makes backlash (play)

		compensation procedure and reach the predetermined position accurately on low speed.)
		0x10 - ENGINE_ACCEL_ON (Acceleration enable flag. If it set, motion begins with acceleration and ends with deceleration.)
		0x20 - ENGINE_LIMIT_VOLT (Maxumum motor voltage limit enable flag(is only used with DC motor).)
		0x40 - ENGINE_LIMIT_CURR (Maxumum motor current limit enable flag(is only used with DC motor).)
		0x80 - ENGINE_LIMIT_RPM (Maxumum motor speed limit enable flag.)
INT16S	Antiplay	Number of pulses or steps for backlash (play) compensation procedure. Used if ENGINE_ANTIPLAY flag is set. Range: -32768..32767
INT8U	MicrostepMode	Settings of microstep mode(Used with steper motor only).
		0x01 - MICROSTEP_MODE_FULL (Full step mode.)
		0x02 - MICROSTEP_MODE_FRAC_2 (1/2 step mode.)
		0x03 - MICROSTEP_MODE_FRAC_4 (1/4 step mode.)
		0x04 - MICROSTEP_MODE_FRAC_8 (1/8 step mode.)
		0x05 - MICROSTEP_MODE_FRAC_16 (1/16 step mode.)
		0x06 - MICROSTEP_MODE_FRAC_32 (1/32 step mode.)
		0x07 - MICROSTEP_MODE_FRAC_64 (1/64 step mode.)
		0x08 - MICROSTEP_MODE_FRAC_128 (1/128 step mode.)
		0x09 - MICROSTEP_MODE_FRAC_256 (1/256 step mode.)
INT16U	StepsPerRev	Number of full steps per revolution(Used with steper motor only). Range: 1..65535.
INT8U	Reserved [12]	Reserved (12 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command GENG

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

Command code (CMD): "geng" or 0x676E6567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (34 bytes)

INT32U	CMD	Command (answer)
INT16U	NomVoltage	Rated voltage. Controller will keep the voltage drop on motor below this value if ENGINE_LIMIT_VOLT flag is set(Used with DC only). Range: 1..65535
INT16U	NomCurrent	Rated current. Controller will keep current consumed by motor below this value if ENGINE_LIMIT_CURR flag is set. Range: 1..65535
INT32U	NomSpeed	Nominal speed (in whole steps / s or rpm for DC and stepper motor as a master encoder). Controller will keep motor shaft RPM below this value if ENGINE_LIMIT_RPM flag is set. Range: 1..1000000.
INT8U	uNomSpeed	The fractional part of a nominal speed in microsteps (is only used with stepper motor). Range: 0..255
INT16U	EngineFlags	Set of flags specify motor shaft movement algorithm and list of limitations

		0x01 - ENGINE_REVERSE (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.)
		0x04 - ENGINE_MAX_SPEED (Max speed flag. If it is set, engine uses maxumum speed achievable with the present engine settings as nominal speed.)
		0x08 - ENGINE_ANTIPLAY (Play compensation flag. If it set, engine makes backlash (play) compensation procedure and reach the predetermined position accurately on low speed.)
		0x10 - ENGINE_ACCEL_ON (Acceleration enable flag. If it set, motion begins with acceleration and ends with deceleration.)
		0x20 - ENGINE_LIMIT_VOLT (Maxumum motor voltage limit enable flag(is only used with DC motor).)
		0x40 - ENGINE_LIMIT_CURR (Maxumum motor current limit enable flag(is only used with DC motor).)
		0x80 - ENGINE_LIMIT_RPM (Maxumum motor speed limit enable flag.)
INT16S	Antiplay	Number of pulses or steps for backlash (play) compensation procedure. Used if ENGINE_ANTIPLAY flag is set. Range: -32768..32767
INT8U	MicrostepMode	Settings of microstep mode(Used with steper motor only).
		0x01 - MICROSTEP_MODE_FULL (Full step mode.)
		0x02 - MICROSTEP_MODE_FRAC_2 (1/2 step mode.)
		0x03 - MICROSTEP_MODE_FRAC_4 (1/4 step mode.)
		0x04 - MICROSTEP_MODE_FRAC_8 (1/8 step mode.)
		0x05 - MICROSTEP_MODE_FRAC_16 (1/16 step mode.)
		0x06 - MICROSTEP_MODE_FRAC_32 (1/32 step mode.)
		0x07 - MICROSTEP_MODE_FRAC_64 (1/64 step mode.)
		0x08 - MICROSTEP_MODE_FRAC_128 (1/128 step mode.)
		0x09 - MICROSTEP_MODE_FRAC_256 (1/256 step mode.)
INT16U	StepsPerRev	Number of full steps per revolution(Used with steper motor only). Range: 1..65535.
INT8U	Reserved [12]	Reserved (12 bytes)
INT16U	CRC	Checksum

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

Command SENT

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

Command code (CMD): "sent" or 0x746E6573.

Request: (14 bytes)

INT32U	CMD	Command
INT8U	EngineType	Engine type
		0x00 - ENGINE_TYPE_NONE (A value that shouldn't be used.)
		0x01 - ENGINE_TYPE_DC (DC motor.)
		0x02 - ENGINE_TYPE_2DC (2 DC motors.)
		0x03 - ENGINE_TYPE_STEP (Step motor.)
		0x05 - ENGINE_TYPE_BRUSHLESS (Brushless motor.)
INT8U	DriverType	Driver type
		0x01 - DRIVER_TYPE_DISCRETE_FET (Driver with discrete FET keys. Default option.)
		0x02 - DRIVER_TYPE_INTEGRATE (Driver with integrated IC.)
		0x03 - DRIVER_TYPE_EXTERNAL (External driver.)

INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set engine type and driver type.

Command GENT

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

Command code (CMD): "gent" or 0x746E6567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (14 bytes)

INT32U	CMD	Command (answer)
INT8U	EngineType	Engine type
		0x00 - ENGINE_TYPE_NONE (A value that shouldn't be used.)
		0x01 - ENGINE_TYPE_DC (DC motor.)
		0x02 - ENGINE_TYPE_2DC (2 DC motors.)
		0x03 - ENGINE_TYPE_STEP (Step motor.)
		0x05 - ENGINE_TYPE_BRUSHLESS (Brushless motor.)
INT8U	DriverType	Driver type
		0x01 - DRIVER_TYPE_DISCRETE_FET (Driver with discrete FET keys. Default option.)
		0x02 - DRIVER_TYPE_INTEGRATE (Driver with integrated IC.)
		0x03 - DRIVER_TYPE_EXTERNAL (External driver.)
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Description:
Return engine type and driver type.

Command SPWR

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

Command code (CMD): "spwr" or 0x72777073.

Request: (20 bytes)

INT32U	CMD	Command
INT8U	HoldCurrent	Current in holding regime, percent of nominal. Range: 0..100.
INT16U	CurrReductDelay	Time in ms from going to STOP state to reducting current. Range: 0..65535.
INT16U	PowerOffDelay	Time in s from going to STOP state to turning power off. Range: 0..65535.
INT16U	CurrentSetTime	Time in ms to reach nominal current. Range: 0..65535.
INT8U	PowerFlags	Flags with parameters of power control.
		0x01 - POWER_REDUCT_ENABLED (Current reduction enabled after CurrReductDelay, if this flag is set.)
		0x02 - POWER_OFF_ENABLED (Power off enabled after PowerOffDelay, if this flag is set.)
		0x04 - POWER_SMOOTH_CURRENT (Current ramp-up/down is performed smoothly during current_set_time, if this flag is set.)

INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set settings of step motor power control. Used with stepper motor only.

Command GPWR

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

Command code (CMD): "gpwr" or 0x72777067.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (20 bytes)

INT32U	CMD	Command (answer)
INT8U	HoldCurrent	Current in holding regime, percent of nominal. Range: 0..100.
INT16U	CurrReductDelay	Time in ms from going to STOP state to reducting current. Range: 0..65535.
INT16U	PowerOffDelay	Time in s from going to STOP state to turning power off. Range: 0..65535.
INT16U	CurrentSetTime	Time in ms to reach nominal current. Range: 0..65535.
INT8U	PowerFlags	Flags with parameters of power control.
		0x01 - POWER_REDUCT_ENABLED (Current reduction enabled after CurrReductDelay, if this flag is set.)
		0x02 - POWER_OFF_ENABLED (Power off enabled after PowerOffDelay, if this flag is set.)
		0x04 - POWER_SMOOTH_CURRENT (Current ramp-up/down is performed smoothly during current_set_time, if this flag is set.)
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Description:
Read settings of step motor power control. Used with stepper motor only.

Command SSEC

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

Command code (CMD): "ssec" or 0x63657373.

Request: (28 bytes)

INT32U	CMD	Command
INT16U	LowUpwrOff	Lower voltage limit to turn off the motor, in mV. Range: 0..65535.
INT16U	CriticalIpwr	Maximum motor current which triggers ALARM state, in mA. Range: 0..65535.
INT16U	CriticalUpwr	Maximum motor voltage which triggers ALARM state, in mV. Range: 0..65535.
INT16U	CriticalT	Maximum temperature, which triggers ALARM state, in tenths of degrees Celcius. Range: 0..65535.
INT16U	CriticalIusb	Maximum USB current which triggers ALARM state, in mA. Range: 0..65535.
INT16U	CriticalUusb	Maximum USB voltage which triggers ALARM state, in mV. Range: 0..65535.
INT16U	MinimumUusb	Minimum USB voltage which triggers ALARM state, in mV. Range: 0..65535.
INT8U	Flags	Critical parameter flags.
		0x01 - ALARM_ON_DRIVER_OVERHEATING (If this flag is set enter Alarm state on driver

		overheat signal.)
		0x02 - LOW_UPWR_PROTECTION (If this flag is set turn off motor when voltage is lower than LowUpwrOff.)
		0x04 - H_BRIDGE_ALERT (If this flag is set then turn off the power unit with a signal problem in one of the transistor bridge.)
		0x08 - ALARM_ON_BORDERS_SWAP_MISSET (If this flag is set enter Alarm state on borders swap misset)
		0x10 - ALARM_FLAGS_STICKING (If this flag is set only a STOP command can turn all alarms to 0)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set protection settings.

Command GSEC

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

Command code (CMD): "gsec" or 0x63657367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (28 bytes)

INT32U	CMD	Command (answer)
INT16U	LowUpwrOff	Lower voltage limit to turn off the motor, in mV. Range: 0..65535.
INT16U	CriticalIpwr	Maximum motor current which triggers ALARM state, in mA. Range: 0..65535.
INT16U	CriticalUpwr	Maximum motor voltage which triggers ALARM state, in mV. Range: 0..65535.
INT16U	CriticalT	Maximum temperature, which triggers ALARM state, in tenths of degrees Celcius. Range: 0..65535.
INT16U	CriticalIusb	Maximum USB current which triggers ALARM state, in mA. Range: 0..65535.
INT16U	CriticalUusb	Maximum USB voltage which triggers ALARM state, in mV. Range: 0..65535.
INT16U	MinimumUusb	Minimum USB voltage which triggers ALARM state, in mV. Range: 0..65535.
INT8U	Flags	Critical parameter flags.
		0x01 - ALARM_ON_DRIVER_OVERHEATING (If this flag is set enter Alarm state on driver overheat signal.)
		0x02 - LOW_UPWR_PROTECTION (If this flag is set turn off motor when voltage is lower than LowUpwrOff.)
		0x04 - H_BRIDGE_ALERT (If this flag is set then turn off the power unit with a signal problem in one of the transistor bridge.)
		0x08 - ALARM_ON_BORDERS_SWAP_MISSET (If this flag is set enter Alarm state on borders swap misset)
		0x10 - ALARM_FLAGS_STICKING (If this flag is set only a STOP command can turn all alarms to 0)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Description:
Read protection settings.

Command SEDS

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

Command code (CMD): "seds" or 0x73646573.

Request: (26 bytes)

INT32U	CMD	Command
INT8U	BorderFlags	Border flags, specify types of borders and motor behaviour on borders.
		0x01 - BORDER_IS_ENCODER (Borders are fixed by predetermined encoder values, if set; borders position on limit switches, if not set.)
		0x02 - BORDER_STOP_LEFT (Motor should stop on left border.)
		0x04 - BORDER_STOP_RIGHT (Motor should stop on right border.)
		0x08 - BORDERS_SWAP_MISSET_DETECTION (Motor should stop on both borders. Need to save motor then wrong border settings is se)
INT8U	EnderFlags	Ender flags, specify electrical behaviour of limit switches like order and pulled positions.
		0x01 - ENDER_SWAP (First limit switch on the right side, if set; otherwise on the left side.)
		0x02 - ENDER_SW1_ACTIVE_LOW (1 - Limit switch connnected to pin SW1 is triggered by a low level on pin.)
		0x04 - ENDER_SW2_ACTIVE_LOW (1 - Limit switch connnected to pin SW2 is triggered by a low level on pin.)
INT32S	LeftBorder	Left border position, used if BORDER_IS_ENCODER flag is set. Range: -2147483647..2147483647.
INT16S	uLeftBorder	Left border position in 1/256 microsteps(used with stepper motor only). Range: -255..255.
INT32S	RightBorder	Right border position, used if BORDER_IS_ENCODER flag is set. Range: -2147483647..2147483647.
INT16S	uRightBorder	Right border position in 1/256 microsteps. Range: -255..255(used with stepper motor only).
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set border and limit switches settings.

Command GEDS

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

Command code (CMD): "geds" or 0x73646567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (26 bytes)

INT32U	CMD	Command (answer)
INT8U	BorderFlags	Border flags, specify types of borders and motor behaviour on borders.
		0x01 - BORDER_IS_ENCODER (Borders are fixed by predetermined encoder values, if set; borders position on limit switches, if not set.)
		0x02 - BORDER_STOP_LEFT (Motor should stop on left border.)
		0x04 - BORDER_STOP_RIGHT (Motor should stop on right border.)
		0x08 - BORDERS_SWAP_MISSET_DETECTION (Motor should stop on both borders. Need to save motor then wrong border settings is se)
INT8U	EnderFlags	Ender flags, specify electrical behaviour of limit switches like order and pulled positions.

		0x01 - ENDER_SWAP (First limit switch on the right side, if set; otherwise on the left side.)
		0x02 - ENDER_SW1_ACTIVE_LOW (1 - Limit switch connnected to pin SW1 is triggered by a low level on pin.)
		0x04 - ENDER_SW2_ACTIVE_LOW (1 - Limit switch connnected to pin SW2 is triggered by a low level on pin.)
INT32S	LeftBorder	Left border position, used if BORDER_IS_ENCODER flag is set. Range: -2147483647..2147483647.
INT16S	uLeftBorder	Left border position in 1/256 microsteps(used with stepper motor only). Range: -255..255.
INT32S	RightBorder	Right border position, used if BORDER_IS_ENCODER flag is set. Range: -2147483647..2147483647.
INT16S	uRightBorder	Right border position in 1/256 microsteps. Range: -255..255(used with stepper motor only).
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Description:
Read border and limit switches settings.

Command SPID

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

Command code (CMD): "spid" or 0x64697073.

Request: (48 bytes)

INT32U	CMD	Command
INT16U	KpU	Proportional gain for voltage PID routine
INT16U	KiU	Integral gain for voltage PID routine
INT16U	KdU	Differential gain for voltage PID routine
INT8U	Reserved [36]	Reserved (36 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command GPID

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

Command code (CMD): "gpid" or 0x64697067.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (48 bytes)

INT32U	CMD	Command (answer)
INT16U	KpU	Proportional gain for voltage PID routine
INT16U	KiU	Integral gain for voltage PID routine
INT16U	KdU	Differential gain for voltage PID routine
INT8U	Reserved [36]	Reserved (36 bytes)

INT16U	CRC	Checksum

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

Command SSNI

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

Command code (CMD): "ssni" or 0x696E7373.

Request: (28 bytes)

INT32U	CMD	Command
INT8U	SyncInFlags	Input synchronization flags
		0x01 - SYNCIN_ENABLED (Synchronization in mode is enabled, if this flag is set.)
		0x02 - SYNCIN_INVERT (Trigger on falling edge if flag is set, on rising edge otherwise.)
		0x04 - SYNCIN_GOTOPOSITION (The engine is go to position specified in Position and uPosition, if this flag is set. And it is shift on the Position and uPosition, if this flag is unset)
INT16U	ClutterTime	Input synchronization pulse dead time (mks). Range: 0..65535
INT32S	Position	Desired position or shift (whole steps)
INT16S	uPosition	The fractional part of a position or shift in microsteps (-255 .. 255)(is only used with stepper motor)
INT32U	Speed	Target speed(for stepper motor: steps / c, for DC: rpm). Range: 0..1000000.
INT8U	uSpeed	Target speed in microsteps/s. Using with stepper motor only. Range: 0..255.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command GSNI

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

Command code (CMD): "gsni" or 0x696E7367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (28 bytes)

INT32U	CMD	Command (answer)
INT8U	SyncInFlags	Input synchronization flags
		0x01 - SYNCIN_ENABLED (Synchronization in mode is enabled, if this flag is set.)
		0x02 - SYNCIN_INVERT (Trigger on falling edge if flag is set, on rising edge otherwise.)
		0x04 - SYNCIN_GOTOPOSITION (The engine is go to position specified in Position and uPosition, if this flag is set. And it is shift on the Position and uPosition, if this flag is unset)
INT16U	ClutterTime	Input synchronization pulse dead time (mks). Range: 0..65535
INT32S	Position	Desired position or shift (whole steps)

INT16S	uPosition	The fractional part of a position or shift in microsteps (-255 .. 255)(is only used with stepper motor)
INT32U	Speed	Target speed(for stepper motor: steps / c, for DC: rpm). Range: 0..1000000.
INT8U	uSpeed	Target speed in microsteps/s. Using with stepper motor only. Range: 0..255.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

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

Command SSNO

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

Command code (CMD): "ssno" or 0x6F6E7373.

Request: (16 bytes)

INT32U	CMD	Command
INT8U	SyncOutFlags	Output synchronization flags
		0x01 - SYNCOUT_ENABLED (Synchronization out pin follows the synchronization logic, if set. It governed by SYNCOUT_STATE flag otherwise.)
		0x02 - SYNCOUT_STATE (When output state is fixed by negative SYNCOUT_ENABLED flag, the pin state is in accordance with this flag state.)
		0x04 - SYNCOUT_INVERT (Low level is active, if set, and high level is active otherwise.)
		0x08 - SYNCOUT_IN_STEPS (Use motor steps/encoder pulses instead of milliseconds for output pulse generation if the flag is set.)
		0x10 - SYNCOUT_ONSTART (Generate synchronization pulse when movement starts.)
		0x20 - SYNCOUT_ONSTOP (Generate synchronization pulse when movement stops.)
		0x40 - SYNCOUT_ONPERIOD (Generate synchronization pulse every SyncOutPeriod encoder pulses.)
INT16U	SyncOutPulseSteps	This value specifies duration of output pulse. It is measured microseconds when SYNCOUT_IN_STEPS flag is cleared or in encoder pulses or motor steps when SYNCOUT_IN_STEPS is set. Range: 0..65535
INT16U	SyncOutPeriod	This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT_ONPERIOD is set. Range: 0..65535
INT32U	Accuracy	This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop. Range: 0..4294967295.
INT8U	uAccuracy	This is the neighborhood around the target coordinates in micro steps (only used with stepper motor). Range: 0 .. 255.
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command GSNO

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

Command code (CMD): "gsno" or 0x6F6E7367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (16 bytes)

INT32U	CMD	Command (answer)
INT8U	SyncOutFlags	Output synchronization flags
		0x01 - SYNCOUT_ENABLED (Synchronization out pin follows the synchronization logic, if set. It governed by SYNCOUT_STATE flag otherwise.)
		0x02 - SYNCOUT_STATE (When output state is fixed by negative SYNCOUT_ENABLED flag, the pin state is in accordance with this flag state.)
		0x04 - SYNCOUT_INVERT (Low level is active, if set, and high level is active otherwise.)
		0x08 - SYNCOUT_IN_STEPS (Use motor steps/encoder pulses instead of milliseconds for output pulse generation if the flag is set.)
		0x10 - SYNCOUT_ONSTART (Generate synchronization pulse when movement starts.)
		0x20 - SYNCOUT_ONSTOP (Generate synchronization pulse when movement stops.)
		0x40 - SYNCOUT_ONPERIOD (Generate synchronization pulse every SyncOutPeriod encoder pulses.)
INT16U	SyncOutPulseSteps	This value specifies duration of output pulse. It is measured microseconds when SYNCOUT_IN_STEPS flag is cleared or in encoder pulses or motor steps when SYNCOUT_IN_STEPS is set. Range: 0..65535
INT16U	SyncOutPeriod	This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT_ONPERIOD is set. Range: 0..65535
INT32U	Accuracy	This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop. Range: 0..4294967295.
INT8U	uAccuracy	This is the neighborhood around the target coordinates in micro steps (only used with stepper motor). Range: 0 .. 255.
INT16U	CRC	Checksum

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

Command SEIO

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

Command code (CMD): "seio" or 0x6F696573.

Request: (18 bytes)

INT32U	CMD	Command
INT8U	EXTIOSetupFlags	Configuration flags of the external I-O
		0x01 - EXTIO_SETUP_OUTPUT (EXTIO works as output if flag is set, works as input otherwise.)
		0x02 - EXTIO_SETUP_INVERT (Interpret EXTIO states and fronts inverted if flag is set. Falling front as input event and low logic level as active state.)
INT8U	EXTIOModeFlags	Flags mode settings external I-O
		0x00 - EXTIO_SETUP_MODE_IN_NOP (Do nothing.)
		0x01 - EXTIO_SETUP_MODE_IN_STOP (Issue STOP command, ceasing the engine movement.)
		0x02 - EXTIO_SETUP_MODE_IN_PWOF (Issue PWOF command, powering off all engine windings.)
		0x03 - EXTIO_SETUP_MODE_IN_MOVR (Issue MOVR command with last used settings.)
		0x04 - EXTIO_SETUP_MODE_IN_HOME (Issue HOME command.)
		0x00 - EXTIO_SETUP_MODE_OUT_OFF (EXTIO pin always set in inactive state.)

		0x10 - EXTIO_SETUP_MODE_OUT_ON (EXTIO pin always set in active state.)
		0x20 - EXTIO_SETUP_MODE_OUT_MOVING (EXTIO pin stays active during moving state.)
		0x30 - EXTIO_SETUP_MODE_OUT_ALARM (EXTIO pin stays active during Alarm state.)
		0x40 - EXTIO_SETUP_MODE_OUT_MOTOR_ON (EXTIO pin stays active when windings are powered.)
		0x50 - EXTIO_SETUP_MODE_OUT_MOTOR_FOUND (EXTIO pin stays active when motor is connected (first winding).)
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command GEIO

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

Command code (CMD): "geio" or 0x6F696567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (18 bytes)

INT32U	CMD	Command (answer)
INT8U	EXTIOSetupFlags	Configuration flags of the external I-O
		0x01 - EXTIO_SETUP_OUTPUT (EXTIO works as output if flag is set, works as input otherwise.)
		0x02 - EXTIO_SETUP_INVERT (Interpret EXTIO states and fronts inverted if flag is set. Falling front as input event and low logic level as active state.)
INT8U	EXTIOModeFlags	Flags mode settings external I-O
		0x00 - EXTIO_SETUP_MODE_IN_NOP (Do nothing.)
		0x01 - EXTIO_SETUP_MODE_IN_STOP (Issue STOP command, ceasing the engine movement.)
		0x02 - EXTIO_SETUP_MODE_IN_PWOF (Issue PWOF command, powering off all engine windings.)
		0x03 - EXTIO_SETUP_MODE_IN_MOVR (Issue MOVR command with last used settings.)
		0x04 - EXTIO_SETUP_MODE_IN_HOME (Issue HOME command.)
		0x00 - EXTIO_SETUP_MODE_OUT_OFF (EXTIO pin always set in inactive state.)
		0x10 - EXTIO_SETUP_MODE_OUT_ON (EXTIO pin always set in active state.)
		0x20 - EXTIO_SETUP_MODE_OUT_MOVING (EXTIO pin stays active during moving state.)
		0x30 - EXTIO_SETUP_MODE_OUT_ALARM (EXTIO pin stays active during Alarm state.)
		0x40 - EXTIO_SETUP_MODE_OUT_MOTOR_ON (EXTIO pin stays active when windings are powered.)
		0x50 - EXTIO_SETUP_MODE_OUT_MOTOR_FOUND (EXTIO pin stays active when motor is connected (first winding).)
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

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

Command SBRK

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

Command code (CMD): "sbrk" or 0x6B726273.

Request: (25 bytes)

INT32U	CMD	Command
INT16U	t1	Time in ms between turn on motor power and turn off brake. Range: 0..65535.
INT16U	t2	Time in ms between turn off brake and moving readiness. All moving commands will execute after this interval. Range: 0..65535.
INT16U	t3	Time in ms between motor stop and turn on brake. Range: 0..65535.
INT16U	t4	Time in ms between turn on brake and turn off motor power. Range: 0..65535.
INT8U	BrakeFlags	Flags.
		0x01 - BRAKE_ENABLED (Brake control is enabled, if this flag is set.)
		0x02 - BRAKE_ENG_PWROFF (Brake turns off power of step motor, if this flag is set.)
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set settings of brake control.

Command GBRK

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

Command code (CMD): "gbrk" or 0x6B726267.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (25 bytes)

INT32U	CMD	Command (answer)
INT16U	t1	Time in ms between turn on motor power and turn off brake. Range: 0..65535.
INT16U	t2	Time in ms between turn off brake and moving readiness. All moving commands will execute after this interval. Range: 0..65535.
INT16U	t3	Time in ms between motor stop and turn on brake. Range: 0..65535.
INT16U	t4	Time in ms between turn on brake and turn off motor power. Range: 0..65535.
INT8U	BrakeFlags	Flags.
		0x01 - BRAKE_ENABLED (Brake control is enabled, if this flag is set.)
		0x02 - BRAKE_ENG_PWROFF (Brake turns off power of step motor, if this flag is set.)
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

Description:
Read settings of brake control.

Command SCTL

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

Command code (CMD): "sctl" or 0x6C746373.

Request: (93 bytes)

INT32U	CMD	Command
INT32U	MaxSpeed [10]	Array of speeds (full step) using with joystick and button control. Range: 0..1000000.
INT8U	uMaxSpeed [10]	Array of speeds (1/256 microstep) using with joystick and button control. Range: 0..255.
INT16U	Timeout [9]	timeout[i] is time in ms, after that max_speed[i+1] is applying. It is using with buttons control only. Range: 0..65535.
INT16U	MaxClickTime	Maximum click time. Prior to the expiration of this time the first speed isn't enabled.
INT16U	Flags	Flags.
		0x03 - CONTROL_MODE_BITS (Bits to control engine by joystick or buttons.)
		0x00 - CONTROL_MODE_OFF (Control is disabled.)
		0x01 - CONTROL_MODE_JOY (Control by joystick.)
		0x02 - CONTROL_MODE_LR (Control by left/right buttons.)
		0x04 - CONTROL_BTN_LEFT_PUSHED_OPEN (Pushed left button corresponds to open contact, if this flag is set.)
		0x08 - CONTROL_BTN_RIGHT_PUSHED_OPEN (Pushed right button corresponds to open contact, if this flag is set.)
INT32S	DeltaPosition	Shift (delta) of position
INT16S	uDeltaPosition	Fractional part of the shift in micro steps (-255 .. 255) is only used with stepper motor
INT8U	Reserved [9]	Reserved (9 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command GCTL

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

Command code (CMD): "gctl" or 0x6C746367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (93 bytes)

INT32U	CMD	Command (answer)
INT32U	MaxSpeed [10]	Array of speeds (full step) using with joystick and button control. Range: 0..1000000.
INT8U	uMaxSpeed [10]	Array of speeds (1/256 microstep) using with joystick and button control. Range: 0..255.
INT16U	Timeout [9]	timeout[i] is time in ms, after that max_speed[i+1] is applying. It is using with buttons control only. Range: 0..65535.

INT16U	MaxClickTime	Maximum click time. Prior to the expiration of this time the first speed isn't enabled.
INT16U	Flags	Flags.
		0x03 - CONTROL_MODE_BITS (Bits to control engine by joystick or buttons.)
		0x00 - CONTROL_MODE_OFF (Control is disabled.)
		0x01 - CONTROL_MODE_JOY (Control by joystick.)
		0x02 - CONTROL_MODE_LR (Control by left/right buttons.)
		0x04 - CONTROL_BTN_LEFT_PUSHED_OPEN (Pushed left button corresponds to open contact, if this flag is set.)
		0x08 - CONTROL_BTN_RIGHT_PUSHED_OPEN (Pushed right button corresponds to open contact, if this flag is set.)
INT32S	DeltaPosition	Shift (delta) of position
INT16S	uDeltaPosition	Fractional part of the shift in micro steps (-255 .. 255) is only used with stepper motor
INT8U	Reserved [9]	Reserved (9 bytes)
INT16U	CRC	Checksum

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

Command SJOY

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

Command code (CMD): "sjoy" or 0x796F6A73.

Request: (22 bytes)

INT32U	CMD	Command
INT16U	JoyLowEnd	Joystick lower end position.
INT16U	JoyCenter	Joystick center position.
INT16U	JoyHighEnd	Joystick higher end position.
INT8U	ExpFactor	Exponential nonlinearity factor.
INT8U	DeadZone	Joystick dead zone.
INT8U	JoyFlags	Joystick control flags.
		0x01 - JOY_REVERSE (Joystick action is reversed. Joystick deviation to the upper values correspond to negative speeds and vice versa.)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set settings of joystick. If joystick position is outside DeadZone limits from the central position a movement with speed, defined by the joystick DeadZone edge to 100% deviation, begins. Joystick positions inside DeadZone limits correspond to zero speed (soft stop of motion) and positions beyond Low and High limits correspond MaxSpeed [i] or -MaxSpeed [i] (see command SCTL), where i = 0 by default and can be changed with left/right buttons (see command SCTL). If next speed in list is zero (both integer and microstep parts), the button press is ignored. First speed in list shouldn't be zero. The DeadZone ranges are illustrated on the following picture.

The relationship between the deviation and the rate is exponential, allowing no switching speed combine high mobility and accuracy. The following picture illustrates this:

The nonlinearity parameter is adjustable. Setting it to zero makes deviation/speed relation linear.

Command GJOY

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

Command code (CMD): "gjoy" or 0x796F6A67.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (22 bytes)

INT32U	CMD	Command (answer)
INT16U	JoyLowEnd	Joystick lower end position.
INT16U	JoyCenter	Joystick center position.
INT16U	JoyHighEnd	Joystick higher end position.
INT8U	ExpFactor	Exponential nonlinearity factor.
INT8U	DeadZone	Joystick dead zone.
INT8U	JoyFlags	Joystick control flags.
		0x01 - JOY_REVERSE (Joystick action is reversed. Joystick deviation to the upper values correspond to negative speeds and vice versa.)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Description:
Read settings of joystick. If joystick position is outside DeadZone limits from the central position a movement with speed, defined by the joystick DeadZone edge to 100% deviation, begins. Joystick positions inside DeadZone limits correspond to zero speed (soft stop of motion) and positions beyond Low and High limits correspond MaxSpeed [i] or -MaxSpeed [i] (see command SCTL), where i = 0 by default and can be changed with left/right buttons (see command SCTL). If next speed in list is zero (both integer and microstep parts), the button press is ignored. First speed in list shouldn't be zero. The DeadZone ranges are illustrated on the following picture.

The relationship between the deviation and the rate is exponential, allowing no switching speed combine high mobility and accuracy. The following picture illustrates this:

The nonlinearity parameter is adjustable. Setting it to zero makes deviation/speed relation linear.

Command SCTP

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

Command code (CMD): "sctp" or 0x70746373.

Request: (18 bytes)

INT32U	CMD	Command
INT8U	CTPMinError	Minimum contrast steps from step motor encoder position, wich set STATE_CTP_ERROR flag. Measured in steps step motor. Range: 0..255.
INT8U	CTPFlags	Flags.
		0x01 - CTP_ENABLED (Position control is enabled, if flag set.)
		0x02 - CTP_BASE (Position control is based on revolution sensor, if this flag is set; otherwise it is based on encoder.)
		0x04 - CTP_ALARM_ON_ERROR (Set ALARM on mismatch, if flag set.)
		0x08 - REV_SENS_INV (Sensor is active when it 0 and invert makes active level 1. That is, if you do not invert, it is normal logic - 0 is the activation.)
INT8U	Reserved [10]	Reserved (10 bytes)

INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command GCTP

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

Command code (CMD): "gctp" or 0x70746367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (18 bytes)

INT32U	CMD	Command (answer)
INT8U	CTPMinError	Minimum contrast steps from step motor encoder position, wich set STATE_CTP_ERROR flag. Measured in steps step motor. Range: 0..255.
INT8U	CTPFlags	Flags.
		0x01 - CTP_ENABLED (Position control is enabled, if flag set.)
		0x02 - CTP_BASE (Position control is based on revolution sensor, if this flag is set; otherwise it is based on encoder.)
		0x04 - CTP_ALARM_ON_ERROR (Set ALARM on mismatch, if flag set.)
		0x08 - REV_SENS_INV (Sensor is active when it 0 and invert makes active level 1. That is, if you do not invert, it is normal logic - 0 is the activation.)
INT8U	Reserved [10]	Reserved (10 bytes)
INT16U	CRC	Checksum

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

Command SURT

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

Command code (CMD): "surt" or 0x74727573.

Request: (16 bytes)

INT32U	CMD	Command
INT32U	Speed	UART speed
INT16U	UARTSetupFlags	UART setup flags

		0x03 - UART_PARITY_BITS (Bits of the parity.)
		0x00 - UART_PARITY_BIT_EVEN (Parity bit 1, if even)
		0x01 - UART_PARITY_BIT_ODD (Parity bit 1, if odd)
		0x02 - UART_PARITY_BIT_SPACE (Parity bit always 0)
		0x03 - UART_PARITY_BIT_MARK (Parity bit always 1)
		0x04 - UART_PARITY_BIT_USE (None parity)
		0x08 - UART_STOP_BIT (If set - one stop bit, else two stop bit)
INT8U	Reserved [4]	Reserved (4 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command GURT

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

Command code (CMD): "gurt" or 0x74727567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (16 bytes)

INT32U	CMD	Command (answer)
INT32U	Speed	UART speed
INT16U	UARTSetupFlags	UART setup flags
		0x03 - UART_PARITY_BITS (Bits of the parity.)
		0x00 - UART_PARITY_BIT_EVEN (Parity bit 1, if even)
		0x01 - UART_PARITY_BIT_ODD (Parity bit 1, if odd)
		0x02 - UART_PARITY_BIT_SPACE (Parity bit always 0)
		0x03 - UART_PARITY_BIT_MARK (Parity bit always 1)
		0x04 - UART_PARITY_BIT_USE (None parity)
		0x08 - UART_STOP_BIT (If set - one stop bit, else two stop bit)
INT8U	Reserved [4]	Reserved (4 bytes)
INT16U	CRC	Checksum

Description:
Read UART settings. This function fill structure with UART settings.

Command SNMF

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

Command code (CMD): "snmf" or 0x666D6E73.

Request: (30 bytes)

INT32U	CMD	Command
CHAR	ControllerName [16]	User conroller name. Can be set by user for his/her convinience. Max string length: 16 chars.
INT8U	CtrlFlags	Internal controller settings.

		0x01 - EEPROM_PRECEDENCE (If the flag is set settings from external EEPROM override controller settings.)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Write user controller name and flags of setting from FRAM.

Command GNMF

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

Command code (CMD): "gnmf" or 0x666D6E67.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (30 bytes)

INT32U	CMD	Command (answer)
CHAR	ControllerName [16]	User conroller name. Can be set by user for his/her convinience. Max string length: 16 chars.
INT8U	CtrlFlags	Internal controller settings.
		0x01 - EEPROM_PRECEDENCE (If the flag is set settings from external EEPROM override controller settings.)
INT8U	Reserved [7]	Reserved (7 bytes)
INT16U	CRC	Checksum

Description:
Read user controller name and flags of setting from FRAM.

Group of commands movement control

Command STOP

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

Command code (CMD): "stop" or 0x706F7473.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command ASIA

```
result_t set_add_sync_in_action (device_t id, const add_sync_in_action_t* add_sync_in_action)
```

Command code (CMD): "asia" or 0x61697361.

Request: (22 bytes)

INT32U	CMD	Command
INT32S	Position	Desired position or shift (whole steps)
INT16S	uPosition	The fractional part of a position or shift in microsteps (-255 .. 255)(is only used with stepper motor)
INT32U	Speed	Target speed(for stepper motor: steps / c, for DC: rpm). Range: 0..1000000.
INT8U	uSpeed	Target speed in microsteps/s. Using with stepper motor only. Range: 0..255.
INT8U	Reserved [5]	Reserved (5 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
This command adds one element of the FIFO commands that are executed when input clock pulse. Each pulse synchronization or perform that action, which is described in SSNI, if the buffer is empty, or the oldest loaded into the buffer action to temporarily replace the speed and coordinate in SSNI. In the latter case this action is erased from the buffer. The number of remaining empty buffer elements can be found in the structure of GETS.

Command PWOF

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

Command code (CMD): "pwof" or 0x666F7770.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command MOVE

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

Command code (CMD): "move" or 0x65766F6D.

Request: (18 bytes)

INT32U	CMD	Command
INT32S	Position	Desired position (whole steps).
INT16S	uPosition	The fractional part of a position in microsteps (-255 .. 255)(is only used with stepper motor)
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Upon receiving the command "move" the engine starts to move with pre-set parameters (speed, acceleration, retention), to the point specified to the Position, uPosition. For stepper motor uPosition sets the microstep for DC motor, this field is not used.

Command MOVR

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

Command code (CMD): "movr" or 0x72766F6D.

Request: (18 bytes)

INT32U	CMD	Command
INT32S	DeltaPosition	Shift (delta) of position
INT16S	uDeltaPosition	Fractional part of the shift in micro steps (-255 .. 255) is only used with stepper motor
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Upon receiving the command "movr" engine starts to move with pre-set parameters (speed, acceleration, hold), left or right (depending on the sign of DeltaPosition) by the number of pulses specified in the fields DeltaPosition, uDeltaPosition. For stepper motor uDeltaPosition sets the microstep for DC motor, this field is not used.

Command HOME

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

Command code (CMD): "home" or 0x656D6F68.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
The positive direction is to the right. A value of zero reverses the direction of the direction of the flag, the set speed. Restriction imposed by the trailer, act the same, except that the limit switch contact does not stop. Limit the maximum speed, acceleration and deceleration function. 1) moves the motor according to the speed FastHome, uFastHome and flag HOME_DIR_FAST until limit switch, if the flag is set HOME_STOP_ENDS, until the signal from the input synchronization if the flag HOME_STOP_SYNC (as accurately as possible is important to catch the moment of operation limit switch) or until the signal is received from the speed sensor, if the flag HOME_STOP_REV_SN 2) then moves according to the speed SlowHome, uSlowHome and flag HOME_DIR_SLOW until signal from the clock input, if the flag HOME_MV_SEC. If the flag HOME_MV_SEC reset skip this paragraph. 3) then move the motor according to the speed FastHome, uFastHome and flag HOME_DIR_SLOW a distance HomeDelta, uHomeDelta. description of flags and variable see in description for commands GHOM/SHOM

Command LEFT

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

Command code (CMD): "left" or 0x7466656C.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:

Start continous moving to the left.

Command RIGT

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

Command code (CMD): "rigt" or 0x74676972.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Start continous moving to the right.

Command LOFT

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

Command code (CMD): "loft" or 0x74666F6C.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command SSTP

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

Command code (CMD): "sstp" or 0x70747373.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
soft stop engine. The motor stops with deceleration speed.

Group of commands set the current position

Command GPOS

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

Command code (CMD): "gpos" or 0x736F7067.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (26 bytes)

INT32U	CMD	Command (answer)
INT32S	Position	The position of the whole steps in the engine
INT16S	uPosition	Microstep position is only used with stepper motors
INT64S	EncPosition	Encoder position.
INT8U	Reserved [6]	Reserved (6 bytes)
INT16U	CRC	Checksum

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

Command SPOS

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

Command code (CMD): "spos" or 0x736F7073.

Request: (26 bytes)

INT32U	CMD	Command
INT32S	Position	The position of the whole steps in the engine
INT16S	uPosition	Microstep position is only used with stepper motors
INT64S	EncPosition	Encoder position.
INT8U	PosFlags	Flags
		0x01 - SETPOS_IGNORE_POSITION (Will not reload position in steps/microsteps if this flag is set.)
		0x02 - SETPOS_IGNORE_ENCODER (Will not reload encoder state if this flag is set.)
INT8U	Reserved [5]	Reserved (5 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Sets any position value in steps and micro for stepper motor and encoder steps of all engines. It means, that changing main indicator of position.

Command ZERO

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

Command code (CMD): "zero" or 0x6F72657A.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Sets the current position and the position in which the traffic moves by the move command and movr zero for all cases, except for movement to the target position. In the latter case, set the zero current position and the target position counted so that the absolute position of the destination is the same. That is, if we were at 400 and moved to 500, then the command Zero makes the current position of 0, and the position of the destination - 100. Does not change the mode of movement that is if the motion is carried, it continues, and if the engine is in the "hold", the type of retention remains.

Group of commands to save and load settings

Command SAVE

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

Command code (CMD): "save" or 0x65766173.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command READ

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

Command code (CMD): "read" or 0x64616572.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Command EESV

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

Command code (CMD): "eesv" or 0x76736565.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Save settings from controller's RAM to stage's EEPROM memory, whitch spontaneity connected to stage and it isn`t change without it mechanical reconstruction. Can be used by manufacturer only.

Command EERD

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

Command code (CMD): "eerd" or 0x64726565.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

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

Group of commands get the status of the controller

Command GETS

```
result_t get_status_impl (device_t id, status_t* status)
```

Command code (CMD): "gets" or 0x73746567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (54 bytes)

INT32U	CMD	Command (answer)
INT8U	MoveSts	Move state.
		0x01 - MOVE_STATE_MOVING (This flag indicates that controller is trying to move the motor.)
		0x02 - MOVE_STATE_TARGET_SPEED (Target speed is reached, if flag set.)
		0x04 - MOVE_STATE_ANTIPLAY (Motor is playing compensation, if flag set.)
INT8U	MvCmdSts	Move command state.
		0x3F - MVCMD_NAME_BITS (Move command bit mask.)
		0x00 - MVCMD_UKNWN (Unknown command.)
		0x01 - MVCMD_MOVE (Command move.)
		0x02 - MVCMD_MOVR (Command movr.)
		0x03 - MVCMD_LEFT (Command left.)
		0x04 - MVCMD_RIGHT (Command rigt.)
		0x05 - MVCMD_STOP (Command stop.)
		0x06 - MVCMD_HOME (Command home.)
		0x07 - MVCMD_LOFT (Command loft.)
		0x08 - MVCMD_SSTP (Command soft stop.)
		0x40 - MVCMD_ERROR (Finish state (1 - move command have finished with an error, 0 - move command have finished correctly). This flags is actual when MVCMD_RUNNING signals movement finish.)
		0x80 - MVCMD_RUNNING (Move command state (0 - move command have finished, 1 - move command is being executed).)
INT8U	PWRSts	Power state of the stepper motor (used only with stepper motor).
		0x00 - PWR_STATE_UNKNOWN (Unknown state, should never happen.)
		0x01 - PWR_STATE_OFF (Motor windings are disconnected from the driver.)
		0x03 - PWR_STATE_NORM (Motor windings are powered by nominal current.)
		0x04 - PWR_STATE_REDUCT (Motor windings are powered by reduced current to lower power consumption.)
		0x05 - PWR_STATE_MAX (Motor windings are powered by maximum current driver can provide at this voltage.)
INT8U	EncSts	Encoder state.
		0x00 - ENC_STATE_ABSENT (Encoder is absent.)
		0x01 - ENC_STATE_UNKNOWN (Encoder state is unknown.)
		0x02 - ENC_STATE_MALFUNC (Encoder is connected and malfunctioning.)
		0x03 - ENC_STATE_REVERS (Encoder is connected and operational but counts in otyher

		direction.)
		0x04 - ENC_STATE_OK (Encoder is connected and working properly.)
INT8U	WindSts	Windings state.
		0x00 - WIND_A_STATE_ABSENT (Winding A is disconnected.)
		0x01 - WIND_A_STATE_UNKNOWN (Winding A state is unknown.)
		0x02 - WIND_A_STATE_MALFUNC (Winding A is short-circuited.)
		0x03 - WIND_A_STATE_OK (Winding A is connected and working properly.)
		0x00 - WIND_B_STATE_ABSENT (Winding B is disconnected.)
		0x10 - WIND_B_STATE_UNKNOWN (Winding B state is unknown.)
		0x20 - WIND_B_STATE_MALFUNC (Winding B is short-circuited.)
		0x30 - WIND_B_STATE_OK (Winding B is connected and working properly.)
INT32S	CurPosition	Current position.
INT16S	uCurPosition	Step motor shaft position in 1/256 microsteps. Used only with stepper motor.
INT64S	EncPosition	Current encoder position.
INT32S	CurSpeed	Motor shaft speed.
INT16S	uCurSpeed	Part of motor shaft speed in 1/256 microsteps. Used only with stepper motor.
INT16S	Ipwr	Engine current.
INT16S	Upwr	Power supply voltage.
INT16S	Iusb	USB current consumption.
INT16S	Uusb	USB voltage.
INT16S	CurT	Temperature in tenths of degrees C.
INT32U	Flags	Set of flags specify motor shaft movement algorithm and list of limitations.
		0x0003F - STATE_CONTR (Flags of controller states.)
		0x00001 - STATE_ERRC (Command error encountered.)
		0x00002 - STATE_ERRD (Data integrity error encountered.)
		0x00004 - STATE_ERRV (Value error encountered.)
		0x00008 - STATE_EXT_POWER (External power is used.)
		0x00010 - STATE_EEPROM_CONNECTED (EEPROM with settings is connected.)
		0x3FFC0 - STATE_SECUR (Flags of security.)
		0x00040 - STATE_ALARM (Controller is in alarm state indicating that something dangerous had happened. Most commands are ignored in this state. To reset the flag a STOP command must be issued.)
		0x00080 - STATE_CTP_ERROR (Control position error(is only used with stepper motor).)
		0x00100 - STATE_POWER_OVERHEAT (Power driver overheat.)
		0x00200 - STATE_CONTROLLER_OVERHEAT (Controller overheat.)
		0x00400 - STATE_OVERLOAD_POWER_VOLTAGE (Power voltage exceeds safe limit.)
		0x00800 - STATE_OVERLOAD_POWER_CURRENT (Power current exceeds safe limit.)
		0x01000 - STATE_OVERLOAD_USB_VOLTAGE (USB voltage exceeds safe limit.)
		0x02000 - STATE_LOW_USB_VOLTAGE (USB voltage is insufficient for normal operation.)
		0x04000 - STATE_OVERLOAD_USB_CURRENT (USB current exceeds safe limit.)
		0x08000 - STATE_BORDERS_SWAP_MISSET (Engine stuck at the wrong edge.)
		0x10000 - STATE_LOW_POWER_VOLTAGE (Power voltage is lower than Low Voltage Protection limit)
		0x20000 - STATE_H_BRIDGE_FAULT (Signal from the driver that fault happened)
INT32U	GPIOFlags	Set of flags of gpio states
		0xFFFF - STATE_DIG_SIGNAL (Flags of digital signals.)
		0x0001 - STATE_RIGHT_EDGE (Engine stuck at the right edge.)
		0x0002 - STATE_LEFT_EDGE (Engine stuck at the left edge.)

		0x0004 - STATE_BUTTON_RIGHT (Button "right" state (1 if pressed).)
		0x0008 - STATE_BUTTON_LEFT (Button "left" state (1 if pressed).)
		0x0010 - STATE_GPIO_PINOUT (External GPIO works as Out, if flag set; otherwise works as In.)
		0x0020 - STATE_GPIO_LEVEL (State of external GPIO pin.)
		0x0040 - STATE_HALL_A (State of Hall_a pin.)
		0x0080 - STATE_HALL_B (State of Hall_b pin.)
		0x0100 - STATE_HALL_C (State of Hall_c pin.)
		0x0200 - STATE_BRAKE (State of Brake pin.)
		0x0400 - STATE_REV_SENSOR (State of Revolution sensor pin.)
		0x0800 - STATE_SYNC_INPUT (State of Sync input pin.)
		0x1000 - STATE_SYNC_OUTPUT (State of Sync output pin.)
		0x2000 - STATE_ENC_A (State of encoder A pin.)
		0x4000 - STATE_ENC_B (State of encoder B pin.)
INT8U	CmdBufFreeSpace	This field shows the amount of free cells buffer synchronization chain.
INT8U	Reserved [4]	Reserved (4 bytes)
INT16U	CRC	Checksum

Description:
Return device state. Useful function that fills structure with snapshot of controller state, including speed, position and boolean flags.

Command GETC

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

Command code (CMD): "getc" or 0x63746567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (38 bytes)

INT32U	CMD	Command (answer)
INT16S	WindingVoltageA	In the case step motor, the voltage across the winding A; in the case of a brushless, the voltage on the first coil, in the case of the only DC.
INT16S	WindingVoltageB	In the case step motor, the voltage across the winding B; in case of a brushless, the voltage on the second winding, and in the case of DC is not used.
INT16S	WindingVoltageC	In the case of a brushless, the voltage on the third winding, in the case step motor and DC is not used.
INT16S	WindingCurrentA	In the case step motor, the current in the coil A; brushless if the current in the first coil, and in the case of a single DC.
INT16S	WindingCurrentB	In the case step motor, the current in the coil B; brushless if the current in the second coil, and in the case of DC is not used.
INT16S	WindingCurrentC	In the case of a brushless, the current in the third winding, in the case step motor and DC is not used.
INT16U	Pot	Potentiometer in ten-thousandths of [0, 10000]
INT16U	Joy	The joystick to the ten-thousandths [0, 10000]
INT16S	DutyCycle	Duty cycle of PWM.
INT8U	Reserved [14]	Reserved (14 bytes)
INT16U	CRC	Checksum

Description:
Return device electrical parameters, useful for charts. Useful function that fill structure with snapshot of controller voltages and currents.

Command GETI

```
result_t get_device_information_impl (device_t id, device_information_t* device_information)
```

Command code (CMD): "geti" or 0x69746567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (36 bytes)

INT32U	CMD	Command (answer)
CHAR	Manufacturer [4]	Manufacturer
CHAR	ManufacturerId [2]	Manufacturer id
CHAR	ProductDescription [8]	Product description
INT8U	Reserved [16]	Reserved (16 bytes)
INT16U	CRC	Checksum

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

Command GSER

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

Command code (CMD): "gser" or 0x72657367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (10 bytes)

INT32U	CMD	Command (answer)
INT32U	SerialNumber	Board serial number.
INT16U	CRC	Checksum

Description:
Read device serial number.

Group of commands to work with the controller firmware

Command GFWV

```
result_t get_firmware_version (device_t id, unsigned int* Major, unsigned int* Minor, unsigne
```

Command code (CMD): "gfwv" or 0x76776667.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (10 bytes)

INT32U	CMD	Command (answer)
INT8U	Major	Firmware major version number
INT8U	Minor	Firmware minor version number
INT16U	Release	Firmware release version number
INT16U	CRC	Checksum

Description:
Read controller's firmware version.

Command UPDF

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

Command code (CMD): "updf" or 0x66647075.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Command puts the controller to update the firmware. After receiving this command, the firmware board sets a flag (for loader), sends echo reply and restarts the controller.

Service commands

Command SSER

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

Command code (CMD): "sser" or 0x72657373.

Request: (50 bytes)

INT32U	CMD	Command
INT32U	SN	New board serial number.
INT8U	Key [32]	Protection key (256 bit).
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Write device serial number to controller's flash memory. Along with the new serial number a "Key" is transmitted. The SN is changed and saved when keys match. Can be used by manufacturer only.

Command RDAN

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

Command code (CMD): "rdan" or 0x6E616472.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (76 bytes)

INT32U	CMD	Command (answer)
INT16U	A1Voltage_ADC	"Voltage on pin 1 winding A" raw data from ADC.
INT16U	A2Voltage_ADC	"Voltage on pin 2 winding A" raw data from ADC.
INT16U	B1Voltage_ADC	"Voltage on pin 1 winding B" raw data from ADC.
INT16U	B2Voltage_ADC	"Voltage on pin 2 winding B" raw data from ADC.

INT16U	SupVoltage_ADC	"Voltage on the top of MOSFET full bridge" raw data from ADC.
INT16U	ACurrent_ADC	"Winding A current" raw data from ADC.
INT16U	BCurrent_ADC	"Winding B current" raw data from ADC.
INT16U	FullCurrent_ADC	"Full current" raw data from ADC.
INT16U	Temp_ADC	Voltage from temperature sensor, raw data from ADC.
INT16U	Joy_ADC	Joystick raw data from ADC.
INT16U	Pot_ADC	Voltage on "Potentiometer", raw data from ADC
INT16U	L5_ADC	USB supply voltage after the current sense resistor, from ADC.
INT16U	H5_ADC	Power supply USB from ADC
INT16S	A1Voltage	"Voltage on pin 1 winding A" calibrated data.
INT16S	A2Voltage	"Voltage on pin 2 winding A" calibrated data.
INT16S	B1Voltage	"Voltage on pin 1 winding B" calibrated data.
INT16S	B2Voltage	"Voltage on pin 2 winding B" calibrated data.
INT16S	SupVoltage	"Voltage on the top of MOSFET full bridge" calibrated data.
INT16S	ACurrent	"Winding A current" calibrated data.
INT16S	BCurrent	"Winding B current" calibrated data.
INT16S	FullCurrent	"Full current" calibrated data.
INT16S	Temp	Temperature, calibrated data.
INT16S	Joy	Joystick, calibrated data.
INT16S	Pot	Potentiometer, calibrated data.
INT16S	L5	USB supply voltage after the current sense resistor.
INT16S	H5	Power supply USB
INT16U	deprecated	
INT32S	R	Motor winding resistance in mOhms(is only used with stepper motor).
INT32S	L	Motor winding pseudo inductance in uHn(is only used with stepper motor).
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Description:
Read analog data structure that contains raw analog data from ADC embedded on board. This function used for device testing and deep recalibraton by manufacturer only.

Command DBGR

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

Command code (CMD): "dbgr" or 0x72676264.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (142 bytes)

INT32U	CMD	Command (answer)
INT8U	DebugData [128]	Arbitrary debug data.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Description:
Read data from firmware for debug purpose. Its use depends on context, firmware version and previous history.

Command CLFR

```
result_t service_command_clear_fram_impl (device_t id)
```

Command code (CMD): "clfr" or 0x72666C63.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (0 bytes)

Description:
The command for clear controller FRAM. The memory is cleared by filling the whole memory byte 0x00. After cleaning, the controller restarts. No response to this command.

Group of commands to work with EEPROM

Command SNME

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

Command code (CMD): "snme" or 0x656D6E73.

Request: (30 bytes)

INT32U	CMD	Command
CHAR	PositionerName [16]	User positioner name. Can be set by user for his/her convinience. Max string length: 16 chars.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Write user stage name from EEPROM.

Command GNME

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

Command code (CMD): "gnme" or 0x656D6E67.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (30 bytes)

INT32U	CMD	Command (answer)
CHAR	PositionerName [16]	User positioner name. Can be set by user for his/her convinience. Max string length: 16 chars.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Description:
Read user stage name from EEPROM.

Command SSTI

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

Command code (CMD): "ssti" or 0x69747373.

Request: (238 bytes)

INT32U	CMD	Command
CHAR	Manufacturer [16]	Manufacturer. Max string length: 16 chars.
CHAR	Series [32]	Series. Max string length: 32 chars.
CHAR	ArticleNumber [32]	Article number. Max string length: 32 chars.
CHAR	Description [128]	Description. Max string length: 128 chars.
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set stage information to EEPROM. Can be used by manufacturer only.

Command GSTI

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

Command code (CMD): "gsti" or 0x69747367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (238 bytes)

INT32U	CMD	Command (answer)
CHAR	Manufacturer [16]	Manufacturer. Max string length: 16 chars.
CHAR	Series [32]	Series. Max string length: 32 chars.
CHAR	ArticleNumber [32]	Article number. Max string length: 32 chars.
CHAR	Description [128]	Description. Max string length: 128 chars.
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Description:
Read stage information from EEPROM.

Command SSTS

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

Command code (CMD): "ssts" or 0x73747373.

Request: (70 bytes)

INT32U	CMD	Command
FLT32	LeadScrewPitch	Lead screw pitch (mm). Data type: float.
CHAR	Units [8]	Units for MaxSpeed and TravelRange fields of the structure (steps, degrees, mm, ...). Max string length: 8 chars.
FLT32	MaxSpeed	Max speed (Units/c). Data type: float.
FLT32	TravelRange	Travel range (Units). Data type: float.
FLT32	SupplyVoltage1	Supply voltage minimum (V). Data type: float.
FLT32	SupplyVoltage2	Supply voltage maximum (V). Data type: float.
FLT32	MaxCurrentConsumption	Max current consumption (A). Data type: float.
FLT32	HorizontalLoadCapacity	Horizontal load capacity (kg). Data type: float.

FLT32	VerticalLoadCapacity	Vertical load capacity (kg). Data type: float.
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set stage information to EEPROM. Can be used by manufacturer only

Command GSTS

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

Command code (CMD): "gsts" or 0x73747367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (70 bytes)

INT32U	CMD	Command (answer)
FLT32	LeadScrewPitch	Lead screw pitch (mm). Data type: float.
CHAR	Units [8]	Units for MaxSpeed and TravelRange fields of the structure (steps, degrees, mm, ...). Max string length: 8 chars.
FLT32	MaxSpeed	Max speed (Units/c). Data type: float.
FLT32	TravelRange	Travel range (Units). Data type: float.
FLT32	SupplyVoltage1	Supply voltage minimum (V). Data type: float.
FLT32	SupplyVoltage2	Supply voltage maximum (V). Data type: float.
FLT32	MaxCurrentConsumption	Max current consumption (A). Data type: float.
FLT32	HorizontalLoadCapacity	Horizontal load capacity (kg). Data type: float.
FLT32	VerticalLoadCapacity	Vertical load capacity (kg). Data type: float.
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Description:
Read stage information from EEPROM.

Command SDCI

```
result_t set_dc_information (device_t id, const dc_information_t* dc_information)
```

Command code (CMD): "sdci" or 0x69636473.

Request: (246 bytes)

INT32U	CMD	Command
CHAR	Manufacturer [16]	Manufacturer. Max string length: 16 chars.
CHAR	Series [32]	Series. Max string length: 32 chars.
CHAR	ArticleNumber [32]	Article number. Max string length: 32 chars.
CHAR	Description [128]	Description. Max string length: 128 chars.
INT8U	Reserved [32]	Reserved (32 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set DC motor information to EEPROM. Can be used by manufacturer only.

Command GDCI

```
result_t get_dc_information (device_t id, dc_information_t* dc_information)
```

Command code (CMD): "gdci" or 0x69636467.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (246 bytes)

INT32U	CMD	Command (answer)
CHAR	Manufacturer [16]	Manufacturer. Max string length: 16 chars.
CHAR	Series [32]	Series. Max string length: 32 chars.
CHAR	ArticleNumber [32]	Article number. Max string length: 32 chars.
CHAR	Description [128]	Description. Max string length: 128 chars.
INT8U	Reserved [32]	Reserved (32 bytes)
INT16U	CRC	Checksum

Description:
Read DC motor information from EEPROM.

Command SDCS

```
result_t set_dc_settings (device_t id, const dc_settings_t* dc_settings)
```

Command code (CMD): "sdcs" or 0x73636473.

Request: (78 bytes)

INT32U	CMD	Command
FLT32	NominalVoltage	Nominal voltage (V). Data type: float.
FLT32	NoLoadSpeed	No load speed (rpm). Data type: float.
FLT32	NoLoadCurrent	No load current (A). Data type: float.
FLT32	StallTorque	Stall torque (mN m). Data type: float.
FLT32	MaxContinuousCurrent	Max continuous current (A). Data type: float.
FLT32	TerminalResistance	Terminal resistance (Ohm). Data type: float.
FLT32	TerminalInductance	Terminal inductance (mH). Data type: float.
FLT32	TorqueConstant	Torque constant (mN m/A). Data type: float.
FLT32	SpeedConstant	Speed constant (rpm/V). Data type: float.
FLT32	SpeedTorqueGradient	Speed torque gradient (rpm/mN m). Data type: float.
FLT32	MechanicalTimeConstant	Mechanical time constant (ms). Data type: float.
FLT32	RotorInertia	Rotor inertia (g cm²). Data type: float.
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set DC motor information to EEPROM. Can be used by manufacturer only.

Command GDCS

```
result_t get_dc_settings (device_t id, dc_settings_t* dc_settings)
```

Command code (CMD): "gdcs" or 0x73636467.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (78 bytes)

INT32U	CMD	Command (answer)
FLT32	NominalVoltage	Nominal voltage (V). Data type: float.
FLT32	NoLoadSpeed	No load speed (rpm). Data type: float.
FLT32	NoLoadCurrent	No load current (A). Data type: float.
FLT32	StallTorque	Stall torque (mN m). Data type: float.
FLT32	MaxContinuousCurrent	Max continuous current (A). Data type: float.
FLT32	TerminalResistance	Terminal resistance (Ohm). Data type: float.
FLT32	TerminalInductance	Terminal inductance (mH). Data type: float.
FLT32	TorqueConstant	Torque constant (mN m/A). Data type: float.
FLT32	SpeedConstant	Speed constant (rpm/V). Data type: float.
FLT32	SpeedTorqueGradient	Speed torque gradient (rpm/mN m). Data type: float.
FLT32	MechanicalTimeConstant	Mechanical time constant (ms). Data type: float.
FLT32	RotorInertia	Rotor inertia (g cm²). Data type: float.
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Description:
Read DC motor information from EEPROM.

Command SSMI

```
result_t set_step_information (device_t id, const step_information_t* step_information)
```

Command code (CMD): "ssmi" or 0x696D7373.

Request: (246 bytes)

INT32U	CMD	Command
CHAR	Manufacturer [16]	Manufacturer. Max string length: 16 chars.
CHAR	Series [32]	Series. Max string length: 32 chars.
CHAR	ArticleNumber [32]	Article number. Max string length: 32 chars.
CHAR	Description [128]	Description. Max string length: 128 chars.
INT8U	Reserved [32]	Reserved (32 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set step motor information to EEPROM. Can be used by manufacturer only.

Command GSMI

```
result_t get_step_information (device_t id, step_information_t* step_information)
```

Command code (CMD): "gsmi" or 0x696D7367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (246 bytes)

INT32U	CMD	Command (answer)
CHAR	Manufacturer [16]	Manufacturer. Max string length: 16 chars.
CHAR	Series [32]	Series. Max string length: 32 chars.
CHAR	ArticleNumber [32]	Article number. Max string length: 32 chars.
CHAR	Description [128]	Description. Max string length: 128 chars.
INT8U	Reserved [32]	Reserved (32 bytes)
INT16U	CRC	Checksum

Description:
Read step motor information from EEPROM.

Command SSMS

```
result_t set_step_settings (device_t id, const step_settings_t* step_settings)
```

Command code (CMD): "ssms" or 0x736D7373.

Request: (58 bytes)

INT32U	CMD	Command
FLT32	NominalVoltage	Nominal voltage (V). Data type: float.
FLT32	CurrentPerWinding	Current per winding (A). Data type: float.
FLT32	HoldingTorque	Holding torque (N m). Data type: float.
FLT32	ResistancePerWinding	Resistance per winding (Ohm). Data type: float.
FLT32	InductancePerWinding	Inductance per winding (mH). Data type: float.
FLT32	RotorInertia	Rotor inertia (g cm²). Data type: float.
FLT32	MaxSpeed	Max speed (steps/s). Data type: float.
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set step motor information to EEPROM. Can be used by manufacturer only.

Command GSMS

```
result_t get_step_settings (device_t id, step_settings_t* step_settings)
```

Command code (CMD): "gsms" or 0x736D7367.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (58 bytes)

INT32U	CMD	Command (answer)
FLT32	NominalVoltage	Nominal voltage (V). Data type: float.
FLT32	CurrentPerWinding	Current per winding (A). Data type: float.

FLT32	HoldingTorque	Holding torque (N m). Data type: float.
FLT32	ResistancePerWinding	Resistance per winding (Ohm). Data type: float.
FLT32	InductancePerWinding	Inductance per winding (mH). Data type: float.
FLT32	RotorInertia	Rotor inertia (g cm²). Data type: float.
FLT32	MaxSpeed	Max speed (steps/s). Data type: float.
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Description:
Read step motor information from EEPROM.

Command SENI

```
result_t set_encoder_information (device_t id, const encoder_information_t* encoder_informati
```

Command code (CMD): "seni" or 0x696E6573.

Request: (246 bytes)

INT32U	CMD	Command
CHAR	Manufacturer [16]	Manufacturer. Max string length: 16 chars.
CHAR	Series [32]	Series. Max string length: 32 chars.
CHAR	ArticleNumber [32]	Article number. Max string length: 32 chars.
CHAR	Description [128]	Description. Max string length: 128 chars.
INT8U	Reserved [32]	Reserved (32 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set encoder information to EEPROM. Can be used by manufacturer only.

Command GENI

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

Command code (CMD): "geni" or 0x696E6567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (246 bytes)

INT32U	CMD	Command (answer)
CHAR	Manufacturer [16]	Manufacturer. Max string length: 16 chars.
CHAR	Series [32]	Series. Max string length: 32 chars.
CHAR	ArticleNumber [32]	Article number. Max string length: 32 chars.
CHAR	Description [128]	Description. Max string length: 128 chars.
INT8U	Reserved [32]	Reserved (32 bytes)
INT16U	CRC	Checksum

Description:
Read encoder information from EEPROM.

Command SENS

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

Command code (CMD): "sens" or 0x736E6573.

Request: (114 bytes)

INT32U	CMD	Command
FLT32	MaxOperatingFrequency	Max operation frequency (kHz). Data type: float.
FLT32	SupplyVoltage1	Minimum supply voltage (V). Data type: float.
FLT32	SupplyVoltage2	Maximum supply voltage (V). Data type: float.
FLT32	MaxCurrentConsumption	Max current consumption (mA). Data type: float.
CHAR	OutputType [64]	Output type. Max string length: 64 chars.
INT8U	Reserved [28]	Reserved (28 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set encoder information to EEPROM. Can be used by manufacturer only.

Command GENS

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

Command code (CMD): "gens" or 0x736E6567.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (114 bytes)

INT32U	CMD	Command (answer)
FLT32	MaxOperatingFrequency	Max operation frequency (kHz). Data type: float.
FLT32	SupplyVoltage1	Minimum supply voltage (V). Data type: float.
FLT32	SupplyVoltage2	Maximum supply voltage (V). Data type: float.
FLT32	MaxCurrentConsumption	Max current consumption (mA). Data type: float.
CHAR	OutputType [64]	Output type. Max string length: 64 chars.
INT8U	Reserved [28]	Reserved (28 bytes)
INT16U	CRC	Checksum

Description:
Read encoder information from EEPROM.

Command SGRI

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

Command code (CMD): "sgri" or 0x69726773.

Request: (246 bytes)

INT32U	CMD	Command
CHAR	Manufacturer [16]	Manufacturer. Max string length: 16 chars.
CHAR	Series [32]	Series. Max string length: 32 chars.
CHAR	ArticleNumber [32]	Article number. Max string length: 32 chars.

CHAR	Description [128]	Description. Max string length: 128 chars.
INT8U	Reserved [32]	Reserved (32 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set gear information to EEPROM. Can be used by manufacturer only.

Command GGRI

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

Command code (CMD): "ggri" or 0x69726767.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (246 bytes)

INT32U	CMD	Command (answer)
CHAR	Manufacturer [16]	Manufacturer. Max string length: 16 chars.
CHAR	Series [32]	Series. Max string length: 32 chars.
CHAR	ArticleNumber [32]	Article number. Max string length: 32 chars.
CHAR	Description [128]	Description. Max string length: 128 chars.
INT8U	Reserved [32]	Reserved (32 bytes)
INT16U	CRC	Checksum

Description:
Read gear information from EEPROM.

Command SGRS

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

Command code (CMD): "sgrs" or 0x73726773.

Request: (46 bytes)

INT32U	CMD	Command
INT32U	Reduction1	Reduction coefficient 1. Data type: unsigned int.
INT32U	Reduction2	Reduction coefficient 2. Data type: unsigned int.
FLT32	MaxContinuousTorque	Max continuous torque (N m). Data type: float.
FLT32	MaxInputSpeed	Max input speed (rpm). Data type: float.
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Set gear information to EEPROM. Can be used by manufacturer only.

Command GGRS

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

Command code (CMD): "ggrs" or 0x73726767.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (46 bytes)

INT32U	CMD	Command (answer)
INT32U	Reduction1	Reduction coefficient 1. Data type: unsigned int.
INT32U	Reduction2	Reduction coefficient 2. Data type: unsigned int.
FLT32	MaxContinuousTorque	Max continuous torque (N m). Data type: float.
FLT32	MaxInputSpeed	Max input speed (rpm). Data type: float.
INT8U	Reserved [24]	Reserved (24 bytes)
INT16U	CRC	Checksum

Description:
Read gear information from EEPROM.

Bootloader commands

Command GBLV

```
result_t get_bootloader_version (device_t id, unsigned int* Major, unsigned int* Minor, unsig
```

Command code (CMD): "gblv" or 0x766C6267.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (10 bytes)

INT32U	CMD	Command (answer)
INT8U	Major	Bootloader major version number
INT8U	Minor	Bootloader minor version number
INT16U	Release	Bootloader release version number
INT16U	CRC	Checksum

Description:
Read controller's firmware version.

Command GOFW

```
result_t service_command_goto_firmware_impl (device_t id, service_result_t*sresult)
```

Command code (CMD): "gofw" or 0x77666F67.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (15 bytes)

INT32U	CMD	Command (answer)
INT8U	sresult	Result of command.
INT8U	Reserved [8]	Reserved (8 bytes)

Description:
Command initiates the transfer of control to firmware. This command is also available from the firmware for compatibility. Result = RESULT_OK, if the transition from the loader in the firmware is possible. After the response to

this command is executed transition. Result = RESULT_NO_FIRMWARE, if the firmware is not found. Result = RESULT_ALREADY_IN_FIRMWARE, if this command is called from the firmware.

Command HASF

```
result_t service_command_has_firmware_impl (device_t id, service_result_t*sresult)
```

Command code (CMD): "hasf" or 0x66736168.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (15 bytes)

INT32U	CMD	Command (answer)
INT8U	sresult	Result of command.
INT8U	Reserved [8]	Reserved (8 bytes)

Description:
Check for firmware on device. Result = RESULT_NO_FIRMWARE, if the firmware is not found. Result = RESULT_HAS_FIRMWARE, if the firmware found.

Command WKEY

```
result_t service_command_write_key_impl (device_t id, unsigned int* Key, unsigned int* sresul
```

Command code (CMD): "wkey" or 0x79656B77.

Request: (46 bytes)

INT32U	CMD	Command
INT8U	Key [32]	Protection key (256 bit).
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Answer: (15 bytes)

INT32U	CMD	Command (answer)
INT8U	sresult	Result of command.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Description:
Write command key to decrypt the firmware. Result = RESULT_OK, if the command loader. Result = RESULT_HARD_ERROR, if at the time of the command there was mistake. Result not available using the library write_key, the field value is processed by the function. Can be used by manufacturer only.

Command CONN

```
result_t service_command_connect_impl (device_t id, unsigned int* sresult)
```

Command code (CMD): "conn" or 0x6E6E6F63.

Request: (14 bytes)

INT32U	CMD	Command
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Answer: (15 bytes)

INT32U	CMD	Command (answer)
INT8U	sresult	Result of command.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Description:
Command to open a session ISP (in-system programming) when downloading the firmware. Result = RESULT_OK, if the command loader. Result = RESULT_SOFT_ERROR, if at the time of the command error occurred. Result not available using the library command_update_firmware, the field value is processed by the function.

Command DISC

```
result_t service_command_disconnect_impl (device_t id, unsigned int* sresult)
```

Command code (CMD): "disc" or 0x63736964.

Request: (14 bytes)

INT32U	CMD	Command
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Answer: (15 bytes)

INT32U	CMD	Command (answer)
INT8U	sresult	Result of command.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Description:
Command to close the session ISP (in-system programming) when loading firmware. Result = RESULT_OK, if the command loader. Result = RESULT_HARD_ERROR, if at the time of the command hardware error occurred. Result = RESULT_SOFT_ERROR, if at the time of the command software error occurred. Result not available using the library command_update_firmware, the field value is processed by the function.

Command WDAT

```
result_t service_command_write_data_impl (device_t id, unsigned int* Data)
```

Command code (CMD): "wdat" or 0x74616477.

Request: (142 bytes)

INT32U	CMD	Command
INT8U	Data [128]	Encoded firmware.
INT8U	Reserved [8]	Reserved (8 bytes)
INT16U	CRC	Checksum

Answer: (4 bytes)

INT32U	CMD	Command (answer)

Description:
Writes encoded firmware in the controller flash memory. Result of each packet write is not available. Overall result is available when firmware upload is finished.

Command REST

```
result_t service_command_reset_impl (device_t id)
```

Command code (CMD): "rest" or 0x74736572.

Request: (4 bytes)

INT32U	CMD	Command

Answer: (0 bytes)

Description:
The reset command controller and go into bootloader mode, for compatibility with the exchange protocol to the loader. No response to this command.

Controller error response types

ERRC

Answer: (4 bytes)
Code: "errc" or 0x63727265

INT32U	"errc"	Command error

Description:
Controller answers with "errc" if the command is either not unrecognized or cannot be processed and sets the correspoding bit in status data structure.

ERRD

Answer: (4 bytes)
Code: "errd" or 0x64727265

INT32U	"errd"	Data error

Description:
Controller answers with "errd" if the CRC of the data section computed by the controller doesn't match the received CRC field and sets the correspoding bit in status data structure.

ERRV

Answer: (4 bytes)
Code: "errv" or 0x76727265

INT32U	"errv"	Value error

Description:
Controller answers with "errv" if any of the values in the command are out of acceptable range and can not be applied. Inacceptable value is replaced by a rounded, truncated or default value. Controller also sets the correspoding bit in status data structure.

6.3. 8SMC1-USBhF software compatibility

8SMC4-USB supports software written for 8SMC1-USBhF controllers through the USMCDLL/libximc compatibility library. The table below lists functional differences: first column lists USMCDLL library functions, second column lists corresponding paragraph in 8SMC1-USBhF controller user's manual, third column contains this function features in the migration library.

Function call	**8SMC1-USBhF**	**8SMC4-USB**
USMC_Init	7.5.3	Uses libximc enumerate_devices, get_enumerate_device_information, get_enumerate_device_serial, get_device_count, get_device_name, open_device functions. Queries all COM-ports present in the system.
USMC_GetState	7.5.4	Uses libximc get_status and get_engine_settings functions. AReset, EMReset, RotTrErr flags in USMC_State structure are always set to false.
USMC_SaveParametersToFlash	7.5.5	Uses libximc command_save_settings function.
USMC_GetMode	7.5.6	Uses libximc get_edges_settings, get_power_settings, get_control_settings, get_ctp_settings, get_sync_out_settings functions. EMReset, ResetRT, SyncOUTR, EncoderEn, EncoderInv, ResBEnc, ResEnc flags in USMC_Mode are always set to false, SyncINOp flag is always set to true.
USMC_SetMode	7.5.7	Uses all functions used by USMC_GetMode and their "set_" equivalents. Ignores ResetD, RotTeEn, RotTrOp, ResetRT, SyncOUTR, SyncINOp, EncoderEn flags.
USMC_GetParameters	7.5.8	Uses libximc get_secure_settings, get_engine_settings, get_move_settings, get_feedback_settings, get_power_settings, get_control_settings, get_ctp_settings, get_home_settings, get_sync_out_settings functions. Returns zeroes in BTimeoutR, BTimeoutD, MinP, MaxLoft, StartPos fields.
USMC_SetParameters	7.5.9	Uses all functions used by USMC_GetParameters and their "set_" equivalents. Ignores BTimeoutR, BTimeoutD, MinP, MaxLoft, StartPos parameters.
USMC_GetStartParameters	7.5.10	Uses libximc get_move_settings, get_engine_settings functions. WSyncIN, SyncOUTR, ForceLoft flags in USMC_StartParameters structure are always set to false.
USMC_Start	7.5.11	Uses libximc get_move_settings, get_engine_settings, set_move_settings, set_engine_settings, command_move functions.
USMC_Stop	7.5.12	Uses libximc command_stop function.
USMC_SetCurrentPosition	7.5.13	Uses libximc set_position function.
USMC_GetEncoderState	7.5.14	Uses libximc get_status function.
USMC_GetLastErr	7.5.15	Does not modify its arguments. Error status is indicated by the nonzero return code of the corresponding USMC_ function.
USMC_Close	7.5.16	Uses libximc close_device function.

Migration library does not require and does not interact with background MicroSMC.exe process. It uses libximc.dll library to interface with 8SMC4-USB controllers.
All composite USMC_ functions containing several libximc function calls terminate if any of the underlying libximc functions returns an error. In this case controller may be left with partially saved settings. Nonzero return code of USMC_ functions is equal to the return code of the libximc function which returned an error.

Application example: usmcdll_libximc_test.zip

6.4. Libximc library timeouts

A number of timeouts and wait times are used when working with XiLab program or your own application using libximc library to detect errors and support robust controller operation. A list of times is provided below, together with reasons. Times are optimized for the USB connection on a modern PC. It is important to take delays into account when designing your own signal transmission lines to avoid erroneous timeouts.

Condition	Name	Time in milliseconds
Enumeration timeout. Happens if device type cannot be determined.	ENUMERATE_TIMEOUT_TIME	100
Port open timeout. Happens if library cannot open port.	DEFAULT_TIMEOUT_TIME	5000
Wait time when no data is received from device.	DEFAULT_TIMEOUT_TIME	5000
Wait time on device open.	RESET_TIME/2	50
Wait time from controller reset to device reappearance on firmware reflash.	RESET_TIME*1.2 + DEFAULT_TIMEOUT_TIME	5120
Wait time on flash sector write.	FLASH_SECTIONWRITE_TIME	100
Reconnect timeout on flash update .	XISM_PORT_DETECT_TIME	60000

7. Files

7.1. Configuration files

1. Translation Stages
2. Rotation Stages
3. Vertical Translation Stages
4. Screws and Actuators
5. Motorized Goniometers
6. Mirror Mounts
7. Motorized Attenuators
8. Motorized Iris Diaphragms

Translation Stages


8MT160 - Motorized Delay Line


8MT295 - Long-Travel Motorized Linear Stages


8MT195 - Long-Travel Motorized Linear Stages


8MT167 - Motorized Translation Stage


8MT173 - Motorized Translation Stages


8MT173DC - Motorized Translation Stages


8MT50 - Motorized Translation Stages


8MT30 - Narrow Motorized Translation Stages


8MT175 - Motorized Translation Stages


8MT177 - Motorized Translation Stage


8MT184 - Motorized Translation Stage


8MT193 - Motorized Translation Stage


8MT200 - Motorized Translation Stages


8MTF - Motorized XY Scanning Stage


8MTF2 - Motorized Fiber Coupling Stage

- 8MTFV - Motorized Translation Stage
- 8MTV - Motorized Translation Stage

8MT160 - Motorized Delay Line

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT160-300	300	mm	8MT160-300_300_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT160	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT160-300-MEn1	300	mm	8MT160-300-MEn1_300_4247_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT160	4247	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MT295 - Long-Travel Motorized Linear Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT295-2.5-X	340	mm	8MT295-2.5-X_340_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT295-2.5	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT295-4-X	540	mm	8MT295-4-X_540_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT295-4	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT295-5-X	740	mm	8MT295-5-X_740_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT295-5	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT295-10-X	840	mm	8MT295-10-X_840_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT295-10	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT295-2.5-Z	340	mm	8MT295-2.5-Z_340_5918-B_NoEnc_AB41_NoAcc_NoPD_XismusbDef.cfg	8MT295-2.5	5918-B	NoEnc	AB41	NoAcc	NoPD	XismusbDef	XIDef
8MT295-4-Z	540	mm	8MT295-4-Z_540_5918-B_NoEnc_AB41_NoAcc_NoPD_XismusbDef.cfg	8MT295-4	5918-B	NoEnc	AB41	NoAcc	NoPD	XismusbDef	XIDef
8MT295-5-Z	740	mm	8MT295-5-Z_740_5918-B_NoEnc_AB41_NoAcc_NoPD_XismusbDef.cfg	8MT295-5	5918-B	NoEnc	AB41	NoAcc	NoPD	XismusbDef	XIDef
8MT295-10-Z	840	mm	8MT295-10-Z_840_5918-B_NoEnc_AB41_NoAcc_NoPD_XismusbDef.cfg	8MT295-10	5918-B	NoEnc	AB41	NoAcc	NoPD	XismusbDef	XIDef

8MT195 - Long-Travel Motorized Linear Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT195-2.5-X	340	mm	8MT195-2.5-X_340_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT195-2.5	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT195-4-X	540	mm	8MT195-4-X_540_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT195-4	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT195-5-X	740	mm	8MT195-5-X_740_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT195-5	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT195-10-X	840	mm	8MT195-10-X_840_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT195-10	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT195-2.5-Z	340	mm	8MT195-2.5-Z_340_5918-B_NoEnc_AB41_NoAcc_NoPD_XismusbDef.cfg	8MT195-2.5	5918-B	NoEnc	AB41	NoAcc	NoPD	XismusbDef	XIDef
8MT195-4-Z	540	mm	8MT195-4-Z_540_5918-B_NoEnc_AB41_NoAcc_NoPD_XismusbDef.cfg	8MT195-4	5918-B	NoEnc	AB41	NoAcc	NoPD	XismusbDef	XIDef
8MT195-5-Z	740	mm	8MT195-5-Z_740_5918-B_NoEnc_AB41_NoAcc_NoPD_XismusbDef.cfg	8MT195-5	5918-B	NoEnc	AB41	NoAcc	NoPD	XismusbDef	XIDef
8MT195-10-Z	840	mm	8MT195-10-Z_840_5918-B_NoEnc_AB41_NoAcc_NoPD_XismusbDef.cfg	8MT195-10	5918-B	NoEnc	AB41	NoAcc	NoPD	XismusbDef	XIDef

8MT167 - Motorized Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT167-25BS1	25	mm	8MT167-25BS1_25_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT167BS	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT167-25LS	25	mm	8MT167-25LS_25_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT167LS	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT167-25LS-MEn1	25	mm	8MT167-25LS-MEn1_25_28_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT167LS	28	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT167M-25BS1	25	mm	8MT167M-25BS1_25_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT167BS	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT167M-25LS	25	mm	8MT167M-25LS_25_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT167LS	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT167-100	100	mm	8MT167-100_100_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT167	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT167-100-28	100	mm	8MT167-100-28_100_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT167	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT167-100C28	100	mm	8MT167-100C28_100_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT167	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT167-100DCE2	100	mm	8MT167-100DCE2_100_DCERE25_HEDS5540_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT167DC	DCERE25	HEDS5540	NoBrake	NoAcc	NoPD	XismusbDef	XIDefDC

8MT173 - Motorized Translation Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT173-10	10	mm	8MT173-10_10_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173-20	20	mm	8MT173-20_20_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173-25	25	mm	8MT173-25_25_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173-30	30	mm	8MT173-30_30_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173-10-MEn1	10	mm	8MT173-10-MEn1_10_28_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173	28	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173-20-MEn1	20	mm	8MT173-20-MEn1_20_28_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173	28	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173-25-MEn1	25	mm	8MT173-25-MEn1_25_28_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173	28	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173-30-MEn1	30	mm	8MT173-30-MEn1_30_28_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173	28	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173V-10-VSS42	10	mm	8MT173V-10-VSS42_10_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173V-20-VSS42	20	mm	8MT173V-20-VSS42_20_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173V-25-VSS42	25	mm	8MT173V-25-VSS42_25_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT173V-30-VSS42	30	mm	8MT173V-30-VSS42_30_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MT173DC - Motorized Translation Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT173-10DCE2	10	mm	8MT173-10DCE2_10_DCERE13_EncMRtypeS_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173DC	DCERE13	EncMRtypeS	NoBrake	NoAcc	NoPD	XismusbDef	XIDefDC
8MT173-20DCE2	20	mm	8MT173-20DCE2_20_DCERE13_EncMRtypeS_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173DC	DCERE13	EncMRtypeS	NoBrake	NoAcc	NoPD	XismusbDef	XIDefDC
8MT173-25DCE2	25	mm	8MT173-25DCE2_25_DCERE13_EncMRtypeS_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173DC	DCERE13	EncMRtypeS	NoBrake	NoAcc	NoPD	XismusbDef	XIDefDC
8MT173-30DCE2	30	mm	8MT173-30DCE2_30_DCERE13_EncMRtypeS_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT173DC	DCERE13	EncMRtypeS	NoBrake	NoAcc	NoPD	XismusbDef	XIDefDC

8MT50 - Motorized Translation Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT50-100BS1	100	mm	8MT50-100BS1_100_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-100BS1-MEn1	100	mm	8MT50-100BS1-MEn1_100_4247_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-100XY	100	mm	8MT50-100XY_100_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-100XYZ	100	mm	8MT50-100XYZ_100_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-150BS1	150	mm	8MT50-150BS1_150_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-150BS1-MEn1	150	mm	8MT50-150BS1-MEn1_150_4247_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-150XY	150	mm	8MT50-150XY_150_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-150XYZ	150	mm	8MT50-150XYZ_150_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-200BS1	200	mm	8MT50-200BS1_200_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-200BS1-MEn1	200	mm	8MT50-200BS1-MEn1_200_4247_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-200XY	200	mm	8MT50-200XY_200_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT50-200XYZ	200	mm	8MT50-200XYZ_200_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT50	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MT30 - Narrow Motorized Translation Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT30-50	50	mm	8MT30-50_50_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT30	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT30V-50-VSS42	50	mm	8MT30V-50-VSS42_50_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT30V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MT175 - Motorized Translation Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT175-50	50	mm	8MT175-50_50_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175-100	100	mm	8MT175-100_100_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175-150	150	mm	8MT175-150_150_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175-200	200	mm	8MT175-200_200_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175-50-MEn1	50	mm	8MT175-50-MEn1_50_4247_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	4247	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175-100-MEn1	100	mm	8MT175-100-MEn1_100_4247_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	4247	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175-150-MEn1	150	mm	8MT175-150-MEn1_150_4247_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	4247	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175-200-MEn1	200	mm	8MT175-200-MEn1_200_4247_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	4247	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175V-50-VSS42	50	mm	8MT175V-50-VSS42_50_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175V-100-VSS42	100	mm	8MT175V-100-VSS42_100_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175V-150-VSS42	150	mm	8MT175V-150-VSS42_150_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT175V-200-VSS42	200	mm	8MT175V-200-VSS42_200_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT175	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MT177 - Motorized Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT177-100	100	mm	8MT177-100_100_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT177	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MT177-100-28	100	mm	8MT177-100-28_100_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT177	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MT184 - Motorized Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT184-13	13	mm	8MT184-13_13_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT184	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MT193 - Motorized Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT193	100	mm	8MT193_100_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT193	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MT200 - Motorized Translation Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MT200-100	100	mm	8MT200-100_100_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MT200	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MTF - Motorized XY Scanning Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MTF-102LS05	102	mm	8MTF-102LS05_102_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MTF-102	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MTF-75LS05	75	mm	8MTF-75LS05_75_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MTF-75	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MTF2 - Motorized Fiber Coupling Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MTF2	2	mm	8MTF2_2_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MTF2	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MTFV - Motorized Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MTFV	40	mm	8MTFV_40_D423_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MTFV	D423	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MTV - Motorized Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MTV	20	mm	8MTV_20_D423_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MTV	D423	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

Rotation Stages


8MR151 - Motorized Rotation Stages


8MR170 - Motorized Rotation Stages


8MR174 - Motorized Rotation Stage


8MR190 - Motorized Rotation Stage


8MR191 - Motorized Rotation Stage


8MRB250 - Large Motorized Rotation Stage


8MRU - Universal Motorized Rotation Stages


8MRH240 - Large High Capacity Rotary Stage

8MR151 - Motorized Rotation Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MR151-1	360	deg	8MR151-1_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR151	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR151R-1	360	deg	8MR151R-1_360_28_RevS_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR151R	28	RevS	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR151-1-MEn1	360	deg	8MR151-1-MEn1_360_28_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR151	28	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR151-30	360	deg	8MR151-30_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR151	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR151R-30	360	deg	8MR151R-30_360_28_RevS_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR151R	28	RevS	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR151-30-MEn1	360	deg	8MR151-30-MEn1_360_28_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR151	28	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR151E-1	360	deg	8MR151E-1_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR151	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR151E-30	360	deg	8MR151E-30_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR151	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR151RE-1	360	deg	8MR151RE-1_360_28_RevS_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR151R	28	RevS	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR151RE-30	360	deg	8MR151RE-30_360_28_RevS_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR151R	28	RevS	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MR170 - Motorized Rotation Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MR170-190	6	deg	8MR170-190_6_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR170	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MR174 - Motorized Rotation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MR174-11-28	360	deg	8MR174-11-28_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR174	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR174-11-28S	360	deg	8MR174-11-28S_360_28S_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR174	28S	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR174-11-20	360	deg	8MR174-11-20_360_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR174	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR174-11-VSS42	360	deg	8MR174-11-VSS42_360_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR174V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR174E-11-28	360	deg	8MR174E-11-28_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR174	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR174E-11-28S	360	deg	8MR174E-11-28S_360_28S_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR174	28S	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR174E-11-20	360	deg	8MR174E-11-20_360_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR174	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR174E-11-VSS42	360	deg	8MR174E-11-VSS42_360_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR174V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MR190 - Motorized Rotation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MR190-2-28	360	deg	8MR190-2-28_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190-2-28-MEn1	360	deg	8MR190-2-28-MEn1_360_28_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190	28	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190-2-4233	360	deg	8MR190-2-4233_360_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190-2-4247	360	deg	8MR190-2-4247_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190-2-ZSS43	360	deg	8MR190-2-ZSS43_360_ZSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190	ZSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190V-2-VSS42	360	deg	8MR190V-2-VSS42_360_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190E-2-28	360	deg	8MR190E-2-28_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190E-2-28-MEn1	360	deg	8MR190E-2-28-MEn1_360_28_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190	28	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190E-2-4233	360	deg	8MR190E-2-4233_360_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190E-2-4247	360	deg	8MR190E-2-4247_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190E-2-ZSS43	360	deg	8MR190E-2-ZSS43_360_ZSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190	ZSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190EV-2-VSS42	360	deg	8MR190EV-2-VSS42_360_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190-90-4247	360	deg	8MR190-90-4247_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190-90-4247	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190-90-4247-MEn1	360	deg	8MR190-90-4247-MEn1_360_4247_HEDM5500_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190-90-4247	4247	HEDM5500	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190-90-59	360	deg	8MR190-90-59_360_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190-90-59	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR190-90-VSS43	360	deg	8MR190-90-VSS43_360_VSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR190-90-VSS43	VSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MR191 - Motorized Rotation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MR191-1-28	360	deg	8MR191-1-28_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-1-4233	360	deg	8MR191-1-4233_360_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-1-4247	360	deg	8MR191-1-4247_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-1-ZSS43	360	deg	8MR191-1-ZSS43_360_ZSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	ZSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-30-28	360	deg	8MR191-30-28_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-30-4233	360	deg	8MR191-30-4233_360_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-30-4247	360	deg	8MR191-30-4247_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-30-ZSS43	360	deg	8MR191-30-ZSS43_360_ZSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	ZSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-28	360	deg	8MR191-28_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-4233	360	deg	8MR191-4233_360_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-4247	360	deg	8MR191-4247_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191-ZSS43	360	deg	8MR191-ZSS43_360_ZSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	ZSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-1-28	360	deg	8MR191E-1-28_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-1-4233	360	deg	8MR191E-1-4233_360_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-1-4247	360	deg	8MR191E-1-4247_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-1-ZSS43	360	deg	8MR191E-1-ZSS43_360_ZSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	ZSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-30-28	360	deg	8MR191E-30-28_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-30-4233	360	deg	8MR191E-30-4233_360_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-30-4247	360	deg	8MR191E-30-4247_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-30-ZSS43	360	deg	8MR191E-30-ZSS43_360_ZSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	ZSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-28	360	deg	8MR191E-28_360_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-4233	360	deg	8MR191E-4233_360_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-4247	360	deg	8MR191E-4247_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191E-ZSS43	360	deg	8MR191E-ZSS43_360_ZSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191	ZSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191V-1-VSS42	360	deg	8MR191V-1-VSS42_360_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191V-30-VSS42	360	deg	8MR191V-30-VSS42_360_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191V-VSS42	360	deg	8MR191V-VSS42_360_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191EV-1-VSS42	360	deg	8MR191EV-1-VSS42_360_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191EV-30-VSS42	360	deg	8MR191EV-30-VSS42_360_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MR191EV-VSS42	360	deg	8MR191EV-VSS42_360_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MR191V	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MRB250 - Large Motorized Rotation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MRB250-152-59	360	deg	8MRB250-152-59_360_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MRB250	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MRB250-152-59D	360	deg	8MRB250-152-59D_360_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MRB250	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MRU - Universal Motorized Rotation Stages

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MRU-1	360	deg	8MRU-1_360_28S_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MRU	28S	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MRU-1TP	360	deg	8MRU-1TP_360_28S_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MRU	28S	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MRU-1WA	360	deg	8MRU-1WA_360_28S_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MRU	28S	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MRH240 - Large High Capacity Rotary Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MRH240-60	360	deg	8MRH240-60_360_5918_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MRH240	5918	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

Vertical Translation Stages


8MVT100 - Vertical Translation Stage


8MVT120 - Vertical Translation Stage


8MVT188 - Vertical Translation Stage


8MVT40 - Vertical Translation Stage


8MVT70 - Vertical Translation Stage

8MVT100 - Vertical Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MVT100-25-1	26	mm	8MVT100-25-1_26_4118L1804R_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MVT100	4118L1804R	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MVT120 - Vertical Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MVT120-5-4247	5	mm	8MVT120-5-4247_5_4247R_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MVT120	4247R	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MVT188 - Vertical Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MVT188-20	8744052	steps	8MVT188-20_8744052_5618R_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MVT188	5618R	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MVT40 - Vertical Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MVT40-13-1	13	mm	8MVT40-13-1_13_28SR_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MVT40	28SR	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MVT70 - Vertical Translation Stage

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MVT70-13-1	13	mm	8MVT70-13-1_13_4118S1404R_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MVT70	4118S1404R	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

Screws and Actuators


8MS00 - Motorized Screws


8CMA06 - Motorized Actuator


8CMA20 - Compact Motorized Actuator


8CMA28 - Motorized Linear Actuator


8CMA16DC - Motorized Linear Actuator

8MS00 - Motorized Screws

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MS00-10	10	mm	8MS00-10_10_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MS00-10	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MS00-25	25	mm	8MS00-25_25_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MS00-25	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MS00-10-28	10	mm	8MS00-10-28_10_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MS00-10	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MS00-25-28	25	mm	8MS00-25-28_25_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MS00-25	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MS00V-10-VSS43	10	mm	8MS00V-10-VSS43_10_VSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MS00V-10	VSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MS00V-25-VSS43	25	mm	8MS00V-25-VSS43_25_VSS43_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MS00V-25	VSS43	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8CMA06 - Motorized Actuator

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8CMA06-13-10	13	mm	8CMA06-13-10_13_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8CMA06-13	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8CMA06-13-15	13	mm	8CMA06-13-15_13_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8CMA06-13	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8CMA06-25-15	25	mm	8CMA06-25-15_25_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8CMA06-25	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8CMA20 - Compact Motorized Actuator

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8CMA20-8-15	8	mm	8CMA20-8-15_8_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8CMA20	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8CMA28 - Motorized Linear Actuator

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8CMA28-10	10	mm	8CMA28-10_10_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8CMA28	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8CMA16DC - Motorized Linear Actuator

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8CMA16DC-13	13	mm	8CMA16DC-13_13_DCE1524_IE216_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8CMA16DC	DCE1524	IE216	NoBrake	NoAcc	NoPD	XismusbDef	XIDefDC
8CMA16DC-25	25	mm	8CMA16DC-25_25_DCE1524_IE216_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8CMA16DC	DCE1524	IE216	NoBrake	NoAcc	NoPD	XismusbDef	XIDefDC

Motorized Goniometers


8MG00 - Motorized Goniometers


8MG99 - Motorized Goniometer

8MG00 - Motorized Goniometers

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MG00-50	0.5	deg	8MG00-50_0.5_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MG00-50	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MG00-80	0.5	deg	8MG00-80_0.5_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MG00-80	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MG00V-50	0.5	deg	8MG00V-50_0.5_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MG00-50	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MG00V-80	0.5	deg	8MG00V-80_0.5_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MG00-80	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MG99 - Motorized Goniometer

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MG99-50	0.5	deg	8MG99-50_0.5_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MG99-50	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MG99-80	0.5	deg	8MG99-80_0.5_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MG99-80	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MG99V-50	0.5	deg	8MG99V-50_0.5_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MG99-50	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MG99V-80	0.5	deg	8MG99V-80_0.5_VSS42_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MG99-80	VSS42	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

Mirror Mounts


8MTOM2 - Motorized Two Axis Translation Optical Mount


8MUP21 - Motorized Optical Mount


8MBM24 - Motorized Mirror Mounts


8MMA60 - Motorized Mirror Mounts


8MBM57 - Large Aperture Motorized Mirror Mount


8MKVDOM - Motorized Vertical drive optical mount

8MTOM2 - Motorized Two Axis Translation Optical Mount

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MTOM2-1	4	mm	8MTOM2-1_4_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MTOM2	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MUP21 - Motorized Optical Mount

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MUP21-2	4	mm	8MUP21-2_4_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MUP21	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MBM24 - Motorized Mirror Mounts

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MBM24-1	4	mm	8MBM24-1_4_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MBM24	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MBM24-2	4	mm	8MBM24-2_4_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MBM24	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MBM24-3	4	mm	8MBM24-3_4_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MBM24	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MMA60 - Motorized Mirror Mounts

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MMA60-1	5	deg	8MMA60-1_5_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MMA60	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MMA60-2	5	deg	8MMA60-2_5_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MMA60	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MMA60-40	5	deg	8MMA60-40_5_4233_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MMA60	4233	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MBM57 - Large Aperture Motorized Mirror Mount

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MBM57-2	5	deg	8MBM57-2_5_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MBM57	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MBM57-4	5	deg	8MBM57-4_5_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MBM57	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MBM57-6	5	deg	8MBM57-6_5_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MBM57	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

8MKVDOM - Motorized Vertical drive optical mount

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MKVDOM-1	6	mm	8MKVDOM-1_6_20_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MKVDOM	20	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
8MKVDOM-2	6	mm	8MKVDOM-2_6_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MKVDOM	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

Motorized Attenuators

10MCWA168 - Motorised Closed Variable Wheel Attenuator

10MWA168 - Motorized Variable Wheel Attenuator

10MCWA168 - Motorised Closed Variable Wheel Attenuator

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
10MCWA168-1	360	deg	10MCWA168-1_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	10MCWA168	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
10MCWA168-20	360	deg	10MCWA168-20_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	10MCWA168	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

10MWA168 - Motorized Variable Wheel Attenuator

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
10MWA168-1	360	deg	10MWA168-1_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	10MWA168	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef
10MWA168-20	360	deg	10MWA168-20_360_4247_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	10MWA168	4247	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

Motorized Iris Diaphragms

8MID98 - Motorized Iris Diaphragm

8MID98 - Motorized Iris Diaphragm

Update: 26 декабря 2013

Part number	Travel range	Units	Profile	Positioner	Motor	Encoder	Brake	Accessories	Peripherals	Controller	XILab
8MID98-4-90	53010	steps	8MID98-4-90_53010_28_NoEnc_NoBrake_NoAcc_NoPD_XismusbDef.cfg	8MID98	28	NoEnc	NoBrake	NoAcc	NoPD	XismusbDef	XIDef

7.2. Software

Full XIMC software package for Windows, last updated 27.12.2013

Includes latest versions of XILab user interface application for Windows only, libximc library development kit, firmware binary and user manual.

Download full software package. Disconnect all controllers from the PC. Launch XILab installer. Full installation manual can be found here.

XILab, version 1.9.14, last updated 11.12.2013

XILab can be downloaded here:

Windows XP, Vista, 7, 8	xilab-1.9.14.exe
Linux Debian/Ubuntu 32-bit	xilab-1.9.14-1_i386.deb
Linux Debian/Ubuntu 64-bit	xilab-1.9.14-1_amd64.deb
Linux RedHat/OpenSUSE 64-bit	xilab-1.9.14-1.x86_64.rpm
MacOS X	xilab-1.9.14-osx64.tar.gz

All XILab versions
Compatibility table
XILab changelog

Development kit, version 2.2.2, last updated 04.12.2013

Download and extract: ximc-2.2.2.tar.gz

Development kit manual may be found here and in the /docs-en/index.html development kit file. Library is located in /ximc-2.2.2 and examples are in /examples.

PDF-version of the development kit manual can be downloaded here

All libximc versions
Compatibility table
libximc changelog

Controller firmware, version 3.8.7, last updated 21.11.2013

Download: xismusb_3.8.7.cod

Firmware update instructions:

1. Download firmware file.
2. Start XILab
3. It is recommended to make a backup of your XILab configuration file before firmware update.
4. Open "Settings... -> Device configuration -> About device", press "Update" button, select new firmware file, press "Open".
5. XILab might stop responding for a while. Please wait while firmware is being uploaded, it should take about 10-15 seconds.
6. If settings have changed then load your configuration file and save it to the controller flash memory by pressing "Save to flash" button.

All firmware versions
Compatibility table
Firmware changelog

Configuration files

Configuration files may be found here.

LabView examples

Labview examples can be downloaded here: XIMC_Labview_2.2.1.7z

Drivers

On Windows controller doesn't need additional drivers, however you need an inf-file for the controller to be

recognized by the operating system. It will be automatically installed with XILab. It can be found in "C:\Program Files\XILab\Driver" after installation. Also you may download it here: XIMC_driver.inf

Linux and MacOS do not require driver files.

8SMC1-USBhF migration library

Migration library may be downloaded here: usmcdll_libximc_2.2.1.zip
Detailed information is available here.

All XILab versions

Version	Windows XP, Vista, 7, 8	Linux Debian/Ubuntu 32-bit	Linux Debian/Ubuntu 64-bit	Linux RedHat/OpenSUSE 64-bit	MacOS X
1.9.14	Download	Download	Download	Download	Download
1.9.13	Download	Download	Download	Download	Download
1.9.12	Download	Download	Download	Download	Download
1.9.11	Download	Download	Download	Download	Download
1.8.30	Download	Download	Download	Download	Download
1.8.29	Download	Download	Download	Download	Download
1.8.28	Download	Download	Download	Download	Download

XILab changelog

- 1.9.13
  - Bugfix: Mac OS X Qwt 6.0 library paths fixed, application error is gone.
  - Bugfix: Floating point number spinboxes round off their values to float-precision on lost focus instead of reopened window.
  - Bugfix: "Save to file" button saves current displayed settings instead of those applied to the controller.
  - Bugfix: Position control page is now grayed out with encoder or hall sensor feedback, like it should be.
  - Bugfix: Z axis background color changes to window background color if this axis is inactive.
  - Bugfix: Movement indicator icon is now shown, added warning if svg resource cannot be loaded.
  - Bugfix: Removed erroneous limits in "power off delay" and "current in hold mode" ranges on power management page.
- 1.9.12
  - Bugfix: Added missing set_controller_name, get_controller_name, set_add_sync_in_action, set_add_sync_in_action_calb, get_chart_data, get_analog_data, get_debug_read, set_stage_name, get_stage_name functions to scripts.
  - Bugfix: Properly handle situations with lost devices and device reordering in multiaxis interface.
  - Bugfix: Grayout settings pages that are not applicable in multiaxis interface.
  - Bugfix: Grayout power management page for DC and BLDC engine types.
  - Bugfix: Fix slider axis multiplier for really small ranges in single-axis interface.
- 1.9.11
  - Major change: Unified Windows installer for both 32-bit and 64-bit systems.
  - Major change: new communication protocol 16.6 is now supported.
  - Feature: Added user units calibration to single-axis interface.
  - Feature: Added power driver failure and borders misset alarm flags visualization.
  - Feature: Added EEPROM precedence option that allows either load settings from memory equipped positioner by default or ignore that settings.
  - Feature: Added working region support for multiaxis interface.
  - Feature: Secondary GUI windows are bought back when application focus is lost and restored. It improves usability.
  - Feature: Multiaxis interface borders are set according to controller settings.
  - Feature: Optimized positioner EEPROM readings to avoid excess communication.
  - Feature: A valid security certificate allows flawless Windows 8 installation now.
  - Feature: Added visualization for free buffer space in controller for synchronization actions.
  - Feature: Default logging settings are optimized to store important data before something happens.
  - Feature: Removed Update borders button because it was confusing to users.
  - Feature: MacOS X version is distributed as .tar.gz archive instead of .dmg volume.
  - Feature: Resistive potentiometer support removed.
  - Feature: Hall sensors are added to graphs.
  - Feature: Scripts now have access to calibrated functions.
  - Feature: Added wait_for_stop function to scripts.
  - Feature: Added get_next_serial function to scripts.
  - Feature: Single-axis interface now supports multiaxis script syntax.

    - Feature: Added log window to multiaxis interface.
    - Feature: Multiaxis calibration is now per-axis.
    - Feature: Script syntax checker.
    - Bugfix: DC engines nominal voltage was disabled in GUI when voltage limiting is off, but it is used in PID control and must never be grayed out.
    - Bugfix: PWM level chart updated values at the wrong times.
    - Bugfix: TTL sync page was skewed on MacOS X.
    - Bugfix: Added default name when saving profile.
    - Bugfix: Scripts in multiaxis interface now work properly.
    - Bugfix: Step division couldn't be changed. Now it can.
    - Bugfix: Slider settings for single axis and multiaxis interface are compatible now.
    - Bugfix: Array elements in settings structures are now supported.
    - Bugfix: Limit_switch_1_pushed_is_closed setting was incorrectly named Limit_switch_1_pushed_is_open.
    - Bugfix: Fixed smart Stop button behaviour.
    - Bugfix: Stage accessories page did not save its settings.
    - Bugfix: Stage settings could not be saved to or loaded from file.
    - Bugfix: Script highlighting.
    - Bugfix: Scripting interface was unresponsive on long script files.
    - Bugfix: Move to and Shift to precision fixed.
    - Bugfix: Shortened example scripts.
    - Bugfix: Settings pages were inactive after unsuccessful firmware update.
    - Bugfix: Correctly save settings after losing connection to the device.
    - Bugfix: Axes control widget in multiaxis interface now doesn't move mouse while controlling with keyboard arrows.
    - Bugfix: Load engine settings for all pages (stepper, DC, BLDC).
    - Bugfix: No default user interface autoselections if no controller settings match.
    - Bugfix: Script editor font is now fixed size.
    - Bugfix: Stage pages use unified motor settings instead of separate motor type pages.
    - Bugfix: Incorrect rounding in spin box controls.
    - Bugfix: Slider uses user units.
    - Bugfix: Multiaxis interface incorrectly interpreted decimal places precision as significant figures.
    - Bugfix: Set_zero script works properly with single-phase homing.
    - Bugfix: Enabled revolution sensor.
    - Bugfix: Encoder output type display on stage page fixed.
    - Bugfix: Main window slider was incorrectly showing percentages instead of absolute values.

- 1.8.30
    - feature: Profile files added and updated

- 1.8.29
    - feature: Profile files added

- 1.8.28
    - feature: Initial autogenerated release

All libximc, usmcdll_libximc and XIMC_Labview versions

Version	Libximc	XIMC_Labview	usmcdll_libximc
2.2.2	Download	-	-
2.2.1	Download	Download	Download
2.2.0	Download	-	-
2.0.5	Download	-	-

Libximc changelog

- 2.2.2
    - Bugfix: Support 64-bit data types for LabVIEW bindings

- 2.2.1
    - Bugfix: Synchronization now works properly.

- 2.2.0
    - Major change: New communication protocol 16.6 is now in use.
    - Feature: Calibrated commands without microstep.
    - Feature: Arrays as calibrated fields.
    - Feature: Calibrated fields of velocities.
    - Feature: Added user units in C# and Delphi programming languages.
    - Feature: Added Java bindings.
    - Bugfix: libximc is built differently.

- 2.0.5
  - Initial public release

XIMC_Labview changelog

- 2.2.1
  - Initial version of XIMC labview library

Usmcdll_libximc changelog

- 2.2.1
  - Initial version of migration library.

All firmware versions

Version	File
3.8.7	Download
3.8.4	Download
3.8.3	Download

Firmware changelog

- 3.8.7
  - Major change: Protocol 16.6 with incompatibilities regarding EEPROM section of the commands.
  - Major change: Completely new motor revers logic. It removes bug with encoder reverse recognition and must be much more stable.
  - Major change: Revised power control logic.
  - Feature: A fixed position shift on button click when the external buttons are in control is implemented.
  - Feature: USB connection can restart itself when connection breaks. It helps to restore connection after statick discharge.
  - Feature: Backlash compensation with LOFT command works during motion with a define behaviour now.
  - Feature: BLDC engines is better supported due to new settings in protocol 16.6.
  - Fixed bug: The initial settings made first movement too power consumping at first 200 ms that can result in power supply overcurrent protection triggering.
  - Fixed bug: Previous version has buggy ZERO command operation. Fixed now.
  - Fixed bug: Border limits may become inoperable when any feedback is used due to zero speed at the moment of border triggering.
- 3.8.4
  - Fixed bug: Possible oscillations around target positions on low acceleration and deceleration
- 3.8.3
  - Added feature: Initial firmware

Compatibility table

Protocol version	XILab	libximc	firmware
16.3	1.8.28-1.8.30	2.0.5	3.8.3-3.8.4
16.6	1.9.11-1.9.13	2.2.1-2.2.2	3.8.7

8. Configuration file sections

8.1. Positioners

Translation Stages

8MT160

Check configuration file 8MT160
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=61&prod=motorized_delay_line

8MT295-2.5

Check configuration file 8MT295-2.5
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=305&prod=long-travel-motorized-stages

8MT295-4

Check configuration file 8MT295-4
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=305&prod=long-travel-motorized-stages

8MT295-5

Check configuration file 8MT295-5
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=305&prod=long-travel-motorized-stages

8MT295-10

Check configuration file 8MT295-10
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=305&prod=long-travel-motorized-stages

8MT195-2.5

Check configuration file 8MT195-2.5
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=259&prod=long-travel_motorized_linear_stage

8MT195-4

Check configuration file 8MT195-4
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=259&prod=long-travel_motorized_linear_stage

8MT195-5

Check configuration file 8MT195-5
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=259&prod=long-travel_motorized_linear_stage

8MT195-10

Check configuration file 8MT195-10
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=259&prod=long-travel_motorized_linear_stage

8MT173

Check configuration file 8MT173

Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=59&prod=motorized_translation_stages

8MT173V

Check configuration file 8MT173V
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=59&prod=motorized_translation_stages

8MT167

Check configuration file 8MT167
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=64&prod=motorized_translation_stage

8MT167LS

Check configuration file 8MT167LS
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=63&prod=linear_motor_positioning_stage

8MT167BS

Check configuration file 8MT167BS
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=63&prod=linear_motor_positioning_stage

8MT30

Check configuration file 8MT30
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=348&prod=narrow_motorized_translation_stages

8MT30V

Check configuration file 8MT30V
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=348&prod=narrow_motorized_translation_stages

8MT175

Check configuration file 8MT175
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=60&prod=motorized_linear_stages

8MT177

Check configuration file 8MT177
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=242&prod=motorized_stage

8MT184

Check configuration file 8MT184
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=292&prod=motorized_translation_stage

8MT193

Check configuration file 8MT193
Download configuration file.

Description http://www.standa.lt/products/catalog/motorised_positioners?item=62&prod=motorized_linear_stage

8MT200

Check configuration file 8MT200
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=282&prod=motorized_translation_stage

8MT50

Check configuration file 8MT50
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=308&prod=motorized_translation_stages

8MTF-75

Check configuration file 8MTF-75
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=311&prod=motrized_xy_scanning_stage

8MTF-102

Check configuration file 8MTF-102
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=311&prod=motrized_xy_scanning_stage

8MTF2

Check configuration file 8MTF2
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=249&prod=motorized_fiber_coupling_stage

Rotation Stages

8MR151

Check configuration file 8MR151
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=9&prod=motorized_rotation_stages

8MR151R

Check configuration file 8MR151R
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=9&prod=motorized_rotation_stages

8MR170

Check configuration file 8MR170
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=312&prod=motorized_rotary_stages

8MR174

Check configuration file 8MR174
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=68&prod=motorized_rotation_stage

8MR174V

Check configuration file 8MR174V
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=68&prod=motorized_rotation_stage

8MR190

Check configuration file 8MR190
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=244&prod=motorized_rotation_stage

8MR190V

Check configuration file 8MR190V
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=244&prod=motorized_rotation_stage

8MR190-90-59

Check configuration file 8MR190-90-59
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=325&prod=motorized_rotation_stage

8MR190-90-4247

Check configuration file 8MR190-90-4247
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=325&prod=motorized_rotation_stage

8MR190-90-VSS43

Check configuration file 8MR190-90-VSS43
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=325&prod=motorized_rotation_stage

8MR191

Check configuration file 8MR191
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=245&prod=motorized_rotary_stage

8MR191V

Check configuration file 8MR191V
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=245&prod=motorized_rotary_stage

8MRB250

Check configuration file 8MRB250
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=397&prod=Large-Motorized-Rotation-Stage

8MRH240

Check configuration file 8MRH240
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=469&prod=large-lhigh-capacity-rotary-stage

8MRU

Check configuration file 8MRU
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=336&prod=Universal_Motorized_Rotation_Stages

Vertical Translation Stages

8MVT100

Check configuration file 8MVT100
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=407&prod=motorized-vertical-stages

8MVT120

Check configuration file 8MVT120
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=385&prod=Motorized-Vertical-Lift-Stage

8MVT188

Check configuration file 8MVT188
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=70&prod=motorized_vertical_translation_stage

8MVT40

Check configuration file 8MVT40
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=315&prod=motorized_vertical_translation_stages

8MVT70

Check configuration file 8MVT70
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=417&prod=motorized-z-stage

Screws and actuators

8MS00-10

Check configuration file 8MS00-10
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=76&prod=motorized_screws

8MS00-25

Check configuration file 8MS00-25
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=76&prod=motorized_screws

8MS00V-10

Check configuration file 8MS00V-10
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=76&prod=motorized_screws

8MS00V-25

Check configuration file 8MS00V-25
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=76&prod=motorized_screws

8CMA06-13

Check configuration file 8CMA06-13
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=241&prod=motorized_actuator

8CMA06-25

Check configuration file 8CMA06-25
Download configuration file.

Description http://www.standa.lt/products/catalog/motorised_positioners?item=241&prod=motorized_actuator

8CMA20

Check configuration file 8CMA20
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=256&prod=compact_motorized_actuator

8CMA28

Check configuration file 8CMA28
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=250&prod=motorized_actuator

Motorized Goniometers

8MG00-50

Check configuration file 8MG00-50
Download configuration file.
Описание на сайте http://www.standa.lt/products/catalog/motorised_positioners?item=73&prod=motorized_goniometers

8MG00-80

Check configuration file 8MG00-80
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=73&prod=motorized_goniometers

8MG99-50

Check configuration file 8MG99-50
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=72&prod=motorized_goniometer

8MG99-80

Check configuration file 8MG99-80
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=72&prod=motorized_goniometer

Mirror Mounts

8MTOM2

Check configuration file 8MTOM2
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=257&prod=motorized_two_axis_translation_optical_mount

8MUP21

Check configuration file 8MUP21
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=484&prod=Motorized-optical-mount

8MBM24

Check configuration file 8MBM24
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=251&prod=motorized_mirror_mounts

8MMA60

Check configuration file 8MMA60
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=71&prod=motorized_mirror_mounts

8MBM57

Check configuration file 8MBM57
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=283&prod=large_aperture_motorized_mirror_mount

8MKVDOM

Check configuration file 8MKVDOM
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=300&prod=motorized_vertical_drive_optical_mount

Motorized Attenuators

10MCWA168

Check configuration file 10MCWA168
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=324&prod=Motorised_Closed_Variable_Wheel_Attenuator

10MWA168

Check configuration file 10MWA168
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=233&prod=motorized_variable_wheel_attenuator

Motorized Iris Diaphragms

8MID98

Check configuration file 8MID98
Download configuration file.
Description http://www.standa.lt/products/catalog/motorised_positioners?item=395&prod=Motorized-iris-diaphragm

8.2. Motors

20

Check configuration file 20
Download configuration file
Motor description file 20.pdf

28

Check configuration file 28
Download configuration file
Motor description file 28.pdf (see bipolar serial)

28S

Check configuration file 28S
Download configuration file
Motor description file 28S.pdf (see bipolar serial)

4118L1804R

Check configuration file 4118L1804R
Download configuration file
Motor description file attachment:4118L1804R.pdf

4118S1404R

Check configuration file 4118S1404R
Download configuration file
Motor description file attachment:4118S1404R.pdf

4247

Check configuration file 4247
Download configuration file
Motor description file 4247.pdf (see FL42STH47-1204B)

4247R

Check configuration file 4247R
Download configuration file
Motor description file 4247.pdf (see FL42STH47-1204B)

D423

Check configuration file D423
Download configuration file
Motor description file Dseries_Data.pdf (see D42.3)

5618

Check configuration file 5618
Download configuration file
Motor description file 5618.pdf

5618R

Check configuration file 5618R
Download configuration file
Motor description file 5618.pdf

5918

Check configuration file 5918
Download configuration file
Motor description file 5918.pdf (see bipolar serial)

5918-B

Check configuration file 5918-B
Download configuration file
Motor description file 5918-B.pdf (see bipolar serial)

VSS42

Check configuration file VSS42
Download configuration file
Motor description file VSS42.pdf (see VSS42.200.1.2)

VSS43

Check configuration file VSS43
Download configuration file
Motor description file VSS43.pdf (see VSS43.200.1.2)

ZSS43

Check configuration file ZSS43
Download configuration file
Motor description file ZSS43.pdf

DCERE25

Check configuration file DCERE25
Download configuration file
Motor description file DCERE25.pdf (see 118752)

DCERE13

Check configuration file DCERE13
Download configuration file
Motor description file RE-13-118512.pdf

DCE1524

Check configuration file DCE1524
Download configuration file
Motor description file EN_1524_SR_DFF.pdf (see 006 SR)

8.3. Encoders

NoEnc

Check configuration file NoEnc
Download configuration file

HEDM-5500

Check configuration file HEDM5500
Download configuration file
Description: HEDM-55xx.pdf

HEDS5540

Check configuration file HEDS5540
Download configuration file
Description: HEDM-55xx.pdf

IE216

Check configuration file IE216
Download configuration file
Description: IE216.pdf

EncMRtypeS

Check configuration file EncMRtypeS
Download configuration file
Description: EncMRtypeS.pdf

RevS

Check configuration file RevS
Download configuration file

MR225780

Description: ENC-MR-1000imp-225780_10_EN_262.pdf

8.4. Brakes

NoBrake

Check configuration file NoBrake
Download configuration file

AB41

Check configuration file AB41
Download configuration file
Description: Bremse-AB-41-228998_09_EN_318.pdf

Accessories

NoAcc

Check configuration file NoAcc
Download configuration file

8.5. Peripherals

NoPD

Check configuration file NoPD
Download configuration file

8.6. Controller

XismusbDef

Check configuration file XismusbDef
Download configuration file

8.7. XILab settings

XIDef

Check configuration file XIDef
Download configuration file