8SMC5-USB
Руководство пользователя

Выпуск 2.24.0

Standa

сент. 12, 2024

Содержание:

1 О продукте **2**
- 1.1 Общие сведения . . . 2
- 1.2 Преимущества . . . 3
  - 1.2.1 Основные преимущества . . . 3
  - 1.2.2 Все преимущества . . . 4
- 1.3 Сводная таблица характеристик . . . 5
- 1.4 Технические характеристики . . . 6
  - 1.4.1 Требования к электродвигателю . . . 6
  - 1.4.2 Электрические характеристики контроллера . . . 6
  - 1.4.3 Возможности управления движением . . . 7
  - 1.4.4 Дополнительные функции прошивки . . . 7
  - 1.4.5 Дополнительные функции, реализованные через разъем подключения мотора . . . 7
  - 1.4.6 Дополнительные функции, реализованные через разъем backplane . . . 7
  - 1.4.7 Программирование . . . 8

2 Техника безопасности **9**
- 2.1 Требования к блоку питания и заземлению. Подключение контроллера . . . 9
- 2.2 Плата контроллера . . . 11
- 2.3 Одноосная и двухосная система в корпусе . . . 11

3 Инструкция по началу работы **12**
- 3.1 Краткое руководство и начало работы . . . 12
  - 3.1.1 Введение . . . 13
  - 3.1.2 Требования . . . 13
  - 3.1.3 Установка ПО и первый пуск . . . 15
  - 3.1.4 Начало работы в ПО XILab . . . 18
  - 3.1.5 Проверка работоспособности . . . 21
  - 3.1.6 Управление из пользовательских приложений . . . 21
- 3.2 Многоосная система . . . 21
  - 3.2.1 Введение . . . 22
  - 3.2.2 Необходимые комплектующие для работы . . . 23
  - 3.2.3 Начало работы с многоосной системой в XILab. . . . 24
  - 3.2.4 Терморегулировка многоосной системы. . . . 24
  - 3.2.5 Известные ошибки и недоработки . . . 24
    - 3.2.5.1 Работа . . . 24
    - 3.2.5.2 Конструкция . . . 24
- 3.3 Пример подключения простого мотора . . . 25

3.3.1 Общий случай . . . 25
3.3.2 Пример . . . 25
3.3.2.1 Подготовка . . . 26
3.3.2.2 Подключение мотора и энкодера к контроллеру . . . 27
3.4 Ручная настройка профиля . . . 31
3.4.1 Введение . . . 32
3.4.2 Подготовка к работе . . . 32
3.4.3 Настройка рабочего тока . . . 32
3.4.4 Настройка базовых параметров . . . 33
3.4.5 Настройка аппаратных концевых выключателей, процедура автокалибровки. . . . 33
3.4.6 Настройка параметров энкодера . . . 38
3.4.7 Настройка кинематических характеристик контроллера . . . 39
3.4.8 Работа с пользовательскими единицами измерения . . . 40
3.5 Расчёт номинального тока . . . 40
3.5.1 Расчеты на базе параметров униполярного полношагового режима . . . 40
3.5.2 Расчеты на базе параметров биполярного полношагового режима . . . 40
3.5.3 Связь со среднеквадратичным током . . . 41
3.5.4 Амплитудный и номинальный ток для BLDC . . . 42
3.5.5 Настройка номинального тока . . . 42

4 Техническое описание устройства 43
4.1 Внешний вид и разъемы . . . 43
4.1.1 Плата контроллера . . . 43
4.1.1.1 Геометрические размеры . . . 43
4.1.1.2 Разъемы подключения плат . . . 44
4.1.1.2.1 Разъем подключения позиционера . . . 44
4.1.1.2.2 Разъем силового питания . . . 46
4.1.1.2.3 Разъемы управления системой . . . 47
4.1.1.3 Разъем подключения к объединительной плате . . . 48
4.1.2 Одноосная система . . . 50
4.1.2.1 Разъёмы . . . 50
4.1.2.1.1 Разъем подключения позиционера . . . 50
4.1.2.1.2 Разъем силового питания для 1- и 2-осных систем . . . 51
4.1.2.1.3 Разъемы управления системой . . . 52
4.1.2.1.4 Дополнительный разъем одноосных систем . . . 53
4.1.3 Двухосная система . . . 54
4.1.3.1 Корпус . . . 54
4.1.3.2 Разъёмы . . . 54
4.1.3.2.1 Разъем подключения позиционера . . . 54
4.1.3.2.2 Разъем силового питания . . . 56
4.1.3.2.3 Разъемы управления системой . . . 57
4.1.3.2.4 Разъём подключения джойстика для 1- и 2-осной систем . . . 58
4.1.3.2.5 Дополнительный разъём двухосной системы . . . 59
4.2 Кинематика и режимы движения . . . 60
4.2.1 Движение с заданной скоростью . . . 60
4.2.2 Движение в заданную точку . . . 61
4.2.3 Смещение на заданное расстояние . . . 61
4.2.4 Движение с ускорением . . . 62
4.2.5 Компенсация люфта . . . 63
4.2.6 Реверсирование движения . . . 64
4.2.7 Рекомендации для точного движения . . . 64
4.2.8 PID алгоритм для управления DC двигателем . . . 64
4.2.8.1 Описание алгоритма . . . 64
4.2.8.2 Особенности работы алгоритма . . . 65

4.2.8.2.1 Коэффициенты PID регулятора . . . 65
4.2.8.2.2 Попадание в целевую позицию . . . 66
4.2.8.3 Рекомендации по настройке PID регулятора . . . 66
4.2.9 Feedback EMF . . . 67
4.2.9.1 Преимущества . . . 67
4.2.9.2 Поведение двигателя при воздействии внешней силы . . . 68
4.2.9.3 Выбор параметров L, R и backEMF для алгоритма EMF . . . 68
4.2.9.4 Выбор коэффициентов PID для EMF . . . 70
4.2.9.4.1 Алгоритм работы . . . 70
4.2.10 Feedback encoder . . . 70
4.2.11 Feedback encoder mediated . . . 71
4.2.12 Режимы остановки движения . . . 71
4.2.12.1 Немедленная остановка . . . 71
4.2.12.2 Остановка с замедлением . . . 71
4.2.13 PID алгоритм для управления BLDC двигателем . . . 72
4.2.13.1 Описание алгоритма . . . 72
4.2.13.2 Ручная настройка коэффициентов PID-регулятора . . . 72
4.2.13.2.1 Шаги по настройке коэффициентов: . . . 73
4.3 Основные возможности контроллера . . . 74
4.3.1 Поддерживаемые типы двигателей . . . 74
4.3.1.1 Шаговые двигатели . . . 74
4.3.1.2 DC двигатели . . . 75
4.3.1.3 BLDC двигатели . . . 76
4.3.1.4 Критерий выбора двигателя . . . 76
4.3.2 Ограничители на двигателях . . . 78
4.3.3 Концевые выключатели . . . 80
4.3.3.1 Задача концевых выключателей . . . 80
4.3.3.2 Общие настройки . . . 80
4.3.3.3 Программное ограничение диапазона движения . . . 80
4.3.3.4 Аппаратные концевые выключатели . . . 80
4.3.3.5 Подключение концевых выключателей . . . 80
4.3.3.6 Расположение концевых выключателей на трансляторах . . . 81
4.3.4 Автокалибровка домашней позиции . . . 82
4.3.4.1 Стандартный алгоритм поиска домашней позиции . . . 82
4.3.4.2 Точная докалибровка . . . 82
4.3.4.3 Быстрый алгоритм автокалибровки . . . 83
4.3.4.4 Особенности автокалиброки . . . 83
4.3.5 Работа с энкодерами . . . 84
4.3.5.1 Область применения энкодеров . . . 84
4.3.5.2 Что такое квадратурный энкодер? . . . 84
4.3.5.3 Возможности контроллера . . . 85
4.3.5.4 Подключение энкодера . . . 85
4.3.5.4.1 Использование длинных кабелей . . . 87
4.3.5.4.2 Автоматическое определение типа энкодера . . . 87
4.3.6 Датчик оборотов . . . 87
4.3.6.1 Схема подключения . . . 87
4.3.7 Обнаружение потери шагов . . . 88
4.3.8 Управление питанием мотора . . . 89
4.3.8.1 Снижение тока потребления . . . 89
4.3.8.2 Отключение питания мотора . . . 90
4.3.8.3 Специфика расчёта временных задержек . . . 90
4.3.8.4 Функция Jerk free . . . 90
4.3.9 Критические параметры . . . 91
4.3.10 Хранение параметров во flash-памяти контроллера . . . 92

4.3.11 Пользовательские единицы координат . . . 92
4.3.12 Использование таблицы коррекции координат для более точного позиционирования . . . 93
4.4 Безопасная работа . . . 94
4.4.1 Границы движения и концевики . . . 94
4.4.2 Ограничители движения . . . 94
4.4.3 Критические параметры . . . 94
4.4.4 Работа с энкодером . . . 95
4.5 Дополнительные функции . . . 95
4.5.1 Индикация режима работы . . . 95
4.5.1.1 Статус контроллера . . . 95
4.5.1.2 Индикация концевых выключателей . . . 96
4.5.1.3 Схема подключения . . . 96
4.5.1.3.1 Плата контроллера . . . 96
4.5.1.3.2 Одноосная и двухосная системы . . . 97
4.5.2 Работа с магнитным тормозом . . . 98
4.5.2.1 Описание работы . . . 98
4.5.2.1.1 Последовательность работы контроллера при отключении подвижки. . . . 98
4.5.2.2 Схема подключения магнитного тормоза . . . 98
4.5.2.2.1 Плата контроллера . . . 98
4.5.2.2.2 Одноосная и двухосная система . . . 99
4.5.3 Управление с помощью джойстика . . . 100
4.5.3.1 Основная информация . . . 100
4.5.3.2 Схема подключения . . . 103
4.5.3.2.1 Плата контроллера . . . 103
4.5.3.2.2 Одноосная и двухосная система . . . 103
4.5.4 Управление кнопками «вправо» и «влево» . . . 104
4.5.4.1 Схема подключения . . . 105
4.5.4.1.1 Плата контроллера . . . 105
4.5.4.1.2 Одноосная или двухосная система в корпусе . . . 105
4.5.5 ТТЛ-синхронизация . . . 106
4.5.5.1 Принцип работы . . . 106
4.5.5.2 Подключение . . . 107
4.5.5.3 Вход синхронизации . . . 107
4.5.5.4 Выход синхронизации . . . 110
4.5.5.5 Схема подключения . . . 114
4.5.5.5.1 Плата контроллера . . . 114
4.5.5.5.2 Одноосная и двухосная система . . . 115
4.5.6 Создание многоосных систем . . . 116
4.5.7 Цифровой вход-выход общего назначения (EXTIO) . . . 119
4.5.7.1 Схема подключения . . . 120
4.5.7.1.1 Плата контроллера . . . 120
4.5.7.1.2 Одноосная и двухосная системы . . . 120
4.5.8 Аналоговый вход общего назначения . . . 121
4.5.8.1 Схема подключения . . . 121
4.5.8.1.1 Плата контроллера . . . 121
4.5.8.1.2 Одноосная и двухосная система . . . 122
4.5.9 Интерфейс управления внешним драйвером . . . 123
4.5.9.1 Схема подключения . . . 123
4.5.9.1.1 Плата контроллера . . . 123
4.5.9.1.2 Одноосная и двухосная система . . . 124
4.5.10 Последовательный порт . . . 125
4.5.10.1 Схема подключения последовательного порта . . . 126

4.5.10.1.1 Подключение платы контроллера . . . 126
4.5.10.1.2 Одноосная и двухосная система . . . 126
4.5.11 Хранение позиции во FRAM-памяти контроллера . . . 127
4.5.12 Опознавание позиционеров Standa . . . 127
4.5.12.1 Для разработчиков . . . 128
4.5.12.2 Схема подключения для проверки внешней памяти . . . 128
4.6 Второстепенные функции . . . 129
4.6.1 Установка нулевой позиции . . . 129
4.6.2 Установка пользовательской позиции . . . 129
4.6.3 Статус контроллера . . . 129
4.6.3.1 Статус движения . . . 129
4.6.3.2 Статус питания мотора . . . 130
4.6.3.3 Статус энкодера . . . 130
4.6.3.4 Статус обмоток двигателя . . . 130
4.6.3.5 Статус положения. . . . 131
4.6.3.6 Статус питания контроллера и температура. . . . 131
4.6.3.7 Статусные флаги . . . 131
4.6.3.8 Статус цифровых сигналов. . . . 132
4.6.4 Автовосстановление соединения . . . 133
4.6.4.1 Автовосстановление USB соединения . . . 133
4.6.4.2 Автовосстановление Ethernet соединения . . . 134
4.7 Совместимость с различным ПО . . . 134
4.7.1 MicroManager . . . 134
4.7.1.1 Подготовка к работе . . . 134
4.7.1.2 Начало работы с MicroManager . . . 135
4.7.1.2.1 Запуск MicroManager . . . 135
4.7.1.2.2 Настройка оборудования . . . 136
4.7.1.2.3 Работа с контроллерами . . . 140
4.7.2 TANGO . . . 141
4.7.2.1 Обзор . . . 141
4.7.2.2 Конфигурация и запуск сервера устройства, объявление устройства . . . 142

5 Руководство по программе XILab 148
5.1 О программе XILab . . . 148
5.2 Основные окна программы XILab . . . 148
5.2.1 Стартовое окно программы XILab . . . 148
5.2.2 Главное окно программы XILab в режиме управления одной осью . . . 152
5.2.2.1 Блок управления движением двигателя . . . 155
5.2.2.1.1 Движение без точного задания конечного положения . . . 155
5.2.2.1.2 Движение в заданную точку . . . 156
5.2.2.1.3 Текущая позиция для команд движения . . . 156
5.2.2.2 Блок управления аттенюатором . . . 157
5.2.2.3 Состояние контроллера и мотора . . . 158
5.2.2.3.1 Электропитание контроллера . . . 158
5.2.2.3.2 Состояние мотора . . . 159
5.2.2.3.3 Состояние программы . . . 159
5.2.2.4 Группа кнопок для управления программой . . . 159
5.2.2.5 Статусная строка . . . 160
5.2.3 Главное окно программы XILab в режиме управления несколькими осями . . . 161
5.2.3.1 Блок позиции и движения . . . 163
5.2.3.2 Блок виртуального джойстика . . . 163
5.2.3.3 Блок управления . . . 164
5.2.3.4 Блок индикаторов состояния контроллеров и моторов . . . 165
5.2.4 Настройки программы . . . 167

5.2.5 Графики . . . 168
5.2.5.1 Отображаемые на графиках величины . . . 170
5.2.5.2 Функции кнопок . . . 170
5.2.5.3 Ограничение значения . . . 171
5.2.6 Скрипты . . . 171
5.2.6.1 Функции кнопок . . . 172
5.2.7 Лог XiLab . . . 173
5.3 Настройки контроллера . . . 174
5.3.1 Настройка кинематики движения (Шаговый двигатель) . . . 174
5.3.1.1 Motor parameters - настройки, непосредственно связанные с электромотором . . . 175
5.3.1.2 Motion setup - настройки, связанные с кинематикой движения . . . 175
5.3.1.3 Настройки обратной связи . . . 175
5.3.2 Настройка диапазона движения и концевых выключателей . . . 176
5.3.3 Настройка предельных параметров контроллера . . . 177
5.3.4 Настройка параметров энергопотребления . . . 179
5.3.5 Настройка исходного положения . . . 181
5.3.6 Настройки синхронизации . . . 182
5.3.7 Настройка тормоза . . . 184
5.3.8 Контроль позиции . . . 186
5.3.9 Настройка внешних управляющих устройств . . . 187
5.3.10 Настройки цифрового входа-выхода общего назначения . . . 190
5.3.11 Настройка типа двигателя . . . 192
5.3.12 Настройка кинематики движения (DC мотор) . . . 194
5.3.12.1 Motor parameters - настройки электромотора . . . 195
5.3.12.2 Motion setup - настройки кинематики движения . . . 195
5.3.12.3 Настройки обратной связи . . . 195
5.3.13 Настройка контуров PID-регулирования . . . 195
5.3.14 О контроллере . . . 197
5.3.15 Настройка кинематики движения (BLDC мотор) . . . 198
5.3.15.1 Motor parameters - настройки электромотора . . . 199
5.3.15.2 Motion setup - настройки кинематики движения . . . 200
5.3.15.3 Feedback - настройки обратной связи . . . 200
5.4 Настройки программы XILab . . . 201
5.4.1 Общие настройки программы XILab . . . 201
5.4.2 Настройки интерфейса абстрактного позиционера . . . 202
5.4.3 Настройки интерфейса аттенюатора . . . 203
5.4.4 Настройка режима циклического движения . . . 204
5.4.5 Настройка логирования . . . 205
5.4.6 Общие настройки отображения графиков . . . 206
5.4.7 Индивидуальные настройки отображения графиков . . . 207
5.4.8 Настройки отображения пользовательских единиц . . . 208
5.4.8.1 Пользовательские единицы . . . 209
5.4.8.2 Таблица коррекции координат для более точного позиционирования . . . 210
5.4.9 О программе . . . 210
5.5 Характеристики позиционера . . . 211
5.5.1 Название позиционера . . . 211
5.5.2 Общие характеристики позиционера . . . 212
5.5.3 Характеристики двигателя . . . 214
5.5.4 Характеристики энкодера . . . 216
5.5.5 Характеристики датчика Холла . . . 218
5.5.6 Характеристики редуктора . . . 220
5.5.7 Характеристики вспомогательных устройств . . . 221
5.6 Корректное завершение работы . . . 223

5.7 Установка XILab . . . 223
5.7.1 Установка под Windows . . . 223
5.7.2 Установка под Linux . . . 228
5.7.3 Установка под MacOS . . . 232

6 Программирование 240
6.1 Руководство по программированию . . . 240
6.1.1 Работа с контроллером в среде Visual Studio . . . 240
6.1.2 Работа с контроллером в среде Delphi . . . 242
6.1.3 Работа с контроллером в среде LabView . . . 244
6.1.4 Работа с контроллером в среде Matlab . . . 248
6.1.5 Работа с контроллером в ScanImage . . . 249
6.1.6 Работа с контроллером в среде LabWindows . . . 249
6.1.7 Краткое описание работы с поддерживаемыми языками программирования . . . 251
6.1.7.1 Visual C++ . . . 252
6.1.7.2 .NET (C# и Visual Basic) . . . 252
6.1.7.3 Delphi . . . 252
6.1.7.4 Matlab . . . 252
6.1.7.5 Java . . . 252
6.1.7.6 Python . . . 253
6.1.7.7 Python (Jupyter Notebook) . . . 254
6.2 Описание протокола обмена . . . 254
6.2.1 Описание протокола . . . 258
6.2.2 Исполнение команд . . . 258
6.2.3 Обработка ошибок на стороне контроллера . . . 259
6.2.3.1 Неверные команды или данные . . . 259
6.2.3.2 Расчёт CRC . . . 259
6.2.3.3 Сбои передачи . . . 260
6.2.3.4 Восстановление синхронизации методом таймаута . . . 261
6.2.3.5 Восстановление синхронизации методом очистительных нулей . . . 261
6.2.4 Обработка ошибок на стороне библиотеки . . . 262
6.2.4.1 Возможные значения ответа библиотеки . . . 262
6.2.4.2 Процедура синхронизации очистительными нулями . . . 263
6.2.5 Коды ошибок ответов контроллера . . . 263
6.2.5.1 ERRC . . . 263
6.2.5.2 ERRD . . . 263
6.2.5.3 ERRV . . . 263
6.2.6 Все команды контроллера . . . 264
6.2.6.1 Команда GACC . . . 264
6.2.6.2 Команда GBRK . . . 265
6.2.6.3 Команда GCAL . . . 266
6.2.6.4 Команда GCTL . . . 267
6.2.6.5 Команда GCTP . . . 268
6.2.6.6 Команда GEAS . . . 269
6.2.6.7 Команда GEDS . . . 270
6.2.6.8 Команда GEIO . . . 271
6.2.6.9 Команда GEMF . . . 272
6.2.6.10 Команда GENG . . . 273
6.2.6.11 Команда GENI . . . 275
6.2.6.12 Команда GENS . . . 276
6.2.6.13 Команда GENT . . . 277
6.2.6.14 Команда GEST . . . 277
6.2.6.15 Команда GFBS . . . 278
6.2.6.16 Команда GGRI . . . 279

6.2.6.17 Команда GGRS . . . 279
6.2.6.18 Команда GHOM . . . 280
6.2.6.19 Команда GHSI . . . 282
6.2.6.20 Команда GHSS . . . 282
6.2.6.21 Команда GJOY . . . 283
6.2.6.22 Команда GMOV . . . 284
6.2.6.23 Команда GMTI . . . 285
6.2.6.24 Команда GMTS . . . 286
6.2.6.25 Команда GNET . . . 288
6.2.6.26 Команда GNME . . . 288
6.2.6.27 Команда GNMF . . . 289
6.2.6.28 Команда GNVM . . . 289
6.2.6.29 Команда GPID . . . 290
6.2.6.30 Команда GPWD . . . 290
6.2.6.31 Команда GPWR . . . 291
6.2.6.32 Команда GSEC . . . 292
6.2.6.33 Команда GSNI . . . 293
6.2.6.34 Команда GSNO . . . 294
6.2.6.35 Команда GSTI . . . 296
6.2.6.36 Команда GSTS . . . 296
6.2.6.37 Команда GURT . . . 297
6.2.6.38 Команда SACC . . . 297
6.2.6.39 Команда SBRK . . . 299
6.2.6.40 Команда SCAL . . . 300
6.2.6.41 Команда SCTL . . . 300
6.2.6.42 Команда SCTP . . . 302
6.2.6.43 Команда SEAS . . . 303
6.2.6.44 Команда SEDS . . . 303
6.2.6.45 Команда SEIO . . . 305
6.2.6.46 Команда SEMF . . . 306
6.2.6.47 Команда SENG . . . 306
6.2.6.48 Команда SENI . . . 309
6.2.6.49 Команда SENS . . . 309
6.2.6.50 Команда SENT . . . 310
6.2.6.51 Команда SEST . . . 311
6.2.6.52 Команда SFBS . . . 311
6.2.6.53 Команда SGRI . . . 312
6.2.6.54 Команда SGRS . . . 313
6.2.6.55 Команда SHOM . . . 313
6.2.6.56 Команда SHSI . . . 315
6.2.6.57 Команда SHSS . . . 316
6.2.6.58 Команда SJOY . . . 316
6.2.6.59 Команда SMOV . . . 317
6.2.6.60 Команда SMTI . . . 318
6.2.6.61 Команда SMTS . . . 319
6.2.6.62 Команда SNET . . . 321
6.2.6.63 Команда SNME . . . 321
6.2.6.64 Команда SNMF . . . 322
6.2.6.65 Команда SNVM . . . 322
6.2.6.66 Команда SPID . . . 323
6.2.6.67 Команда SPWD . . . 323
6.2.6.68 Команда SPWR . . . 324
6.2.6.69 Команда SSEC . . . 325
6.2.6.70 Команда SSNI . . . 326

6.2.6.71 Команда SSNO . . . 327
6.2.6.72 Команда SSTI . . . 328
6.2.6.73 Команда SSTS . . . 329
6.2.6.74 Команда SURT . . . 329
6.2.6.75 Команда ASIA . . . 330
6.2.6.76 Команда CLFR . . . 331
6.2.6.77 Команда CONN . . . 331
6.2.6.78 Команда DBGR . . . 331
6.2.6.79 Команда DBGW . . . 332
6.2.6.80 Команда DISC . . . 332
6.2.6.81 Команда EERD . . . 333
6.2.6.82 Команда EESV . . . 333
6.2.6.83 Команда GBLV . . . 333
6.2.6.84 Команда GETC . . . 334
6.2.6.85 Команда GETI . . . 334
6.2.6.86 Команда GETM . . . 335
6.2.6.87 Команда GETS . . . 335
6.2.6.88 Команда GFWV . . . 341
6.2.6.89 Команда GOFW . . . 341
6.2.6.90 Команда GPOS . . . 342
6.2.6.91 Команда GSER . . . 342
6.2.6.92 Команда GUID . . . 343
6.2.6.93 Команда HASF . . . 343
6.2.6.94 Команда HOME . . . 343
6.2.6.95 Команда IRND . . . 344
6.2.6.96 Команда LEFT . . . 344
6.2.6.97 Команда LOFT . . . 344
6.2.6.98 Команда MOVE . . . 345
6.2.6.99 Команда MOVR . . . 345
6.2.6.100 Команда PWOF . . . 346
6.2.6.101 Команда RDAN . . . 346
6.2.6.102 Команда READ . . . 348
6.2.6.103 Команда RERS . . . 348
6.2.6.104 Команда REST . . . 349
6.2.6.105 Команда RIGT . . . 349
6.2.6.106 Команда SARS . . . 349
6.2.6.107 Команда SAVE . . . 350
6.2.6.108 Команда SPOS . . . 350
6.2.6.109 Команда SSER . . . 350
6.2.6.110 Команда SSTP . . . 351
6.2.6.111 Команда STMS . . . 351
6.2.6.112 Команда STOP . . . 352
6.2.6.113 Команда UPDF . . . 352
6.2.6.114 Команда WDAT . . . 352
6.2.6.115 Команда WKEY . . . 353
6.2.6.116 Команда ZERO . . . 353
6.3 Совместимость с ПО для 8SMC1-USBhF . . . 353
6.4 Таймауты libximc . . . 356
6.5 Скрипты XILab . . . 356
6.5.1 Краткое описание языка . . . 358
6.5.1.1 Типы данных . . . 358
6.5.1.2 Инструкции . . . 358
6.5.1.3 Объявление переменных . . . 359
6.5.1.4 Ключевые и зарезервированные слова . . . 359

6.5.1.5 Функции . . . 359
6.5.2 Подсветка синтаксиса . . . 360
6.5.3 Дополнительные функции, предоставляемые XILab . . . 360
6.5.3.1 Запись в лог XILab . . . 360
6.5.3.2 Задержка выполнения скрипта . . . 361
6.5.3.3 Создание объекта типа «ось» . . . 361
6.5.3.4 Создание объекта типа «файл» . . . 361
6.5.3.5 Создание структуры калибровки . . . 362
6.5.3.6 Получение следующего серийного номера . . . 363
6.5.3.7 Ожидание остановки движения . . . 363
6.5.3.8 функции библиотеки libximc . . . 363
6.5.4 Примеры . . . 364
6.5.4.1 Скрипт-пример работы с битовыми масками . . . 364
6.5.4.2 Скрипт сканирования и записи в файл . . . 365
6.5.4.3 Многоосный скрипт циклического движения . . . 366
6.5.4.4 Одноосный скрипт циклического движения . . . 367
6.5.4.5 Скрипт проверки калибровки домашней позиции . . . 368
6.5.4.6 Скрипт для поиска серийных номеров контроллеров . . . 369
6.5.4.7 Скрипт перемещения и ожидания . . . 369
6.5.4.8 Скрипт случайного сдвига . . . 370
6.5.4.9 Скрипт установки нулевой позиции . . . 371
6.5.4.10 Скрипт для автотестирования . . . 373
6.5.4.11 Тест на пересечение границ . . . 373
6.5.4.12 Тест настройки с замкнутым контуром . . . 374
6.5.4.13 Скрипт дискретного движения . . . 378
6.5.4.14 Экспоненциальное изменение позиции использующие user units . . . 379
6.5.4.15 Шаговый скрипт использующий user units . . . 381
6.5.4.16 Шаговый скрипт . . . 382
6.5.4.17 Тест калибровки домашней позиции сигналу со входа EXTIO . . . 383
6.5.4.18 Скрипт движения по sin . . . 384
6.5.4.19 Скрипт перемещения по сигналу со входа EXTIO. Движение осуществляется в user units . . . 386
6.5.4.20 Вероятные тесты . . . 387
6.5.4.21 Скрипт выполняющий ряд смещений с калибровкой . . . 388
6.5.4.22 Тест на пропуск шагов . . . 389
6.5.4.23 Скрипт тестирования синхронизации . . . 391
6.5.4.24 Скрипт тестирования ошибок синхронизации . . . 394
6.6 Неподдерживаемые примеры . . . 395
6.6.1 Пример шестиосного интерфейса XILab . . . 395
6.6.2 Примеры для работы с аттенюатором для Python и LabView . . . 395
6.6.3 Программа для отправки команд на контроллер XIMC с контроллера AVR . . . 396
6.6.4 Многоосевой интерфейс на Python . . . 396

7 Управление контроллером по Ethernet 397
7.1 8Eth1 адаптер . . . 397
7.1.1 Обзор Ethernet адаптера . . . 397
7.1.1.1 Общая информация и внешний вид . . . 397
7.1.2 Администрирование . . . 401
7.1.2.1 Конфигурация сети . . . 401
7.1.2.1.1 Подключение контроллера по сети, в которой нет DHCP-сервера 401
7.1.2.1.2 Подключение контроллера по сети, в которой есть DHCP-сервер 402
7.1.2.2 Автоматическое обнаружение устройств . . . 405
7.1.2.3 Обзор . . . 406
7.1.2.3.1 Секция «Common» . . . 408

7.1.2.3.2 Секция «Motion Control» . . . 408
7.1.2.3.3 Секция «Cameras» . . . 409
7.1.2.4 Управление сервисами . . . 409
7.1.2.5 Начало работы с помощью XILab . . . 410
7.1.2.6 Обновление прошивки . . . 411
7.2 Serial to Ethernet конвертер . . . 412
7.2.1 Общая информация . . . 412
7.2.2 Подключение конвертера к контроллеру 8SMC5 . . . 413
7.2.3 Настройка MOXA NPort 5110 конвертера через web интерфейс . . . 414
7.2.4 Начало работы с помощью XILab . . . 416
7.3 8SMC5 Ethernet . . . 417
7.3.1 Общая информация . . . 417
7.3.2 Примеры конфигурации для контроллера 8SMC5 с поддержкой Ethernet . . . 418
7.3.3 Автоматическое обнаружение устройства . . . 419
7.3.4 Панель администрирования . . . 420
7.3.5 Начало работы с XILab . . . 421

8 FAQ **423**
8.1 No device found / Can't open device . . . 423
8.1.1 Подключение через USB . . . 423
8.1.2 Подключение через Ethernet . . . 428
8.1.3 Подключение через 8Eth1 адаптер . . . 429
8.1.4 Подключение через Serial to Ethernet конвертер . . . 430
8.2 Не удаётся вращать мотором при помощи контроллера . . . 430
8.2.1 Контроллер в состоянии Alarm . . . 430
8.2.2 Мотор вибрирует, вращения нет . . . 431
8.2.3 Механическое заклинивание . . . 435
8.2.4 Двигатель не реагирует на команды движения . . . 435
8.3 Потеря USB-соединения . . . 435
8.4 Самодиагностика адаптера 8Eth1 . . . 436
8.4.1 Инструкция по установке статического ip образа . . . 437
8.4.2 Самодиагностика статического ip-адреса на 8Eth1 . . . 437
8.5 Как реализовать кнопку экстренной остановки? . . . 439
8.6 Список паролей панели администратора 8SMC5 по умолчанию . . . 439
8.7 Как вернуть окно xilab, которое скрылось за пределами экрана? . . . 440
8.8 Где я могу найти руководство по программированию для контроллера 8SMC5? . . . 440
8.9 CRC алгоритм на Python . . . 440
8.10 Виртуальный контроллер, как в XILab . . . 441
8.11 *probe_flag* - что это? . . . 443
8.12 Как проверить, установлено ли соединение с 8SMC5-USB и активно ли оно еще во время моего сеанса с помощью библиотеки libximc? . . . 443
8.13 Проблема компенсации люфта (пример из техподдержки) . . . 443
8.14 Управление Raspberry Pi . . . 444
8.14.1 Работа с программным обеспечением XILab на процессоре ARM . . . 444
8.14.2 Работа с библиотекой libximc на процессоре ARM . . . 444
8.15 Зависание операционной системы при использовании библиотеки libximc и ядра Linux с версией менее 3.16 . . . 445

8SMC4/5 technical support

- Задать вопрос
- Написать письмо: **8smc4@standa.lt**
- Для быстрых простых вопросов свяжитесь с нами:
- **Telegram**: @SMC5TechSupport
- **WhatsApp**: +1 (530) 584-4117
- *Онлайн Пн-Пт 8:00-16:00 UTC*

For controllers purchase

- По вопросам покупки контроллеров, пожалуйста, обращайтесь по e-mail: **sales@standa.lt**

Глава 1

О продукте

1.1 Общие сведения

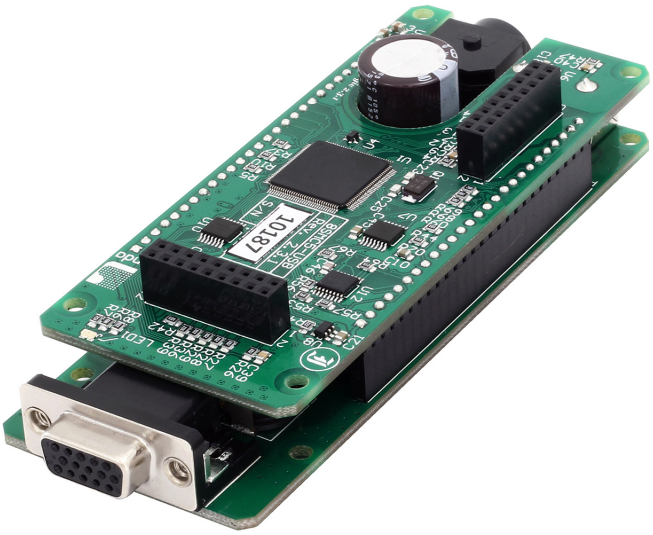
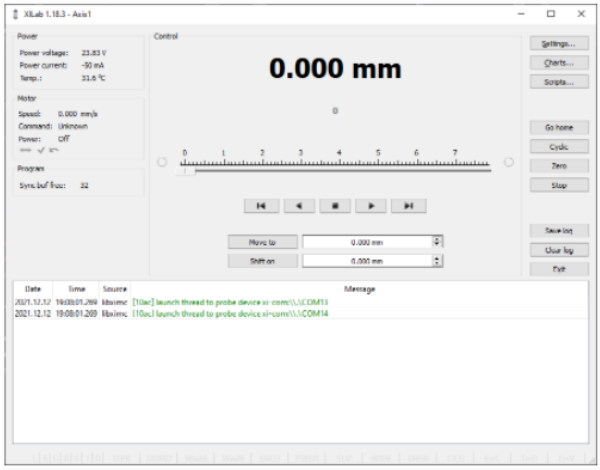


Рис. 1.1: Плата контроллера без корпуса

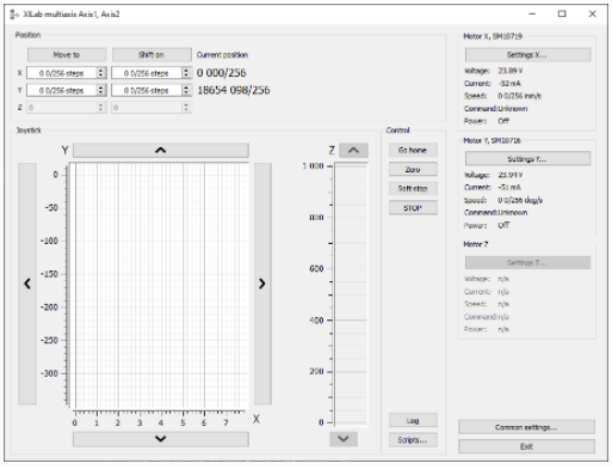
Мы предлагаем недорогой ультра-компактный сервопривод с интерфейсом USB/Ethernet для шаговых, DC и BLDC двигателей с внешним питанием.

Забудьте о громоздких и дорогих сервоприводах! Все, что вам нужно, это шаговый, DC или BLDC двигатель, контроллер, USB/Ethernet кабель и любой стабилизированный внешний источник питания. И все! Не нужно никакого активного охлаждения. Благодаря современной конструкции контроллера можно использовать даже простые и недорогие подвижки для достижения высокой скорости и точности. Неважно, какой двигатель вы предпочитаете, потому что один контроллер может управлять ими всеми. Контроллер отлично подходит для управления шаговыми двигателями с номинальным

током обмотки до 3А и DC двигателями с номинальным током до 6А. Контроллер работает как со шаговыми двигателями без обратной связи, так и с шаговыми двигателями, оснащенными энкодером в цепи обратной связи, в том числе линейным энкодером на позиционере. Разъем для мотора на контроллере соответствует разъему, который использует компания Standa, и подходит для всех позиционеров Standa. USB/Ethernet соединение обеспечивает легкость подключения и простоту работы с компьютером. Несколько контроллеров могут быть подключены к одному компьютеру через несколько USB/Ethernet портов или с помощью специальной объединительной платы, поставляемой в составе многоосных систем. Контроллер совместим практически со всеми операционными системами, Windows (XP SP3, Vista, 7, 8, 10, 11), Linux, MacOS Sierra и новее для intel и Apple Silicon (с использованием Rosetta 2).



XILab main window (Single Axis mode)



XILab main window (Multi Axis mode)

С контроллером предоставляется все необходимое программное обеспечение включая конфигурационные файлы (профили в формате .cfg), чтобы быстро начать работу по принципу «подключил и работает». Поэтому все, что вам нужно, это открыть контроллер в программном обеспечении XILab, загрузить .cfg файл для своей подвижки и нажать «Apply». Теперь ваш контроллер полностью настроен! Введите команды перемещения и контроллер их выполнит.

1.2 Преимущества

1.2.1 Основные преимущества

- *Самый компактный!* Габариты127 x 46 x 32 мм со всеми разъемами, подходит для всех шаговых двигателей с номинальным током обмоток до 3 А.
- *Совместимость с 8SMC1-USBh* Все программы, написанные для 8SMC1-USBh, смогут работать с 8SMC5-USB после обновления MicroSMC драйвера.
- Поддержка всех позиционеров Standa! У вас позиционер Standa? Просто подключите, все остальное мы уже сделали за вас!
- *Помнит все!* Не нужно сохранять текущую позицию на компьютере, контроллер сделает это сам, используя собственную энергонезависимую память даже при неожиданной потере питания.
- Умеет работать с периферией! Поддержка *энкодера*, *магнитного тормоза*, *джойстика*, *концевиков*, датчика нулевой позиции - все встроено, просто подключите и пользуйтесь!
- *Встроенная калибровка нуля!* Одной *командой*, по *концевикам*, *датчику оборотов*, *внешнему сигналу* или по их комбинациям!

1.2.2 Все преимущества

- Действительно мощный! Контроллер отлично подходит для управления шаговыми двигателями с номинальным током обмотки до 3А и двигателями постоянного тока с номинальным током до 6А.
- *Совместимость с 8SMC1-USBh* Все программы, написанные для 8SMC1-USBh, смогут работать с 8SMC5-USB после обновления MicroSMC драйвера.
- Поддержка всех позиционеров Standa! У вас позиционер Standa? Просто подключите, все остальное мы уже сделали за вас!
- *Знает своих!* Доступна встроенная функция загрузки всех конфигурационных данных непосредственно из подвижки при поддержке такой памяти в позиционере.
- Выбери свой интерфейс! USB или *последовательный порт* встроены и готовы к использованию.
- По настоящему быстрый! До 35 000 полных шагов/сек при любом делении шага!
- Точный! Режимы деления шага от полношагового до 1/256 шага на всех скоростях.
- *Помнит все!* Не нужно сохранять текущую позицию на компьютере, контроллер сделает это сам, используя собственную энергонезависимую память даже при неожиданной потере питания.
- Умеет работать с периферией! Поддержка *квадратурного энкодера*, *магнитного тормоза*, *джойстика*, *концевиков*, датчика нулевой позиции - все встроено, просто подключите и пользуйтесь. Отдельный стабилизированный выход 5 В 500 мА для периферии.
- *Встроенная калибровка нуля!* Одной *командой*, по *концевикам*, *датчику оборотов*, *внешнему сигналу* или по их комбинациям!
- Автономный! Хотите работать автономно - пожалуйста. Поддерживается внешний *джойстик*, *кнопочное управление* или их комбинация.
- *Экономный!* Снижение тока через обмотку двигателя в режиме удержания. Программно, с шагом 1%.
- Бесшумный! Плавное движение на малых скоростях, отсутствие дополнительных шумов на больших.
- Защищенный! ESD защита по всем контактам внешних разъемов, защита от КЗ в цепях мотора.
- *Внимательный!* Контролирует температуры процессора, силовой части; токи и напряжения как силового питания, так и USB.
- Современный! *Обновление* микропрограммы в энергонезависимой памяти контроллера по USB.
- Управляющий и управляемый! Встроен *вход и выход синхронизации*, позволяющий начать движение в заданную позицию по внешнему сигналу и/или выдать сигнал при достижении заданной позиции. Встроен *аналоговый вход* общего назначения, *цифровой вх/вых общего назначения*.
- Понятный! *Статусный светодиод* отражает состояние контроллера и наличие электропитания. Для удобства оба сигнала и состояние концевиков также выводятся на внешние светодиоды.
- *Работает на любом компьютере!* Все программное обеспечение, поставляемое с контроллером, совместимо с *Windows (XP SP3, Vista, 7, 8, 10, 11)* , *Linux*, *MacOS Sierra и новее для intel и Apple Silicon (с использованием Rosetta 2)*.
- Примеры для всех языков! Контроллер поставляется с кросс-платформенной библиотекой и примерами, которые позволяют быстро начать программирование с использованием C++, C#, .NET, Delphi, Visual Basic, Xcode, Python, Matlab, Java, LabWindows и LabVIEW.

- *Полнофункциональный интерфейс!* Контроллер поставляется с интерфейсом пользователя XILab, позволяющим легко управлять всеми функциями устройства без необходимости в разработке собственных программ.
- *Собственный скриптовый язык!* В XILab интегрирован скриптовый язык, позволяющий легко, без компиляции и освоения какого-либо языка программирования задавать последовательности действий, включая циклы и условные переходы.
- *Реализован новый алгоритм управления шаговым двигателем с ведущим энкодером!* На всех контроллерах 8SMC5-USB управление движением с использованием обратной связи по энкодеру делает перемещения мотора более быстрыми и более плавными. Никаких проскальзываний, заминок и срывов движения!

1.3 Сводная таблица характеристик

Тип питания контроллера	внешнее 12 - 48 В
Потребляемый контроллером ток	до 5 А от внешнего источника питания, зависит от напряжения источника питания
Ток в обмотке шагового мотора, BLDC мотора	до 3 А
Ток в обмотке DC мотора	до 6 А
Защиты	от перегрузки по току, напряжению, короткого замыкания, горячего отключения двигателя
Режимы работы	движение в направлении, движение в заданную точку, смещение на заданную дельту, поддержание заданной скорости, трапецевидный профиль скорости, режим компенсации люфта, автоматическая калибровка домашней позиции
Режимы деления шага	полношаговый, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256
Скорость движения	до 35 000 полных шагов/сек
Профиль скорости	трапециевидный
Счетчик позиции	40 бит
Обратная связь	Квадратурный несимметричный и дифференциальный энкодер (опционально)
Концевые выключатели	2 шт
Ширина полосы обратной связи	200 кГц для несимметричного и 5МГц для дифференциального
Частота дискретизации выходного сигнала синхронизации	1 мс

Примечание: Рабочий диапазон напряжений питания 12 - 48 В. На нём контроллер работает согласно данной инструкции. Предельный диапазон напряжений питания 12 - 50 В. При превышении напряжения 50 В контроллер гарантированно выходит из строя. При напряжении менее 12 В контроллер выключается.

Название контроллера	Вес контроллеров	Габариты контроллеров	Диапазон температур хранения контроллера	Рабочий диапазон контроллера

Continued on next page

Таблица 1.2 – continued from previous page

8SMC5 (контроллер без корпуса)	93 г	*46x126,75x31,81 мм*	от -40 °C до 80 °C	от 5 °C до 60 °C
8SMC5-USB-B8-1	500 г	*66x46x114 мм*	от -20 °C до 70 °C	
8SMC5-ETH-B8-1	510 г			
8SMC5-USB-B9-2	621 г	*122x45x106 мм*		
8SMC5-ETH-B9-2	631 г			

Примечание: Контроллер не тестировался в вакууме. Скорее всего, контроллер будет работать в вакууме, но важно, как тепло будет отводиться от корпуса.

1.4 Технические характеристики

1.4.1 Требования к электродвигателю

- Тип электродвигателя: биполярный шаговый, DC, BLDC.
- Номинальный ток в обмотке: не менее 100 мА.
- Номинальное напряжение на обмотке: не менее 2 В.

1.4.2 Электрические характеристики контроллера

- Режимы электропитания: от внешнего источника питания.
- Ток в каждой обмотке шагового мотора, BLDC мотора: до 3 А.
- Ток в каждой обмотке DC мотора: до 6 А.
- Максимальная частота следования импульсов с энкодера: 200 кГц для несимметричного и 5МГц для дифференциального.
- Стабилизированный выход 5 В (для питания энкодера и прочей внешней электроники): выходной ток не более 100 мА, стабильность выходного напряжения не хуже 5%.
- ESD защита по всем контактам внешних разъемов (D-Sub 15 pin, mini-USB, питание).
- Защита от замыкания обмоток мотора на «землю».
- Защита от замыкания обмоток мотора между собой.
- Защита от подключения/отключения мотора к контроллеру в процессе работы.
- Защита от подачи питания неверной полярности (не более 1 сек).
- Защита от подачи напряжения питания больше допустимого (не более 1 сек).
- Ограничение тока, потребляемого от внешнего источника питания.
- Ограничение скорости вращения мотора.
- Установка программируемого рабочего тока протекающего через обмотки шагового двигателя с точностью до 10 мА.
- Программируемое уменьшение рабочего тока через обмотки шагового двигателя: с математической точностью *1%* для режима удержания.

1.4.3 Возможности управления движением

- Режимы деления шага: полношаговый, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256.
- Бесшумное движение на малых скоростях.
- Минимальная скорость: 1/256 полного шага/сек.
- Максимальная скорость: до 35 000 полных шагов/сек при всех режимах деления шага.
- Минимальное смещение: 1/256 шага.
- Максимальное смещение: 2 147 483 647 полных шагов при всех режимах деления шага.
- Режим плавного начала/остановки движения.
- Счетчик позиции: 40 бит (32 бит - полный шаг, 8 бит - микрошаг).
- Режимы движения: движение в направлении, движение в заданную точку, смещение на заданную дельту, поддержание заданной скорости, трапециевидный профиль скорости, режим компенсации люфта.

1.4.4 Дополнительные функции прошивки

- Режим автокалибровки «HOME» на уровне прошивки.
- Сохранение/загрузка настроек в энергонезависимую память контроллера.
- Обновление прошивки в энергонезависимой памяти контроллера по USB.
- Автоматическое сохранение позиции по счетчику шагов и по энкодеру с защитой от неожиданного отключения питания.

1.4.5 Дополнительные функции, реализованные через разъем подключения мотора

- Отработка сигналов с одного или двух концевиков. Конфигурируется программно.
- «Опознавание» позиционеров Standa и автоматическая загрузка всех конфигурационных данных для них (при поддержке соответствующей возможности в позиционере).
- Определение «потери шагов» и восстановление правильной позиции с помощью датчика оборотов или квадратурного энкодера (при поддержке соответствующей возможности в позиционере).
- Определение положения с помощью квадратурного энкодера. Режим x4.
- Движение шагового двигателя в режиме «ведущего энкодера», т.е. с опорой на показания квадратурного энкодера, обеспечивающее движение без потери шагов на максимальной доступной скорости реализовано начиная с прошивки 4.1.

1.4.6 Дополнительные функции, реализованные через разъем backplane

- Дубликат входа USB. Используется совместно с объединительной платой.
- Последовательный порт RS-232. Доступны линии TX, RX. Параметры: скорость – 9600 - 921600 бод, ТТЛ 3.3 В. На базе последовательного порта по запросу доступны конфигурации с интерфейсом Ethernet, Bluetooth, WiFi, ZigBee и другие.
- Вход синхронизации – при подаче импульса на этот вход (срабатывание, полярность и длительность настраиваются пользователем) контроллер начинает движение в заранее заданную позицию или на заданное смещение. Параметры: ТТЛ 3.3 В.

- Выход синхронизации – выдача импульса с этого выхода контроллера (срабатывание, полярность и длительность настраиваются пользователем) производится либо по началу движения, либо завершению движения, либо через заданное перемещение (задается пользователем). Параметры: ТТЛ 3.3 В.
- Кнопки «вправо», «влево». Нажатие на кнопку приводит к движению в заданную сторону с постепенным настраиваемым ростом скорости и в соответствии с настройками ускорения и другими параметрами. Параметры: ТТЛ 3.3 В.
- Вход «джойстик». Позволяет работать с джойстиками с диапазоном выдаваемых напряжений не шире 0 - 3.3 В.
- Выход для управления магнитным тормозом. Позволяет управлять магнитным тормозом, установленным на ось шагового двигателя. Параметры: ТТЛ 3.3 В, 5 мА.
- Аналоговый вход общего назначения. Позволяет работать с сигналами 0-3.3 В. Частота считывания 1 кГц. Конфигурируется программно.
- Цифровой вход/выход общего назначения. Частота обновления 1 кГц. Конфигурируется программно. Параметры: ТТЛ 3.3 В, 5 мА.
- Индикация концевых выключателей. TTL 3.3 В, 2 мА. Предназначены для непосредственного подключения светодиодов.
- Цифровые контакты «Power» и «Status» дублируют LED индикатор и предназначены для прямого подключения светодиодов. Технические характеристики: TTL 3,3 В, 2 мА.
- Интерфейс управления внешним драйвером. Позволяет управлять любым внешним драйвером с помощью трех сигналов: enable, direction, clock.
- Создание многоосных систем. Многоосные системы создаются с помощью стандартных USB/Ethernet хабов, внешних или установленных на специальную объединительную плату.

1.4.7 Программирование

- Все программное обеспечение, поставляемое с контроллером, совместимо с Windows (XP SP3, Vista, 7, 8, 10, 11), Linux, MacOS Sierra и новее для intel и Apple Silicon (с использованием Rosetta 2).
- Контроллер поставляется с кросс-платформенной библиотекой и примерами, которые позволяют быстро начать программирование с использованием C++, C#, .NET, Delphi, Visual Basic, Xcode, Python, Matlab, Java, LabWindows и LabVIEW.
- Контроллер поставляется с интерфейсом пользователя XILab, позволяющим легко управлять всеми функциями устройства без необходимости разработки собственных программ.
- В XILab интегрирован скриптовый язык, диалект EcmaScript, позволяющий легко, без компиляции и освоения какого-либо языка программирования задавать последовательности действий, включая циклы и условные переходы.

Внимание: Руководство по программированию можно найти на нашем втором сайте libximc.xisupport.com или Вы можете скачать его PDF версию

Глава 2

Техника безопасности

2.1 Требования к блоку питания и заземлению. Подключение контроллера

Ниже перечислены основные требования к блоку питания для работы как с *платой контроллера*, так и с контроллерами в корпусе (*одноосная* и *двухосная системы*).

Во время работы, потребление тока будет зависеть от того, как используется контроллер. Наши контроллеры откалиброваны по номинальному току двигателей, с которыми они будут использоваться. Благодаря широтно-импульсной модуляции (ШИМ) наши контроллеры потребляют меньше тока, чем номинальный ток двигателей. Однако во избежание проблем в наихудших сценариях мы рекомендуем выбирать источник питания с максимальным током не менее номинального тока двигателей, которые будут подключены к контроллеру. В случае многоосевых контроллеров вам нужно будет суммировать ток всех контроллеров, подключенных к источнику питания.

Наши контроллеры требуют напряжения 12 - 48В. Потребляемый контроллером ток до 5 А от внешнего источника питания, зависит от напряжения источника питания. Рекомендуемые блоки питания к контроллеру: PS12-1.5-4 (12V, 1.5A), PS 12-3-4 (12V, 3A), PS24-2.5-4 (24V, 2.5A), PS36-4.4-4 (36V, 4.15A).

Для того, чтобы определить, какой блок питания из перечисленных выше подойдет для вашей задачи, необходимо изучить рекомендации к питанию используемого мотора или позиционера. Информацию можно найти на сайте standa.lt.

Важно: Блок питания должен быть заземлен через розетку 220В (трехпроводная схема подключения). Убедитесь, что используемый вами блок питания имеет заземленный минусовой выход. Если вы используете блок питания с отвязанным от земли минусовым проводом, заземляйте **контроллер** через **клемму заземления**, которая находится на плате контроллера. Если контроллер поставляется внутри металлического корпуса, корпус должен быть заземлен. Несоблюдение этих правил может привести к снижению стабильности работы контроллера и помехоустойчивости.

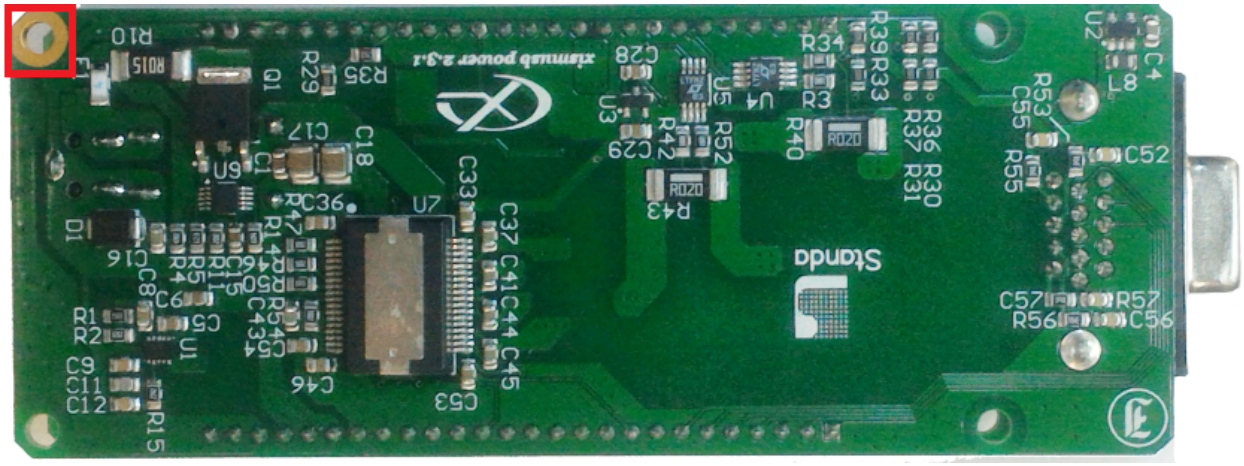


Рис. 2.1: Внешний вид платы контроллера сверху. Красным квадратом отмечена клемма заземления

Типовые схемы подключения контроллера (как коробочных версий, так и платы без корпуса):

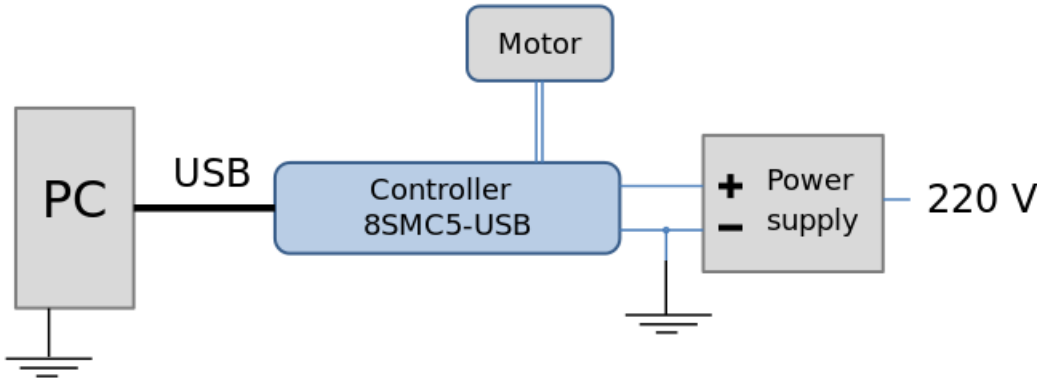


Рис. 2.2: Схема подключения контроллера с заземлением через минусовый провод блока питания

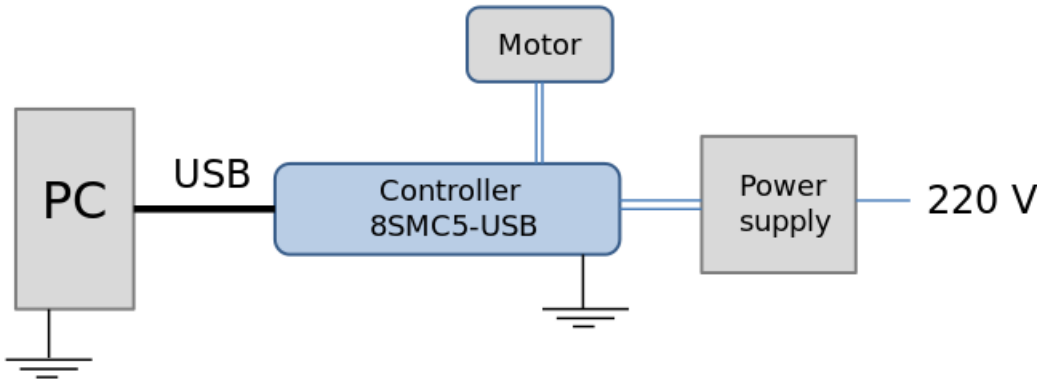


Рис. 2.3: Схема подключения контроллера с заземлением через клемму заземления

Предупреждение: Блок питания должен обеспечивать необходимый ток для вращения двигателя. Абсолютным минимумом является ток, рассчитываемый по формуле

$$I_{power.min} = \frac{2 * I_{motor} * U_{motor}}{U_{power}}$$

где $I_{power.min}$ это минимальный рабочий ток блока питания, I_{motor} это рабочий ток через обмотку двигателя, U_{power} это стабилизируемое напряжение блока питания, а U_{motor} это номинальное рабочее напряжение мотора. Рекомендуется использовать блок питания с рабочим током $I_{power} \geq 2 * I_{power.min}$. Напряжение U_{power} должно быть выше U_{motor}. Чем выше напряжение питания, тем большей скорости вращения можно достичь.

Можно использовать параметр мощности блока питания вместо рабочего тока. Абсолютным минимумом мощности блока питания будет:

$$W_{power.min} = I_{power.min} * U_{power} = 2 * I_{motor} * U_{motor}$$

Например, для мотора с рабочим током в обмотке 1 А и рабочим напряжением 5 В (номинальная мощность 5 Вт), рабочее напряжение блока питания можно взять 20 В с выдаваемой мощностью не менее 10 Вт (максимальный рабочий ток блока питания не менее 0.5 А).

2.2 Плата контроллера

Важно: При работе с *платой контроллера* запрещено трогать плату руками, т.к. статический разряд может повредить компоненты на ней. Рекомендуется пользоваться антистатическими браслетами. **Запрещено превышать максимально допустимое напряжение 48В**, превышение этого значения более чем на 2В может немедленно привести к выходу системы из строя.

Важно: При работе с *платой контроллера* категорически запрещено подключать плюс силового источника питания к контроллеру, когда провод заземления не подключен. Категорически запрещается подключать или отключать кабель питания, когда источник питания включен, контроллер подключен к ПК, а источник питания и ПК заземлены. **Это может привести к выходу из строя компьютера!** Данное требование является общим для любых электронных устройств с собственным электропитанием.

Предупреждение: Рекомендуется сначала подключить *плату контроллера* к силовому электропитанию с заземлением, либо отдельно заземлить плату с помощью специально маркированной клеммы (см. *пункт выше*), прежде чем подключать контроллер к мотору или к компьютеру по интерфейсу USB.

2.3 Одноосная и двухосная система в корпусе

Важно: При работе с контроллером в корпусе (одноосная или двухосная системы) **запрещено превышать максимальное напряжение питания, равное 48В**, превышение этого значения более чем на 2В может немедленно привести к выходу системы из строя.

Глава 3

Инструкция по началу работы

Данное руководство описывает работу с контроллером многоосных и одноосных систем, настройку основных параметров и начало работы с программным обеспечением XILab для Windows.

Внимание: Для быстрого начала работы с контроллером рекомендуем прочитать *Краткое руководство и начало работы*. Руководство по программированию можно найти на нашем втором сайте libximc.xisupport.com или Вы можете скачать его PDF версию

- *Краткое руководство и начало работы* - краткое описание начала работы с контроллером 8SMC4-USB для одной оси. Рассмотрена быстрая настройка XiLab, перечислено всё необходимое оборудование.
- *Многоосная система* - краткое описание начала работы с многоосной системой на базе контроллера 8SMC4-USB. Рассмотрена настройка XiLab, перечислено всё необходимое оборудование и особенности работы системы.
- *Пример подключения простого мотора* - подключение к 8SMC4-USB шагового мотора на примере Nanotec ST5918L3008-B с энкодером CUI INC AMT112S-V. Описано, как изготовить собственный кабель, руководствуясь спецификацией на двигатель, даны пояснения характеристикам из спецификации.
- *Ручная настройка профиля* - настройка рабочего профиля для XiLab. Обзор основных функциональных возможностей.
- *Расчёт номинального тока* - установка амплитуды номинального тока для шаговых двигателей.

3.1 Краткое руководство и начало работы

- *Введение*
- *Требования*

- *Установка ПО и первый пуск*
- *Начало работы в ПО XILab*
- *Проверка работоспособности*
- *Управление из пользовательских приложений*

Внимание: Это руководство является универсальным как для одноосных так и для двухосных контроллеров 8SMC5

https://youtu.be/cjMuT1hpRFI

Видео быстрого начала работы с контроллером подключенным по USB (для windows)

https://youtu.be/Iu_n6cTIiQU

Видео быстрого начала работы с контроллером подключенным через 8Eth1 adapter (для windows)

3.1.1 Введение

Данное руководство описывает установку контроллера и начало работы с программным обеспечением XILab для Windows. Установка программы на другие ОС описана в разделе *Установка XILab*. Подробно с характеристиками контроллера вы можете ознакомиться в разделе *Технические характеристики*. Для разработки собственных приложений для контроллера рекомендуем ознакомиться с разделом *Руководство по программированию* и скачать пакет программ для разработки приложений в разделе Программное обеспечение.

3.1.2 Требования

Для успешной настройки контроллера необходимо иметь:

- **Компьютер с наличием USB/Ethernet разъёма**



- **Программное обеспечение** Всё необходимое ПО для работы с контроллером можно скачать по ссылке Программное обеспечение.
- **USB или Ethernet кабель**





- **8SMC5 контроллер**. В корпусе или без, с Ethernet разъемом или нет - это не важно.
- **Позиционер или мотор**



Рис. 3.1: Позиционер на базе шагового двигателя

На рисунке представлен используемый в работе позиционер на базе шагового двигателя, более подробно о требованиях к электромотору написано в разделе *Технические характеристики*. Если подразумевается использование собственных кабелей для подключения контроллера к позиционеру, пожалуйста, руководствуйтесь *схемой соединения контроллера и позиционера*, а также *схемой расположения выводов* контроллера. Для позиционеров с конечным диапазоном перемещения следует использовать *концевые выключатели*. Данные контакты используются для определения границ движения подвижки.

- **Блок питания**



- Используйте стабилизированный источник питания 12 - 36 В. Подробней смотрите *Техника безопасности*.
- Если контроллер поставляется в металлическом корпусе, то корпус требует заземления. Заземление контроллера, в случае, если он поставляется без корпуса, происходит через «землю» блока питания. Более подробно о заземлении смотри в разделе *Техника безопасности*.
- Убедитесь, что работающая без корпуса плата контроллера лежит на диэлектрической поверхности.

3.1.3 Установка ПО и первый пуск

Вы можете скачать необходимое программное обеспечение здесь. Выберите файл «xilab-<version_name>.exe». Инсталлятор автоматически определит, запущен ли он на 32-битной или 64-битной системе и установит соответствующую версию XILab. Запустите программу установки, появится окно установки (версии программного обеспечения могут немного отличаться).



Рис. 3.2: Первое окно установки XILab

Нажмите кнопку «Next>» и следуйте инструкциям на экране. Все необходимое программное обеспечение, включая драйвера, пакеты и программы, будут установлены автоматически. Программа XILab автоматически запуститься по окончанию установки. Появится следующее окно:

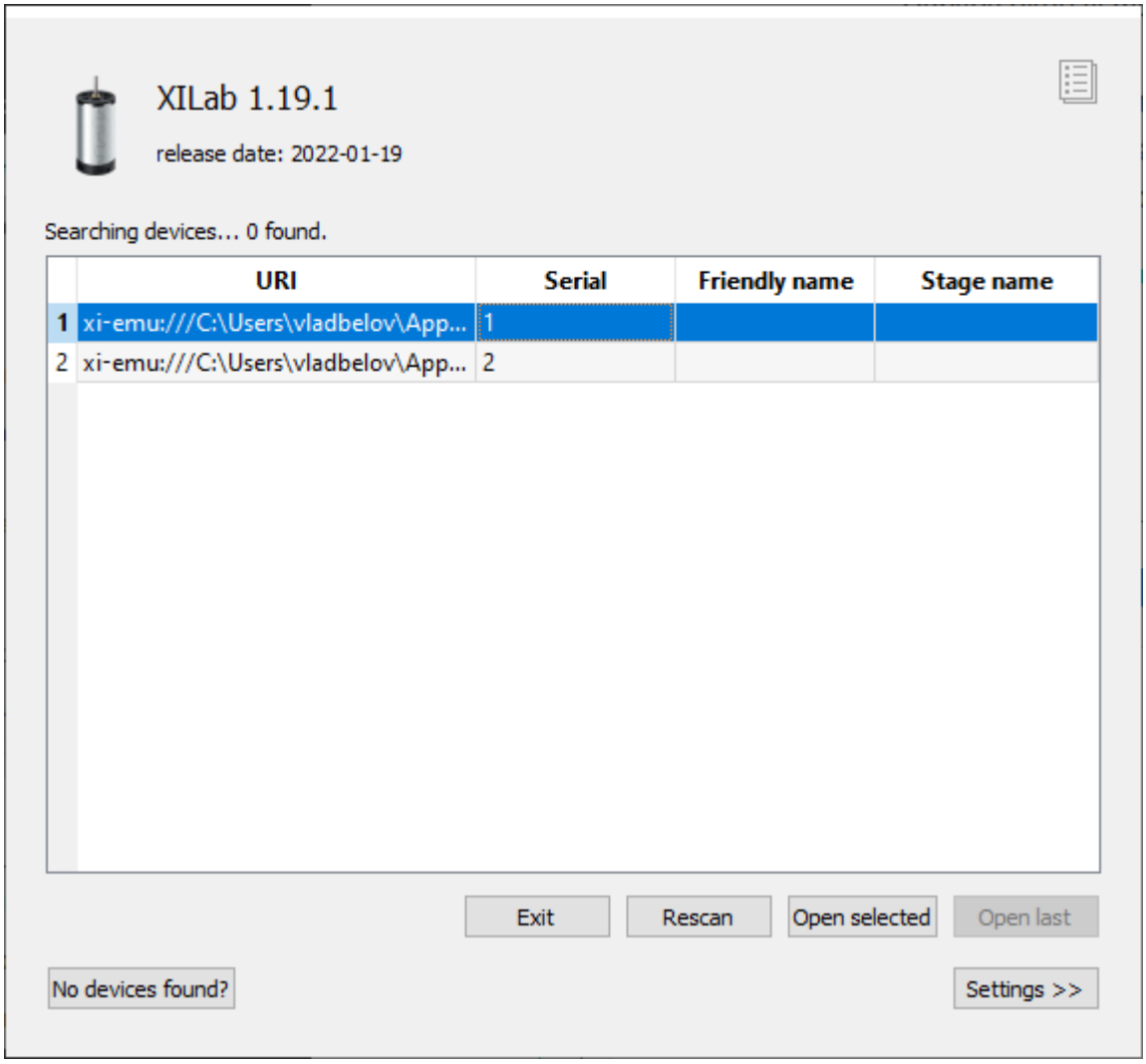


Рис. 3.3: Диалоговое окно программы XILab «Virtual controllers found» (найдены виртуальные контроллеры)

Подключите позиционер к контроллеру. Подключите стабилизированный источник питания к контроллеру. Заземлите контроллер или блок питания. Подключите контроллер к компьютеру используя USB или Ethernet кабель. LED индикатор на плате контроллера начнет *мигать*.

Если контроллер подключен через USB , «New Hardware Wizard» начинает работать после первого подключения контроллера к ПК. Пожалуйста, подождите, пока Windows обнаружит новое устройство и установит для него все необходимые драйверы.

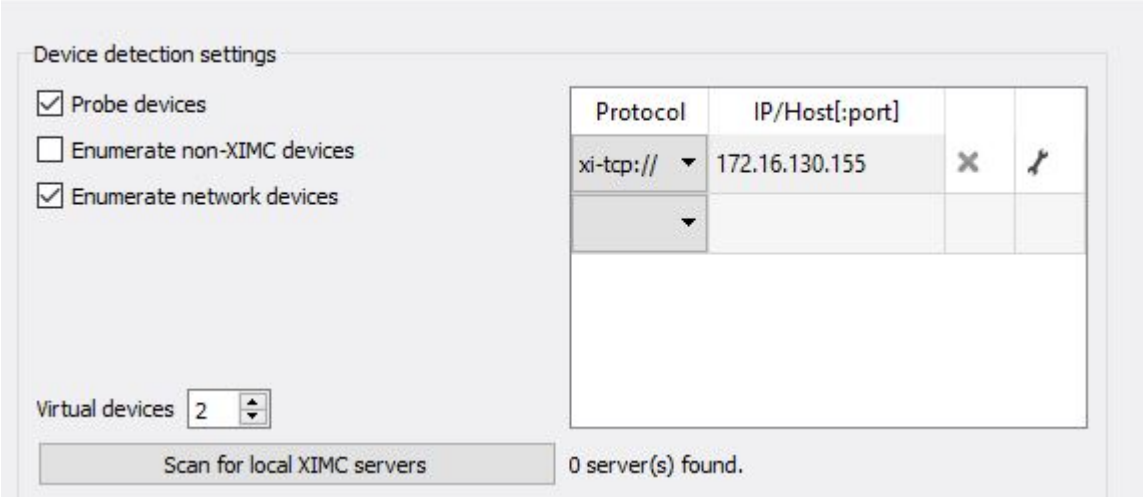
Если драйвер автоматически не установился, то в появившемся окне выберите «No, not this time», затем нажмите «Next>». В следующем окне выберите «Install from a list or specific location (Advanced)» и нажмите «Next>». Выберите **'*.inf'** файл на диске с ПО, поставляемым с контроллером, или в директории **C:\Program Files\XILab\driver** и подождите, пока установка будет завершена.

Вернитесь к диалоговому окну программы XILab «Virtual controllers found» и нажмите «Rescan». Если это окно было закрыто, пожалуйста, снова откройте программное обеспечение XILab. Диалоговое окно откроется снова.

Если вы используете Ethernet-соединение, в стартовом окне XILab нажмите *«Settings >>»* и

включите флаг *«Enumerate network devices»* . Нажмите кнопку *«Rescan»*. Дождитесь окончания поиска контроллера в сети. Обычно контроллер определяется автоматически, но если этого не произошло:

- Убедитесь, что в вашей сети работает DHCP-сервер. Простой DHCP сервер, удовлетворяющий всем требованиям, можно скачать отсюда.
- Используйте *revealer*, чтобы узнать IP адрес, выданный вашему контроллеру. Откройте контроллер в браузере. При необходимости измените настройки сети.
- В стартовом окне XILab нажмите *«Settings >>»*, выберите протокол подключения, введите IP адрес вашего контроллера в поле «IP/Host[:port]».



3.1.4 Начало работы в ПО XILab

XILab представляет собой удобный графический интерфейс пользователя для контроля работы, диагностики и настройки двигателей. Он также может быть использован для легкой установки и сохранения/загрузки параметров для любого двигателя. В этом разделе рассмотрено начало работы с XILab. Для получения полной информации о работе программы, рекомендуем Вам ознакомиться с разделом *Руководство по программе XILab*.

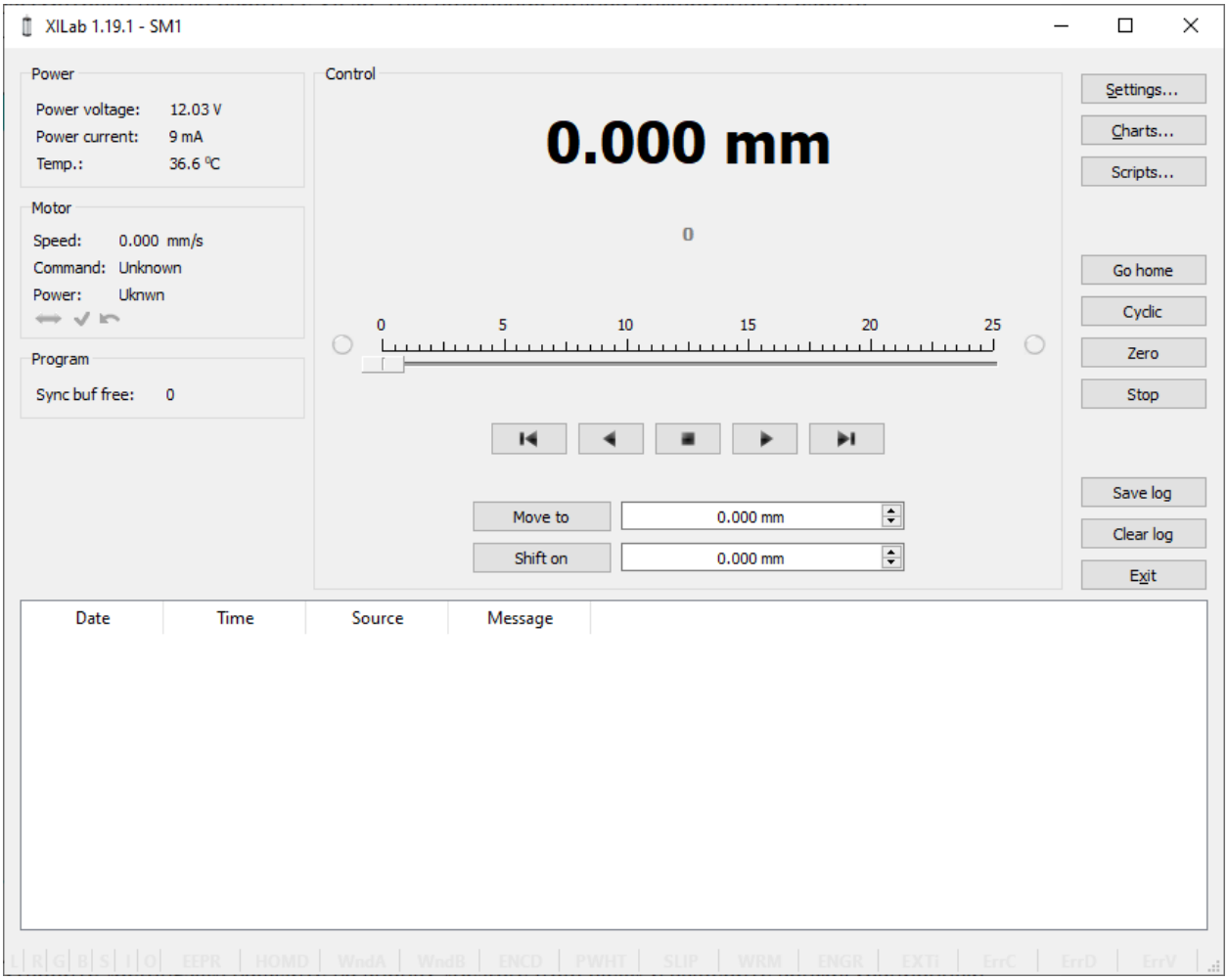


Рис. 3.4: Главное окно программы XILab

Откройте «Settings...», нажмите на кнопку «Load setting from file...» и выберите конфигурационный файл (профиль) для вашего позиционера из открывшейся папки **C:\Program Files\XILab\profiles**. Все поля меню «Settings...» автоматически будут заполнены значениями, подобранными для вашего позиционера. Если нужный файл не найден, оставьте запрос на сайте поддержки.

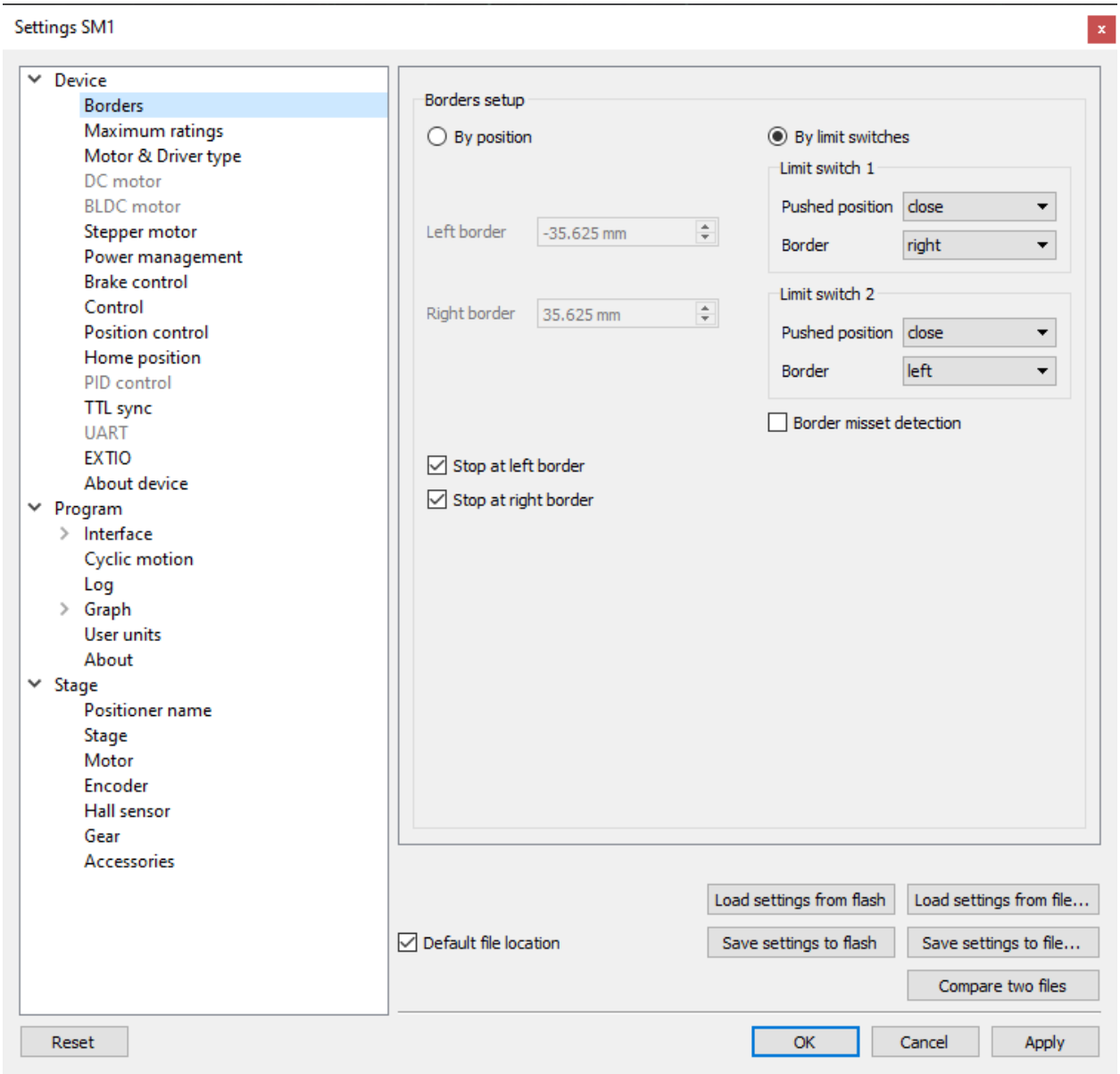


Рис. 3.5: XILab, меню Settings

Предупреждение: Для работы контроллера с двигателями обязательна корректная установка:

- *рабочего тока*;
- *границ движения и концевиков*;
- *критических параметров*;
- *ограничителей*;
- *режима питания мотора*.

Если вы решили настроить контроллер самостоятельно, обязательно проверьте эти настройки!

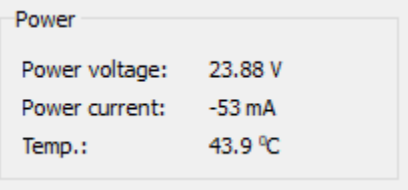
Контроллер готов к работе!

3.1.5 Проверка работоспособности

Для проверки правильной настройки контроллера нажмите в главном окне XILab клавишу влево или вправо в центральном ряду клавиш управления. Позиционер должен начать двигаться. Для остановки вращения используйте центральную клавишу плавной остановки.



Обратите внимание на параметры питания контроллера в блоке *Power*. Там можно увидеть напряжение питания, потребляемый ток и температуру контроллера.



Если при старте движения главное окно XILab окрасилось в красный цвет, то это свидетельствует о сработавшей защите и попадании в режим *ALARM*. Это может быть вызвано неправильными настройками, неверным подключением позиционера или неисправностью контроллера. Подробней смотрите в разделе *Критические параметры*.

3.1.6 Управление из пользовательских приложений

Для удобного управления контроллером 8SMC5 вы можете использовать программу *XIlab*. Однако, если у вас есть необходимость управлять контроллером из собственных приложений, вы легко сможете это сделать, используя функции библиотеки libximc. В *комплекте разработчика библиотеки* представлены примеры на различных языках программирования: *C*, *C#, VB.net*, *Delphi*, *LabView*, *Matlab*, *LabWindows*, *Java*, *Python*, включая *Python (Jupyter Notebook)*.Если Вам необходимо автоматизировать небольшое количество действий, то, возможно, вместо разработки собственной программы вы сочтете более целесообразным использовать для этого *скриптовый язык* программы Xilab.

3.2 Многоосная система

- *Введение*
- *Необходимые комплектующие для работы*
- *Начало работы с многоосной системой в XILab.*
- *Терморегулировка многоосной системы.*
- *Известные ошибки и недоработки*
  - *Работа*
  - *Конструкция*



Рис. 3.6: Внешний вид многоосной системы

3.2.1 Введение

Данное руководство предназначено для работы с многоосной системой 8SMC4-36, построенной на базе контроллеров двигателей 8SMC4-USB. Начало работы описано в разделе *Инструкция по началу работы*. Многоосная система работает от сети 220 В и позволяет одновременно управлять 36 осями.

3.2.2 Необходимые комплектующие для работы



Рис. 3.7: Силовой кабель 220 В



Рис. 3.8: Ethernet кабель. Позволяет подсоединиться к многоосной системе удалённо

Для управления по локальной сети необходимо наличие *DHCP сервера*. Вы должны знать ip-адреса ваших устройств в сети.

3.2.3 Начало работы с многоосной системой в XILab.

Подключите к многоосной системе кабель 220 В и Ethernet кабель. Включите её, нажав кнопку Power On/Off на задней панели. Вы услышите характерный звук вращающихся вентиляторов. Необходимо подождать пару минут, чтобы загрузилась операционная система на управляющем одноплатном компьютере внутри коробки, после чего 8SMC-B36 готова к работе.

Для начала работы потребуется наличие программы XILab с поддержкой многоосных систем версии 1.13.x или выше (см. раздел). Процесс установки и первый запуск программы описан *здесь*. Для получения более детальной информации рекомендуем вам ознакомиться с *руководством по программе XILab*.

Настройка доступа к удаленным контроллерам через Ethernet описана в главе *Работа с сетевыми контроллерами*. В появившемся списке можно выбрать интересующую ось и управлять ей, как в случае с одним *8SMC4-USB*. Также можно выбрать несколько осей и управлять ими в *режиме управления несколькими осями*. Подробнее см. *Начало работы в ПО XILab* и *Руководство по программе XILab*

3.2.4 Терморегулировка многоосной системы.

В многоосную систему 8SMC4-36 встроен термодатчик, позволяющий отслеживать нагрев контроллеров осей в процессе работы, путём измерения температуры проходящего через них воздуха. В случае повышения температуры проходящего воздуха, вентиляторы в системе охлаждения будут увеличивать обороты.

При температуре проходящего воздуха менее 25^oC система охлаждения работает на 10% от своей максимально возможной мощности. Наибольший обдув достигается при температуре воздуха внутри коробки более 45^oC, что фактически означает несколько большую температуру самих контроллеров.

3.2.5 Известные ошибки и недоработки

Все перечисленные замечания указаны для многоосной системы версии 1.1.

3.2.5.1 Работа

Основные замеченные проблемы, которые касаются работоспособности многоосной системы:

- При силовом питании 110В полная потребляемая мощность ограничена значением 1кВт;
- При одновременной экстренной остановке нескольких осей на моторах большой мощности возможна кратковременная потеря связи многоосной системы с интерфейсом управления;
- Возможны сбои с повторным обнаружением осей при горячем включении модулей;
- При перезаливке прошивки контроллеры могут терять USB соединение и становятся недоступными в Xilab вплоть до перезапуска системы;
- Возможна ситуация, когда при добавлении во включенную систему модуля, он оказывается неработоспособен до перезапуска системы.

3.2.5.2 Конструкция

Конструктивные недоработки многоосной системы, с которыми вы можете столкнуться при разборке и сборке:

- Плата ic9xbb. Шелкография P1 - P5 должна быть с обратной стороны соответствующих разъёмов;
- Плата ic9xbb. Перепутана местами шелкография разъёмов P1 и P.

3.3 Пример подключения простого мотора

- *Общий случай*
- *Пример*
  - *Подготовка*
  - *Подключение мотора и энкодера к контроллеру*

3.3.1 Общий случай

Для того, чтобы подключить мотор к контроллеру, необходимо знать распиновку разъёма *подключения позиционера*, а также типовую схему подключения мотора к контроллеру:

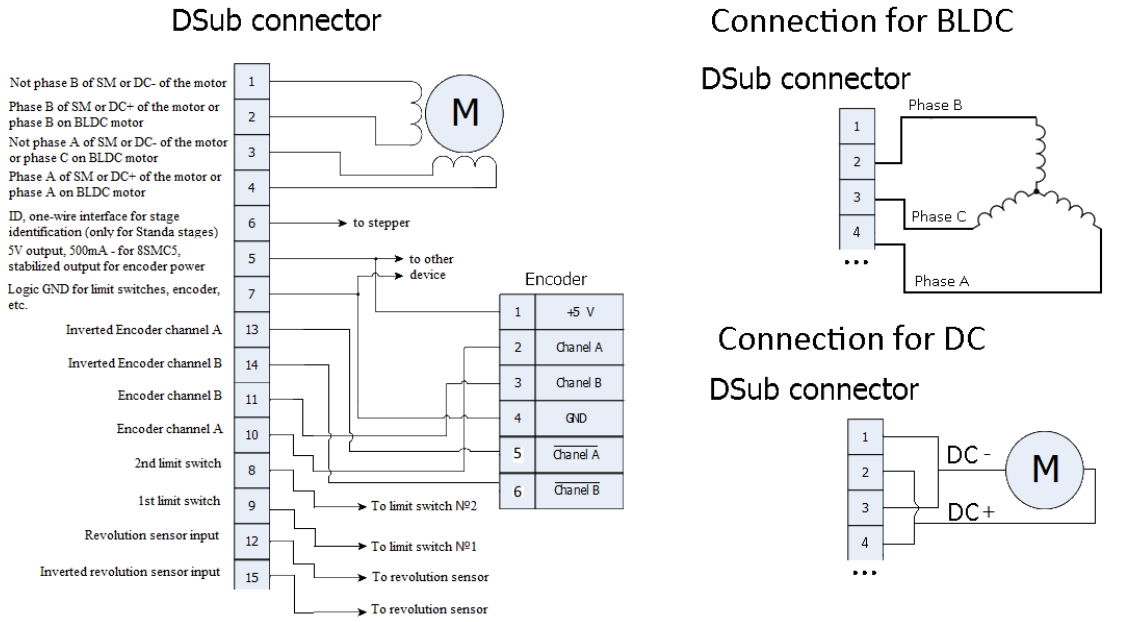


Рис. 3.9: Общая схема подключения позиционера с энкодером через разъем D-sub

Примечание: Если каналы A и B энкодера работают в режиме открытого коллектора, то для обеспечения максимальной частоты передачи сигнала на высоких скоростях вращения могут потребоваться дополнительные подтяжки выходов энкодера к питанию 5 В резисторами (см. *Работа с энкодерами*).

Примечание: Управление BLDC моторами возможно только в прошивках 4.1.0 и новее.

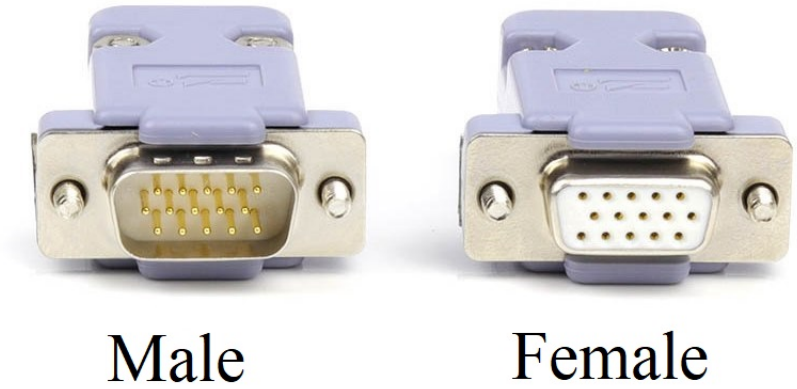
3.3.2 Пример

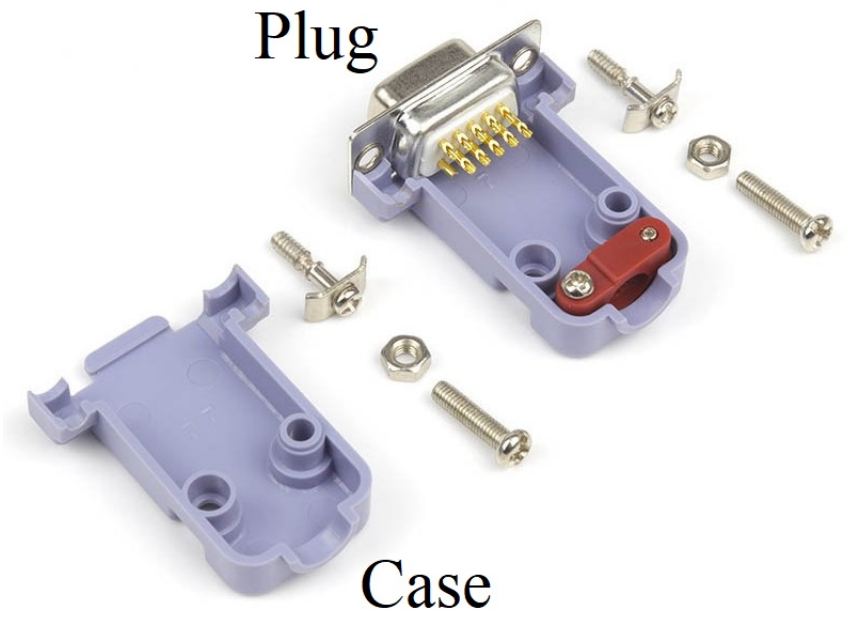
Рассмотрим подключение двигателя с энкодером CUI INC AMT112S-V к контроллеру 8SMC5-USB на примере двухфазного шагового мотора Nanotec ST5918L3008-B.

3.3.2.1 Подготовка

Для начала работы нам понадобятся:

- Мотор;
- Энкодер;
- *Распиновка разъёма D-SUB* для 8SMC5-USB;
- Спецификация на мотор ;
- Спецификация на энкодер ;
- Паяльное оборудование: паяльник, провода, флюс, припой, кусачки, термоусадочные трубки разных размеров;
- Винты M2.5x6 для крепления энкодера;
- D-SUB корпус + разъём (male) и провода для изготовления своего кабеля;





3.3.2.2 Подключение мотора и энкодера к контроллеру

- Прежде чем начать работу, соберите энкодер в соответствие с инструкцией по сборке, прилагаемой к нему.

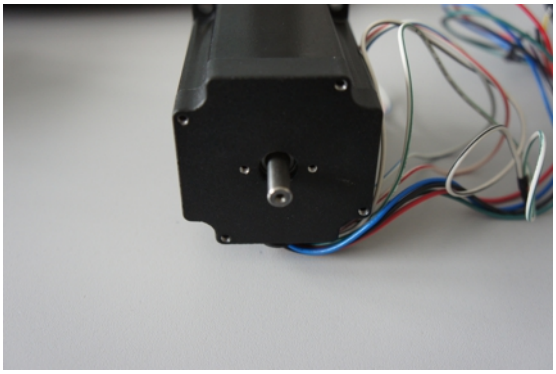


Рис. 3.10: Мотор без энкодера. Обратите внимание на 2 отверстия M2.5 к которым обычно крепится энкодер

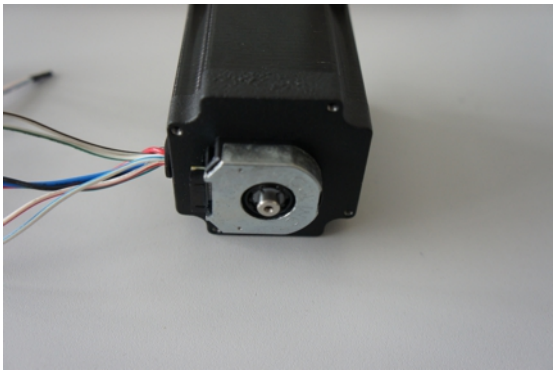


Рис. 3.11: Мотор с прикреплённым энкодером

- Смотрим в спецификацию мотора и находим маркировку выводов (для Nanotec ST5918L3008-B она находится справа внизу в спецификации):

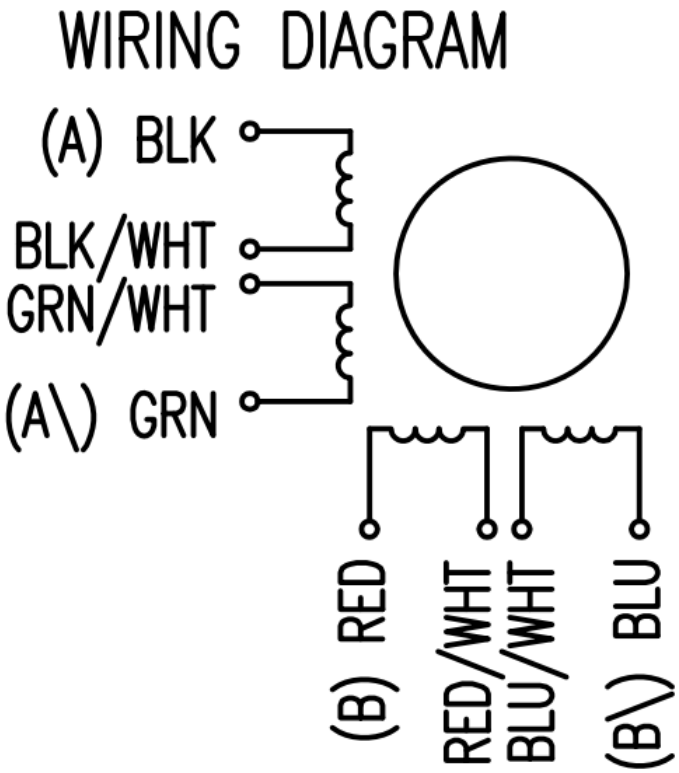


Рис. 3.12: Контакты мотора

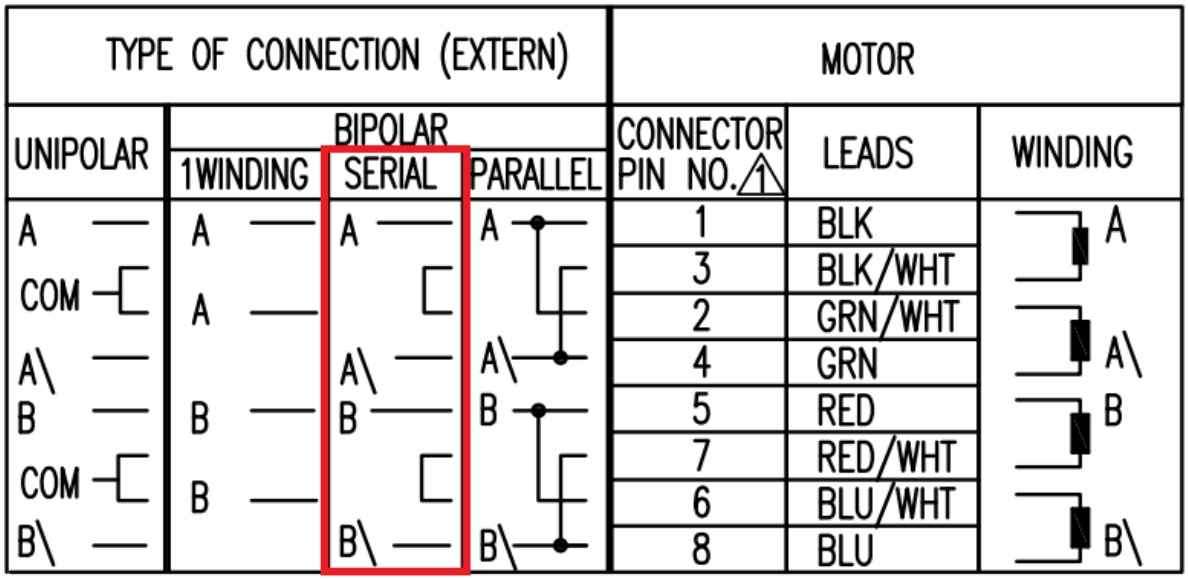
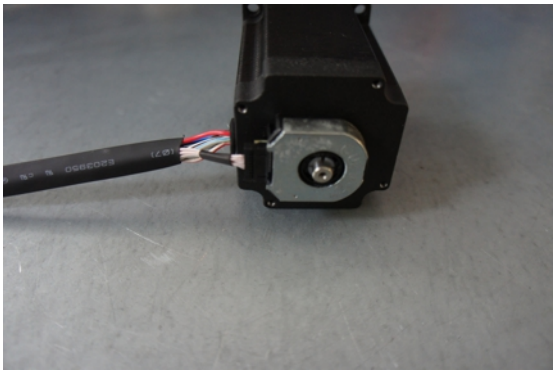


Рис. 3.13: Тип соединения

– Существует последовательное и параллельное соединение обмоток, причём каждое из соединений позволяет получить свои характеристики для мотора. Мы соединим обмотки последовательно (на картинке выше обозначено красным). Для этого провода, имеющие два цвета BLK/WHT и GRN/WHT, а также RED/WHT и BLU/WHT надо соединить между собой попарно. Далее необходимо поставить в соответствие *A*, *не A*, *B*, *не B* контактам разъёма контроллера, контакты обмоток мотора ST5918L3008-B: чёрный, зелёный, красный, голубой. Одна обмотка - это соединение *A* - *не A* или *B* - *не B*. После соединения между собой двухцветных проводов, получится что одна обмотка мотора - это соединение чёрный - зелёный, а другая красный - голубой. Поэтому соответствие контактов будет таким: чёрный- *A*, зелёный - *не A*, красный - *B*, голубой - *не B*. Это же соответствие видно на картинке Тип соединения выше.

– Для подключения энкодера откройте спецификацию на энкодер и найдите на его разъёме 5 контактов: *A+* (канал *A*), *B+* (канал *B*, сдвинутый относительно *A* на 90 град), *Z+* (счётчик оборотов), *5 V*, *GND*. Их надо вывести от энкодера 5 проводами и пустить вместе с проводами от мотора, т.к. далее они пойдут на один разъём. Энкодер CUI INC AMT112S-V имеет 18-пиновый вход, поэтому надо сделать кабель с таким же разъёмом на конце, чтобы вывести необходимые сигналы:



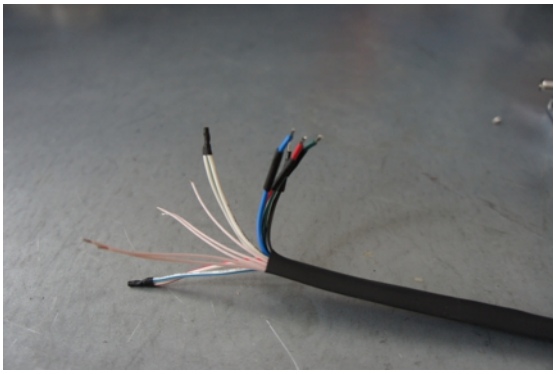
Контакты энкодера A+, B+, Z+, 5 V и GND соответствуют контактам 10, 11, 12, 5, 7 *D-SUB male разъёма* соответственно.

Для удобства воспользуйтесь таблицами подключения к D-SUB разъёму (в скобках указан номер пина на соответствующем разъёме):

Контакты энкодера	**Контакты D-SUB**
A+ (10)	Энкодер A (10)
B+ (8)	Энкодер B (11)
Z+ (12)	Вход датчика оборотов (12)
5 В (6)	Выход 5 В, 100 мА (5)
GND (4)	Земля логическая (7)

Контакты мотора	**Контакты D-SUB**
A (BLK)	ШД фаза *A* (4)
not A (GRN)	ШД фаза *не A* (3)
B (RED)	ШД фаза *B* (2)
not B (BLU)	ШД фаза *не B* (1)

- Припаяйте вышеуказанные контакты к D-SUB male разъёму:



Провода от мотора и энкодера, собранные при помощи термоусадочной трубки. Обратите внимание на наличие термоусадочных трубок малого размера на проводах, идущих к обмоткам мотора (BLK, GRN, RED и BLU), а также на соединённые вместе двухцветные провода (BLK/WHT и GRN/WHT, RED/WHT и BLU/WHT). Тоненькие проводки - это контакты энкодера. Их 5 штук.

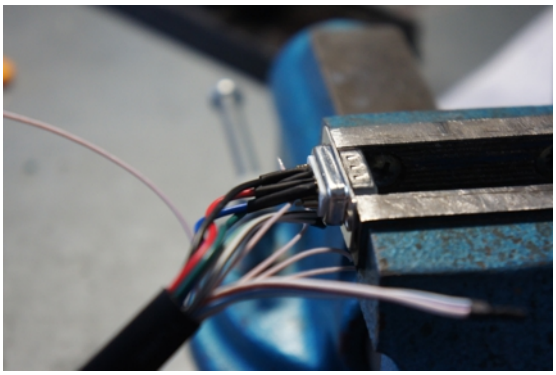


Рис. 3.14: Припаянные контакты обмоток мотора

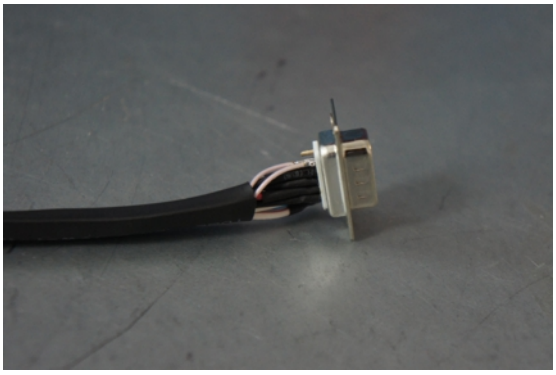
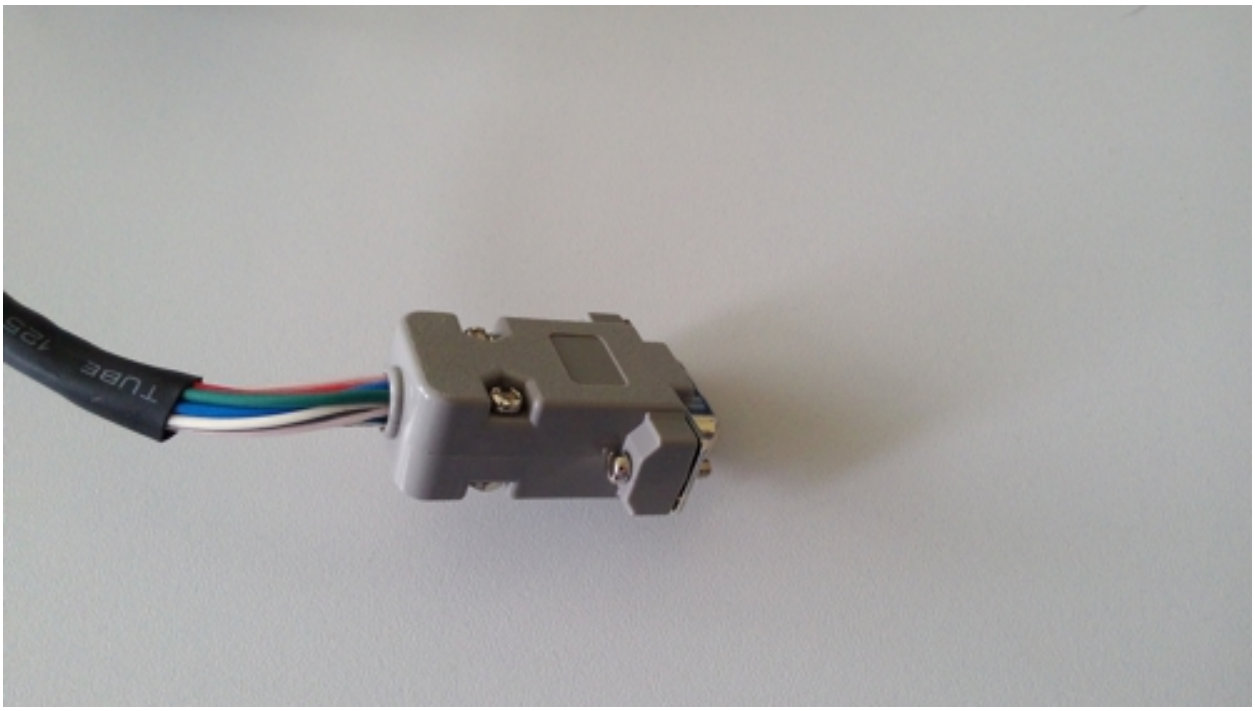


Рис. 3.15: Готовый провод, идущий от мотора с D-SUB разъёмом на конце

Рекомендация: используйте термоусадочные трубки малого диаметра (2-3 мм) при пайке контактов к D-SUB разъёму и большого диаметра - для того, чтобы через них пропустить все провода, идущие от мотора и энкодера. Надевайте трубки до пайки.

- Сверху на разъём одевается защитный кожух



- Теперь мотор можно подключить к контроллеру 8SMC5-USB.

Описание и настройки профиля дана в следующей главе *Ручная настройка профиля*.

3.4 Ручная настройка профиля

- *Введение*

- *Подготовка к работе*
- *Настройка рабочего тока*
- *Настройка базовых параметров*
- *Настройка аппаратных концевых выключателей, процедура автокалибровки.*
- *Настройка параметров энкодера*
- *Настройка кинематических характеристик контроллера*
- *Работа с пользовательскими единицами измерения*

3.4.1 Введение

После подключения мотора, настраиваются необходимые для работы параметры (см. *Пример подключения двигателя*. Рассмотрим настройку профиля на примере шагового мотора **Nanotec ST5918L3008-B**.

3.4.2 Подготовка к работе

- Устанавливаем и запускаем XILab (см. раздел *Краткое руководство и начало работы*).
- Загружаем профиль с выставленными по умолчанию настройками. Для этого откройте **Settings -> Load setting from file...** и выберите xilabdefault.cfg, лежащий в корне папки с XILab .

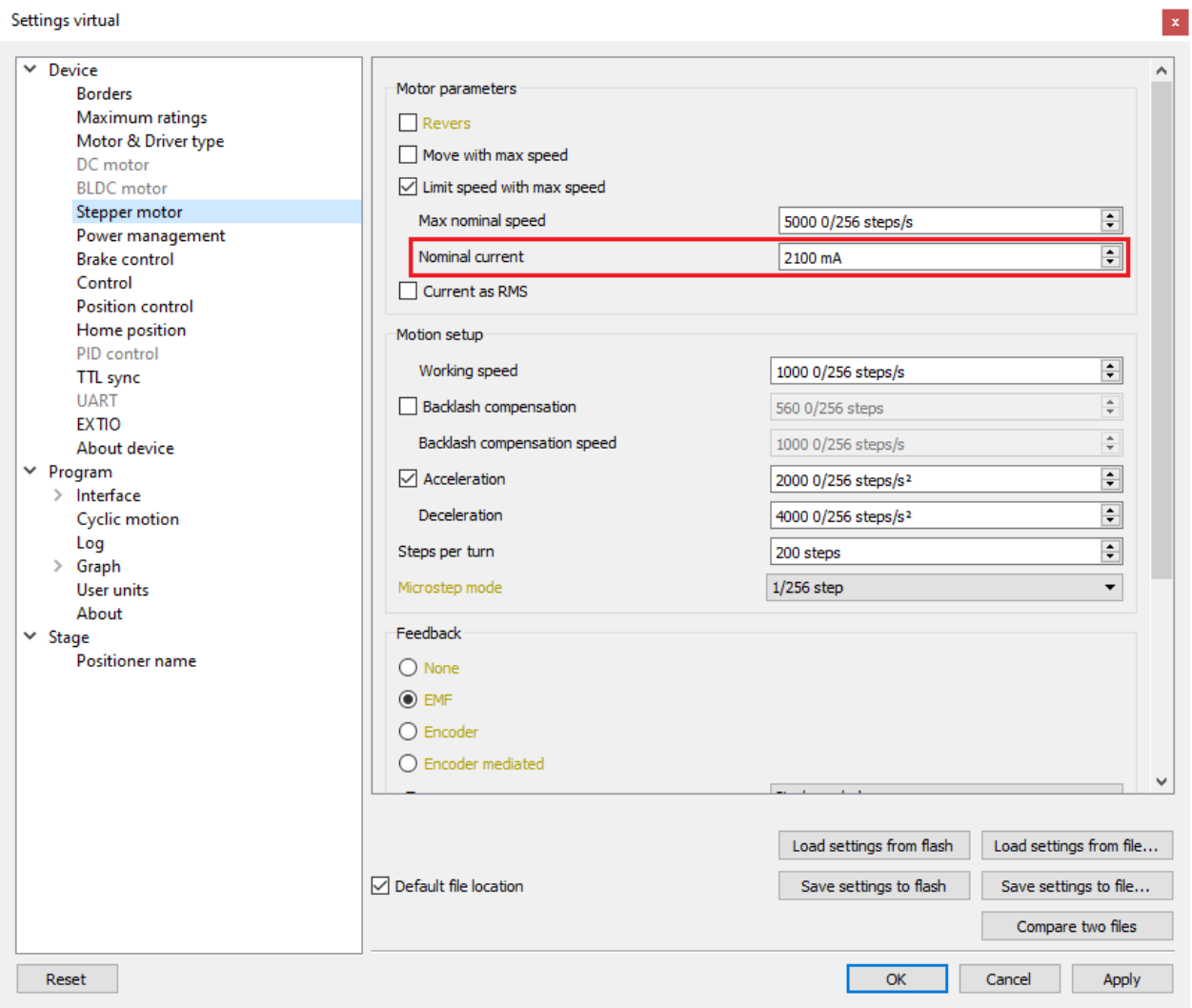
3.4.3 Настройка рабочего тока

Первоначально необходимо настроить правильный ток в обмотках двигателя:

- Из спецификации находим **ток фазы 2.1 А** - это максимальное значение тока для данного мотора при соединении обмоток **последовательно**:

SPECIFICATION \ CONNECTION	UNIPOLAR OR BIPOLAR–1 WINDING	BIPOLAR	
		SERIAL	PARALLEL
VOLTAGE (VDC)	3.0		
AMPS/PHASE	3.0	2.1	4.2
RESISTANCE/PHASE (Ohms)@25°C	1.0±10%	2.0±10%	0.5±10%
INDUCTANCE/PHASE (mH) @1KHz	2.2±20%	8.8±20%	2.2±20%

- Находясь в окне *Settings*, откройте вкладку *Stepper motor*. Тут задаются такие параметры, как скорость вращения, ускорение, режим движения и др. (подробнее см. *Настройка кинематики движения (Шаговый двигатель)*). В поле *Motor parameters-> Nominal current* выставляется ток фазы для мотора. В это поле нужно вписать значение **не превышающее 2.1 А**:



3.4.4 Настройка базовых параметров

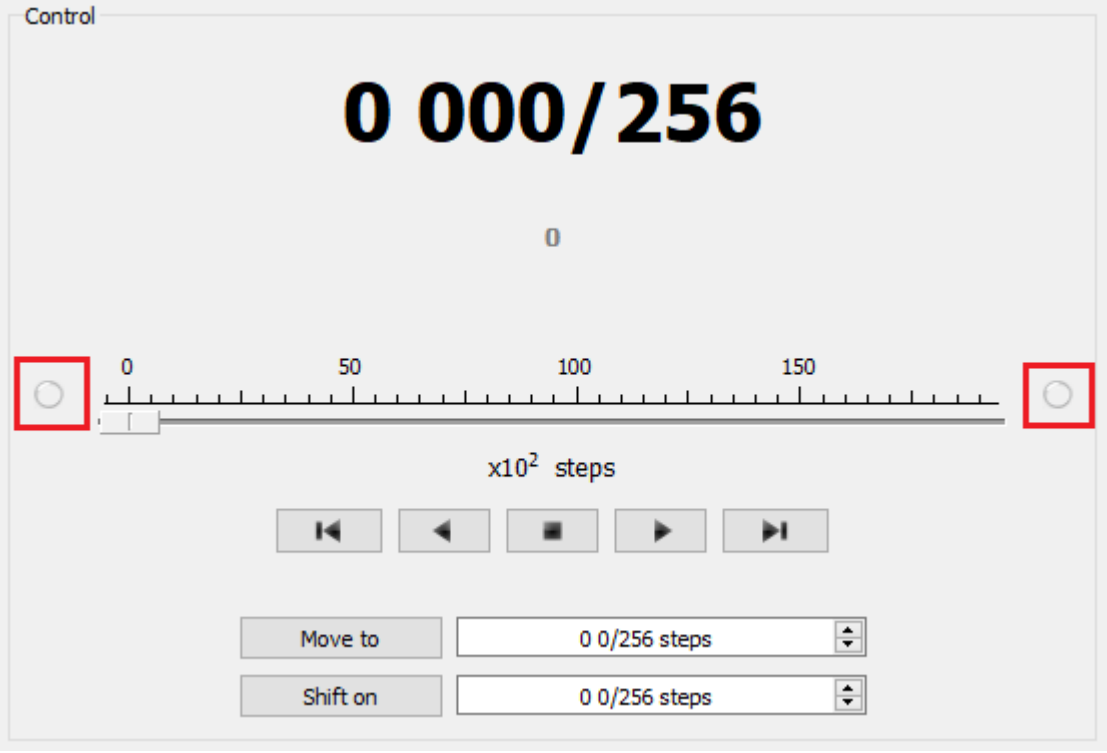
- Укажем скорость вращения в поле *Working speed*. Рекомендуемая величина скорости не более **1000 steps/s** при первом запуске. В том же окне укажите *Max Nominal Speed* (**5000 steps/s** для большинства моторов и позиционеров является разумным значением) и установите галочку напротив *Limit speed with max speed*. Данная настройка нужна, чтобы ограничить скорость мотора, т.к. некоторые механические системы могут быть не рассчитаны на высокие скорости и слишком быстрое вращение может привести к сильному износу механики подвижки/мотора.
- Из спецификации мотора найдём количество шагов на оборот. Для нашего мотора это значение равно **200 ш**. Укажем его в поле *Steps per turn*. Обычно в описаниях к мотору приводится величина одного шага в градусах, исходя из которой можно расчитать количество шагов на оборот, зная, что в одном обороте 360 градусов
- Проверьте, что при старте движения вправо из главного окна XILab подвижка физически тоже движется вправо. Если это не так, то поставьте галочку *Reverse* в поле *Stepper motor -> Motor parameters*.

3.4.5 Настройка аппаратных концевых выключателей, процедура автокалибровки.

Примечание: Данный пункт подразумевает использование позиционеров с аппаратными *концевыми выключателями*. Если в вашей системе аппаратные концевые выключатели не предусмотрены, то рекомендуется отключить остановку по концевикам в настройках. Для этого следует убрать галочки *Stop at right border* и *Stop at left border* во вкладке *Borders*.

Позиционеры бывают с ограниченным (трансляторы) и с неограниченным диапазоном движения (ротаторы). Ограничение диапазона перемещения может осуществляться с помощью концевых выключателей или по позиции. При работе с позиционерами первого типа при неправильной настройке концевых выключателей существует риск сломать механику, т.к. подвижная часть может попытаться выехать за пределы допустимого диапазона движения. У ротаторов такой проблемы нет. Также следует иметь ввиду, что у ротаторов может быть всего один концевик.

- Для работы с концевыми выключателями контроллеру необходимо указать какой из них будет левым, а какой - правым. Иногда это заранее неизвестно, а известно лишь, что оба концевика подключены и срабатывают каждый по достижении своей границы перемещения. Если неправильно настроить концевики, то позиционер может заклинить. Поэтому контроллер поддерживает простую функцию обнаружения неверно настроенных концевиков, останавливаясь по обоим из них. Убедитесь, что:
  - Подвижная часть находится вдали от концевиков;
  - Полярность концевиков настроена правильно (индикаторы концевиков не горят в главном окне XILab). В случае неправильной настройки, поменяйте их полярность (*Borders -> Pushed position*), индикаторы должны погаснуть.



  - Включена остановка по обоим концевикам (галочки напротив *Stop at right border* и *Stop at left border* во вкладке *Borders*).
- Включите флаг обнаружения неправильного подключения концевиков *Border misset detection* во

вкладке *Borders*.

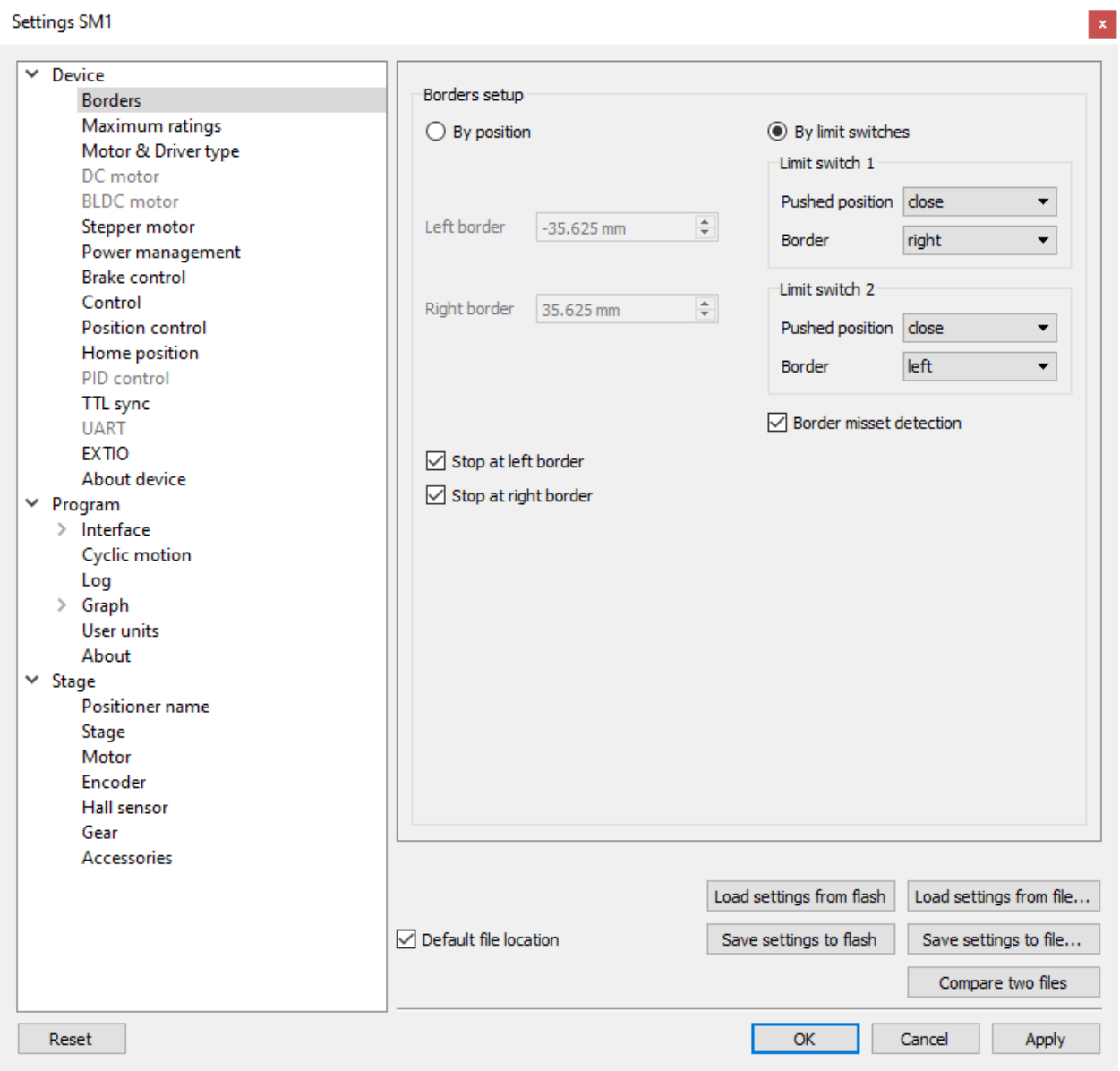


Рис. 3.16: Вкладка с настройками концевиков

- При обнаружении неверного срабатывания концевика, контроллер может перейти в режим Alarm, если включена настройка *Enter Alarm state when edge misset is detected* в меню *Maximum ratings*. Рекомендуется включить эту опцию. Начните движение в любую сторону из *главного окна XILab* до остановки по концевику или перехода в режим Alarm. При возникновении Alarm нужно поменять концевики местами, изменив значения в полях *Borders->Border* на противоположные.

Предупреждение: Защита от перепутанных концевиков не гарантирует, что о проблеме перепутанности можно забыть. Она лишь облегчает первоначальную настройку. При перепутанных концевиках нельзя начинать движение, если какой-либо концевик активен, даже при включенной

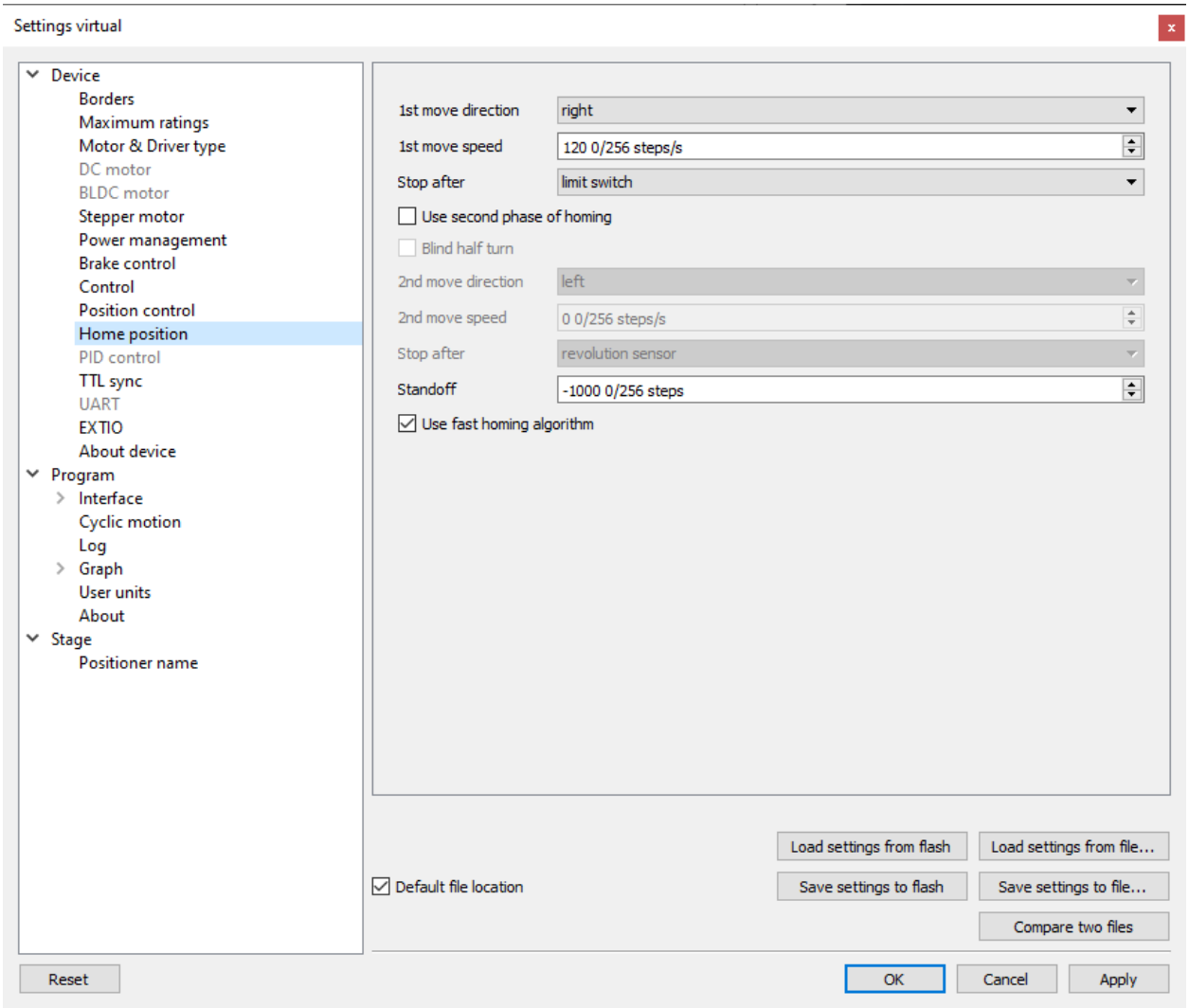
функции защиты.

Существует ещё 2 способа определения какой из концевиков правый, а какой левый:

- Необходимо знать, куда подсоединён каждый из концевиков в позиционере. При загрузке профиля с настройками по умолчанию, концевик, подсоединённый к пину 9 D-SUB разъёма контроллера, считается левым, а к пину 8 - правым. Их расположение относительно позиционера настраивается в полях *Limit switch 1* и *Limit switch 2* (см. скриншот выше). Запустите систему на маленькой скорости (<100 steps/s) вдали от концевиков. Если направление движения к концевику отличается от ожидаемого, измените значения в полях *Borders->Border* на противоположные.
- Если возможно подлезть к концевикам, то попробуйте активировать их и увидеть в главном окне XILab загорающиеся индикаторы. Запомните, какой из индикаторов, какому концевику соответствует. Затем, вдали от концевиков, запустите систему на маленькой скорости (<100 steps/s) и убедитесь, что система движется в направлении срабатывания правильного выключателя. Соотнесите это с тем, что видите в главном окне XILab. Если направление движения к концевику в реальной установке и в главном окне различаются, измените значения в полях *Borders->Border* на противоположные.

Для более подробной информации обратитесь к *соответствующему пункту документации*.

- В контроллере предусмотрена полезная функция *автокалибровки домашней позиции* для установки начального положения позиционера.



Рассмотрим наиболее простой вариант настройки с одной фазой движения. Начать следует с установки скорости первой фазы (*1st move speed*) примерно в 5-10 раз меньшей, чем *Working speed*. Это нужно для более высокой точности процедуры автокалиборовки. В поле *Stop after* укажите *limit switch*, чтобы в процессе автокалибровки подвижка доезжала до одного из концевиков (до правого или левого - выбирается в поле *1st move direction*). В поле *Standoff* укажите число в шагах, на которое подвижка должна отъехать от концевика. Нажмите *Ok* или *Apply*.

Примечание: Значение в поле *Standoff* является знаковым. Положительное направление - право. То есть, если процедура автокалибровки настроена по правому концевику, то для того, чтобы подвижка отъехала от него влево в поле *Standoff* должно быть отрицательное значение.

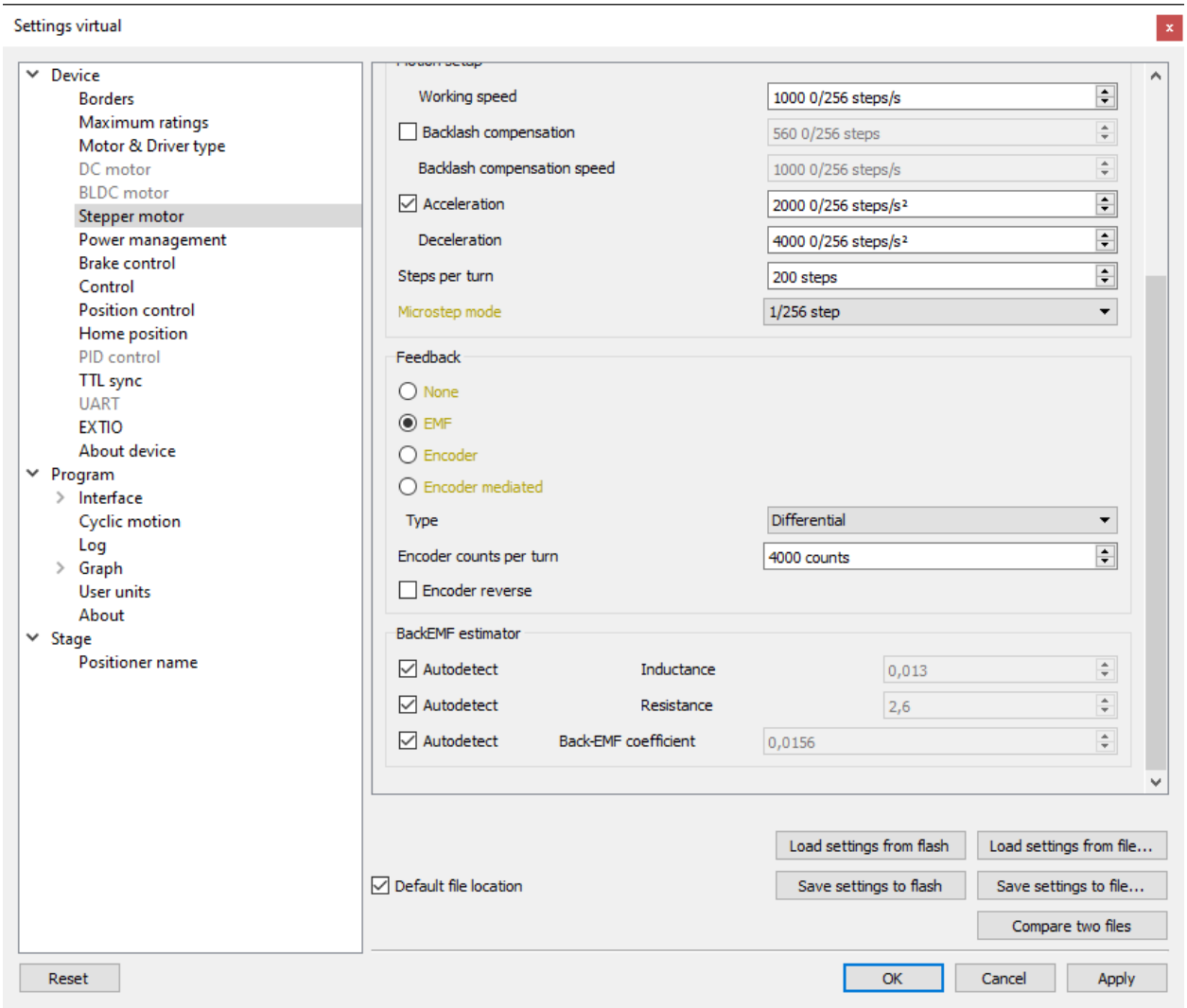
- Запустите процедуру автокалибровки, нажав на кнопку *Go home* в главном окне XILab. Результатом будет движение подвижки до указанного концевика с относительно низкой скоростью и смещение от него в сторону на значение, указанное в поле *Standoff*.
- После выполнения автокалибровки нажмите *ZERO* в главном окне XILab для установки начала системы отсчёта.
- Повторите процедуру калибровки второй раз. Подвижка, при этом, должна снова вернуться в

нулевую позицию. Обратите внимание, что могут быть небольшие отклонения от нуля, связанные с погрешностью процедуры автокалибровки.

3.4.6 Настройка параметров энкодера

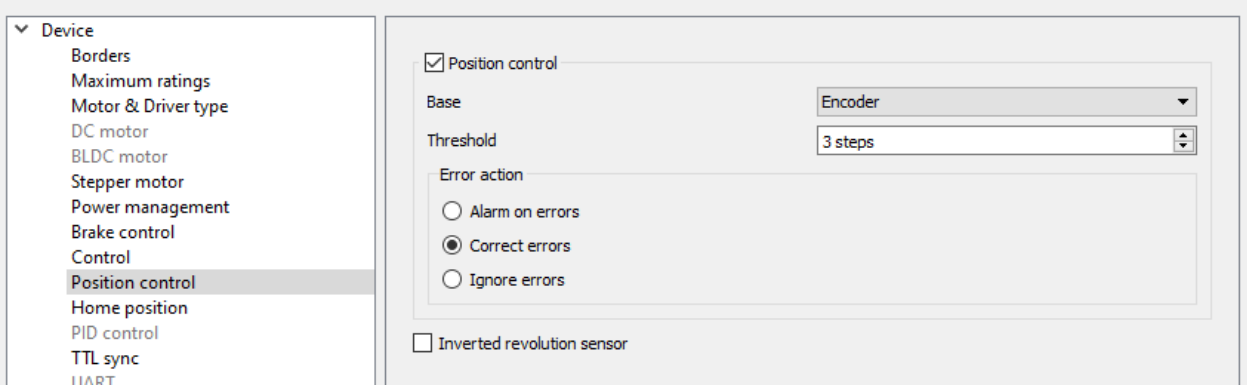
Примечание: Данный пункт подразумевает использование мотора с энкодером. Если у вас двигатель без энкодера, то описанные ниже параметры можно оставить без изменений.

- Каждый энкодер имеет характеристику, указывающую количество импульсов на оборот (**Pulse Per Turn - PPT**). Для корректной работы энкодера с контроллером, необходимо в интерфейс XILab, в поле **Encoder counts per turn** (вкладка *Stepper motor*) вписать количество отсчётов энкодера на оборот, которое равно **4xPPT**. Например, если ваш энкодер имеет **1024** импульса на оборот, то в поле *Counts per turn* вписывается число **4096**:



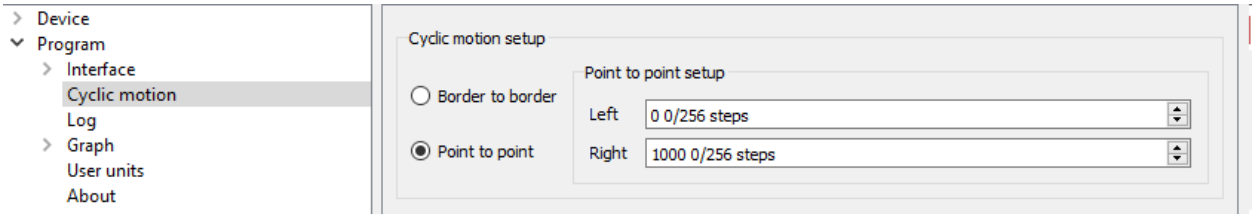
- Запустите вращение из *главного окна XILab*. Если всё настроено правильно, то внизу окна загорится зелёным индикатор ENCD. В случае, если ENCD имеет жёлтый цвет, то следует поставить галочку *Encoder reverse* во вкладке *Stepper motor*, в поле с настройками энкодера. Красный цвет ENCD указывает на наличие проблем с пересчётом позиции по энкодеру.

- Существует возможность активировать контроль позиции по энкодеру. Для этого во вкладке *Position control* следует поставить галочку *Position control* и в поле *Threshold* указать допустимую ошибку в отсчётах энкодера. Тогда при рассогласовании позиции с отсчётами энкодера, в главном окне XILab будет загораться индикатор SLIP и, если стоит галочка *Alarm on errors*, контроллер будет переходить в состояние Alarm. Установка *Correct errors* позволяет контроллеру работать в режиме ведущего энкодера, компенсируя разницу между реальной позицией и позицией, соответствующей отсчётам энкодера.



3.4.7 Настройка кинематических характеристик контроллера

- Во вкладке *Stepper motor* можно указать необходимое ускорение (*Acceleration*) и замедление (*Deceleration*) для используемого шагового двигателя. Процедура выбора оптимальных значений следующая:
  - Начиная со значений по умолчанию, делайте небольшие сдвиги мотора (старт и быстрый стоп), постепенно увеличивая *Acceleration* до тех пор, пока движение при этом не начнёт срываться или быть нестабильным. Примите ускорение равным примерно *половине* этого значения.
  - Замедление можно настроить примерно в **1.5 - 2 раза больше**, чем ускорение
- Если в вашей механической системе подъезд в желаемую позицию слева и справа неодинаков и присутствует люфт, то есть возможность устранить эту неоднозначность. Для этого поставьте галочку напротив *Backlash compensation* в Stepper motor и укажите значение, превышающее величину люфта. Знак этой настройки определяет направление подхода. Положительный знак означает подход слева, а отрицательный - справа. В поле *Backlash compensation speed* настройте скорость, с которой будет выполняться компенсационное движение. Её величина должна быть маленькой (**значения 50 steps/s достаточно**), чтобы не было «заносов» во время антилюфта.
- После основной настройки позиционера/мотора можно увеличить рабочую скорость. Процедуру настройки рабочей скорости можно провести экспериментально, подобно процедуре настройки ускорения, т.е. выбрать значение примерно в **2 раза меньшее**, чем то, при котором наблюдается нестабильное движение. Для проверки стабильности вращения рекомендуется воспользоваться функцией *Cyclic* в интерфейсе *главного окна*, предварительно её *настроив*.



- В поле *Microstep mode* мы рекомендуем оставить значение **1/256 шага**.

3.4.8 Работа с пользовательскими единицами измерения

Зачастую неудобно работать с шагами и микрошагами и хочется работать с более удобными единицами измерения. По этой причине в контроллере есть возможность пересчитывать координаты в привычные единицы измерения, например в миллиметры или градусы. Это делается во вкладке *User units*, где указывается величина шага и соответствующая ей единица измерения. Для более подробной информации обратитесь к *соответствующему пункту документации*.

Настройка рабочего профиля завершена.

3.5 Расчёт номинального тока

Для того, чтобы шаговый двигатель выдавал максимальный вращающий момент, но при этом не перегревался, важно правильно задать такую техническую характеристику, как номинальный ток.

Чем больше ток в обмотке двигателя, тем больше вращающий момент на оси. Важно помнить, что с увеличением протекающего через обмотки тока, выделяемая тепловая мощность двигателя увеличивается. Чтобы двигатель мог работать длительное время, выделяемая тепловая мощность (Закон Джоуля — Ленца) должна быть меньше мощности рассеяния. Мощность рассеяния можно рассчитать исходя из документации на двигатель.

3.5.1 Расчеты на базе параметров униполярного полношагового режима

Мощность рассеяния равна

$$P = n \cdot R_u I_u^2,$$

где R_u - сопротивление обмотки в униполярном режиме, I_u - ток через одну обмотку в униполярном режиме, n - колличество одновременно работающих обмоток.

Рассмотрим для примера ST2818M1006. Таблица в документации показывает, что в полношаговом режиме одновременно работает две обмотки (n = 2) в униполярном режиме, т.е. $P = 2R_u I_u^2$. Контроллеры моторов поддерживают только биполярный режим управления. Чтобы перейти от униполярного в биполярный режим, соединим обмотки каждой фазы последовательно, сопротивление возрастёт, $R_b = 2R_u$, где R_b - сопротивление последовательно соединенных обмоток для биполярного режима управления.

Алгоритм управления в контроллерах моторов работает в микрошаговом режиме и поддерживает ток так, что в одной обмотке ток меняется по функции $I_a \sin(\phi)$, в другой обмотке ток меняется по функции $I_a \cos(\phi)$, где I_a - амплитуда тока. Тепловая мощность, выделяемая двумя обмотками в любой момент времени

$$P = R_b I_a^2 \sin^2(\phi) + R_b I_a^2 \cos^2(\phi) = R_b I_a^2$$

Получим уравнение, приравняв мощности, из которого найдём, что $I_a = I_u$.

3.5.2 Расчеты на базе параметров биполярного полношагового режима

Мощность рассеяния равна $P = n \cdot R_b I_b^2$, где R_b - сопротивление обмотки в биполярном режиме, I_b - ток через одну обмотку в биполярном режиме, n - колличество одновременно работающих обмоток.

Рассмотрим для примера ST2018S0604. Таблица в документации показывает, что в полношаговом режиме одновременно работает две обмотки (n = 2) в биполярном режиме, т.е. $P = 2R_b I_b^2$.

Тепловая мощность, выделяемая на обмотках двигателя, управляемого контроллерами моторов, по-прежнему

$$P = R_b I_a^2 \sin^2(\phi) + R_b I_a^2 \cos^2(\phi) = R_b I_a^2$$

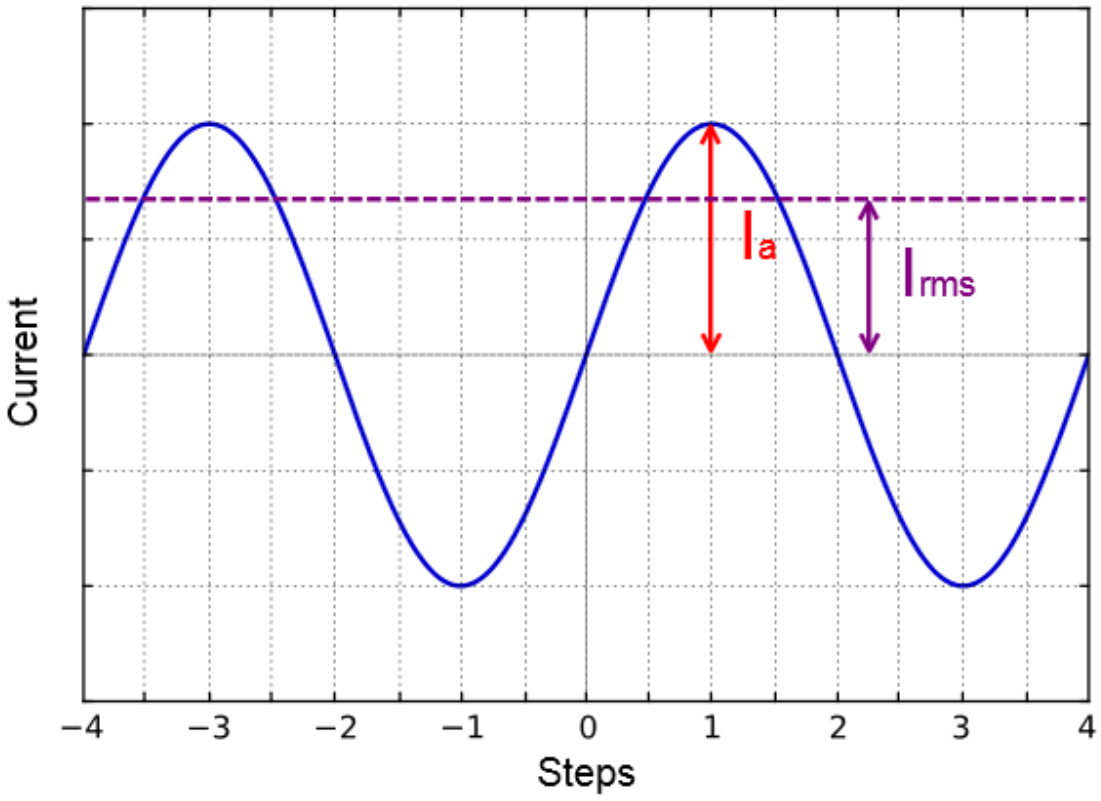
Получим уравнение, приравняв мощности $2R_b I_b^2 = R_b I_a^2$. Найдем, что $I_a = \sqrt{2} \cdot I_b$.

3.5.3 Связь со среднеквадратичным током

Переменный ток в каждой обмотке двигателя может характеризоваться своим среднеквадратичным значением за период

$$I_{rms} = \sqrt{\frac{1}{2\pi} \int_0^{2\pi} (I_a \sin(\phi))^2 \, \mathrm{d}\phi} = \frac{I_a}{\sqrt{2}}$$

Тепловое выделение **одной** обмотки связано со среднеквадратичным током через неё $P_1 = R_b I_{rms}^2$. Обе обмотки идентичны $P_1 = P_2$. Общая тепловая мощность двигателя под управлением контроллера моторов $P = P_1 + P_2 = 2R_b I_{rms}^2$.



Из вышеописанного следует, что $I_{rms} = \frac{I_u}{\sqrt{2}}$, а также $I_{rms} = I_b$.

3.5.4 Амплитудный и номинальный ток для BLDC

Номинальный ток двигателя рассчитывается из максимально допустимого тепловыделения. Номинальный ток, написанный в документации, рассчитан из ограничения на мощность, выделяемую при подключении источника питания к двум обмоткам.

Запишем формулу для мощности при таком подключении:

$$P_{chop} = 2R_{phase}I_{rate}^2$$

Формула для мощности, выделяемой обмотками для синусоидального управления:

$$P_{sin} = 3R_{phase}I_{rms}^2$$

Номинальный ток двигателя рассчитывается, исходя из ограничения на мощность. Приравняем правые части формул:

$$I_{rms} = \frac{\sqrt{2}}{\sqrt{3}}I_{rate}$$

Итак,

$$I_{amp} = \frac{2I_{rate}}{\sqrt{3}}$$

Это означает: если в документации на ваш двигатель сказано, что номинальный ток равен, например, 0.88A, то в контроллер можно записать значение:

$$I_{amp} = \frac{2 * 0.88}{\sqrt{3}} = 1A$$

3.5.5 Настройка номинального тока

Контроллеры моторов способны принимать значение номинального тока в виде амплитуды тока I_a или в виде среднеквадратичного значения I_{rms}. Выбор того, каким способом интерпретировать входное значение номинального тока, определяется отсутствием/наличием соответственно флага *ENGINE_CURRENT_AS_RMS* в поле *EngineFlags* структуры *engine_settings* . При *настройке номинального тока в XILab* следует правильно указывать способ интерпретации тока. Контроллеры моторов в этом случае будут обеспечивать максимальный допустимый момент, не перегревая двигатель.

Этот же флаг определяет смысл значения тока BLDC.

Как и для шагового двигателя, в XiLab есть специальная галочка, которая определяет, как трактовать введённое в поле Nominal current значение. Если галочка «Amplitude current» отмечена, введённое значение тока будет амплитудным: максимальная амплитуда синуса будет всегда меньше этого значения. Если галочка «Amplitude current» снята, ведённое значение будет пересчитано по формуле (3) и амплитуда тока будет ограничена уже этим пересчитанным значением

Для всех моторизованных позиционеров Standa подготовленные конфигурационные файлы содержат номинальный ток, заданный среднеквадратичным значением. Соответсвующий флаг установлен. Таким образом, двигатели работают на оптимальных параметрах.

Глава 4

Техническое описание устройства

4.1 Внешний вид и разъемы

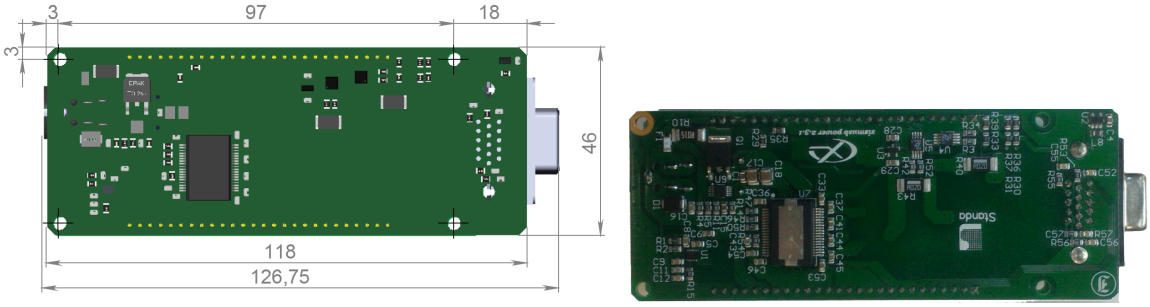
Контроллеры 8SMC5-USB выпускаются в 3 различных версиях:

- *Плата контроллера*
- *Одноосная система*
- *Двухосная система*

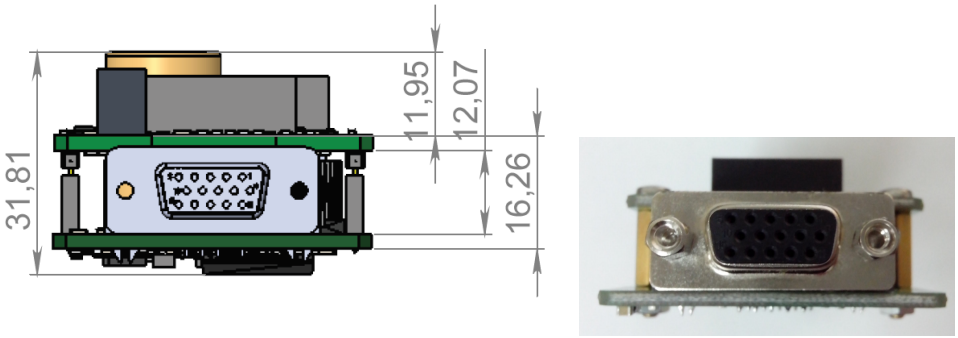
4.1.1 Плата контроллера

4.1.1.1 Геометрические размеры

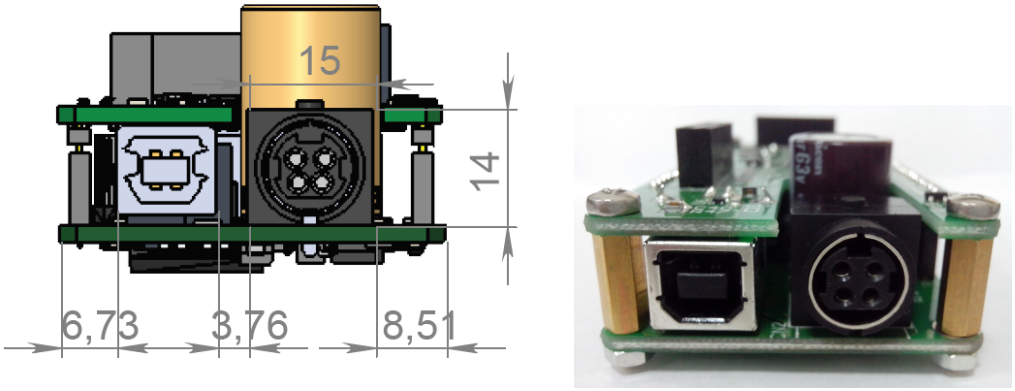
Конструктивно контроллер представляет собой две жестко соединенные печатные платы размером 127 x 46 x 32 мм. Верхняя плата содержит логический контроллер и схемы управления, нижняя - силовую часть.



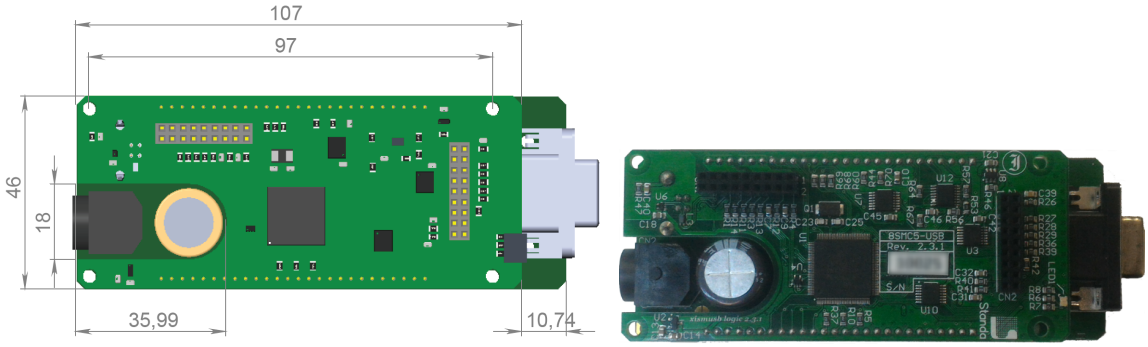
Контроллер. Вид снизу. Со стороны силового модуля и радиатора.



Контроллер. Вид спереди. Со стороны кабеля для подключения позиционера.



Контроллер. Вид сзади. Со стороны разъема для подключения питания USB type-B.



Контроллер. Вид сверху. Со стороны разъемов для подключения к объединительной плате.

Важно: В случае самостоятельной установки радиатора на силовой драйвер, убедитесь, что нет контакта между теплопроводящей поверхностью силового драйвера и другими электропроводящими частями установки. Если такой контакт будет присутствовать, возможно повреждение силовой схемы! Данное предупреждение относится только к контроллерам, поставляемым без внешнего корпуса.

4.1.1.2 Разъемы подключения плат

4.1.1.2.1 Разъем подключения позиционера

Для подключения позиционера используется разъем DSub 15 типа «мама».

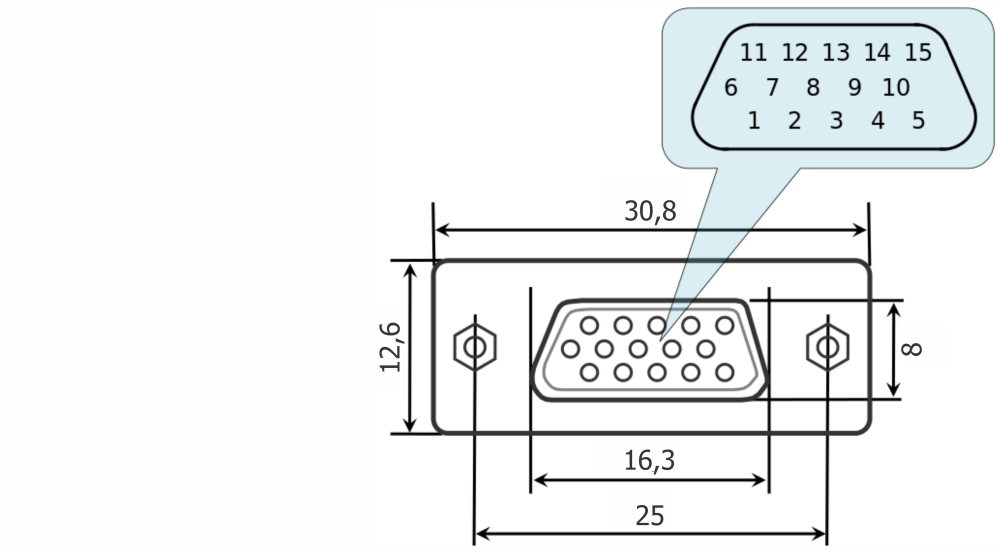


Рис. 4.1: Размеры и нумерация выводов в разъеме DSub, вид спереди

Назначение выводов:

1. ШД фаза не B или -DC мотора
2. ШД фаза B или +DC мотора или фаза B BLDC мотора
3. ШД фаза не A или -DC мотора или фаза C BLDC мотора
4. ШД фаза B или +DC мотора или фаза B BLDC мотора
5. Выход 5 В, 500 мА, стабилизированный выход для питания энкодера
6. ID, однопроводной интерфейс опознавания подвижки (работает только с позиционерами Standa)
7. Земля логическая для концевиков, энкодера и прочего.
8. Концевик №2
9. Концевик №1
10. Энкодер A
11. Энкодер B
12. Вход датчика оборотов
13. Инверсный канал энкодера A
14. Инверсный канал энкодера B
15. Инверсный вход датчика оборотов

Примечание: BLDC моторы поддерживаются в прошивке версий 4.1.0 и старше.

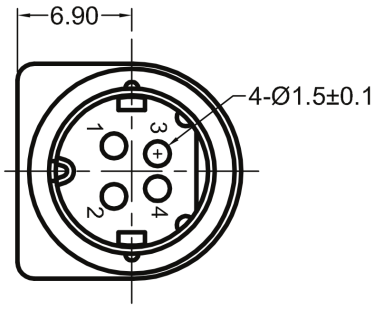
Примечание: Для подключения DC мотора с рабочим током выше 3A нужно соединить выходы 1 и 3, а также выходы 2 и 4.

Предупреждение: Не рекомендуется подключать к контроллеру или отключать от контроллера мотор, пока на его обмотках есть питание.

4.1.1.2.2 Разъем силового питания

Одноосные и двухосные системы используют разъём Kycon 4-pin DC power (part number по каталогу KPPX-4P, www.kycon.com).





Назначение выводов:

1. Силовое питание, '-'.
2. Силовое питание, '+'. 12-48В.
3. Силовое питание, '-'.
4. Силовое питание, '+'. 12-48В.

Важно: Никогда не подавайте электропитание на контроллер и не подключайтесь к разъему силового электропитания, если вы не уверены, что параметры вашего блока питания соответствуют требуемым. Не пытайтесь подключить электропитание к контроллеру если не уверены, что разъемы на блоке питания и на корпусе контроллера совместимы друг с другом! Допустимые параметры подключения указаны в *Техника безопасности*.

Важно: Горячее присоединение/отсоединение, а также ненадежное подсоединение силового питания через разъём типа Kycon может вывести из строя контроллер и/или компьютер. Подробнее смотрите в *Техника безопасности*.

4.1.1.2.3 Разъемы управления системой

Контроллеры подключаются через разъем USB type-B или Ethernet.



Рис. 4.2: Кабель USB type-A - USB type-B

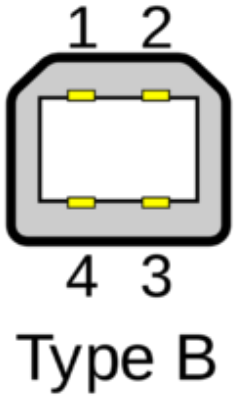


Рис. 4.3: Разъем USB type-B

Таблица 4.1: Таблица расположения выводов кабеля USB type-B

Номер вывода	Название	Цвет кабеля	Описание
1	VCC	Красный	+5 В DC
2	D-	Белый	Data -
3	D+	Зеленый	Data +
4	GND	Черный	Земля

Предупреждение: Используйте только проверенные и заведомо работоспособные USB-кабели! Неисправный или некачественный USB кабель может стать причиной неправильной работы контроллера, в том числе ошибок при вращении мотором или при распознавании устройства операцион-

ной системой. Для стабильного соединения используйте короткие кабели с толстыми проводниками и экранировкой.

4.1.1.3 Разъем подключения к объединительной плате

Для подключения к объединительной плате на плате контроллера установлен 20-ти контактный двухрядный штыревой соединитель (PBD-20R) с шагом 2.54 мм типа «мама».

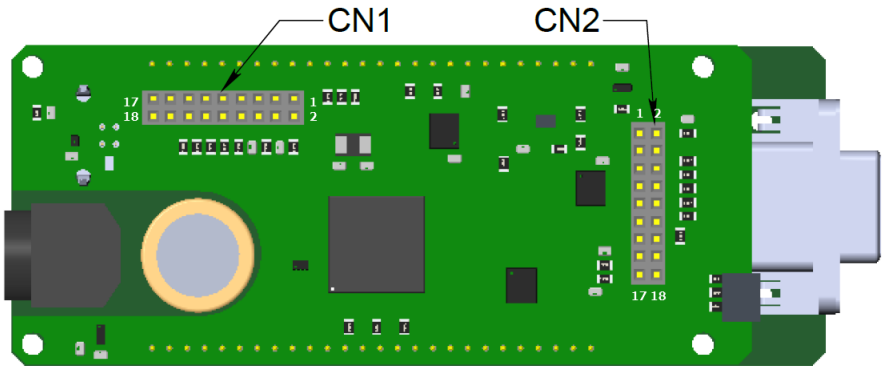


Рис. 4.4: Размеры и нумерация выводов в разъеме BPC (BackPlane Connector), вид со стороны разъема

Назначение выводов разъема CN1:

1. Силовая земля 12-48В.
2. JOY, аналоговый вход 0-3.3 В, используется для подключения внешнего *джойстика*.
3. Силовая земля 12-48В.
4. POT, аналоговый вход 0-3.3 В *общего назначения*.
5. Силовая земля 12-48В.
6. EXTGPIO0, вход-выход общего назначения (или сигнал Enable при использовании внешнего драйвера), 3.3 В логика.
7. Силовое питание 12-48 В.
8. U_LED, светодиод индикации режима работы контроллера (статус).
9. Силовое питание 12-48 В.
10. L_LIM, светодиод индикации левого концевика.
11. Силовое питание 12-48 В.
12. R_LIM, светодиод индикации правого концевика.
13. USB_SEL, выбор USB устройства в многоосной системе.
14. EXTGPIO1, вход-выход альтернативной функции, 3.3 В логика.
15. USB_D0_N, отрицательный выход шины USB Slave.
16. H_USB_D0_N, отрицательный выход шины USB Master.
17. USB_D0_P, положительный выход шины USB Slave.

18. H_USB_D0_P, положительный выход шины USB Master.

Назначение выводов разъема CN2:

1. Зарезервировано.
2. TX1 *последовательного порта*, 3.3 В логика.
3. Зарезервировано.
4. TX1 *последовательного порта*, 3.3 В логика.
5. Зарезервировано.
6. DIR, сигнал direction для управления внешним драйвером, 3.3 В логика.
7. Зарезервировано.
8. STEP, сигнал step для управления внешним драйвером, 3.3 В логика.
9. Зарезервировано.
10. SYNC_OUT, выход синхронизации, 3.3 В логика.
11. Зарезервировано.
12. SYNC_IN, вход синхронизации, 3.3 В логика.
13. EMBRAKE, выход для управления электромагнитным тормозом, 3.3 В логика.
14. 3.3 В логическое питание (было реализовано для логических плат версии 2.3.6 и выше).
15. BUT_R, вход с внешней *кнопки «Вправо»*.
16. Выход питания 5 В.
17. BUT_L, вход с внешней *кнопки «Влево»*.
18. Цифровая земля для логики 3.3 В и 5 В.

Примечание: Зарезервированные контакты внутреннего разъема не требуют какого-либо дополнительного подключения или подтяжки к земле/питанию. **Просто не используйте их.**

Внимание: Аналоговые входы Joy, Pot рассчитаны на работу с напряжением до 3.3 В. Никогда не подавайте на эти входы большее напряжение, в том числе 3.3 В. Это может нарушить правильную работу всех аналоговых каналов контроллера и вывести из строя контроллер или двигатель.

Выводы, доступные на внутреннем разъеме не имеют защит, установка дополнительных буферов и фильтрующих цепочек **целиком и полностью находится на ответственности пользователя или разработчика**, создающего предназначенную для использования этих линий объединительную плату.

Внутренний разъем не допускает подачу каких-либо напряжений свыше 0.3 В относительно ножки DGND на обесточенный контроллер. Установка дополнительных защит, предотвращающих такие ситуации, если они возможны, **целиком и полностью находится на ответственности пользователя или разработчика**, создающего предназначенную для использования этих линий объединительную плату.

4.1.2 Одноосная система

Одноосная версия контроллера представляет собой *плату контроллера* в металлическом корпусе. Размеры корпуса: 124 x 68 x 48,5 мм.

На передней панели расположен *разъем силового питания*, *разъём для подключения к компьютеру тип USB type-B* , светодиоды «стасус контроллера», «питание», «левый концевик», «правый концевик», кнопки для движения вправо и влево.

На задней панели расположен *разъем подключения позиционера*.

Левая боковая панель содержит два Ethernet порта.

4.1.2.1 Разъёмы

4.1.2.1.1 Разъем подключения позиционера

Для подключения позиционера используется разъем DSub 15 типа «мама».

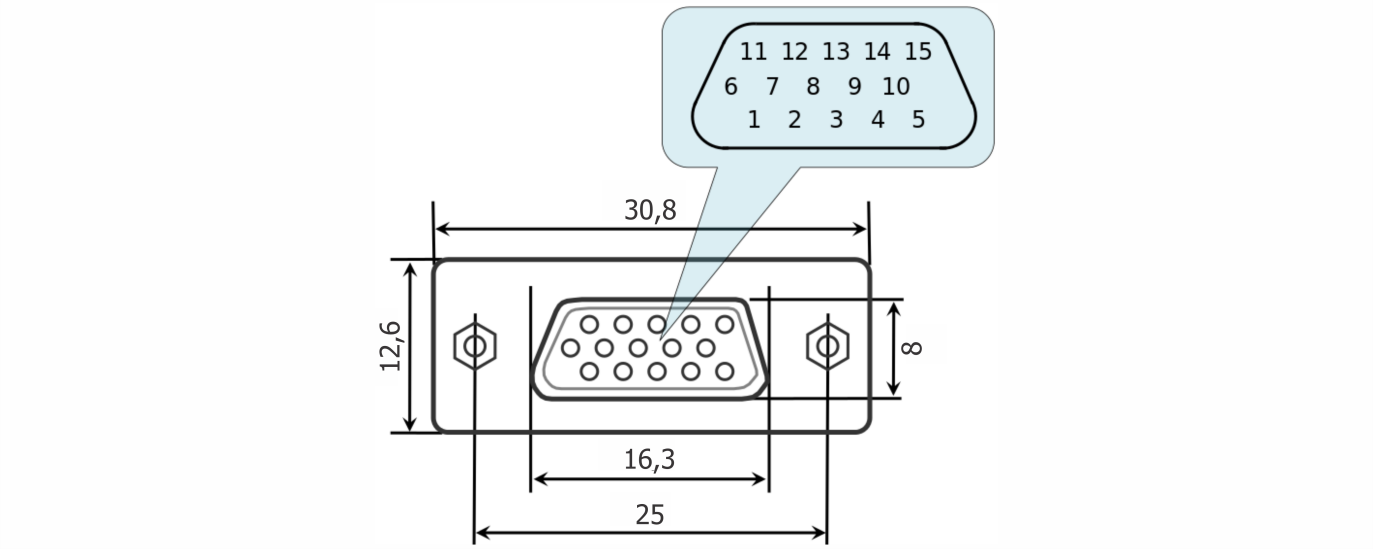


Рис. 4.5: Размеры и нумерация выводов в разъеме DSub, вид спереди

Назначение выводов:

1. ШД фаза не B или -DC мотора
2. ШД фаза B или +DC мотора или фаза B BLDC мотора
3. ШД фаза не A или -DC мотора или фаза C BLDC мотора
4. ШД фаза B или +DC мотора или фаза B BLDC мотора
5. Выход 5 В, 500 мА, стабилизированный выход для питания энкодера
6. ID, однопроводной интерфейс опознавания подвижки (работает только с позиционерами Standa)
7. Земля логическая для концевиков, энкодера и прочего
8. Концевик №2
9. Концевик №1
10. Энкодер A

11. Энкодер B
12. Вход датчика оборотов
13. Инверсный канал энкодера A
14. Инверсный канал энкодера B
15. Инверсный вход датчика оборотов

Примечание: BLDC моторы поддерживаются в прошивке версий 4.1.0 и старше.

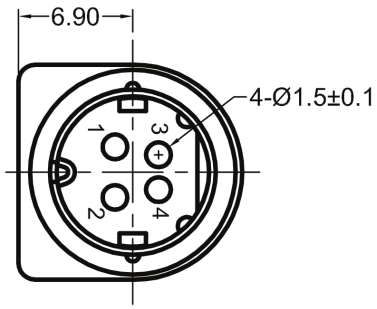
Примечание: Для подключения DC мотора с рабочим током выше 3A нужно соединить выходы 1 и 3, а также выходы 2 и 4.

Предупреждение: Не рекомендуется присоединять к/отсоединять от контроллера двигатель пока есть питание на обмотках мотора.

4.1.2.1.2 Разъем силового питания для 1- и 2-осных систем

Одноосные и двухосные системы, поставляемые в корпусе, используют разъём Kycon 4-pin DC power (part number по каталогу KPPX-4P, www.kycon.com).





Назначение выводов:

1. Силовое питание, '-'.
2. Силовое питание, '+'. 12-48В.
3. Силовое питание, '-'.

4. Силовое питание, '+'. 12-48В.

Важно: Никогда не подавайте электропитание на контроллер и не подключайтесь к разъему силового электропитания, если вы не уверены, что параметры вашего блока питания соответствуют требуемым. Не пытайтесь подключить электропитание к контроллеру если не уверены, что разъемы на блоке питания и на корпусе контроллера совместимы друг с другом! Допустимые параметры подключения указаны в разделе *Техника безопасности*.

Важно: Горячее присоединение/отсоединение, а также ненадежное подсоединение силового питания через разъём типа Kycon может вывести из строя контроллер и/или компьютер. Подробнее смотрите в разделе *Техника безопасности*.

4.1.2.1.3 Разъемы управления системой

Контроллеры подключаются через разъём USB type-B или Ethernet.



Рис. 4.6: Кабель USB type-A - USB type-B

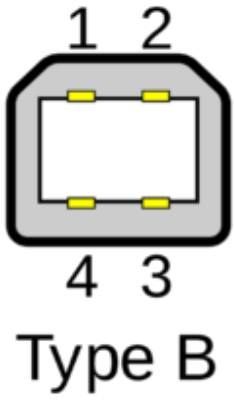


Рис. 4.7: Разъем USB type-B

Таблица 4.2: Таблица расположения выводов кабеля USB type-B

Номер вывода	Название	Цвет кабеля	Описание
1	VCC	Красный	+5 В DC
2	D-	Белый	Data -
3	D+	Зеленый	Data +
4	GND	Черный	Земля

Предупреждение: Используйте только проверенные и заведомо работоспособные USB-кабели! Неисправный или некачественный USB кабель может стать причиной неправильной работы контроллера, в том числе ошибок при вращении мотором или при опознавании устройства операционной системой. Для стабильного соединения используйте короткие кабели с толстыми проводниками и экранировкой.

4.1.2.1.4 Дополнительный разъем одноосных систем

На корпусе одноосной системы расположен порт HDB-26 типа «мама».

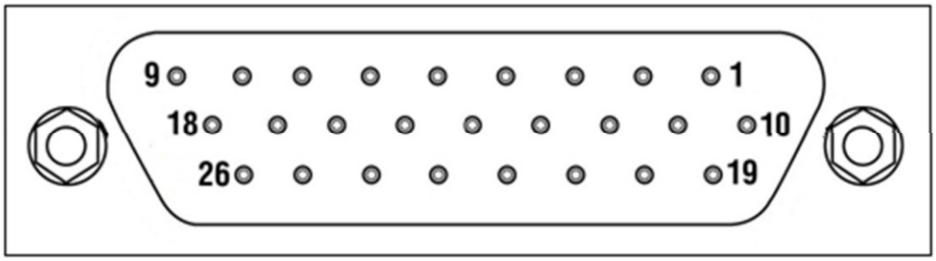


Рис. 4.8: Нумерация выводов в разъеме HDB-26, вид спереди

Назначение выводов:

1. NC, не используется
2. NC, не используется
3. NC, не используется
4. DGND, цифровая земля
5. NC, не используется
6. SYNC_IN, вход синхронизации
7. SYNC_OUT, выход синхронизации
8. RX, вход последовательного порта
9. NC, не используется
10. NC, не используется
11. NC, не используется
12. NC, не используется
13. DIR, сигнал direction для управления внешним драйвером

14. DGND, цифровая земля
15. +5 В
16. BRAKE, выход управления тормозом
17. CLK, сигнал clock для управления внешним драйвером
18. TX, выход последовательного порта
19. NC, не используется
20. NC, не используется
21. NC, не используется
22. PGND, силовая земля
23. PBRK (24В), выход магнитного тормоза
24. +5 В
25. EXTGPIO_0, *цифровой вход-выход общего назначения*
26. POT, *аналоговый вход общего назначения*

4.1.3 Двухосная система

4.1.3.1 Корпус

Двухосная версия контроллера представляет собой две *платы контроллера* в металлическом корпусе. Размеры корпуса: 122 x 45 x 106 мм.

На передней панели расположен *разъем силового питания*, *разъём для подключения к компьютеру тип USB type-B*, каскадный разъем USB. На передней панели также расположены светодиоды «статус контроллера», «левый концевик», «правый концевик», кнопки для движения вправо и влево для каждого из двух контроллеров двухосной системы.

На задней панели расположены *разъем подключения позиционера* для каждого из двух контроллеров, *дополнительный разъём двухосных системы* и разъем *подключения джойстика*.

Левая боковая панель содержит два Ethernet порта .

Каскадный разъем USB type-A и разъем Ethernet применяется для соединения нескольких двухосных конфигураций в цепочку, заканчивающуюся двухосным или одноосным контроллером. Таким образом, требуемое количество осей может быть подсоединено к компьютеру посредством одного кабеля.

4.1.3.2 Разъёмы

4.1.3.2.1 Разъем подключения позиционера

Для подключения позиционера используется разъем DSub 15 типа «мама».

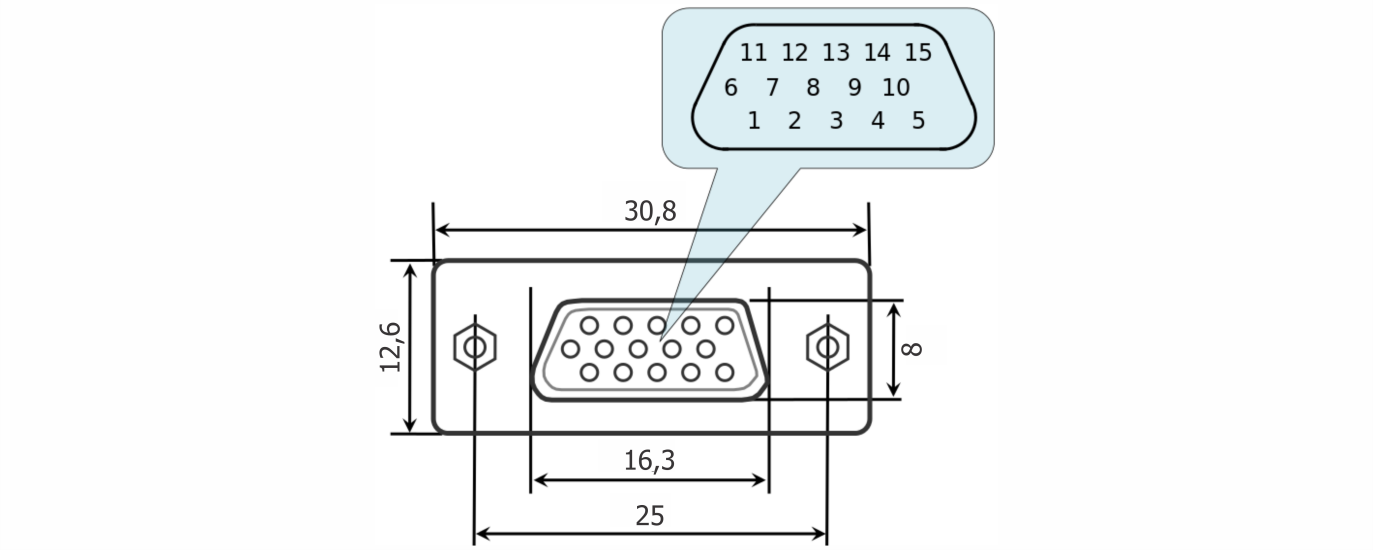


Рис. 4.9: Размеры и нумерация выводов в разъеме DSub, вид спереди

Назначение выводов:

1. ШД фаза не B или -DC мотора
2. ШД фаза B или +DC мотора или фаза B BLDC мотора
3. ШД фаза не A или -DC мотора или фаза C BLDC мотора
4. ШД фаза A или +DC мотора или фаза A BLDC мотора
5. Выход 5 В, 500 мА, стабилизированный выход для питания энкодера
6. ID, однопроводной интерфейс опознавания подвижки (работает только с позиционерами Standa)
7. Земля логическая для концевиков, энкодера и прочего
8. Концевик №2
9. Концевик №1
10. Энкодер A
11. Энкодер B
12. Вход датчика оборотов
13. Инверсный канал энкодера A
14. Инверсный канал энкодера B
15. Инверсный вход датчика оборотов

Примечание: BLDC моторы поддерживаются в прошивке версий 4.1.0 и старше.

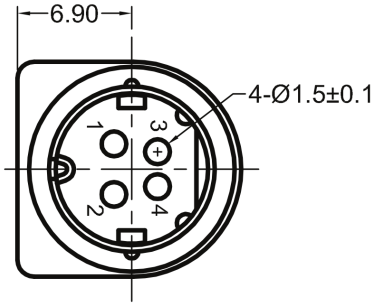
Примечание: Для подключения DC мотора с рабочим током выше 3A нужно соединить выходы 1 и 3, а также выходы 2 и 4.

Предупреждение: Не рекомендуется присоединять к/отсоединять от контроллера двигатель пока есть питание на обмотках мотора.

4.1.3.2.2 Разъем силового питания

Одноосные и двухосные системы, поставляемые в корпусе, используют разъём Kycon 4-pin DC power (part number по каталогу KPPX-4P, www.kycon.com).





Назначение выводов:

1. Силовое питание, '-'.
2. Силовое питание, '+'. 12-48В.
3. Силовое питание, '-'.
4. Силовое питание, '+'. 12-48В.

Важно: Никогда не подавайте электропитание на контроллер и не подключайтесь к разъему силового электропитания, если вы не уверены, что параметры вашего блока питания соответствуют требуемым. Не пытайтесь подключить электропитание к контроллеру если не уверены, что разъемы на блоке питания и на корпусе контроллера совместимы друг с другом! Допустимые параметры подключения указаны в разделе *Техника безопасности*.

Важно: Горячее присоединение/отсоединение, а также ненадежное подсоединение силового питания через разъём типа Kycon может вывести из строя контроллер и/или компьютер. Подробнее смотрите в разделе *Техника безопасности*.

4.1.3.2.3 Разъемы управления системой

Контроллеры подключаются через разъём USB type-B или Ethernet.



Рис. 4.10: Кабель USB type-A - USB type-B

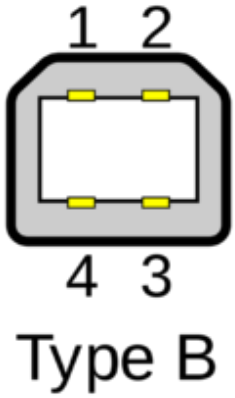


Рис. 4.11: Разъем USB type-B

Таблица 4.3: Таблица расположения выводов кабеля USB type-B

Номер вывода	Название	Цвет кабеля	Описание
1	VCC	Красный	+5 В DC
2	D-	Белый	Data -
3	D+	Зеленый	Data +
4	GND	Черный	Земля

Предупреждение: Используйте только проверенные и заведомо работоспособные USB-кабели! Неисправный или некачественный USB кабель может стать причиной неправильной работы контроллера, в том числе ошибок при вращении мотором или при опознавании устройства операцион-

ной системой. Для стабильного соединения используйте короткие кабели с толстыми проводниками и экранировкой.

4.1.3.2.4 Разъём подключения джойстика для 1- и 2-осной систем

Для подключения джойстика на корпусе одноосной или двухосной системы системы может быть установлен DSub 9 пин разъем типа «папа».

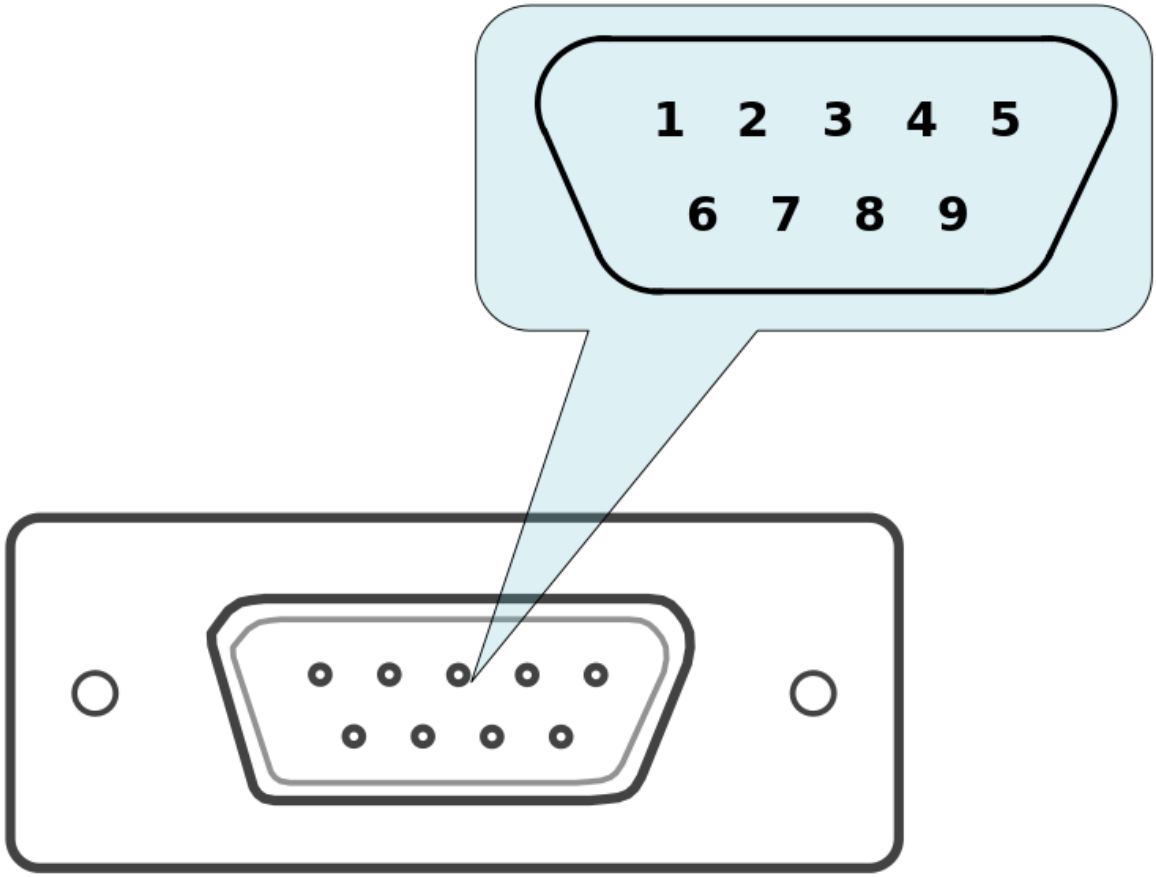


Рис. 4.12: Размеры и нумерация выводов в разъеме DSub, вид спереди

Назначение выводов:

1. AXIS1_BUT_R, вход для *кнопки «Вправо»*, ось 1.
2. AXIS2_Joy, аналоговый вход 0-3.3 В, используется для подключения внешнего *джойстика*, ось 2.
3. DGND, земля, общая для всех осей.
4. AXIS2_BUT_R, вход для *кнопки «Вправо»*, ось 2.
5. DGND, земля, общая для всех осей.
6. AXIS1_Joy, аналоговый вход 0-3.3 В, используется для подключения внешнего *джойстика*, ось 1.

7. Выход 3.3 В.
8. AXIS1_BUT_L, вход для *кнопки «Влево»*, ось 1.
9. AXIS2_BUT_L, вход для *кнопки «Влево»*, ось 2.

Примечание: Если разъем установлен на одноосной системе, то контакты 2, 4, 9 не используются.

Примечание: Неиспользуемые контакты внутреннего разъема не требуют никакого дополнительного подключения или подтяжки к земле/питанию. Просто не используйте их.

Важно: Аналоговые входы Joy, Pot рассчитаны на работу с напряжением ДО 3.3 В. Превышение напряжения может нарушить правильную работу всех аналоговых каналов контроллера и вывести из строя контроллер или двигатель.

4.1.3.2.5 Дополнительный разъём двухосной системы

На корпусе двухосной системы расположен порт HDB-26 типа «мама».

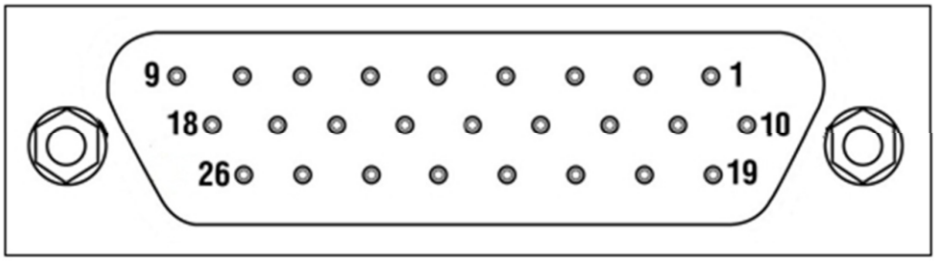


Рис. 4.13: Нумерация выводов в разъеме HDB-26, вид спереди

Назначение выводов:

1. AXIS_2_RX, вход последовательного порта второй оси
2. AXIS_2_SYNC_IN, вход синхронизации второй оси
3. AXIS_2_TX, выход последовательного порта второй оси
4. DGND, цифровая земля
5. AXIS_2_SYNC_OUT, выход синхронизации второй оси
6. AXIS_1_SYNC_IN, вход синхронизации первой оси
7. AXIS_1_SYNC_OUT, выход синхронизации первой оси
8. AXIS_1_RX, вход последовательного порта первой оси
9. NC, не используется
10. AXIS_2_CLOCK, сигнал clock для управления внешним драйвером, вторая ось
11. AXIS_2_POT, *аналоговый вход общего назначения*, вторая ось
12. AXIS_2_EXTGPIO_0, *цифровой вход-выход общего назначения*, вторая ось

13. AXIS_1_DIR_1, сигнал direction для управления внешним драйвером, вторая ось
14. DGND, цифровая земля
15. +5 В
16. AXIS_1_EMBRAKE, выход управления тормозом для первой оси
17. AXIS_1_CLOCK, сигнал clock для управления внешним драйвером, первая ось
18. AXIS_1_TX, выход последовательного порта для первой оси
19. AXIS_2_DIR_1, сигнал direction для управления внешним драйвером первой оси
20. AXIS_2_EMBRAKE, выход управления тормозом второй оси
21. AXIS_2_PBRK, выход магнитного тормоза второй оси
22. PGND, силовая земля
23. AXIS_1_PBRK, выход магнитного тормоза первой оси
24. +5 В
25. AXIS_1_EXTGPIO_0, *цифровой вход-выход общего назначения*, первая ось
26. AXIS_1_POT, *аналоговый вход общего назначения*, первая ось

4.2 Кинематика и режимы движения

4.2.1 Движение с заданной скоростью

Режим движения с заданной скоростью является основным режимом работы контроллера со всеми типами двигателей. Он позволяет поддерживать заданную скорость движения вдали от точки назначения, обычно применяется совместно с режимами *Движение в заданную точку* или *Смещение на заданное расстояние*. Этот режим может быть также вызван командами движения влево или вправо.

Скорость измеряется в шагах шагового двигателя или в отсчётах *энкодера* при его наличии (работает для всех типов двигателей) за единицу времени. Для DC-моторов постоянная скорость поддерживается даже при меняющейся внешней нагрузке.

Включение режима *Движение с ускорением* временно дезактивирует *режим движения с заданной скоростью*.

При получении команды для начала движения контроллер перемещает двигатель с заранее настроенной пользователем скоростью. Скорость может быть задана из *соответствующего раздела меню «Settings...» программы XILab* или с помощью вызова функции *set_move_settings()*, (см. раздел *Руководство по программированию*). **Значение скорости** при работе с шаговыми моторами можно задать в целых шагах и микрошагах в секунду, для DC-моторов скорость задается в оборотах в минуту (RPM).

Скорость специальных движений, таких как *компенсация люфта* или *автоматическая калибровка нулевой позиции*, отличается от общей скорости движения и настраивается отдельно.

Контроллер может ограничивать максимальную скорость, если соответствующая настройка сделана пользователем. В этом случае любое движение, которое бы происходило со скоростью выше максимальной, происходит с максимальной скоростью. Отдельно можно настроить контроллер, чтобы максимальная скорость использовалась для всех обычных движений, за исключением специальных, таких как *компенсация люфта* или *автоматическая калибровка нулевой позиции*. Настроить максимальную скорость и режимы её использования можно на *вкладке меню «Settings...» программы XILab*.

Текущую скорость вы можете увидеть в *Главном окне программы XILab* в поле *Speed* или воспользоваться *графиками отображения основных параметров* работы.

Примечание: *Замечание.* Если **стабильность поддержания скорости** при использовании *энкодера* кажется вам недостаточной, то обратитесь к *рекомендациям для точного движения*.

Примечание: Максимально допустимая скорость, которую можно задать это **100000 шагов/сек.** или **100000 оборотов/сек.** в зависимости от типа двигателя.

4.2.2 Движение в заданную точку

Режим *движения в заданную точку* является основным режимом работы контроллера со всеми типами двигателей, обычно используется совместно с *режимом движения с заданной скоростью*. Он позволяет перемещать позиционер в заданное положение, причем координата точки назначения имеет **абсолютное значение**, в отличие от режима смещения на *Смещение на заданное расстояние*. В режиме *Компенсация люфта* может производится дополнительное возвратно-поступательное движение вблизи заданной точки.

При использовании *энкодера* для определения текущей позиции, возможно несколько малозаметных «колебаний» прежде чем мотор остановится в заданной точке.

Также движение в заданную точку, для шаговых двигателей, может осуществляться в режиме обратной связи *Encoder mediated*. В этом случае движение осуществляется за несколько итераций с контролем положения по завершению каждой итерации по энкодеру, до попадания в заданную координату с определенной точностью.

При получении команды для начала движения контроллер либо переходит в *режим движения с ускорением*, при включении соответствующей настройки, либо сразу поворачивает ось двигателя с заранее настроенной пользователем скоростью. Движение останавливается при достижении заданной точки с переходом в *режим замедления*, если соответствующая настройка включена. **Позиция назначения** задается в *главном окне программы XiLab*. Позицию назначения можно задать в целых шагах и микрошагах при работе с шаговыми моторами, или отсчетах *энкодера* для всех двигателей.

Текущую позицию вы можете увидеть в *главном окне программы XILab* в блоке *Control* или воспользоваться *графиками* отображения основных параметров работы.

Примечание: Если **точность позиционирования** при использовании *энкодера* кажется вам недостаточной, то обратитесь к *рекомендациям для точного движения*.

4.2.3 Смещение на заданное расстояние

Режим смещение на заданное расстояние обеспечивает смещение для предопределенного значения относительно нуля, если это первая команда с момента запуска контроллера или относительно положения, достигнутого двигателем после завершения предыдущих команд, то есть координата назначения имеет **относительное значение**.

Этот режим полезен, когда абсолютное положение неизвестно или не имеет значения.

Примечание: Режим смещения на заданное расстояние может быть активирован как соответствующей командой, так и входным синхроимпульсом, подробнее см. раздел *ТТЛ-синхронизация*.

4.2.4 Движение с ускорением

Функция движения с ускорением активирована по умолчанию. Движение с ускорением используется для плавного начала и завершения движения без «толчков», обязательно сопровождающих мгновенный выход на заданную скорость. Кроме того, инерция ротора мотора и остальных компонентов позиционера обычно просто не позволяет мгновенно набрать высокую скорость, что приводит к потере шагов и срыву движения шагового двигателя в режиме работы без обратной связи. В режиме работы с обратной связью по энкодеру скорость будет набираться максимально быстро, как того позволяют *ограничители движения*. Быстрый набор скорости делает движение менее стабильным и создаёт больше шума и вибраций. Поэтому мы рекомендуем использовать движение с ускорением. Функция движения с ускорением позволяет достигать максимальных скоростей и стабильного движения даже на моторах со средним значением крутящего момента.

Режим движения с *ускорением/замедлением* работает следующим образом: при разгоне, когда требуемая скорость движения выше текущей по модулю, происходит постепенное ускорение движения на величину *Acceleration*, измеряемую в шагах на секунду в квадрате. При достижении требуемой скорости контроллер переходит в режим *движения с заданной скоростью*. При подходе к позиции назначения контроллер начинает снижать скорость движения так, чтобы замедление равнялось *Deceleration* и остановка произошла ровно в позиции назначения. Таким образом, этот режим обеспечивает трапецеидальный профиль скорости. Если расстояние, на которое требуется сдвинуться мало, то ускорение может непосредственно смениться замедлением, что приведёт к треугольному профилю скорости. Включение/отключение режима движения с ускорением, а также настройку величины ускорения и замедления можно сделать в программе XiLab (см. раздел *Настройка кинематики движения (Шаговый двигатель)*) или командой *set_move_settings()*, описанной в *Руководстве по программированию*.

Ускорение *Acceleration* настраивается независимо от замедления *Deceleration*. Это сделано неспроста. Обычно максимальное возможное ускорения меньше чем максимальное замедление из-за трения, которое препятствует ускорению, но способствует замедлению. Поэтому для максимально быстрого отклика позиционера нужно либо воспользоваться готовыми профилями, либо экспериментально подобрать те значения ускорения и замедления, которые способен обеспечить Ваш позиционер. Для шаговых двигателей, работающих без обратной связи это те значения, при которых не происходит потери шагов. Для двигателей с обратной связью нужно проконтролировать трапецеидальность скорости на *графиках XiLab*. Стоит брать значения ускорения/замедления в полтора-два раза ниже тех, где наблюдаются искажения профилей скорости или потери шагов.

Примечание: Отключение ускорения/замедления может быть полезно для управления многоосными системами, где движение по многомерным траекториям требует постоянной проекции скорости на каждую из осей.

Примечание: В *Главном окне программы XILab* не отображается значение ускорения.

Примечание: Устанавливаемое ускорение/замедление должно быть рассчитано так, чтобы обеспечить выход на требуемую скорость или замедление с максимально возможной скрости не более чем за **5 мин.** Если установить ускорение/замедление не придерживаясь этого правила в *настройках кинематики* движения, то контроллер вернет ошибку связанную с попыткой задания недопустимых значений, а значение ускорения/замедления будет изменено в контроллере для попадания в допустимый диапазон.

4.2.5 Компенсация люфта

В любом механическом устройстве, например редукторе или червячной передаче, существует люфт, наличие которого приводит к тому, что при подходе к одной и той же позиции с разных сторон реальное положение позиционера будет отличаться, при том, что ось мотора находится точно в заданном положении.

Для устранения такой неоднозначности используется режим компенсации люфта, активация которого позволяет пользователю выбрать, с какой стороны нужно приближать позиционер к заданной точке. В дальнейшем при любых движениях позиционер будет подходить к точке останова только с выбранной стороны, устраняя механический люфт. Если естественное направление подхода к заданной точке не совпадает с выбранным направлением подхода, то контроллер заводит двигатель на некоторое расстояние, определяемое пользователем, за заданную точку, разворачивает двигатель и завершает подход к заданной точке с требуемой стороны.

При движении нагруженной механической системы в зоне люфта её динамические характеристики отличаются от обычного движения. Поэтому движение в зоне люфта выполняется с задаваемой пользователем скоростью.

Пользователь может настраивать следующие параметры системы компенсации люфта:

- Флаг включения/выключения компенации люфта.
- Скорость движения при выполнении компенсационного движения.
- Расстояние, на которое достаточно проехать, чтобы коомпенсировать люфт. Знак этой настройки определяет направление подхода. Положительный знак означает подход слева, а отрицательный - справа.

Контроллер сигнализирует о моментах, когда происходит отработка компенсации люфта в структуре состояния с помощью флага MOVE_STATE_ANTIPLAY. Он также выводится на *главное окно XiLab*.

Если нет уверенности, что текущее положение свободно от люфта, то можно совершить вынужденную компенсацию люфта с помощью команды *LOFT*. При выполнении этой команды в режиме остановки происходит сдвиг из текущей позиции на расстояние компенсации антилюфта и возвращение назад. Вызов данной команды во время движения приведет к плавной остановке двигателя. Применение этой команды имеет смысл только при включенной системе компенсации антилюфта.

Примечание: *Режим компенсации люфта* не предусматривает никакой коррекции положения оси, он только позволяет пользователю выбрать, с какой стороны нужно приближать позиционер к заданной точке и всегда придерживаться указанного направления подхода.

Настройка компенсации люфта в программе XILab описана в *настройках кинематики* движения шагового двигателя. Команды включения и определения параметров компенсации люфта описаны в *руководстве по программированию*.

Люфт будет минимальным в том случае, если подход к заданной точке осуществляется с одинаковыми параметрами движения, поэтому оптимальными будут являться следующие значения параметров: скорость в зоне люфта должна быть равна номинальной, расстояние компенсации антилюфта должно быть таково, чтобы устройство успевало набрать номинальную скорость.

По умолчанию в профилях двигателей компенсация люфта задаётся по формуле:

$$S = \frac{U^2}{2}\left[\frac{1}{Ac} + \frac{1}{Dc}\right] + 0.2U$$

где S - компенсация люфта, U - номинальная скорость, Ac, Dc - ускорение и замедление, 0.2 - время, в течение которого двигатель будет ехать с постоянной скоростью.

4.2.6 Реверсирование движения

Считается, что рост координаты соответствует движению вправо, а убывание координаты - движению влево. Если физически позиционер расположен так, что это правило не выполняется, либо на позиционере нанесен репер, направление возрастания которого не совпадает с ростом координаты, то необходимо реверсировать движение.

Реверс движения можно включить в *меню XiLab* в блоке Motor parameters. Включение этой настройки поменяет знак текущей координаты и понятия «влево» и «вправо» поменяются местами. Например, первое движение при *калибровке нулевой позиции*, будет выполняться теперь физически в другую сторону, команды *влево* и *вправо* на *главном окне XiLab* поменяются местами и т. п.

Предупреждение: Реверс относится к настройкам, изменение которых сказывается на всей работе контроллера. Работавшие ранее *скрипты XiLab* или *собственные программы управления* будут вести себя по иному.

В частности *концевые выключатели* настраиваются независимо от реверса движения. При включении или отключении реверса концевые выключатели обязательно нужно перенастроить.

4.2.7 Рекомендации для точного движения

Контроллер способен автоматически подстраиваться под необходимый режим, будь то поддержание скорости или координаты. Но скорость и качество подстройки зависят от настроек контроллера. Практически мгновенно выходить на требуемый режим способен шаговый двигатель в режиме позиционирования по шагам и микрошагам. Если шаговый двигатель физически неспособен обеспечить требуемую скорость или ускорение, то скорее всего движение вовсе остановится (сорвётся). При использовании датчика обратной связи, такого как квадратурный энкодер, в качестве опорного, движение шагового двигателя не сорвётся, но возможно, что контроллер не сможет обеспечить требуемые параметры движения.

DC-моторы требуют использования дополнительного датчика положения (энкодера). Непрямая связь между воздействием управляющей схемы и смещением положения позиционера с DC-мотором приводит к замедлению выхода на требуемую координату или на требуемую скорость. Ускорить этот процесс и сделать его более стабильным можно используя следующие рекомендации:

- В контроллер загружен и используется профиль, соответствующий используемому позиционеру, если вы не уверены, что профиль верный, загрузите профиль из раздела Конфигурационные файлы.
- Мотор не попадает в режим ограничения одного из рабочих параметров (ток или напряжение), см. подробнее разделы *Ограничители на двигателях*, *Управление питанием мотора*. Такое ограничение вы можете заметить по горизонтальной полоске над индикатором *Current* в блоке *Power*, *Voltage* или *Speed* в блоке *Motor* *Главное окно программы XILab в режиме управления одной осью*, подробнее см. разделы *Ограничители на двигателях*, *Управление питанием мотора*.
- Отсутствуют механические препятствия для движения, заклинивание оси или позиционера.
- Мощность применяемого блока питания достаточна (см. *Техника безопасности*).

4.2.8 PID алгоритм для управления DC двигателем

4.2.8.1 Описание алгоритма

Управление DC двигателем осуществляется с помощью PID регулятора. Регулируемой величиной является координата. Для обеспечения возможности движения, сама регулируемая координата изменяется в соответствии с установленными настройками движения и поступившими командами. Изменяющу-

юся во времени регулируемую координату далее будем называть бегущей позицией. Управляющим сигналом регулятора является фактор заполнения ШИМ сигнала, подаваемого на обмотку двигателя.

Формула для вычисления управляющего воздействия:

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

$U(t)$ - управляющее воздействие

$E(t)$ - разница между бегущей координатой и текущей координатой двигателя

K_P, K_I, K_D - пропорциональный, интегральный и дифференциальный коэффициенты регулятора. Коэффициенты регулятора задаются с помощью соответствующего *меню программы* XiLab или с помощью вызова функции set_pid_settings(), (см. раздел *Руководство по программированию*).

Для того, чтобы результат работы PID регулятора не был зависим от двигателя, датчика обратной связи и текущего напряжения питания, производится нормировка полученного значения по следующей формуле:

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

$DC(t)$ - фактор заполнения ШИМ

U_{nom} - номинальное(максимальное) напряжение питания двигателя, (см. раздел *Ограничители на двигателях*).

$U_{supp}(t)$ - текущее напряжение питания

IPS - разрешение датчика обратной связи в отсчетах/оборот

Данный подход позволяет менять двигатель, датчик обратной связи и источник питания без перенастройки PID регулятора.

Предупреждение: Если необходимо изменить настройки максимального напряжение питания мотора, не забудьте изменить *коэффициенты PID регулятора* в соответствии с приведенной выше формулой.

4.2.8.2 Особенности работы алгоритма

4.2.8.2.1 Коэффициенты PID регулятора

Для того, чтобы оптимальные коэффициенты PID регулятора не выходили из диапазона [0..65535], *задаваемые пользователем значения* нормируются.

Для лучшего понимания работы регулятора, рассмотрим какое влияние оказывают различные составляющие.

Будем считать, что напряжение питания $U_{supp}(t)$ постоянно и равно номинальному напряжению мотора U_{nom}. При выполнении данного предположения фактор заполнения ШИМ сигнала будет равен 1 в следующих случаях:

1. $K_p = 1,\ K_I = 0,\ K_D = 0$ - если целевая позиция превышает реальную позицию на 256 оборотов ротора
2. $K_p = 0,\ K_I = 1,\ K_D = 0$ - если интеграл, приведенный в формуле выше, равен 52,5 оборотовcdotсек
3. $K_p = 0,\ K_I = 0,\ K_D = 1$ - если реальная скорость вращения ротора отличается правильной скорости на 96000 об/мин.

4.2.8.2.2 Попадание в целевую позицию

Попадание в целевую позицию считается успешным как только ось двигателя попадает в целевую позицию. При этом, возможно наличие некоторых переколебаний около целевой позиции. Если движение производится без ускорения, пришла команда немедленной остановки или происходит экстренная остановка по достижению концевого датчика, то мотору понадобится некоторое время до полной остановки и возвращения в правильную позицию.

Предупреждение: Если PID регулятор настроен неправильно, возможно возникновение длительных колебаний около целевой позиции, хотя движение и будет считаться завершённым.

4.2.8.3 Рекомендации по настройке PID регулятора

При настройке PID регулятора следует пользоваться тремя критериями качества его работы:

- Точность поддержания скорости - определяется средним отклонением скорости от требуемой. При этом, если во время движения скорость принимает не более 3 различных значений, можно считать, что режим поддержания скорости работает оптимальным образом. Более аккуратное поддержание требуемой скорости не представляется возможным из-за эффекта квантования значения скорости.
- Качество попадания в целевую позицию определяется следующими критериями:
  - Время до окончательной остановки в целевой позиции.
  - Отсутствие проскока целевой позиции при подходе к ней.
  - Отсутствие нескольких переколебаний около целевой позиции перед остановкой в ней.
  - Отсутствие самопроизвольных отклонений около целевой позиции после остановки в ней.
- Отсутствие шумов при работе. Шум увеличивается только от увеличения одного из трёх коэффициентов PID.

При настройке PID регулятора каждый может сам выбирать приоритет отдельных критериев качества в зависимости от решаемой задачи.

1. Приступая к настройке PID регулятора рекомендуется отключить все ограничения работы DC двигателя, в том числе ускорение.
2. Настройку регулятора лучше начинать с режима поддержания скорости.
3. Настройку PID регулятора следует начинать с нулевых значений интегрального и дифференциального коэффициентов.
4. Устанавливаем значение пропорционального коэффициента $K_P \leq 10$
5. Устанавливаем требуемую скорость движения и начинаем движение вправо или влево вдали от концевых выключателей.
6. Постепенно увеличиваем значение K_P и наблюдаем за графиком текущей скорости. Таким образом получаем оптимальное значение K_P при котором скорость движения поддерживается наилучшим образом и время выхода на неё перестаёт заметно уменьшаться с увеличением K_P. При этом следует обращать внимание на возможный рост шума.
7. После настройки значения K_I стоит приступить к настройке значения K_I. Интегральный коэффициент PID регулятора в большей степени влияет на процесс попадания в целевую позицию.
8. Устанавливаем значение интегрального коэффициента $K_I \leq 10$ и приступаем к движениям к позиции. Удобнее всего использовать команду смещения на заданное расстояние.

9. Лучше всего, чтобы при движении к позиции двигатель успевал разогнаться хотя бы до 20% от требуемой скорости.
10. Увеличение значения K_I приводит к более быстрой остановке в целевой позиции.
11. Постепенно увеличивая значение K_I добиваемся быстрого попадания в целевую позицию. Увеличение коэффициента следует прекратить, когда заметно снизится качество попадания в позицию (переколебания, осцилляции) или вырастут шумы. При этом, проскок позиции будет наблюдаться всё равно.
12. Для улучшения качества попадания в позицию понадобится изменение K_P в большую или меньшую сторону в зависимости от того где переколебаний меньше.
13. После подстройки K_P стоит проверить качество поддержания постоянной скорости и уровень шумов. При неудовлетворительных показателях качества коэффициенты подстраиваются в направлении предыдущих значений.
14. После настройки коэффициентов K_I и K_P можно приступить к настройке коэффициента K_D.
15. Настройку коэффициента K_D лучше начинать в режиме поддержания скорости, проверяя затем остальные параметры качества.
16. Коэффициент K_D увеличивается пока заметно уменьшение колебаний скорости около требуемого значения.
17. В случае наличия проскока позиции коэффициент K_D увеличивают. Однако дальнейшее увеличение приводит к колебанием около целевой позиции. Необходимо соблюсти баланс между скоростью попадания и отсутствием колебаний.

Если планируется работа с включенным ускорением, то его необходимо включить. При этом может возникнуть проскакивание целевой позиции. Для компенсации этого эффекта необходимо увеличение значения K_I.

На этом этапе первичная настройка PID контура завершена. Полученные коэффициенты в большинстве случаев годятся для работы. Для дальнейшей оптимизации коэффициентов они вариируются с постоянным контролем качества по выбранным критериям скорости, позиции и шума с учетом их важности в конкретной задаче.

Примечание: Не рекомендуется одновременно изменять более одного коэффициента при настройке PID регулятора.

Важно: ВАЖНО. Крайне не рекомендуется устанавливать сразу большие значения коэффициентов PID регулятора или резко их изменять. Это может привести к самовозбуждению колебаний скорости на вашем двигателе и привести к поломке.

4.2.9 Feedback EMF

4.2.9.1 Преимущества

- Всегда сохраняет синосуидальную форму тока, что обеспечивает бесшумную работу;
- На высоких скоростях он может динамически адаптироваться к внешним нагрузкам, ограничениям тока и напряжения (автоматически снижает скорость);
- На низких скоростях использует частотное управление без контроля позиции ротора. Когда положение ротора начинает выдавать корректные показания алгоритм переключается на полеориентировнное управление с обратной связью по позиции ротора. Порог переключения индивидуален

для каждого двигателя, он определяется качеством оценки выдаваемой наблюдателем позиции ротора;

- Не использует датчик позиции (encoder);
- Может работать в трех режимах:
  - **MTPA** - наиболее экономичный режим, характеризующийся минимальным током, но напряжение быстро растет со скоростью, $I_d = 0$;
  - **FW** - режим понижения потокосцепления, активен, когда заданной скорости невозможно достигнуть при текущих ограничениях напряжения, используя MTPA, $I_d < 0$;
  - **Limit** - режим насыщения, когда движение с заданной скоростью невозможно. Возникает при насыщении напряжения и силы тока. В нем привод выдает максимальный момент определяемый текущей скоростью и ограничениями по току и выставляет флаг *PowerLimited*.

Важно: Алгоритм не должен использоваться с включенным флагом «Position Control». Для плавности хода в EMF алгоритме реализовано расхождение с реальной позиции и позиции по профилю. Если флаг «Position Control «включен, могут быть вызваны ложные срабатывания Alarm. Чтобы избежать ложных срабатываний следует во вкладке «Position control» поставить галочку «Position Control» и в поле «Threshold» указать допустимую ошибку. Тогда при рассогласовании позиции, в главном окне XiLab будет загораться индикатор SLIP и, если стоит галочка «Alarm on errors», контроллер будет переходить в состояние Alarm.

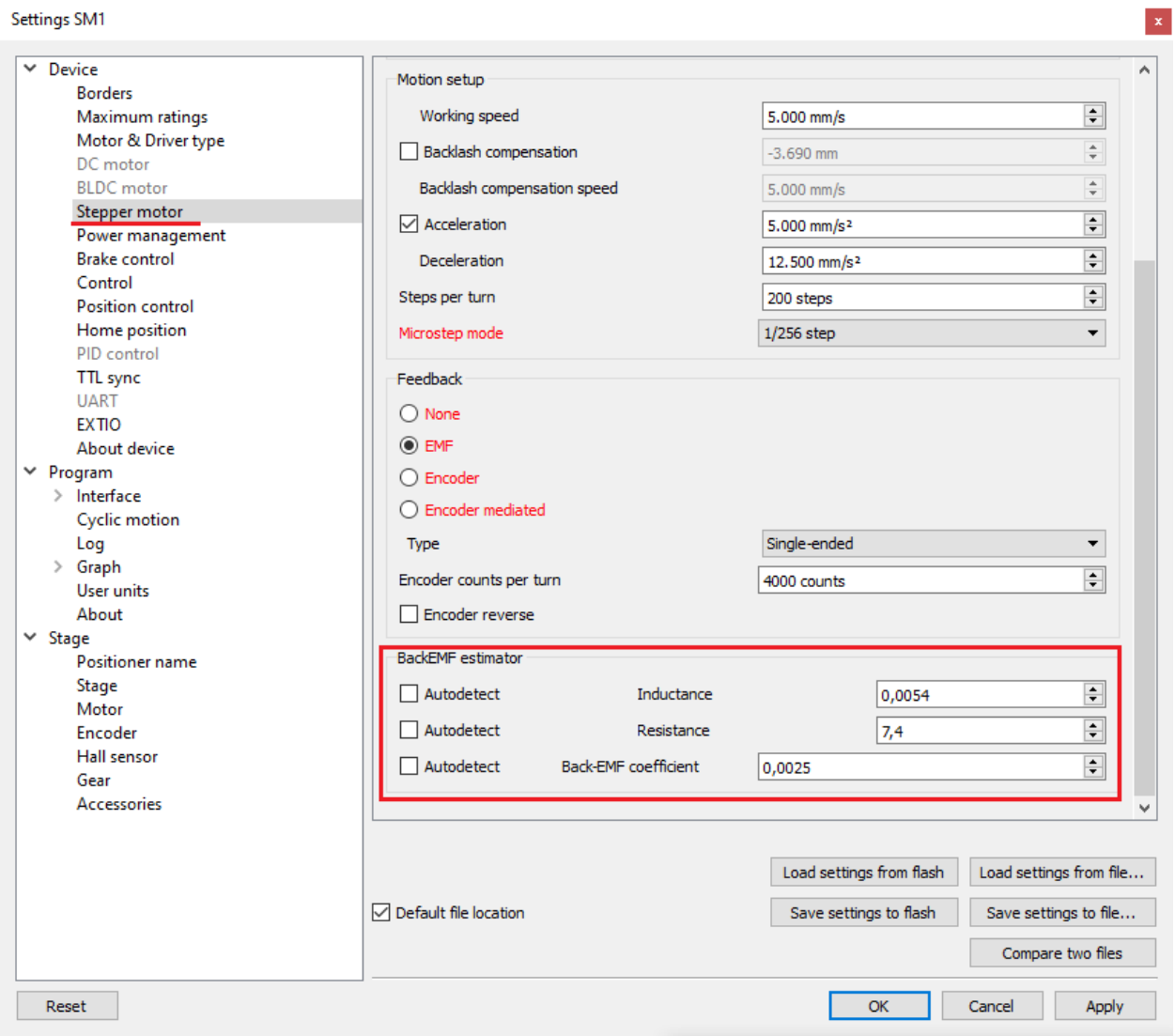
4.2.9.2 Поведение двигателя при воздействии внешней силы

В режиме частотного управления: - Положение ротора не контролируется, но ток равен номинальному значению. Только внешняя сила, превышающая момент удержания, может привести к потере шагов

В режиме полеориентированного управления: - Если сила может быть преодолена, то движение продолжается с заданной скоростью. - Если сила не может быть преодолена, то устанавливается флаг PowerLimited и установленное значение скорости начинает уменьшаться в соответствии со значением замедления, уставка положения, определяемая логикой генератора профиля скорости (интеграл от скорости), изменяется соответственно; - Если действие силы прекращается, то расхождение между уставкой и реальной позицией будет компенсировано за счет PID регулятора позиции; - Если сила не может быть преодолена приводом (превышает удерживающую силу), то на пороге скорости происходит переключение с последующим отказом ротора и потерей шагов.

4.2.9.3 Выбор параметров L, R и backEMF для алгоритма EMF

После включения питания (переключения из состояния Power: Off) и перед началом перемещения параметры R и L автоматически определяются (слышен короткий звуковой сигнал). Если эти параметры устанавливаются через интерфейс xilab, этап пропускается. Для повторной оценки параметров двигатель должен быть возвращен в состояние Power: off.



На вкладке *Stepper motor* можно дополнительно назначить следующие параметры:

1. Resistance - сопротивление обмотки R Ω
2. Inductance - индуктивность обмотки L Н
3. Back EMF coefficient -потокосцепление ротора λ_m Нм/А
4. Значение всех этих параметров сохраняется в профиле для двигателя.

Для большинства двигателей удовлетворительно работает автоопределение параметра *back-EMF parameter*. Однако назначение параметров через профиль обеспечивает большую устойчивость алгоритма.

Значения R Ω и L Н следует брать непосредственно из datasheet.

Значение потокосцепления ротора λ_m Нм/а может быть получено следующим образом:

В datasheet дано значение электромеханического коэффициента двигателя K_m (torque constant, Нм/А) или коэффициента противоЭДС K_{emf} (backEMF constant, Vs)„ тогда

$\lambda_m = \frac{4K}{n}$, где K значение K_{emf} или K_m, а n - число шагов на оборот.

В datasheet дана номинальная сила тока i_n (nominal current, A) и момент удержания T_n (holding torque, Hm), тогда $\lambda_m = \frac{4T_n}{I_n n}$,где n - число шагов на оборот.

4.2.9.4 Выбор коэффициентов PID для EMF

В режиме полеориенированного управления, когда доступна оценка положения ротора, управление позицией осуществляется при помощи стандартного PID-регулятора. Его коэффициенты обеспечивают устойчивость двигателя в области высоких скоростей.

Точность позиции по профилю определяется многими факторами:

- В режиме удержания - соотношением силы удержания и мешающей силы, так же как у всех алгоритмов открытого контура;
- На низких скоростях - расхождение реальной позиции и позиции по профилю не должно превышать одного шага;
- На высоких скоростях расхождение может составлять несколько шагов, что обусловлено переходными процессами, коэффициентами обратной связи и наличием внешних сил.

Мы предлагаем использовать стандартный набор коэффициентов: $K_p = 3.6, \quad K_d = 0.028, \quad K_i = 38$

- K_p увеличение коэффициента увеличивает точность (уменьшает ошибку θ_e), уменьшает время регулирования;
- K_d увеличение коэффициента, увеличивает демпфирование системы и снижает вибрации. Неустойчивость может быть следствием слишком маленького значения K_d;
- K_i мало влияет на точность в переходных режимах, но уменьшает постоянную ошибку при движении с постоянной скоростью и ускорением, а также позволяет компенсировать постоянные внешние силы. Во многих приложениях может быть принят нулевым. **Слишком большое значение приводит к потери устойчивости**.

4.2.9.4.1 Алгоритм работы

Входом регулятора является is:

- θ_r - желаемое положение ротора (рад), значение формируется генератором профиля скорости
- ω_r - желаемая угловая скорость ротора (рад/с), значение формируется генератором профиля скорости
- θ_m, ω_r - положение ротора (рад) и его скорость (рад/с), вычисленные оценщиком положения ротора

Выход: I_{qr} - значение тока (A), определяющее момент создаваемый двигателем: $M = k_m I_{qr}$, где k_m электромеханический коэффициент (torque constant) двигателя

Параметры: K_p, K_i, K_d - коэффициенты обратной связи (значения заданные во вкладке *«PID Control»*).

Закон управления: 1. Вычисление ошибки регулирования: $\theta_e = \theta_r - \theta_m, \quad \omega_e = \omega_r - \omega_m$, 2. Расчет силы тока: $I_{qr} = K_p\theta_e + K_d\omega_e + K_i \int_0^t \theta_e d\tau$

Регулятор снабжен контуром противонакопления, на основе алгоритма условного интегрирование. Рост интегральной части прекращается, если произошло насыщение силы тока ($|I_{qr}| > I_{max} I_{qr} \cdot \theta_e < 0$)

4.2.10 Feedback encoder

В этом случае все параметры мотора, в том числе положение и скорость движения, измеряются непосредственно с помощью энкодера и имеют размерности, основанные на отсчетах энкодера. Положение отображается непосредственно в отсчетах энкодера, скорости выражаются в RPM - оборотах в минуту.

Скорость движения рассчитывается контроллером на основе данных об измеренной скорости и значении количества импульсов энкодера на один полный оборот оси мотора, указанных в блоке настроек обратной связи во вкладке Настройка кинематики движения (*Шаговый двигатель*), (*DC мотор*). Отметим, что в случае использования DC мотора *режимы поддержания заданной скорости*, *движения в заданную точку* и все производные от них работают с помощью алгоритмов PID-регулирования и требуют *соответствующих настроек*. Для шаговых двигателей режим ведущего энкодера оптимизирует управление двигателем, за счет чего уменьшаются шумы при движении, стабильно проходятся резонансные скорости, достигается большая скорость вращения по сравнению с режимом работы без энкодера, без риска потери шагов, при которых сбивается координата и требуется повторная *калибровка*.

Новый алгоритм доступен с прошивкой 4.0.7+. Используйте XiLab 1.13.13+ установите параметр Feedback в Encoder в настройках Device -> *Stepper motor page*.

4.2.11 Feedback encoder mediated

Данный режим работы оптимально использовать в системах с большими люфтами для попадания в необходимую координату за несколько итераций, в среднем от 3 до 10. Этот режим используется для шаговых двигателей.

Для этого режима все параметры мотора, в том числе положение и скорость движения, задаются и отображаются непосредственно с помощью энкодера и имеют размерности, основанные на отсчетах энкодера. Однако в контроллере само движение осуществляется в шагах.

Принцип работы: Полученные от внешнего интерфейса значения координат пересчитываются в шаги и производится первая итерация движения,по завершению которой осуществляется проверка положения по энкодеру. Далее определяются значения отклонения для новой итерации и осуществляется новый цикл подъезда. Это происходит до момента попадания с заданной точностью в указанную координату.

Новый алгоритм доступен с прошивкой 4.3+. Используйте XiLab 1.16+ установите параметр Feedback в Encoder mediated в настройках Device -> *Stepper motor page*.

4.2.12 Режимы остановки движения

В контроллере существует два режима прекращения движения:

- Немедленная остановка;
- остановка с замедлением.

4.2.12.1 Немедленная остановка

Немедленная остановка движения происходит по команде *STOP*. Контроллер старается мгновенно остановить вращение вала двигателя. Это может привести к пропуску шагов в шаговом двигателе, если не используется обратная связь. Резкое прекращение движения может пагубно влиять на оборудование, например, может произойти сдвиг образцов на предметных столиках микроскопов или потребовать дополнительной юстировки оптической линии после резкой остановки.

Предупреждение: Когда контроллер настроен на остановку по срабатыванию левого/правого *концевого выключателя*, то всегда происходит экстренная остановка при достижении концевого выключателя. Этого надо избегать.

4.2.12.2 Остановка с замедлением

Остановка движения с замедлением выполняется по команде *SSTP*. В этом режиме происходит плавная остановка с замедлением *Deceleration*, если оно не отключено в настройках *движения с ускорением* .

Предупреждение: Если *движение с ускорением* отключено, то разницы между режимами остановки с замедлением и немедленной остановки не будет. По команде *SSTP* будет происходить резкая остановка.

4.2.13 PID алгоритм для управления BLDC двигателем

4.2.13.1 Описание алгоритма

Управление BLDC двигателем осуществляется с помощью PID регулятора. Регулируемой величиной является координата. Для обеспечения возможности движения, сама регулируемая координата изменяется в соответствии с установленными настройками движения и поступившими командами. Изменяющуюся во времени регулируемую координату далее будем называть бегущей позицией. Управляющим сигналом регулятора является модуль вектора тока, который (вектор) удерживается перпендикулярно ротору. Формула для вычисления управляющего воздействия:

$$U(t) = I + P + D = K_p \cdot E(t) + K_i \int E(t)dt + K_d \frac{dE(t)}{dt}, where:$$

$U(t)$- управляющее воздействие

$E(t)$- разница между бегущей координатой и текущей координатой двигателя

K_p, K_i, K_d - коэффициенты усиления пропорциональной, интегральной и дифференциальной составляющих регулятора, соответственно. Коэффициенты регулятора задаются с помощью *соответствующего меню* программы XiLab или с помощью вызова функции set_pid_settings(), (см. раздел *Руководство по программированию*).

PID-коэффициенты в алгоритме управления BLDC имеют тот же смысл, что и в управлении DC. См. влияние различных составляющих PID, рекомендации по настройке и замечания в главе *PID алгоритм для управления DC двигателем*.

4.2.13.2 Ручная настройка коэффициентов PID-регулятора

Для тонкой настройки коэффициентов PID-регулятора существует специальное окно программы XiLab. В окне выводится зависимость от времени скорости BLDC-двигателя и ошибки следования соответсвующей координате, вид окна показан на скриншоте ниже.

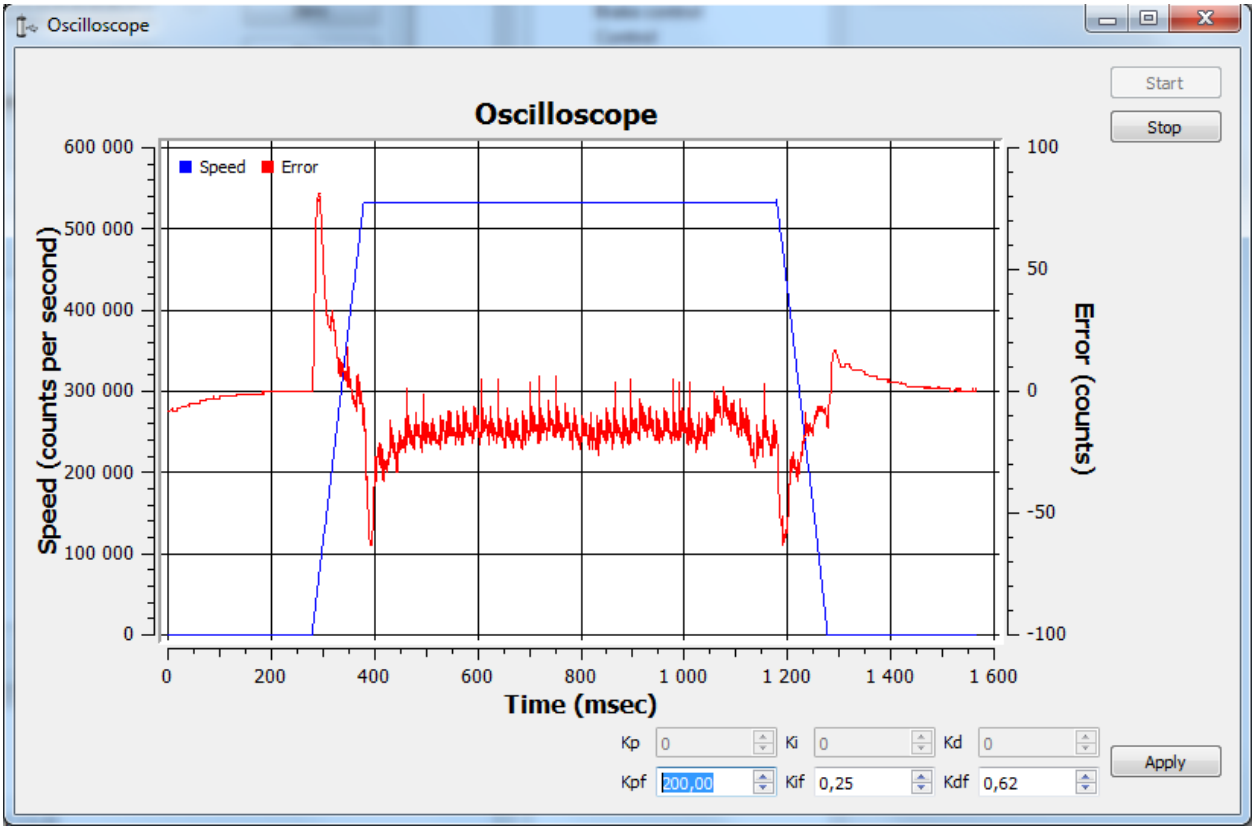


Рис. 4.14: Окно настройки PID регулятора.

Для корректной работы двигателя нужно добиться устойчивой компенсации ошибки следования.

4.2.13.2.1 Шаги по настройке коэффициентов:

1. Для начала нужно оценить PID-коэффициенты. Учитывая структуру управляемой системы, их можно вычислить по упрощенным формулам. Для этого используются параметры из документации на соответствующий двигатель и позиционер.

   - K_m - электромеханический коэффициент двигателя [Н / А] (момент создаваемый силой тока 1 А). Может быть вычислен как отношение $K_m = \frac{F_n}{I_n}$, где F_n - номинальная (максимальное) усилие создаваемое двигателем, I_n - номинальная (максимальная) сила тока.
   - M - масса нагрузки (кг).
   - $\sigma = \frac{M}{K_m}$.

   Пример для подвижки 8MTL1301-170:

   - Загружаем профиль для этой подвижки. Из него узнаем дефолтные PID-коэффициенты: *Kp 200 000, Ki 250, Kd 625.*
   - *Коэффициент km* рассчитывался по формуле выше, для данного позиционера он равен *12,9.*
   - Используя формулу посчитаем PID коэффициенты для массы *4,7 кг.* $\sigma = 4.7/12.9 = 0,3643$.
   - Значение σ, которое мы нашли ранее по формуле, умножаем на дефолтные PID коэффициенты из профиля и получаем:

- $200000 \cdot 0,3643 = 72860$;
- $250 \cdot 0,3643 = 91,075$, округляем до 91;
- $625 \cdot 0,3643 = 227,6875$, округляем до 228.

2. Выставить коэффициенты, рассчитанные по формулам, нажать Apply. В *главном окне XiLab* нажать кнопку Zero. В поле Move to выставить 0, отправить команду Move to. Двигатель должен остановиться. Попробовать сдвинуть позицию руками, убедиться, что отклик правильный - двигатель старается вернуться в нулевую позицию (реверс энкодера настроен верно).
3. В *настройках движения*, выставить маленькую скорость, нажать Apply. В *главном окне* начать движение в сторону. Если начинаются вибрации и срывы, нужно увеличивать дифференциальный коэффициент (Kdf), увеличивать, пока заметно уменьшение колебаний скорости около требуемого значения.
4. Если вибрации имеют звуковые частоты (позиционер издаёт громкий звук при движении), возможно, следует уменьшить коэффициент Kd или все коэффициенты пропорционально.
5. Интегральный коэффициент (Kif) отвечает за попадание в целевую позицию, для проверки удобно использовать команду Shift on.
6. Для более точной настройки коэффициентов используйте окно Oscilloscope, в этом окне визуализируется ошибка следования для данных параметров движения. Чтобы открыть окно нужно нажать кнопку PID tuning.
7. После того, как коэффициенты настроены, можно их пропорционально менять, это соответствует увеличению/уменьшению массы, отклик на воздействие становится более/менее мощным. Добиться того, чтобы резкие остановки не приводили к срывам движения.

4.3 Основные возможности контроллера

4.3.1 Поддерживаемые типы двигателей

В настоящий момент контроллер поддерживает шаговые и DC двигатели. Характеристики поддерживаемых двигателей можно посмотреть в разделе *Технические характеристики* .

4.3.1.1 Шаговые двигатели

Основным параметром шагового двигателя является его номинальный ток. Номинальный ток двигателя можно установить во вкладке *Настройка кинематики движения (Шаговый двигатель)*.

Важно: Установка завышенного тока постепенно приведёт к перегреву двигателя и его физической поломке. Обязательно проконтролируйте, чтобы был установлен номинальный ток, соответствующий используемой подвижке. В предустановленных профилях подвижек все настройки уже сделаны правильно.

Другим важным параметром является режим деления шага. В полношаговом режиме двигатель перемещается на величину угла шага (например, двигатель с шагом 1,8° совершает 200 шагов за один полный оборот). В режиме деления шага основной шаг двигателя может делиться до 256 раз. Деление шага улучшает гладкость перемещений и минимизирует эффекты резонанса на низких скоростях.

Доступны следующие режимы деления шага:

- 1 (полный) шаг
- 1/2 шага
- 1/4 шага

- 1/8 шага
- 1/16 шага
- 1/32 шага
- 1/64 шага
- 1/128 шага
- 1/256 шага

Режим микрошага устанавливается во вкладке *Настройка кинематики движения (Шаговый двигатель)* или командами настройки мотора, см. раздел *Описание протокола обмена* и описание соответствующих функций в разделе *Руководство по программированию*.

Примечание: Контроллер всегда использует внутреннее деление шага 1/256. При смене пользователем деления шага на более грубый, в ПО отображаются только кратные более грубому делению позиции, установка и передача становится возможна только в таких, более грубых делениях. Это сделано для поддержки устаревшего и совместимости с уже существующим ПО, работающим на делениях шага малой кратности. С другой стороны, работа на наибольшем делении шага позволяет двигаться наиболее плавно и тихо на малых скоростях.

Еще один непосредственным параметром шагового двигателя является количество полных шагов на оборот. Эта настройка не влияет на движение, но используется в блоке *контроля проскальзывания* или при работе с обратной связью по *энкодеру*.

Примечание: Контроллер поддерживает шаговые двигатели с датчиком обратной связи - энкодером. *Энкодер* может использоваться как основной датчик положения (*подробнее*) или для обнаружения проскальзывания, люфта или потери шагов (*подробнее*). Использование энкодера способствует стабильному прохождению резонансных скоростей без срыва движения.

4.3.1.2 DC двигатели

В отличии от шагового двигателя, для управления DC двигателем контроллеру необходимо наличии обратной связи позиции двигателя. В настоящее время, в качестве датчика обратной связи поддерживается только *энкодер*.

Основными параметрами двигателя являются максимальный ток и напряжение, которые устанавливаются во вкладке *Настройка кинематики движения (DC мотор)*. Основным параметром *энкодера* является количество отсчетов на оборот.

Важно: Установка завышенного тока постепенно приведёт к перегреву двигателя и его физической поломке. Обязательно проконтролируйте, чтобы был установлен номинальный ток, соответствующий используемой подвижке. В предустановленных профилях подвижек все настройки уже сделаны правильно.

Важно: Установка неправильного значения количества отсчетов энкодера на оборот приведет к тому, что установленное значение скорости не будет выдерживаться правильно. В некоторых случаях это может привести к поломке подвижки или редуктора.

Управление DC двигателем производится с помощью *PID регулятора*. Перед началом работы рекомендуется внимательно ознакомиться с разделом *PID алгоритм для управления DC двигателем*.

Важно: Неправильные настройки PID регулятора могут привести к поломке подвижки. В предустановленных профилях подвижек все настройки уже сделаны правильно. Без крайней необходимости не рекомендуется самостоятельно изменять данные настройки.

4.3.1.3 BLDC двигатели

Прошивки версии 4.1.x и новее поддерживают управление BLDC. Как и для DC, для управления BLDC необходим датчик обратной связи. В настоящее время в качестве датчика обратной связи поддерживается только энкодер.

Основными параметрами двигателя являются максимальный ток и напряжение, которые устанавливаются во вкладке *Настройка кинематики движения (BLDC мотор)*. Основным параметром *энкодера* является количество отсчетов на оборот.

Управление BLDC двигателем производится с помощью *PID регулятора*. Перед началом работы рекомендуется внимательно ознакомиться с разделом *PID алгоритм для управления BLDC двигателем*.

Важно: Как и для DC двигателя, важно установить правильное значение максимального тока, количества импульсов на оборот, PID-коэффициентов. Также важно установить правильное значение количества полюсов. Неправильные значения могут привести к поломке

4.3.1.4 Критерий выбора двигателя

Для управление током в обмотках двигателях используется принцип широтно-импульсной модуляции, приводящий к колебаниям тока на частоте модуляции (так называемый «токовый риппл»). В зависимости от параметров используемого двигателя (индуктивность его обмоток, омическое сопротивление) риппл может быть разным. Такие резкие колебания тока могут приводить к тому, что мотор нагреется сильнее, чем ожидается при номинальном токе, т.е. $\frac{P_{real}}{RI_s^2} > 1$, где RI_s^2 - мощность, которая ожидалась бы при прохождении постоянного тока I_s, P_{real} - действительная мощность, выделяемая в двигателе. Чтобы оценить перегрев, рекомендуем воспользоваться следующим графиком:

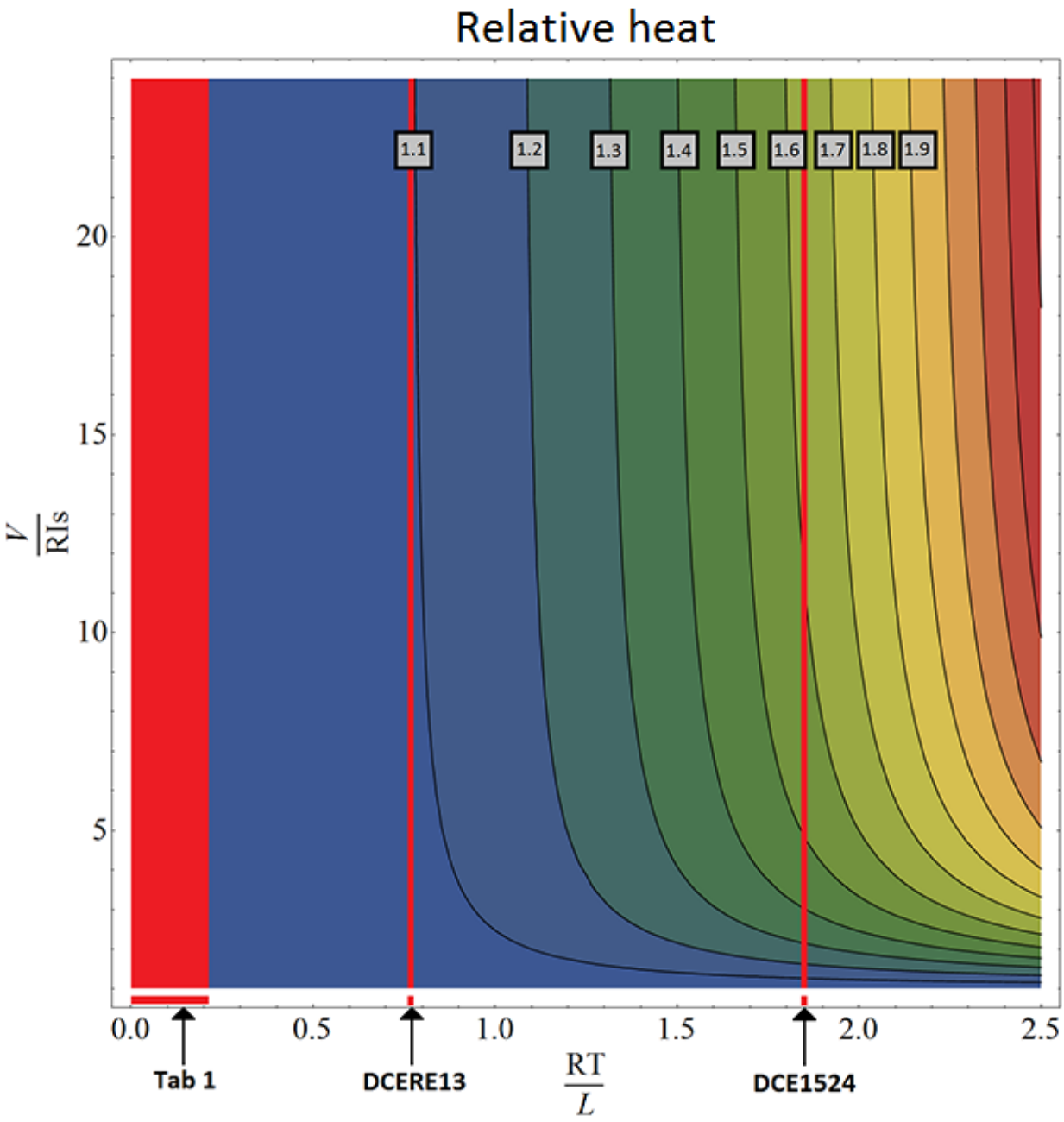

Relative heat
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
20
15
10
5
V / RIs
0.0
0.5
1.0
1.5
2.0
2.5
RT / L
Tab 1
DCERE13
DCE1524

Таблица 4.4: Значения параметра RT/L для некоторых моторов

Мотор	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

Последовательность действий:

- Расчёт параметра $\frac{RT}{L}$, где R, L - сопротивление и индуктивность обмотки (см. документацию на соответствующий двигатель), T - время периода модуляции. Его следует взять равным 51.2 мкс для шаговых двигателей и 25.6 мкс для двигателей постоянного тока.
- Расчёт параметра $\frac{V}{RI_s}$, характеризующего превышение питающего напряжения над номинальным. Тут V - напряжение питания, R - сопротивление обмотки, I_s - ток стабилизации.
- Определение перегрева. После первых двух шагов на график можно нанести соответствующую точку. Теперь нужно определить области, которые соответствуют степеням перегрева. Например, области между линиями 1.1 и 1.2 соответствуют значениям перегрева $1.1 * RI_s^2$ and $1.2 * RI_s^2$.

Пример расчёта для двигателя DCE1524:

- $T = 25.6$ мкс
- $R = 5.1$ Ом
- $L = 70$ мкГн
- $\frac{RT}{L} = 1.86$

Далее проводим вертикальную линию, по которой можно сделать вывод какой будет перегрев при разных питающих напряжениях (эта линия проведена на графике выше). Допустим, что желаемый ток стабилизации $I_s = 500$. Номинальное напряжение в таком случае: $R * I_s = 2.55$ В. Тогда при превышении этого напряжения больше чем в 5, но меньше чем в 10 раз, перегрев будет между 1.5 и 1.6. А при питающем напряжении в 30 В двигатель нагреется в $\simeq 1.65$ раз сильнее чем ожидается.

Для облегчения работы, все основные двигатели и их параметры были рассчитаны (см. *Значения параметра RT/L для некоторых моторов*).

4.3.2 Ограничители на двигателях

Для обеспечения безопасной работы двигателей предусмотрены ограничители по току и напряжению на обмотках двигателя, а так же ограничение максимальных оборотов оси мотора. Данные ограничения, если они активированы приводят к плавному снижению мощности и скорости вращения до значений, приводящих к снижению ограничиваемых параметров до установленных значений. Данная функция оперирует со значениями токов и напряжений непосредственно на моторе, в отличие от критических

параметров, которые оперируют с токами и напряжениями на входе контроллера. Другим отличием «ограничителей» от *критических параметров* является то, что первые не приводят к остановке мотора и переходу в состояние **Alarm**, а лишь ограничивают рост тока, напряжения или оборотов двигателя.

Для DC моторов:

- *Max voltage* - номинальное напряжение питания мотора. Определяет максимальное напряжение питания на обмотках двигателя. Обычно используется для ограничения роста напряжения при заклинивании или нештатной работе подвижки. *Стоит использовать только в случае, если не известно значение максимального тока на обмотках двигателя.* Данный параметр используется при работе *PID регулятора*.
- *Max current* - определяет максимальное значение тока на обмотках двигателя. Обычно используется для ограничения роста тока при заклинивании или нештатной работе подвижки. Данное ограничение стоит выбирать исходя из того, какой ток может в течении длительного времени течь через обмотку, не вызывая повреждения двигателя (в первую очередь от перегрева).
- *Max RPM* - максимальная скорость вращения вала мотора. Обычно используется для ограничения скорости вращения при работе с редукторами и прочими механизмами, обладающими жесткими ограничениями на максимальную скорость вращения.

Примечание: Не стоит путать понятия максимального тока двигателя и номинального тока. В общем случае, они могут отличаться в зависимости от охлаждения мотора и условий его эксплуатации. Также не стоит путать максимальный ток и стартовый ток, который развивается при неподвижном вале.

Важно: Изменение максимального напряжения питания мотора может привести к расстройке PID регулятора. Подробнее смотрите в разделе *PID алгоритм для управления DC двигателем*.

Важно: Значение максимального напряжения питания мотора может превышать значение номинального напряжения питания (обычно на 10-15%). Если вы используете двигатель с малой нагрузкой и вам нужна высокая скорость движения двигателя, то можно повысить максимальное напряжение питания мотора.

Работа ограничителя тока.

Важно помнить, что ограничитель максимального тока *Max current* при работе с DC двигателями работает не мгновенно. При возникновении превышения тока в обмотке двигателя, напряжение, подаваемое на двигатель, начинает постепенно уменьшаться до тех пор, пока ток через обмотку не будет меньше *Max current*. В случае, если во время быстрого движения произошло резкое заклинивание двигателя (самый худший случай) напряжение на обмотке двигателя может спадать в течение максимум 370мс. При правильно выбранном ограничении тока, за это время двигатель не перегреется.

Примечание: Если поставить значение максимального тока *Max current* слишком маленьким, то возможно, что при большой нагрузке или высоком трении DC двигатель не сможет сдвинуться с места.

Для ШД:

- *Max(nominal) Speed* - максимальная скорость вращения вала мотора в шагах в секунду. Текущая скорость ШД определяется параметром *Speed* (см. *Движение с заданной скоростью*)
- *Nominal current* - определяет номинальное значение тока на обмотках двигателя. Это значение не превышается в силу особенностей управления шаговыми двигателями.

В программе XILab установки ограничителей описаны в разделах *Настройка кинематики движения (DC мотор)* и в *Настройка кинематики движения (Шаговый двигатель)*.

4.3.3 Концевые выключатели

4.3.3.1 Задача концевых выключателей

Задача концевых выключателей - предотвратить выход позиционера за допустимые физические границы его перемещения или ограничить диапазон перемещения в соответствии с требованиями пользователя. Неправильная настройка концевых выключателей может привести к заклиниванию позиционера, если контроллер выйдет за границы допустимого диапазона.

https://youtu.be/obMC2t8Xrho

Отличие механических и оптических концевых выключателей. В этом видео наглядно показано, как механические и оптические переключатели влияют на точность и как «уплывает» позиция при повторении калибровки в программном обеспечении XILab.

4.3.3.2 Общие настройки

Если концевик считается активным, то в структуре состояния выставляется соответствующий флаг, а на *главном окне XiLab* выводится соответствующий значок (левый или правый). Контроллер способен останавливать любое движение в сторону обоих активных концевых выключателей (левого и правого), только одного (левого или правого) или не ограничивать движение. Настройка концевиков может быть выполнена в XILab (см. *Настройка диапазона движения и концевых выключателей*).

4.3.3.3 Программное ограничение диапазона движения

Если аппаратных ограничителей на диапазон движения нет, а позиционер требует такого ограничения, то можно использовать программные ограничители. Для этого концевики переводятся в режим ограничения по отсчётам позиции (см. *Настройка диапазона движения и концевых выключателей*).

Предупреждение: Программное ограничение диапазона работает надежно только, если не происходит непосредственного задания новой позиции командами ZERO или SPOS, нет потери шагов или неисправности энкодера, при его использовании для позиционирования, а также не происходит частой потери питания во время движения. Если возникла одна из таких проблем, то программный диапазон надо перенастроить. Автоматически это можно сделать если есть подходящий опорный датчик с помощью *автоматической калибровки нулевой позиции*.

4.3.3.4 Аппаратные концевые выключатели

Контроллер может работать с концевыми выключателями на базе сухих контактов, оптопар, герконов и датчиками любых других типов, способных выдавать электрический сигнал «логическая единица» стандарта TTL 5 В в одном состоянии и «логический нуль» в другом. Причем каждый концевик может быть сконфигурирован независимо. Также есть возможность программно менять местами концевые выключатели и изменять их полярность.

Примечание: Концевые выключатели также удобно использовать для *автоматической калибровки нулевой позиции*.

4.3.3.5 Подключение концевых выключателей

Концевые выключатели подключаются к выводам в *разъеме DSub*. Ниже приведены типовые схемы подключения:

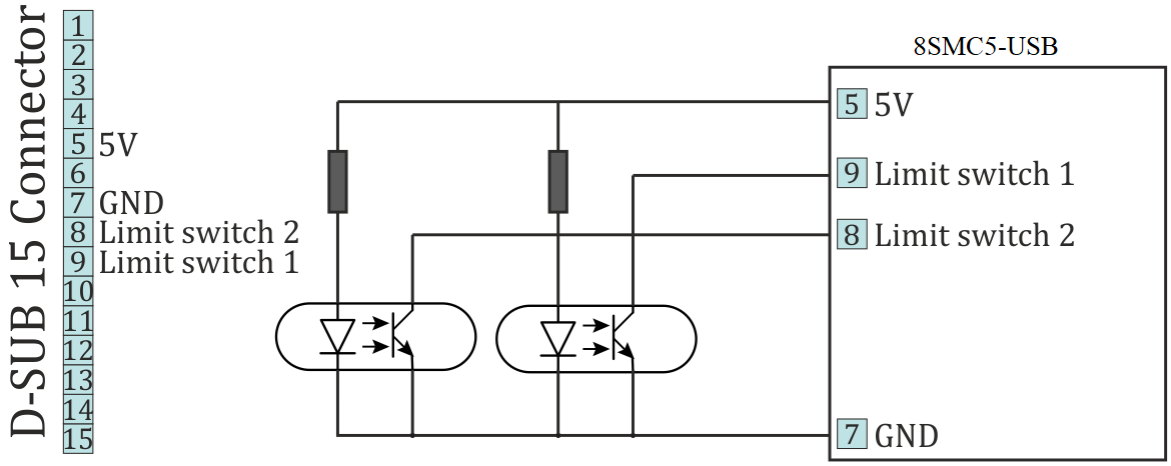


Рис. 4.15: Подключение концевиков типа «оптопара»

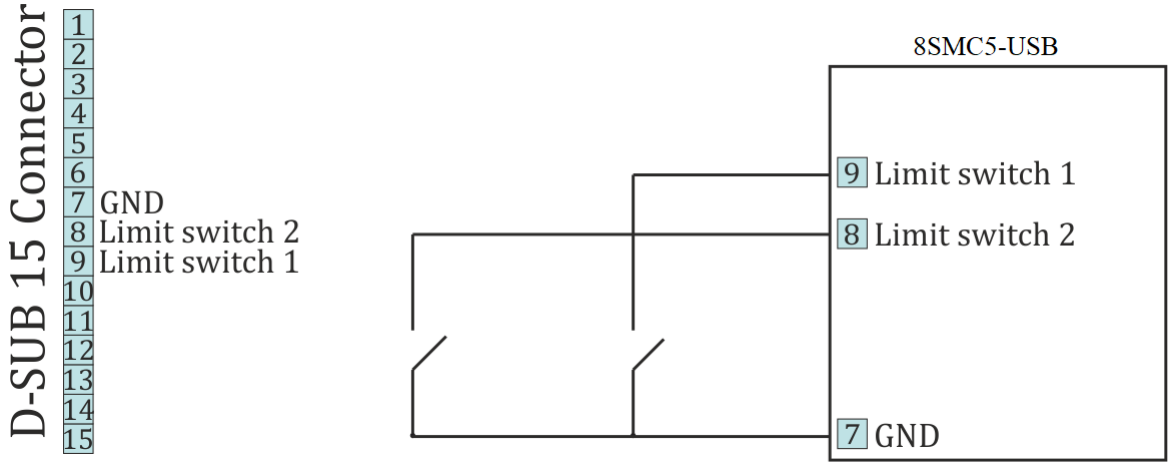


Рис. 4.16: Подключение концевиков типа «сухой контакт»

4.3.3.6 Расположение концевых выключателей на трансляторах

Контроллеру необходимо указать какой из концевых выключателей будет левым, а какой - правым. Иногда это заранее неизвестно, а известно лишь, что оба концевика подключены и срабатывают каждый по достижении своей границы перемещения. Если неправильно настроить концевики, то позиционер может заклинить. Поэтому контроллер поддерживает простую функцию обнаружения неверно настроенных концевиков, останавливаясь по обоим из них. Убедитесь, что оба концевика не активны, их полярность настроена правильно и включена остановка по обоим концевикам. Включите флаг обнаружения неправильного подключения концевиков в соответствующем *меню XiLab*. Начните движение в любую сторону до остановки движения по концевику. Если движение было вправо, а левый концевик стал активным, или наоборот, то нужно поменять концевики местами (см. *Настройка диапазона движения и концевых выключателей*). При обнаружении неверного срабатывания концевика контроллер может перейти в режим Alarm, если включена соответствующая настройка в меню *критических параметров* .

Предупреждение: Защита от перепутанных концевиков не гарантирует, что о проблеме перепутанности можно забыть. Она лишь облегчает первоначальную настройку. Нельзя, в частности,

начинать движение если какой-либо концевик активен, даже при включенной функции защиты.

4.3.4 Автокалибровка домашней позиции

Автокалибровка используется для определения и установки подвижки в начальную позицию (также ее можно назвать «домашней» или «нулевой»). Калибровка сводится к автоматическому точному обнаружению *концевика*, сигнала *датчика оборотов* или момента поступления *внешнего сигнала*, определяющего нулевую позицию, и смещению от нее на заданное значение отступа. Это позволяет начать работу в ситуации, когда текущее положение позиционера не известно, но известно расположение одной реперной точки (исходного положения) относительно концевика или какого-либо другого сигнала. Процесс автокалибровки не требует от пользователя навыков программирования.

Реперная точка (сигнал остановки) определяется одним из трех способов в зависимости от выбранных пользователем настроек:

- Движение до *концевика* - в этом случае используются текущие настройки концевиков (расположение, полярность), см. раздел *Настройка диапазона движения и концевых выключателей*.
- Движение до поступления сигнала с *датчика оборотов* - в этом случае используются текущие настройки датчика оборотов, см. раздел *Контроль позиции*.
- Движение до поступления сигнала на *вход синхронизации* - в этом случае используются текущие настройки входа синхронизации, см. раздел *Настройки синхронизации*, если вход синхронизации программно выключен в соответствующих настройках, то обработка сигнала со входа синхронизации никогда не произойдет.

Предупреждение: Если вход синхронизации выключен программно в соответствующих настройках, то обработка сигнала со входа синхронизации никогда не произойдет.

4.3.4.1 Стандартный алгоритм поиска домашней позиции

В зависимости от настроек процесс автокалибровки может происходить по трем алгоритмам. Стандартный поиск домашней позиции заключается в том, что контроллер начинает перемещение с заданными параметрами скорости и направления движения до получения сигнала остановки. Скорость автокалибровки, как правило, задается более низкой, чем скорость обычного движения, чтобы «не пропустить» приход сигнала и повысить точность калибровки. Затем производится безусловное смещение на обычной рабочей скорости на заданное расстояние отступа.

Полученная точка называется исходным положением или «домашней позицией». Важно отметить, что ее расположение на позиционере не зависит от начального положения, из которого началась калибровка.

4.3.4.2 Точная докалибровка

После поступления сигнала остановки позиция контроллера уже определена. Но прежде чем сделать сдвиг до домашней позиции, можно включить дополнительное движение до следующего сигнала остановки (вторую фазу автокалибровки). Это позволяет выставить домашнюю позицию с точностью, достигающей на некоторых позиционерах 1/256 шага для шаговых двигателей или 1 отсчета энкодера для DC и BLDC двигателей. Если установлен соответствующий флаг, контроллер вращает мотор в заданную пользователем сторону с настроенной скоростью до получения сигнала остановки от источника, выбранного пользователем. Затем, как и при использовании стандартного алгоритма, производится смещение на обычной рабочей скорости на заданное расстояние отступа.

Параметры второй фазы автокалибровки (скорость, направление движения и источник сигнала остановки) задаются независимо от настроек параметров первой фазы. При этом разумно использовать

сигнал с датчика оборота на валу двигателя до редуктора и совершать движение на маленькой скорости - это обеспечит максимальную точность. Так как сигналы окончания первого движения и второго движения могут совпадать, то в ПО предусмотрен специальный флаг для начала отслеживания сигнала окончания второго движения только после совершения полуоборота вала двигателя. Это позволяет избежать неоднозначной последовательности получения сигналов о завершении первого и второго движений. В результате опционального второго движения калиброванная позиция уточняется.

Примечание: Если используется вторая фаза автокалибровки, то первое перемещение можно выполнять с высокой скоростью, так как оно лишь грубо калибрует позицию, и точность там не требуется. Если использовать для второй фазы движения второй концевик, повышение точности не произойдет, так как его физические параметры не отличаются от параметров первого концевика.

4.3.4.3 Быстрый алгоритм автокалибровки

При активации функции быстрого алгоритма автокалибровки контроллер начинает вращать мотор в заданную сторону с обычной рабочей скоростью для быстрого поиска положения реперной точки. После поступления сигнала остановки контроллер отводит мотор назад на половину оборота и вновь начинает движение в заданном направлении, но уже со скоростью, заданной в настройках первой фазы автокалибровки. После повторного получения сигнала остановки производится смещение на рабочей скорости на заданное расстояние отступа. Источник сигнала остановки также задается настройками стандартного алгоритма.

Быстрый алгоритм автокалибровки является оптимальным для большинства двигателей и позиционеров и, как правило, выставлен по умолчанию в настройках профилей для большинства позиционеров фирмы Standa.

4.3.4.4 Особенности автокалиброки

После успешного завершения калибровки домашней позиции в *структуре состояния контроллера* устанавливается флаг STATE_IS_HOMED. Если после этого домашняя позиция каким-либо образом сбилась (*остановка по концевому выключателю*, *немедленная остановка* во время движения, *обнаружение потери шагов*, переход в *режим Alarm*), то соответствующий флаг снимается и нужно вновь провести калибровку домашней позиции.

Примечание: Получаемая в результате калибровки позиция будет немного зависеть от скорости, с которой выполнялось последнее движение до срабатывания выбранного датчика. Поэтому для точного попадания в ту же позицию не меняйте параметры скорости.

Примечание: Если команда *немедленной остановки движения* или команда *отключения питания* мотора выполняются в момент, когда мотор не вращается, то это не сбивает калибровку домашней позиции и флаг STATE_IS_HOMED не снимается.

Описание функций автокалибровки домашней позиции приведено в *руководстве по программированию*.

Команды автокалибровки приведены в разделе *Описание протокола обмена*.

Автокалибровка может быть настроена пользователем в программе XILab на вкладке *Device -> Home position* см. раздел *Настройка исходного положения*, а запущена кнопкой **Go home** в *главном окне XiLab*.

Для **автоматической установки** настроек домашней позиции в комплект XILab входит *скрипт set_zero*. Этот скрипт изменяет настройку Standoff вкладки *Home position settings* так, что текущая

позиция становится домашней.

Использование скрипта:

- установите подвижку в желаемую позицию,
- включите скрипт и дождитесь окончания его выполнения.

В результате подвижка окажется в той же позиции, в которой был запущен скрипт, и все последующие вызовы функции автокалибровки будут приводить её в эту позицию. Не забудьте *сохранить настройки* в энергонезависимую память контроллера.

4.3.5 Работа с энкодерами

4.3.5.1 Область применения энкодеров

Энкодеры применяются для создания точной и быстродействующей обратной связи по координате со всеми типами электродвигателей. Причем обратная связь может осуществляться по положению оси мотора, по линейному положению позиционера, углу поворота моторизованного столика или по любому параметру, непосредственно связанному с положением оси мотора и измеряемому с помощью двухканального квадратурного энкодера, удовлетворяющего требованиям описанным в разделе *Технические характеристики* для соответствующего типа контроллера. Контроллер **8SMC5** поддерживает как дифференциальные энкодеры, так и простые (single-ended) энкодеры, с возможностью автоопределения типа энкодера.

Предупреждение: Автоопределение типа энкодера работает только с энкодерами на 3.3 В и 5 В (с погрешностью 0.2 В).

4.3.5.2 Что такое квадратурный энкодер?

Энкодер - это датчик механического движения. Квадратурный энкодер предназначен для прямого определения позиции оси. Датчик передает относительное положение оси в виде двух электрических сигналов по каналам CH A и CH B, смещенных относительно друг друга на четверть периода.



Рис. 4.17: Сигналы на выходе CH A и CH B квадратурного энкодера

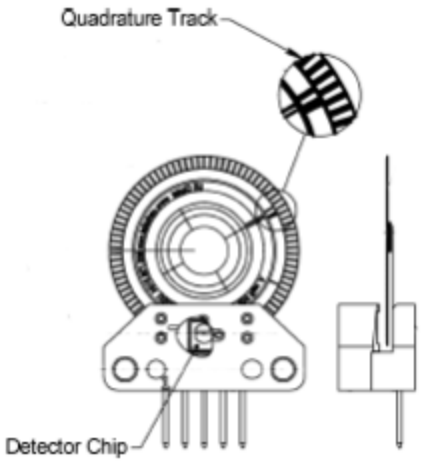


Рис. 4.18: Механика оптического квадратурного энкодера

Механика оптического квадратурного энкодера представлена на рисунке. Используются две оптопары. Принцип работы оптопары: светодиод и детектор расположены напротив друг друга с разных сторон от диска. Когда «окно» диска попадает на детектор, оптопара «открыта» (выходной сигнал - логический 0). Если детектор закрыт непрозрачной частью диска, то выходной сигнал датчика – логическая 1.

Основная характеристика квадратурного энкодера – число шагов на один оборот (CPR). Стандартные значения разрешения для энкодера – от 24 до 1024 CPR. Каждый период изменения сигнала может быть расшифрован 1, 2 или 4 кодами, что соответствует режимам работы X1, X2 и X4. В данном контроллере используется наиболее точный режим X4. Максимальная частота импульсов энкодера: 200 кГц для несимметричного и 5 МГц для дифференциального энкодера. Максимальная частота каждого из сигналов энкодера, зависит от выбранного энкодера, так для 200 кГц и режима x4 контроллер способен считывать 800 000 отсчётов энкодера в секунду. Для энкодера с частотой 5 МГц в режиме X4 контроллер может считывать до 20 миллионов отсчетов энкодера в секунду.

4.3.5.3 Возможности контроллера

Контроллер имеет два режима работы с энкодером:

- использование энкодера как основного датчика положения (основной режим работы с двигателями постоянного тока; для шаговых двигателей режим ведущего энкодера доступен в прошивках с версий 3.10.x).
- обнаружение проскальзывания, люфта или потери шагов (рекомендуемый режим работы совместно с шаговыми двигателями, если энкодер не используется как основной датчик положения, *подробнее*).

4.3.5.4 Подключение энкодера

Подключение энкодера к контроллеру осуществляется через *разъем DSub* , который есть во всех системах: *плата контроллера*, *одноосная* и *двухосная* в корпусе и многоосная

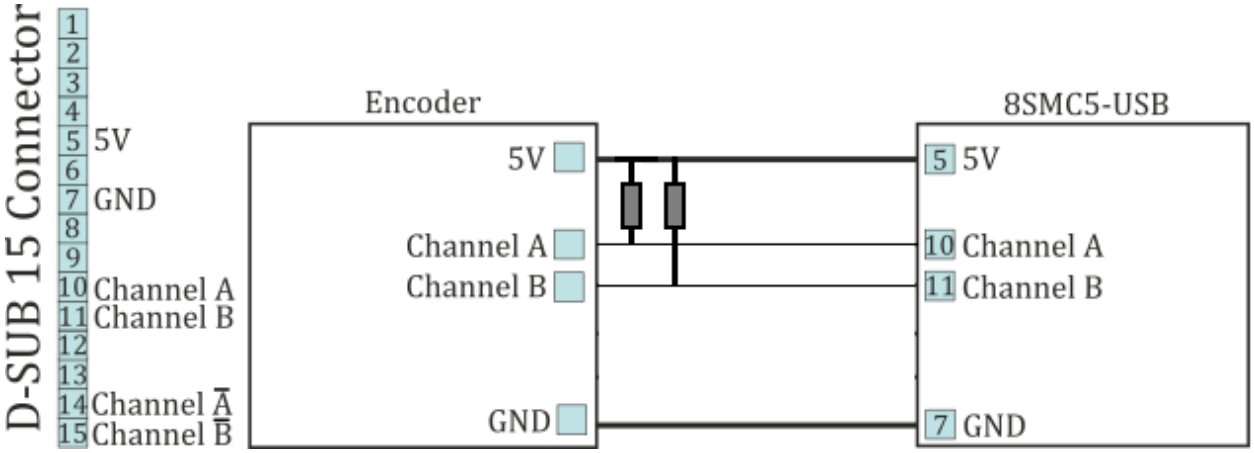


Рис. 4.19: Схема подключения простого энкодера к разъему DSub.

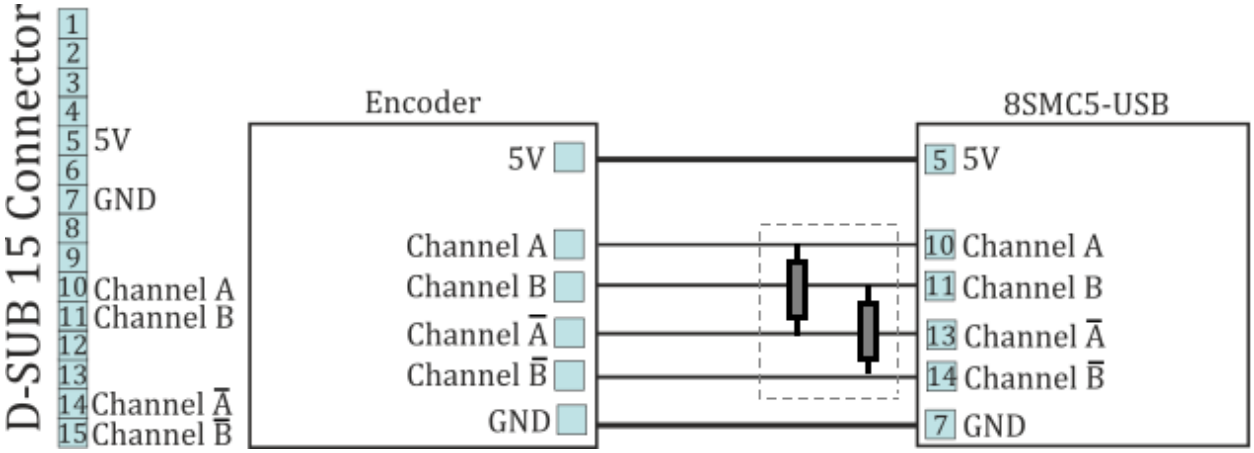


Рис. 4.20: Схема подключения дифференциального энкодера к разъему DSub.

Также смотрите раздел *пример подключения простого мотора*.

Предупреждение: Входы энкодеров контроллера внутренне подтянуты к логической единице сопротивлением 5.1 кОм. Обычно выходы энкодера имеют тип «открытый коллектор» с внутренним подтягивающим резистором. При передаче данных, они обеспечивают хорошие показатели перехода из высокого логического уровня в низкий. Но переход из логического 0 в логическую 1 оказывается более плавным. Он происходит через RC цепь, образованную сопротивлением подтяжки и ёмкостью кабеля. Это особенно важно для длинных кабелей *(до 5 метров)*. Если встроенной подтяжки недостаточно, то для улучшения показателей скорости перехода 0 - 1 можно добавить подтягивающий резистор R=1.5k Ом к +5 В на каждый выход, проверив, что открытый коллерктор энкодера способен пропускать ток 5 мА. Схема включения резисторов показана выше. Максимальной скорости работы инкрементального квадратурного энкодера можно достичь добавив к его выходу драйвер push-pull с выходным током более 10 мА, который обеспечивает резкие фронты переходов 0 - 1 и 1 - 0.

4.3.5.4.1 Использование длинных кабелей

Для корректной работы энкодеров при использовании кабелей длиннее 5 метров рекомендуется использовать энкодеры с дифференциальным выходом типа RS485 для снижения влияния электромагнитных наводок. При использовании интерфейса RS485 все дифференциальные пары должны быть терминированы резистором номиналом 120 Ом, который должен располагаться в разъёме подключения к контроллеру.

Кабель должен иметь дополнительный внутренний экран для цифровых сигналов (пины 5-15), подключенный к DGND (пин 7) на стороне контроллера и на стороне позиционера. Внешний экран должен быть подключен к металлическому корпусу разъёма напрямую на стороне позиционера и к металлическому корпусу разъёма через конденсатор номиналом 47 нФ на стороне позиционера.

4.3.5.4.2 Автоматическое определение типа энкодера

Контроллер 8SMC5 может автоматически определять тип подключенного энкодера, *если соответствующая опция активирована в настройках*. Эта система сконструирована для работы со стандартными кабелями типа CAT-5E длиной до 50 метров и с сопротивлением проводников 80 мОм на метр. Автоматическое определение может работать неправильно, если длина кабеля превышает 50 метров или при использовании нестандартного кабеля с большим удельным сопротивлением. В случае проблем с автоматически определением типа энкодера, тип экондера может быть принудительно установлен *в настройках обратной связи*.

4.3.6 Датчик оборотов

Датчик оборотов предназначен для обнаружения *потери шагов* шагового двигателя (ШД) и более точной калибровки домашней позиции (см. *Автокалибровка «домашней» позиции*).

Контроллер может получать данные о текущей позиции с внешнего датчика оборотов, установленного на оси ШД. Датчик передает сигналы в контроллер один или несколько раз за один оборот двигателя.

Обычно датчик оборотов представляет собой маленький диск с точной шкалой деления, который устанавливается на ось ШД. С разных сторон диска, напротив друг друга, расположены источник (светодиод) и регистратор оптопары. Когда отсечка шкалы не находится между светодиодом и регистратором, датчик «открыт» (на выход оптопары подается логический ноль). Когда отсечка закрывает источник света от детектора, то датчик на выход подает логическую единицу.

Контроллер по умолчанию воспринимает низкий логический уровень как активное состояние датчика оборотов. Вход контроллера притянут к единичному логическому уровню, так что неподключенный датчик оборотов считается неактивным состоянием. При необходимости можно инвертировать вход контроллера и активным будет считаться логический уровень единицы.

4.3.6.1 Схема подключения

Выводы для подключения датчика оборотов во всех системах (*плата контроллера*, *одноосная* и *двухосная* в корпусе и многоосная) расположены на *разъёме D-SUB*.

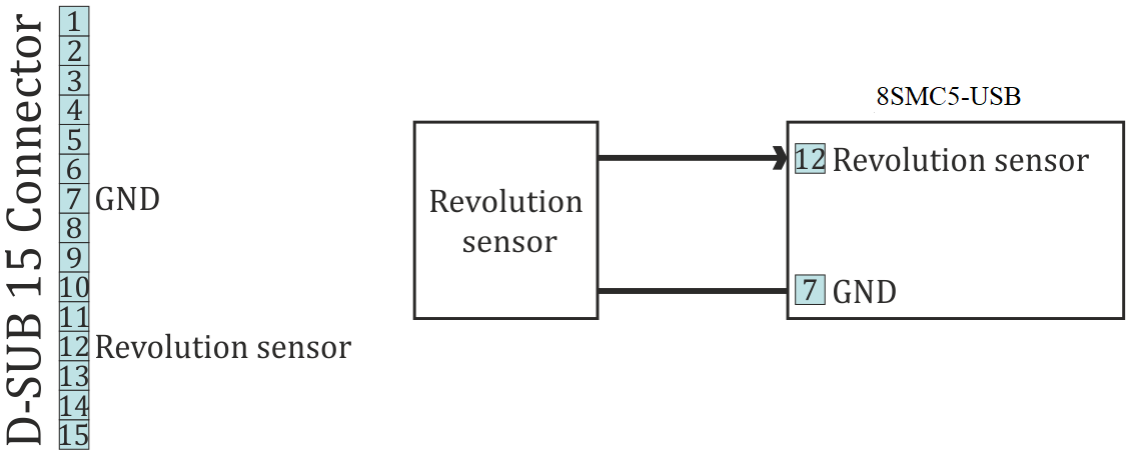


Рис. 4.21: Схема подключения датчика оборотов к системе на базе 8SMC4-USB

4.3.7 Обнаружение потери шагов

Данный режим используется главным образом если производится работа с шаговым двигателем на предельных скоростях или нагрузках, где возможно застревание оси, приводящее к потере шагов. В этом случае дополнительный датчик положения (*датчик оборотов* или *энкодер*) позволяет отследить этот момент уведомить пользователя. Данная функция используется **только совместно с шаговыми двигателями** и позволяет обнаруживать потерю шагов. Все координаты и положения оси мотора измеряются в шагах и микрошагах.

При использовании энкодера в контроллере сохраняется значение количества шагов двигателя и отсчётов энкодера на оборот (см. вкладку XiLab *Настройка кинематики движения*). При включении функции, контроллер сохраняет текущую позицию в шагах ШД и текущую позицию по данным энкодера. Далее, в ходе движения, позиция по энкодеру преобразовывается в шаги и, если разница оказывается больше заданного значения, осуществляется индикация проскальзывания и переход в *режим Alarm*, при включении соответствующей настройки. Подробно использование энкодера как датчика потери шагов описано в разделе *Работа с энкодерами*.

При использовании датчика оборотов позиция контролируется по нему. По активному и неактивному фронтам на входе от датчика оборотов контроллер запоминает текущее положение в шагах. Далее, при каждом обороте (количество шагов на один полный оборот мотора устанавливается параметром *Steps per turn*, см. *Настройка кинематики движения (Шаговый двигатель)*) контроллер проверяет, на сколько шагов сместилась ось. При рассогласовании более чем на заданное значение ошибки Minimal error (устанавливается в настройках контроля позиции, см. *Контроль позиции*) осуществляется индикация проскальзывания флагом структуры состояния. Если в настройках установлен соответствующий флаг, то при выявлении ошибки контроллер переходит в *режим Alarm* и мотор останавливается, иначе продолжает свое движение. Если флаг индикации проскальзывания активен, то контроллер переходит в *режим Alarm* при включении соответствующего параметра в настройках.

Так же в настройках контроля позиции может быть включен режим *автоматической коррекции позиции* в случае потери шагов. Если эта опция включена, то при обнаружении проскальзывания контроллер останавливает вращение, корректирует шаговую позицию на основании данных энкодера и пытается запустить вращение заново. *Флаг ошибки контроля позиции* устанавливается, когда позиция сбивается и автоматически снимается после корректировки. В случае невозможности скорректировать позицию устанавливается *флаг ошибки контроля позиции* и контроллер переходит в *режим Alarm*. Если потеря шагов происходит в процессе движения, то *статус движения* не сбрасывается во время коррекции позиции. Если потеря шагов происходит в режиме удержания позиции, то для восстановления корректной позиции подаётся команда *движения в позицию*, в которой ось мотора находилась до потери шагов.

Примечание: Для использования функции коррекции позиции нужен энкодер с разрешением не менее двух отсчётов на шаг мотора.

Примечание: Для корректной работы коррекции позиции контроллеру нужно дать постоять с запитанными обмотками в течение 1 секунды для калибровки. После перехода конторллера в *режим Alarm* или изменения настроек требуется повторная калибровка.

Примечание: При использовании автоматической коррекции позиции не рекомендуется устанавливать значение *Threshold* более 3 шагов т.к. в этом случае не любое проскальзывание будет скорректировано.

Примечание: Команды резкой и плавной остановки могут быть проигнорированы контроллером во время коррекции позиции. В этом случае можно послать команду плавной остановки дважды, что приведёт к снятию питания с обмоток мотора.

Примечание: *Замечание.* Если Вы пользуетесь *программными концевиками*, то использовать автоматическую коррекцию позиции не рекомендуется, т.к. положения программных концевиков будут изменяться в процессе коррекции позиции.

Примечание: Команда резкой остановки запускает процесс перекалибровки положения датчика оборотов, причём калибровка происходит при срабатывании датчика оборотов во время движения, управляемого двигателем. Это значит, что если сразу после резкой остановки повернуть ось руками, то после начала движения проскальзывание не будет обнаружено, т.к. калибровка ещё не была произведена.

Примечание: Если датчик оборота двигателя подвержен дребезгу (механическому), то на очень малых скоростях возможны ложные срабатывания контроля по датчику оборотов.

Примечание: Контроль позиции по датчику оборотов не может обнаружить вращение оси при нулевой внутренней скорости. Т.е. если остановить двигатель и руками провернуть ось, то это не будет обнаружено.

4.3.8 Управление питанием мотора

4.3.8.1 Снижение тока потребления

Для уменьшения энергопотребления шагового двигателя в режиме ожидания контроллер позволяет задавать уровень потребляемого тока в состоянии остановки ниже номинального значения. Этот режим по умолчанию активирован. Он повсеместно используется для снижения нагрева шагового двигателя в режиме удержания при сохранении точности нахождения в заданной позиции. Уровень тока удержания задаётся в процентах от номинального уровня тока в обмотках. Также определяется время в миллисекундах, через которое ток будет снижен. Опцию снижения тока можно отключить специальным флагом. Для настройки функции снижения тока удержания смотрите функцию *set_power_settings*

(см. раздел *Руководство по программированию*) или вкладку настроек XiLab - *Настройка параметров энергопотребления*. Установка номинального тока шагового двигателя осуществляется командой *set_engine_settings* (см. *Руководство по программированию* или раздел *Настройка кинематики движения (Шаговый двигатель)*).

Разумным значением уровня сниженного тока удержания является 40-70%. Это снижает энергопотребление в 2-4 раза, а сила удержания обычно остаётся достаточной. Время, через которое ток снижается разумно выбирать в диапазоне 50-500 мс. Это время окончания низкочастотных колебаний механической системы, которые могут сбить позицию удержания в некоторых системах.

4.3.8.2 Отключение питания мотора

Также для уменьшения энергопотребления шагового двигателя существует режим отключения питания мотора по таймеру. Это необходимо в основном для предотвращения траты энергии на удержание позиции, когда работа с установкой закончена и никаких движений не происходит долгое время. Этот режим по умолчанию активирован, но может быть отключен пользователем. Время от остановки до отключения настраивается в секундах. Разумным временем является 3600 секунд (один час). Для настройки функции отключения питания мотора смотрите функцию *set_power_settings* (см. раздел *Руководство по программированию*) или вкладку настроек XiLab - *Настройка параметров энергопотребления*.

4.3.8.3 Специфика расчёта временных задержек

Все временные задержки работают следующим образом: при каждом переходе в состояние остановки двигателя запоминается время с точностью до миллисекунды. Далее, при достижении заданных пользователем таймаутов и включенности функций PowerOff/CurrentReduce происходит отключение питания двигателя или снижение тока в обмотках. Все настройки можно менять онлайн. Например, если увеличить время таймаута PowerOff после того, как он уже случился, то обмотки запитаются и функция PowerOff сработает снова по достижению таймаута от момента остановки двигателя. То же самое касается включения и отключения флагов использования режимов PowerOff/CurrentReduce. Отсчёт таймаутов останавливается и функции PowerOff/CurrentReduce отменяются при любом начале движения.

4.3.8.4 Функция Jerk free

Иногда необходимо плавно менять ток в обмотках двигателя для устранения вибраций механической системы. Для этого в контроллере предусмотрена опция *Jerk free*, где можно задать скорость выхода тока через обмотки с нуля на номинальное значение с точностью до миллисекунды. Опция включается соответствующим флагом. При этом все изменения тока стабилизации или отключения обмоток будут проходить с предварительным плавным набором или сбросом тока удержания. Например, если установлена скорость набора тока 100 мс и происходит событие снижение тока удержания до 50%, то он будет снижен плавно за 50 мс (а не 100 мс, ведь 100 мс нужно чтобы полностью сбросить ток до нуля). Также за 50 мс ток будет снова набран до номинального при новом движении. Для настройки функции *Jerk free* смотрите функцию *set_power_settings* (см. раздел *Руководство по программированию*) или вкладку настроек XiLab - *Настройка кинематики движения (Шаговый двигатель)*.

Функция плавного набора тока работает при любом изменении амплитуды тока в обмотках, например при смене номинального тока удержания. При этом скорость увеличения или уменьшения тока рассчитывается на основе максимального из введённых токов удержания: старого или нового. Если обмотки нужно отключить, то ток снижается до нуля, а только затем силовые выходные цепи контроллера обесточиваются. Если обмотки нужно запитать номинальным током, то они запитываются нулевым током и далее ток растёт до номинального.

Существуют исключения, когда ток мгновенно сбрасывается до нуля и обмотки отключаются, даже когда функция Jerk free включена. Это события опасности и попадание в состояние Alarm (см. *Критические параметры*), а также моменты перезагрузки контроллера для обновления программного

обеспечения. Все эти события редки и не должны происходить во время работы с позиционером.

Разумным значение времени Jerk free будет 50-200 мс, так как это приведёт лишь к низкоэнергетичным вибрациям на частоте 3-10 Гц, которые будут значительно меньше вибраций от бытовых шумов (шагов, сквозняка). Установка большого времени Jerk free и функции снижения тока для экономии электроэнергии и снижения нагрева мотора приведёт к постоянным задержкам для сброса и набора тока. Поэтому время Jerk free не стоит делать большим.

4.3.9 Критические параметры

Для безопасности работы контроллера и двигателя устанавливаются максимальные и минимальные значения токов, напряжений, температур. Выход из допустимого диапазона для любого из этих параметров приводит к тому, что движение прекращается, обмотки мотора обесточиваются, контроллер переходит в состояние Alarm. Выход из состояния Alarm возможен только при устранении причины превышения критического параметра и посылки команды STOP. Настройки используются для всех типов двигателей.

Доступны следующие параметры:

- *Low voltage off* - определяет минимальное значение напряжения силового питания контроллера (измеряется десятками мВ). Включается флагом *Low voltage protection*. Иначе минимальный порог отключения не действует. Разумное значение 6000-8000 мВ, для рабочего диапазона питания 12-36В. Эта защита помогает обнаружить момент, когда блок питания отключился из-за срабатывания одной из его защит. Такое может произойти со стабилизированным блоком питания при превышении его рабочей мощности.
- *Max current (power)* - определяет максимальное значение тока силового питания контроллера (измеряется в мА). Устанавливать разумно в два раза выше, чем максимальный зарегистрированный рабочий ток потребления во время тестов. Для регистрации тока потребления используйте *графики XiLab*.
- *Max voltage (power)* - определяет максимальное значение напряжения силового питания контроллера (измеряется десятками мВ). Это ограничение разумно брать на 20% выше, чем рабочее напряжение блока питания.
- *Temperature* - определяет максимальное значение температуры микропроцессора (измеряется десятыми долями градуса Цельсия). Микропроцессор выдерживает рабочую температуру 75 градусов Цельсия, но не перегревается сам по себе. Повышение его температуры может косвенно свидетельствовать о перегреве силовой части платы. Значение порога перегрева разумно выбирать в диапазоне 40-75 градусов.

Флаги:

- *ALARM_ON_DRIVER_OVERHEATING* - Входить в состояние Alarm по превышению критической температуры драйвера (>125 градусов). Силовой драйвер сигнализирует о приближении его температуры близко к критической. Если драйвер не отключить, то при дальнейшем нагреве он отключит себя сам. Рекомендуется не доводить до принудительного отключения и установить флаг добровольного отключения.
- *H_BRIDGE_ALERT* - Входить в состояние Alarm при неполадках в силовом драйвере, вызванных безусловным отключение по перегреву или повреждением платы контроллера. Этот флаг должен быть установлен.
- *ALARM_ON_BORDERS_SWAP_MISSET* - Входить в состояние Alarm обнаружении срабатывания не того концевика, к которому осуществлялось движение (см. *Концевые выключатели*). Служит для более понятной индикации срабатывая подсистемы обнаружения перепутанности концевиков. Рекомендуется держать флаг включенным.
- *ALARM_FLAGS_STICKING* - Этот флаг настраивает «залипание» индикаторов произошедшей ошибки в статусной структуре контроллера. Иначе флаги ошибок активны только пока происхо-

дит событие, вызывающее ошибку. Если ошибка носила кратковременный характер и её причина самостоятельно исчезла, то иногда неясна причина попадания в состояние Alarm. Для этого удобно включить «залипание» и на главном окне XiLab диагностировать причину попадания в Alarm.

- *USB_BREAK_RECONNECT* - Этот флаг настраивает работу блока перезагрузки USB шины при потере связи. При установке этого флага данный блок начинает функционировать и отслеживать потерю связи по USB шине (к примеру, в случае удара статическим разрядом).

Установка параметров описана в меню программы XILab *«Настройка предельных параметров контроллера»*. Команды установки максимально допустимых значений описаны в *руководстве по программированию*.

4.3.10 Хранение параметров во flash-памяти контроллера

Контроллер позволяет сохранять все свои настройки в энергонезависимую память. При подаче питания на контроллер он восстанавливает настройки из этой памяти и мгновенно готов к работе. Не нужно каждое включение питания настраиваться на позиционер заново. Контроллер хранит в своих настройках своё имя, вводимое пользователем. Это удобно для его последующей идентификации.

Важно: В энергонезависимую память сохраняются все текущие рабочие параметры контроллера, относящиеся к вкладке Device из меню настроек программы XILab. Поле *friendly_name* является единственным исключением. Любые изменения, внесенные в настройки XILab из других вкладок, не сохранятся!

Это делается с помощью кнопки **Save settings to flash** в программе XILab или с помощью функции *command_save_settings* (см. *Руководство по программированию*).

Можно восстановить в ОЗУ контроллера все настройки из энергонезависимой памяти не только при подаче питания, но и при нажатии в XILab на кнопку Load setting from flash, что позволяет работать с сохраненными во flash данными. Для загрузки можно использовать функцию command_read_settings, см. *Руководство по программированию*. Восстановленные настройки станут активны немедленно. При этом произойдёт переинициализация всех блоков контроллера.

4.3.11 Пользовательские единицы координат

Текущая координата контроллера выводится и задается в шагах шагового двигателя или отсчетах энкодера, если энкодер присутствует и включен. При работе с подвижками может быть удобно задавать позицию в миллиметрах для трансляторов, в градусах для ротаторов, или в других естественных единицах. Для этого программное обеспечение контроллера позволяет пересчитывать координаты в пользовательские единицы. Если пользователь знает какому линейному перемещению соответствует смещение шагового двигателя на определенное количество шагов, он может задать это соотношение как коэффициент пересчета и далее отдавать команды движения и наблюдать за координатой подвижки в этих единицах. Это касается и интерфейса XiLab, и использования в собственных программах и скриптах. Скорость и ускорение также задаются в единицах, производных от пользовательских (например в миллиметрах в секунду). *Установка нулевой позиции* делается одинаково для отсчёта в шагах или в пользовательских единицах.

В *XiLab* отображение пользовательских единиц можно включить на вкладке *Настройки отображения пользовательских единиц*. Можно установить подходящее имя пользовательских единиц.

При работе с библиотекой libximc функции, принимающие и возвращающие величины в пользовательских единицах имеют имена оканчивающиеся на _calb. Эти функции дополнительно принимают как параметр калибровочную структуру calibration_t, см. *Руководство по программированию*.

4.3.12 Использование таблицы коррекции координат для более точного позиционирования

Если используется подвижка без линейного энкодера, то точное положение не всегда будет соответствовать показаниям координат по осям. Это связано с точностью изготовления механических деталей, люфтами, температурным расширением. В этом случае для более точного позиционирования можно воспользоваться корректирующей таблицей.

Важно: Таблица является индивидуальной для каждой подвижки. Таблица формируется производителем на высокоточном стенде.

Принцип работы:

Через определенные расстояния, не обязательно равные, начиная с 0 промеряется реальное положение подвижки. Разность между заданным и реальным положением заносится в таблицу. По полученным значениям, с использованием линейной интерполяции, производится пересчет координат при использовании определенных _calb функций. В результате, компенсируются неточности изготовления и другие возможные отклонения положения.

Пример: Предположим для позиционера задана следующая корректирующая таблица.

X	0	5	10	15	20	25
dX	0	0.05	0.02	-0.003	0.01	-0.04

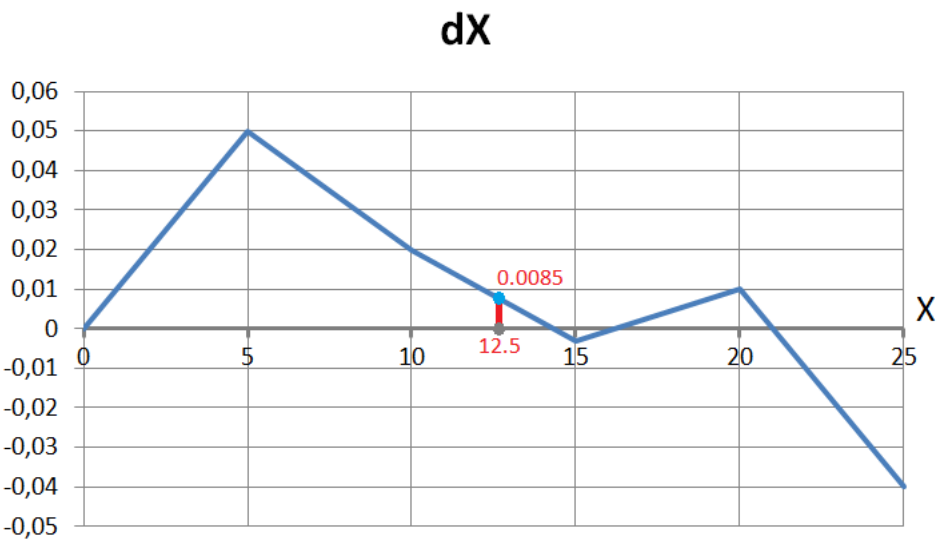


Рис. 4.22: На графике показаны отклонения координат соответствующие таблице

Для перемещения в позицию 12.5 необходимо задать координату на 0.0085 большую то есть 12.5085. Это именно то, что делают алгоритмы некоторых _calb команд, которые используют таблицу коррекции координат.

В *XiLab* загрузить и очистить корректирующую таблицу можно на вкладке *Настройки отображения пользовательских единиц*.

Для просмотра списка функций, структур и параметров, которые корректируются при использовании корректирующей таблицы, смотрите руководство по библиотеке libximc.

4.4 Безопасная работа

Несколько настроек контроллера непосредственно связаны с безопасностью работы. Неправильная их установка может повредить позиционер или контроллер. Позиционер можно повредить превышением мощности, скорости вращения и выходом за пределы допустимого диапазона движения. Обычно для безопасной работы достаточно загрузить заранее подготовленный профиль для Вашего позиционера, где все необходимые настройки уже сделаны.

4.4.1 Границы движения и концевики

Линейные позицинеры имеют ограниченный диапазон перемещения в отличие от круговых ротаторов. Выход такого позиционера за допустимые физические границы его перемещения - основная причина заклинивания или выхода позиционеров из строя. Для предотвращения таких поломок диапазон перемещения позиционера ограничивается в соответствии с требованиями пользователя. Для этого используются *Концевые выключатели*, но в части случаев, например, когда позиционер не оборудован концевиками или имеет только один концевик, границы движения могут определяться программно (см. *Концевые выключатели*). Часто бывает, что *концевики* перепутаны местами. В этом случае воспользуйтесь механизмом обнаружения перепутанности концевиков, описанным в разделе *Концевые выключатели*, чтобы первое же движение до границы не привело к заклиниванию позиционера. Настройка диапазона движения и концевых выключателей описана в соответствующем *разделе* . Команды настройки описаны в *Руководство по программированию*.

4.4.2 Ограничители движения

Шаговый двигатель имеет основную настройку безопасности - номинальный ток в обмотках. Это основной параметр, определяющий мощность, подаваемую на двигатель. Номинальный ток должен быть установлен не выше допустимого для данного двигателя. Подробней смотрите главу *Ограничители на двигателях*. Для DC двигателей максимальный ток является ограничивающим и должен быть установлен согласно максимально допустимому току через DC двигатель. Если не известен максимальный ток, то может быть ограничено максимальное напряжение, подаваемое на двигатель. Это также будет препятствовать его перегреву, хотя ограничение напряжения это более грубый способ, чем ограничение тока. Подробней смотрите главу *Ограничители на двигателях*.

Повредить позиционер или способствовать его быстрому износу может превышение скорости вращения. Необходимо поставить флаг ограничения скорости не выше максимальной и установить правильную максимальную скорость для данного позиционера. Подробней смотрите главу *Ограничители на двигателях*.

4.4.3 Критические параметры

Контроллер отслеживает токи и напряжения, которые возникают в его цепях и способен реагировать на их подозрительные значения. Реакция обесточивает двигатель и препятствует дальнейшему движению, пока причина проблемы не устранена. Это позволяет отследить замыкания обмоток двигателя на друг друга или на землю, которое может возникнуть при повреждении кабеля позиционера или самого позиционера. Также реакция носит информативный характер, позволяя отследить некорректные значения напряжения питания или приближающийся перегрев. Поэтому прочтите главу *Критические параметры* и установите необходимые защиты. При возникновении опасного состояния контроллер переходит в режим Alarm и *главное окно программы XiLab* приобретает красный оттенок. Если такое произошло, то отследите и устраните причину опасности прежде чем отключать режим Alarm. Если Вы пользуетесь собственным приложением для управления двигателем, то обращайте повышенное внимание на флаг состояния Alarm (см. *Статус контроллера*).

4.4.4 Работа с энкодером

Если при подключении энкодера перепутать каналы датчика, то при движении двигателя в положительном направлении, энкодер будет показывать уменьшение координаты. При работе с DC двигателем, данная ситуация появится если перепутать каналы управления двигателем (DC+ и DC-). Для того, чтобы исправить эти ошибки достаточно установить флаг *Encoder Reverse*, указанный в блоке настроек обратной связи во вкладке *Настройка кинематики движения (Шаговый двигатель)* для шагового двигателя и *Настройка кинематики движения (DC мотор)* для DC двигателя.

Так же возможна ситуация, когда нет контакта с одним из каналов энкодера. В этом случае, при движении двигателя показания датчика будут колебаться в диапазоне [-1..1] от начальной позиции.

При работе с DC двигателем обе эти ошибки приведет к некорректной работе алгоритма управления, описанном в пункте *PID алгоритм для управления DC двигателем*. Если вы впервые подключили новый DC двигатель, то перед началом работы настоятельно рекомендуется выполнить проверку подключения энкодера. Для этого, установите следующие значения коэффициентов регулирования: $K_P = 1, K_I = 0, K_D = 0$ и попытайтесь выполнить движение вправо или влево с небольшой скоростью. После начала движения проверьте, что показания энкодера изменяются в соответствии с выбранным направлением. Если необходимо установите флаг *Encoder Reverse*.

4.5 Дополнительные функции

4.5.1 Индикация режима работы

4.5.1.1 Статус контроллера

В контроллере предусмотрена индикация режима работы. Для этого на плате расположен один двухцветный светодиод.



8SMC5-USB 8SMC5-USB-B8-1

Зелёный индикатор Power показывает наличие питания у контроллера.

Красный индикатор Status - отображает режим работы контроллера. Одновременное горение двух цветов выглядит как **жёлтое свечение**.

Таблица 4.5: Режимы работы индикатора Power/Status

Частота мерцания, Гц	Описание
LED индикаторы не светятся	контроллер выключен, нет питания
Power LED индикатор светится зеленым	на контроллер подается питание
Status LED индикатор светится зеленым	в контроллер не загружена прошивка
Status LED индикатор светится желтым	контроллер в состоянии *Alarm*
Status LED индикатор мигает желтым, 0,25 Гц	контроллер работает, но нет связи с ПК по USB
Status LED мигает желтым, 1 Гц	контроллер работает, ожидается команда движения
Status LED мигает желтым, 4 Гц	контроллер работает, выполняется команда движения
Status LED мигает желтым, 8 Гц	контроллер находится в режиме перепрошивки
Status LED мигает желтым, 10 Гц	контроллер находится в режиме повторного переподключения шины USB

4.5.1.2 Индикация концевых выключателей

В контроллере реализована индикация срабатывания концевых выключателей. Высокий логический уровень появляется на соответствующем выводе в момент активности концевого выключателя. Активное состояние определяется исходя из настроек концевых выключателей (см. *Настройка диапазона движения и концевых выключателей*).

4.5.1.3 Схема подключения

Примечание: При использовании дополнительных светодиодов, они должны быть рассчитаны на рабочий ток 4 мА. Дополнительных резисторов, ограничивающих ток, не требуется. Для типовых светодиодов рабочий ток будет около 2 мА. Не рекомендуется использовать светодиоды синего и фиолетового цвета из-за их высокого запирающего напряжения и, как следствие, низкой яркости.

4.5.1.3.1 Плата контроллера

Status индикатор продублирован выходом на многоцелевом BPC разъеме CN1 *коннекторе BPC*. Светодиод подключается к ним в соответствие со схемой ниже:

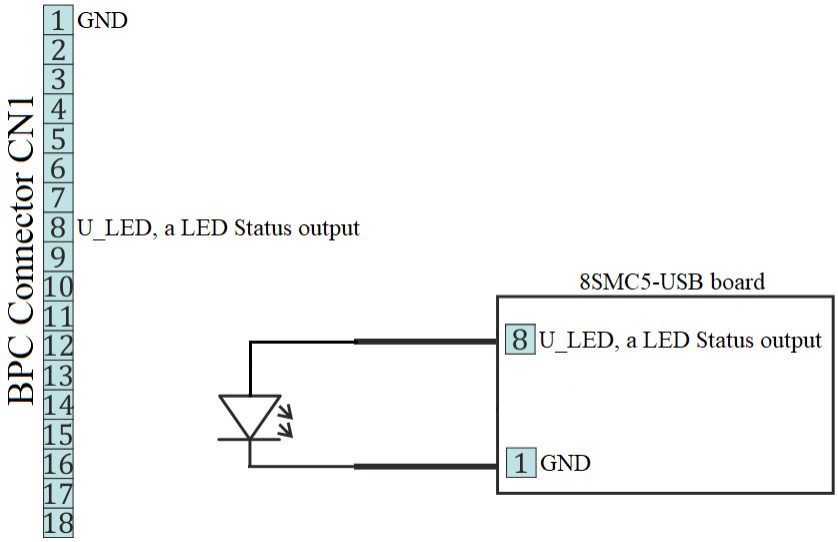


Рис. 4.23: Схема подключения индикаторов Power и Status к плате контроллера

Концевые выключатели расположены на том же самом разъёме. Схема подключения указана ниже. Для индикации срабатывания концевых выключателей удобно использовать светодиоды рассчитанные на нужный ток.

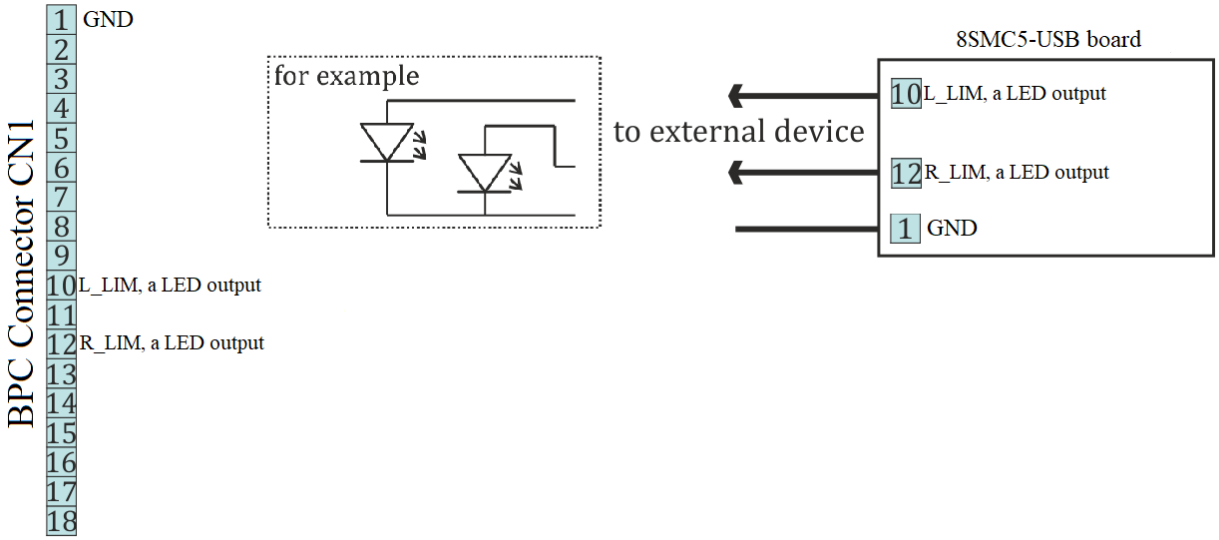


Рис. 4.24: Схема подключения индикаторов концевых выключателей к плате контроллера

4.5.1.3.2 Одноосная и двухосная системы

В коробочных версиях контроллеров (*одноосная* и *двухосная системы*) светодиоды на передней панели представляют собой индикаторы питания, статуса и концевых выключателей, поэтому никакой схемы подключения не требуется.

4.5.2 Работа с магнитным тормозом

На разъеме BPC есть вывод для управления магнитным тормозом, установленным на ось шагового двигателя. Магнитный тормоз используется для удержания положения мотора при отсутствии питания.

4.5.2.1 Описание работы

Магнитный тормоз состоит из магнита и пружины, осуществляющей остановку оси мотора. При отсутствии напряжения на магните пружина зажимает ось в текущем состоянии, что позволяет сохранять необходимое положение мотора. После подачи напряжения на магнит, пружина освобождает ось.

4.5.2.1.1 Последовательность работы контроллера при отключении подвижки.

Остановка мотора (время остановки запоминается в контроллере) -> Отключение магнита от питания, фиксация вала -> Отключение питания платы

При включении подвижки последовательность работы контроллера обратная.

Поскольку любое движение инерционно, для управления магнитным тормозом и процессом фиксации положения устанавливаются следующие параметры:

- Время между включением питания мотора и отжатием тормоза (мс)
- Время между отжатием тормоза и готовностью к движению (мс)
- Время между включением питания мотора и зажатием тормоза (мс)
- Время между зажатием тормоза и отключением питания мотора (мс)

При отключении функции магнитного тормоза контроллер непрерывно подаёт сигнал отжатия тормоза. Это позволяет двигать мотор, оснащённый магнитным тормозом, не используя фиксацию ротора при остановках. При отключении функции обесточивания обмоток контроллер отрабатывает только задержки между переключением тормоза и началом/остановкой движения.

Все настройки магнитного тормоза можно изменять онлайн и тормоз будет переключаться в такой режим, который был бы если бы настройка всегда имела новое значение. Например, значительное увеличение задержки срабатывания тормоза, когда тормоз уже сработал, приведёт к тому, что тормоз снова будет отведён и по достижению новой задержки от момента остановки снова сработает. Так же можно отключать и включать сам магнитный тормоз или функцию запитывания обмоток.

Таблица 4.6: Электрические параметры вывода

Тип	ТТЛ
Тормоз отжат	3.3 В
Тормоз зажат	0 В
Рабочий ток	не более 4 мА

Настройка магнитного тормоза в программе XILab описана в *Настройка тормоза*.

4.5.2.2 Схема подключения магнитного тормоза

Для работы с магнитным тормозом используется специальная плата, управляемая цифровым сигналом. Одноосные и двухосные системы, оборудованные такой платой и, соответственно, имеющие возможность работать с магнитным тормозом, поставляются отдельно (см. *ниже*).

4.5.2.2.1 Плата контроллера

Контакт, отвечающий за управление магнитным тормозом, расположен на *разъеме BPC*.

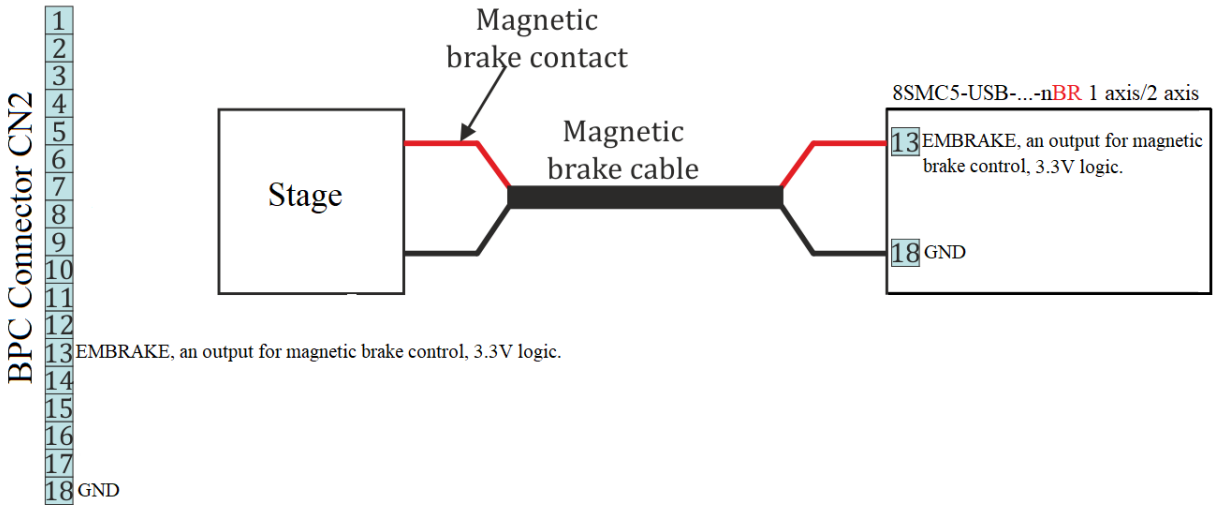


Рис. 4.25: Схема подключения магнитного тормоза к плате контроллера

Power converter - это преобразователь цифровых сигналов в силовые. При высоком уровне на ножке Magnetic brake output, на контакт магнитного тормоза подвижки подаётся 24В, при низком - с него убирается напряжение. В простейшем случае это схема, построенная с использованием транзистроного ключа и диода.

Важно: Гарантируется, что магнитный тормоз будет работать от Power supply PS36-4.4-4

4.5.2.2.2 Одноосная и двухосная система

Для того, чтобы использовать магнитный тормоз, необходимо, чтобы система с контроллером была оборудована специальной платой-преобразователем. Модели, отвечающие этому требованию, можно опознать по буквам **BR** в названии, например **8SMC5-USB-B8-1BR**.v

Контакт, отвечающий за управление магнитным тормозом в коробочных версиях контроллера, расположен на *26 пиновом D-SUB разъеме*. Схема подключения для двух разных систем указана ниже. *Двухосные системы* поставляются только с одной осью, которая может работать с магнитным тормозом.

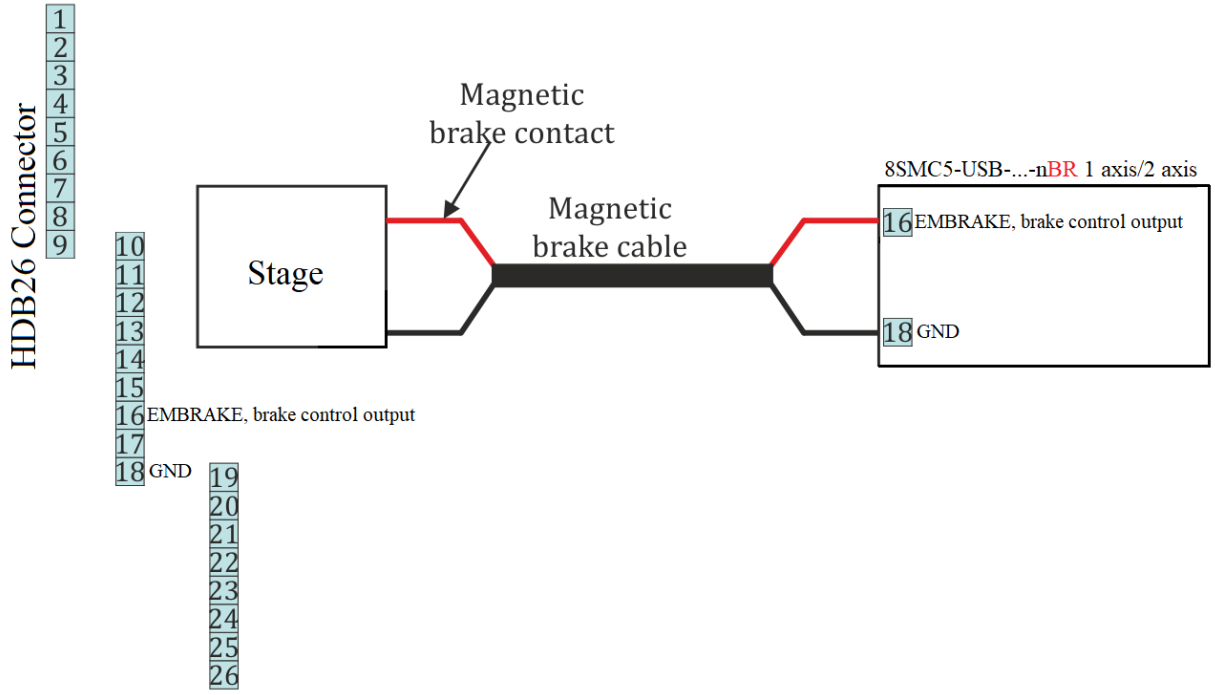


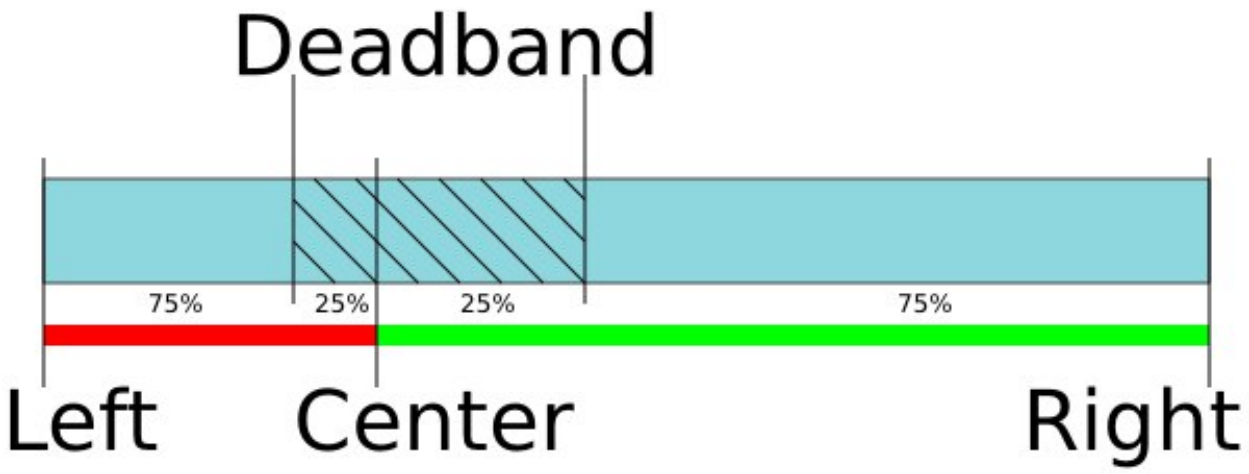
Рис. 4.26: Схема подключения магнитного тормоза к одноосной или двухосной системе

4.5.3 Управление с помощью джойстика

4.5.3.1 Основная информация

Контроллер позволяет работать с джойстиком, который выдает аналоговое напряжение в диапазоне 0-3.3 В. При этом напряжение в равновесном (центральном) положении, а также напряжения максимального и минимального отклонений могут быть заданы любыми в рабочем диапазоне напряжений, соблюдая условие: минимальное отклонение < центральной позиции < максимальное отклонение. Контроллер использует числовое представление напряжения на выходе джойстика: 0 В контроллер сопоставляет значению 0, а напряжению 3.3 В - 10000.

Для того, чтобы движение могло остановиться в центральном положении, предусмотрена мертвая зона DeadZone, отсчитываемая от центрального положения и измеряемая в процентах. Внутри DeadZone контроллер вызывает остановку движения. При отклонении джойстика от центрального положения, выводящем его из DeadZone начинается движение со скоростью, определяемой отклонением джойстика от границы DeadZone до максимального отклонения. Диапазоны DeadZone показаны на следующем рисунке.



Связью направления движения джойстика и его отклонения можно управлять с помощью флага реверса, что может быть удобно для соответствия: «отклонение вправо» - «движение вправо», - независимо от физической ориентации джойстика и подвижной части.

Скорость движения экспоненциально зависит от отклонения джойстика. Это позволяет небольшими отклонениями достигать высокой точности подводки позиции, а сильными отклонениями джойстика вызывать быстрые перемещения. Параметр нелинейности (Exp factor) можно менять. При установке параметра нелинейности в 0 скорость движения двигателя начинает линейно зависеть от отклонения джойстика.

На графике приведён пример зависимости скорости движения от отклонения джойстика для следующих настроек:

Центральное отклонение	4500
Минимальное отклонение	500
Максимальное отклонение	9500
Мёртвая зона	10%
Максимальная скорость движения	100

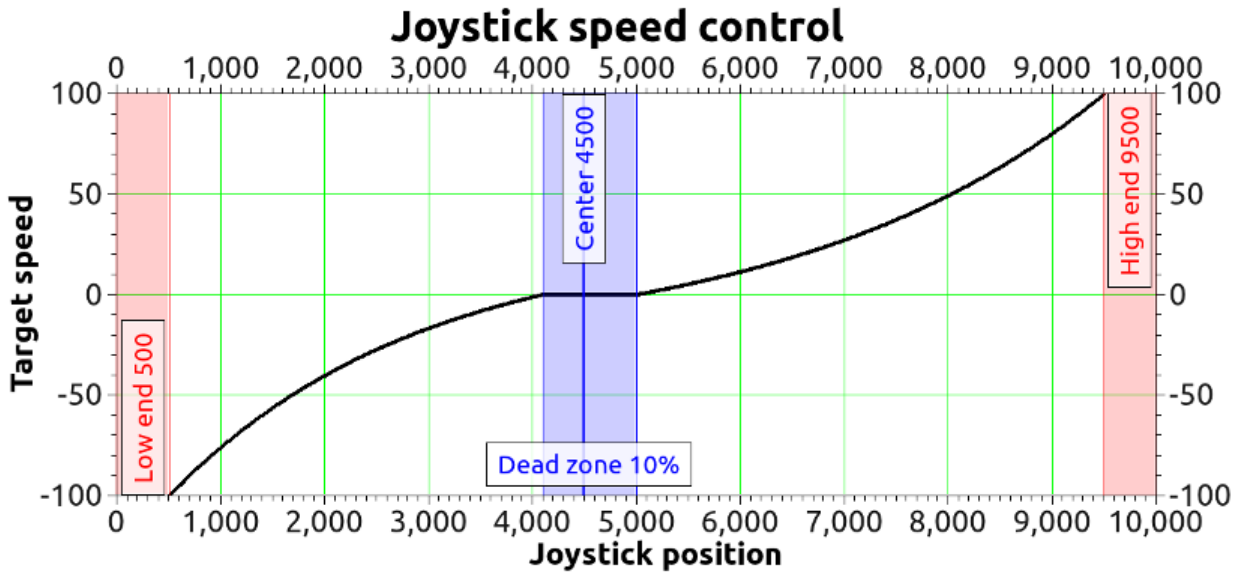


Рис. 4.27: Пример зависимости скорости движения от отклонения джойстика

Экспоненциального отклика джойстика, сочетающего высокую точность и скорость, может быть недостаточно. Поэтому контроллер поддерживает таблицу максимальных скоростей, между которыми можно переключаться кнопками управления «влево»/ «вправо» (см. главу *Управление кнопками «вправо» и «влево»*). В этом случае при нажатии на кнопку «вправо» скорость, соответствующая 100% отклонения, меняется с *MaxSpeed*[i] на *MaxSpeed*[i+1]. Если нажата кнопка «влево», максимум скорости меняется с *MaxSpeed*[i+1] на *MaxSpeed*[i]. При старте контроллера i=0. Количество скоростей в таблице - 10. Если *MaxSpeed*[x] равна нулю (целая и дробная части), то перейти на эту скорость с *MaxSpeed*[x-1] нельзя. Это сделано для возможности ограничить таблицу меньшим количеством скоростей. Попытка выйти за границы индекса таблицы скоростей (0-9) также ни к чему не приводит.

Контроллер подавляет дребезг контактов на кнопках управления. Для срабатывания кнопок длительность нажатия должна превышать 3 мс.

Если джойстик находится внутри мёртвой зоны более 5 секунд, то он не будет считаться вышедшим из неё, пока он не пробудет вне *DeadZone* более 100 мс. Это позволяет отпустить джойстик и быть уверенным, что даже случайный шум на выходе джойстика не приведет к ненужным сдвигам мотора. Пока джойстик находится внутри *DeadZone* контроллер способен принимать любые команды с компьютера, в том числе и команды движения, калибровки домашней позиции и т.п. Если при выполнении команды джойстик выводится из DeadZone, то команда движения отменяется и двигатель начинает подчиняться управляющему воздействию джойстика. Это позволяет включить режим управления двигателем с помощью джойстика, но не пользоваться им без надобности. А при касании джойстика он перехватит управление.

К режиму джойстика применимо все, что относится и к движению под воздействием управляющих команд: подчинение ускорению, ограничение максимальной скорости, режимы отключения обмоток при простое, работа с магнитным тормозом, компенсация люфта и т. д. Например, если резко бросить ручку джойстика внутрь DeadZone, то, при включении соответствующих режимов, контроллер плавно замедлит двигатель, отъедет в сторону для компенсации люфта, остановит двигатель, зафиксирует вал двигателя магнитным тормозом, плавно снизит ток и отключит питание обмоток.

Изменение параметров MaxSpeed[i], DeadZone описано в разделе *Настройка внешних управляющих устройств*.

Важно: В режиме управления джойстиком физические и виртуальные кнопки остаются в рабочем

состоянии

Предупреждение: Не отсоединяйте и не подсоединяйте джойстик к включенному контроллеру! При отключении/подключении джойстика к включенному контроллеру джойстик или контроллер не сгорят, но подвижки, подключенные к контроллеру, начнут движение к концевым выключателям.

4.5.3.2 Схема подключения

Важно: Аналоговые входы для подключения джойстика расчитаны на диапазон 0-3.3 В. Будьте внимательны и не подавайте на контакты джойстика напряжение больше 3.3 В.

4.5.3.2.1 Плата контроллера

Контакты джойстика на плате контроллера расположены на *разъеме BPC*. Обратите внимание, что джойстику необходимо питание +3В.

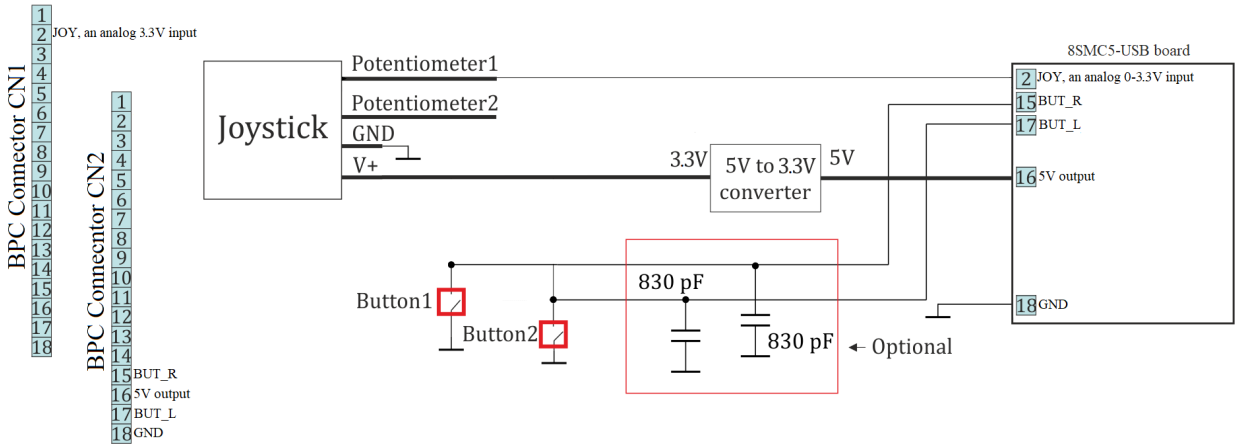


Рис. 4.28: Схема подключения джойстика к плате контроллера через разъём BPC

4.5.3.2.2 Одноосная и двухосная система

Разъём для подключения джойстика есть только в *двухосной системе*. Схема подключения к нему представлена ниже.

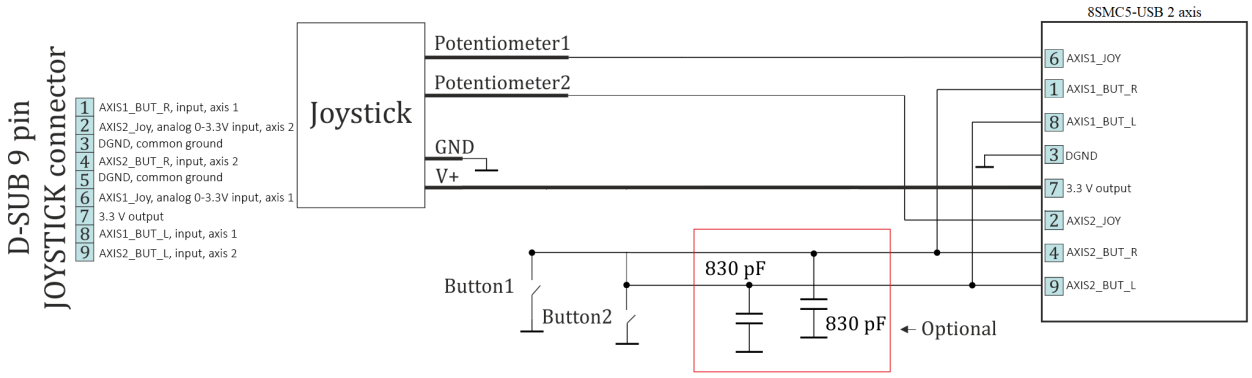


Рис. 4.29: Схема подключения джойстика к двухосной системе в корпусе через разъём D-SUB9

4.5.4 Управление кнопками «вправо» и «влево»

Для каждой системы существует возможность управлять движением моторов при помощи кнопок. Контроллер поддерживает таблицу из 10 скоростей движения MaxSpeed[0-9], которые используются и для управления *джойстиком*, и при управлении кнопками.

Настройки кнопок передаются/считываются командами SCTL/GCTL (*set_control_settings*/*get_control_settings*).

- При кратковременном нажатии (менее MaxClickTime) на кнопку вправо или влево мотор сдвигается на заданное расстояние, если DeltaPosition и uDeltaPosition отличны от нуля.
- При длительном нажатии одной из кнопок, по истечении времени MaxClickTime контроллер запускает движение со скоростью MaxSpeed[0] и начинает отсчитываться таймаут Timeout[0]. По истечению каждого таймаута Timeout[i] скорость меняется на с MaxSpeed[i] на MaxSpeed[i+1].
- При одновременном нажатии двух кнопок контроллер совершает *остановку с замедлением*. Удержание двух кнопок в течении 3 секунд запускает *автокалибровку домашней позиции*.

Примечание: Если вся таблица из 10 скоростей не нужна, то достаточно заполнить только её верхнюю часть. Контроллер не будет менять скорость на следующую, если она равна нулю или если таймаут, который для этого надо отсчитать, равен нулю. Например, если MaxSpeed[0] и MaxSpeed[1] ненулевые, а MaxSpeed[2] равно нулю (включая микрошаговую часть), то контроллер начнёт движение на скорости MaxSpeed[0], перейдёт на скорость MaxSpeed[1] и продолжит движенее с крайней скоростью до момента отпускания кнопки. Для той же функциональности можно сделать Timeout[1] равным нулю, величина скорости MaxSpeed[2] не будет иметь значения. Движение мотора подчиняется настройкам движения (за исключением устанавливаемой скорости). Например, при переходе от MaxSpeed[i] на MaxSpeed[i+1] мотор может ускоряться до достижения нового значения скорости или менять её скачком, если ускорение отключено.

По умолчанию состояние кнопки задаётся уровнями напряжений согласно таблице *Параметры вывода*. Состояние каждой кнопки может быть программно инвертировано. При активном состоянии кнопка считается нажатой. Не имеет значения каким образом состояние становится активным (после изменения настройки инвертирования состояний или же при смене уровня напряжения при физическом воздействии на кнопку). Контроллер использует программное подавление дребезга контактов на кнопках. Кнопка считается нажатой, если активное состояние на входе кнопки длилось более 3-х миллисекунд.

Таблица 4.7: Параметры вывода

Тип	ТТЛ уровень
Логический нуль	0 В
Логическая единица	3.3 В

Предупреждение: Если при включении контроллера или его перезагрузке на входе кнопки присутствует уровень напряжения, который считается активным, то контроллер воспримет это как сигнал нажатия кнопки и начнёт подчиняться *правилам описанным выше*.

4.5.4.1 Схема подключения

4.5.4.1.1 Плата контроллера

К плате контроллера могут быть подключены кнопки управления («вправо», «влево») через *разъём BPC*.

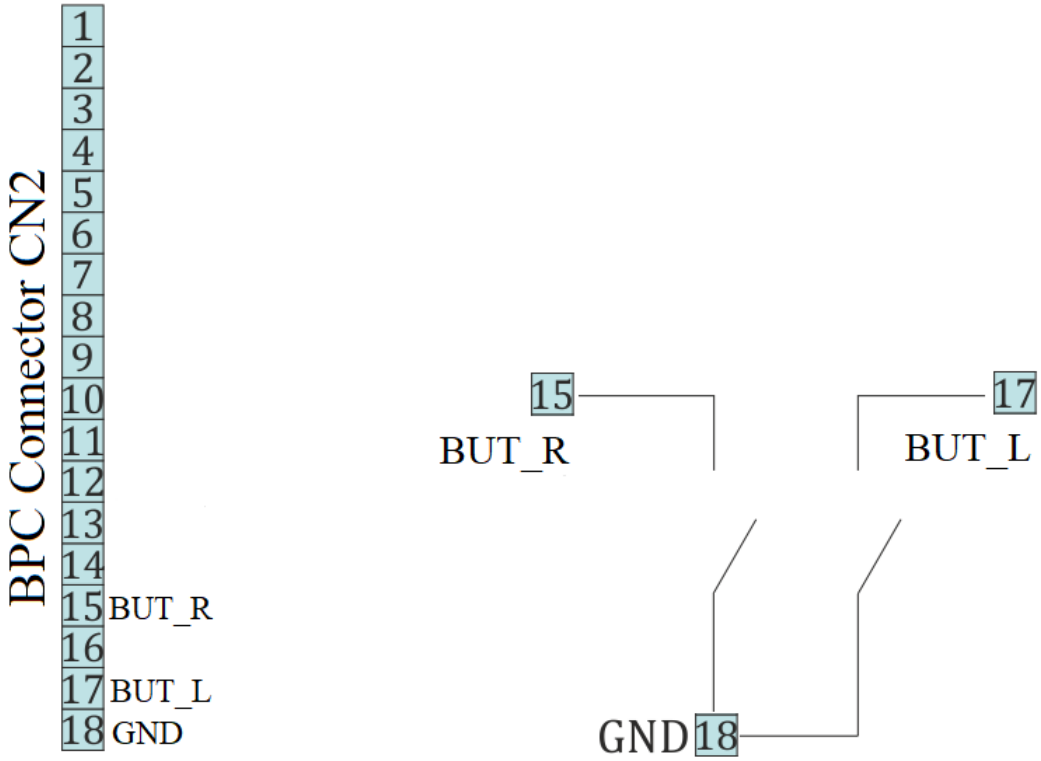


Рис. 4.30: Схема подключения кнопок к разъёму BPC в плате контроллера

4.5.4.1.2 Одноосная или двухосная система в корпусе

Для контроллеров в корпусе кнопки уже выведены наружу. Однако есть возможность подключить свои кнопки управления к соответствующим контактам. Они находятся в *D-SUB 9 разъёме* и есть только в *двухосной* системе. Схема подключения приведена ниже.

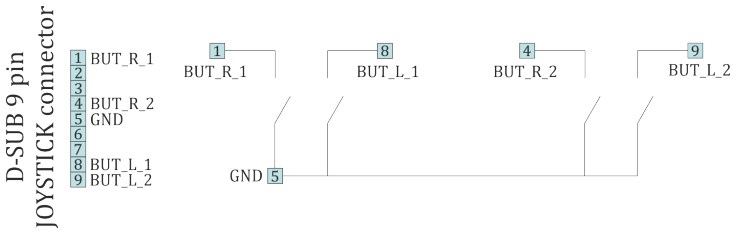


Рис. 4.31: Схема подключения кнопок к разъёму двухосной системы

4.5.5 ТТЛ-синхронизация

4.5.5.1 Принцип работы

ТТЛ-синхронизация предназначена для синхронизации производимых контроллером движений с внешними устройствами и/или событиями. Например, контроллер может каждый раз при перемещении на заданное расстояние выдавать импульс синхронизации, запускающий какое-либо измерение. И наоборот, при получении импульса синхронизации от внешнего устройства, например означающего, что экспериментальная установка готова к перемещению в следующую измерительную позицию, контроллер может выполнить смещение на заранее заданное расстояние.

Для работы с входными сигналами синхронизации, генерируемыми с помощью механических контактов, предусмотрена защита от дребезга контактов. Можно установить минимальную длительность входного импульса, после которого сигнал синхронизации считается полученным. По умолчанию активным считается состояние логической единицы (см. *Параметры вывода*), а запускающим фронтом - нарастающий. Если такая логика работы входа и выхода синхронизации не подходит, то её можно инвертировать.

Таблица 4.8: Параметры вывода

Тип	ТТЛ уровень
Логический нуль	0 В
Логическая единица	3.3 В

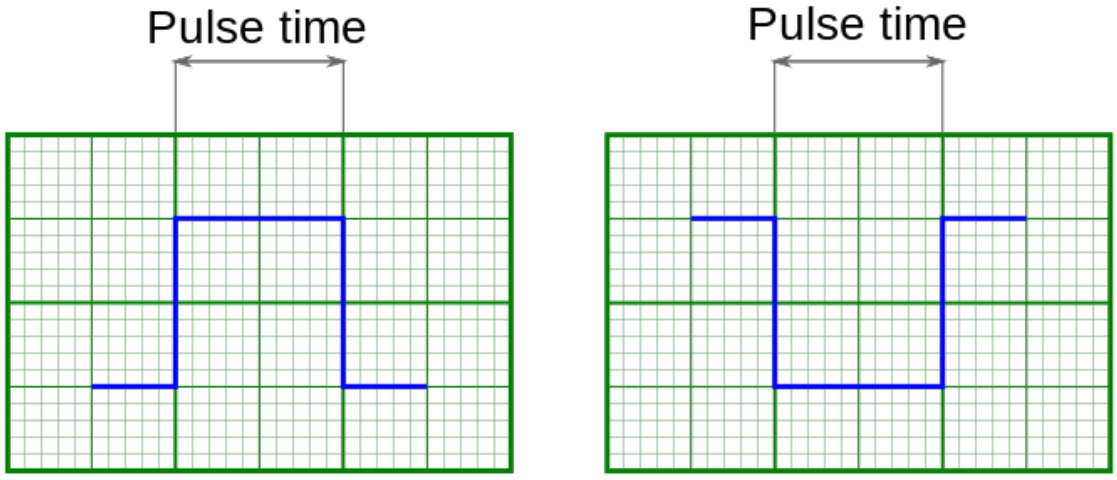


Рис. 4.32: Иллюстрация инвертирования входного или выходного импульса

Примечание: Для одновременности старта многоосных систем минимальная длительность входного сигнала должна быть одинаковой у всех контроллеров. Не следует использовать подавление дребезга механических контактов в системах, где такого эффекта нет, но есть короткие наводки на линию входа синхронизации. Вместо этого достаточно добавить RC цепь, фильтрующую такие фантомные входные импульсы.

Синхронизация важна при создании многоосных систем, так как позволяет нескольким осям начать движение в один и тот же момент времени, для этого все оси подготавливают к началу движения, во всех ведомых осях устанавливают режим начала движения по входному синхроимпульсу, одну ось, ведущую, настраивают на отправку импульса синхронизации, при начале движения. Выход синхронизации ведущей оси соединяют со входами ведомых. После такой настройки и подключения, движение ведущей оси вызывает мгновенный отклик и начало движения всех ведомых.

Примечание: При таком соединении необходимо установить минимальную длительность входного импульса синхронизации на 0. Это отключает защиту от дребезга входного сигнала, но в описанной конфигурации механических контактов нет, следовательно нет и дребезга. Если минимальная длительность входного сигнала не равна нулю, то чтобы избежать неодновременного старта движения ведущей и ведомых осей, необходимо поставить эту длительность одинаковой у всех контроллеров, соединить выход синхронизации не только со входами ведомых контроллеров, но и со входом ведущего контроллера, а дальше подавать запускающий импульс ручным переключением состояния выхода синхронизации.

Вход и выход синхронизации полностью независимы друг от друга и иных способов управления движением. Так управление движением через XILab (см. *Главное окно* программы XILab в режиме управления одной осью) или пользовательскую программу, управление от джойстика (см. *Управление с помощью джойстика*), от кнопок ручного управления (см. *Управление кнопками «вправо» и «влево»*), происходит независимо от состояния входа и выхода синхронизации. Всегда выполняется принцип «приоритет у команды пришедшей позднее». То есть, например, команда движения, поданная через XILab, отменит выполняемое по импульсу входной синхронизации движение, но не повлияет на работу выходной синхронизации, а приход входного синхроимпульса, при соответствующей настройке, отменит текущее движение, инициированное пользовательской программой, заменив его на движение, определяемое настройками работы синхровхода.

Примечание: Настройки синхронизации могут быть сохранены энергонезависимой памяти контроллера, в этом случае, все, что касается работы синхронизации будет относиться и к случаю автономной работы контроллера т.е. вы можете, например, настроить смещение на заданное расстояние по приходу синхроимпульса с выдачей синхроимпульса по завершению смещения, подключить контроллер к измерительной установке, начинающей измерение по входному синхросигналу и выдающему синхроимпульс по завершению измерения и запустить такую измерительную систему без компьютера. После поступления первого импульса, измерения и смещения будут производиться автоматически без участия компьютера.

4.5.5.2 Подключение

В плате контроллера предусмотрены два ТТЛ-канала синхронизации на *разъеме BPC*.

4.5.5.3 Вход синхронизации

Для входа синхронизации имеется настройка минимальной длительности входного синхроимпульса, который может быть зарегистрирован. Эта длительность задается в микросекундах. Используйте эту

настройку для увеличения помехоустойчивости контроллера. Вход синхронизации может быть включен или выключен. Если он включен, то переход из не активного состояния в активное приводит к движению аналогичному выполнению команды *Смещение на заданное расстояние*, в которой значение смещения задается знаковым *Position* с микрошаговой частью, а скорость *Speed* с микрошаговой частью. Изменение настроек входа синхронизации во время выполнения движения не приводит к изменению параметров движения (скорости, целевой координаты). Эти параметры будут применены при приходе следующего фронта активного состояния на вход синхронизации. Это сделано специально, чтобы при выполнении согласованного многокоординатного движения можно было запрограммировать следующий сдвиг для каждой оси во время выполнения текущего сдвига. Тогда не потребуется останавливать движение осей каждый сдвиг.

Предупреждение: Если при включении контроллера или его перезагрузке на входе синхронизации присутствует уровень напряжения, который считается активным, то контроллер воспримет это как сигнал для инициирования движения аналогичного выполнению команды *Смещение на заданное расстояние*.

Примечание: *Position* и *Speed* - отдельные переменные, которые могут быть сохранены в энергонезавиимой памяти контроллера, используются только при работе синхровхода.

Примечание: Движение по входному импульсу синхронизации подчиняется настройкам ускорения, максимальной скорости, и другим подсистемам, связанным с движением. Неправильная их настройка может мешать согласованному синхронному движению в многоосной системе.

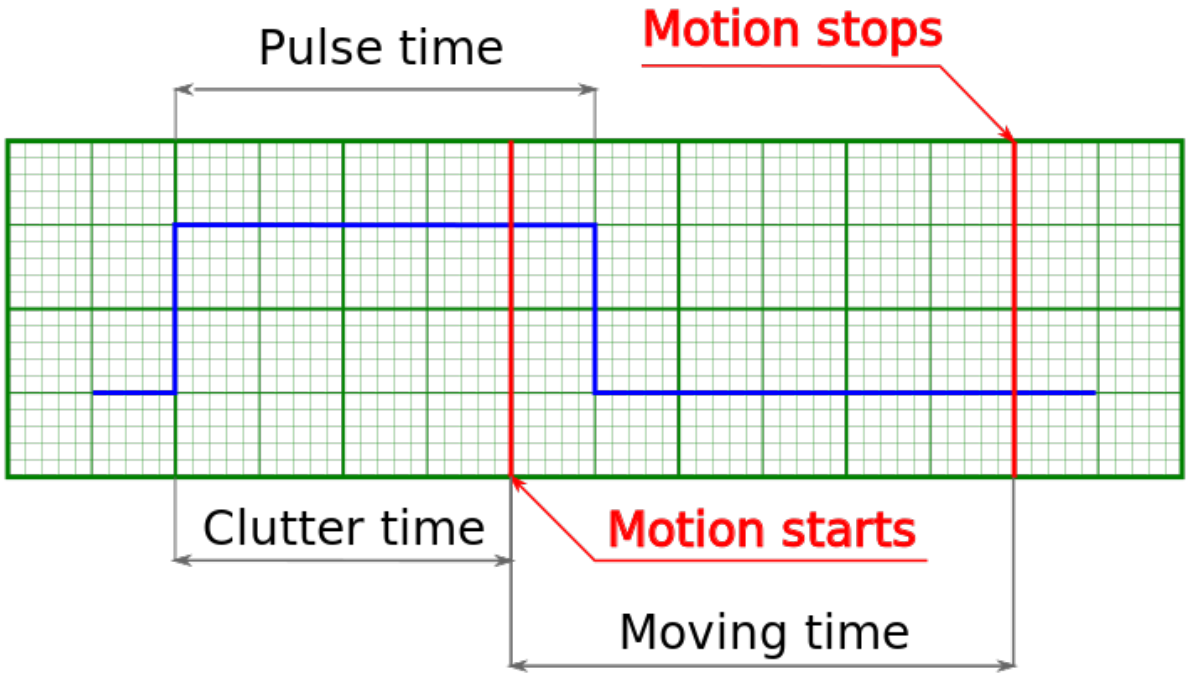


Рис. 4.33: Движение начнётся раз входной импульс дольше времени подавления дребезга

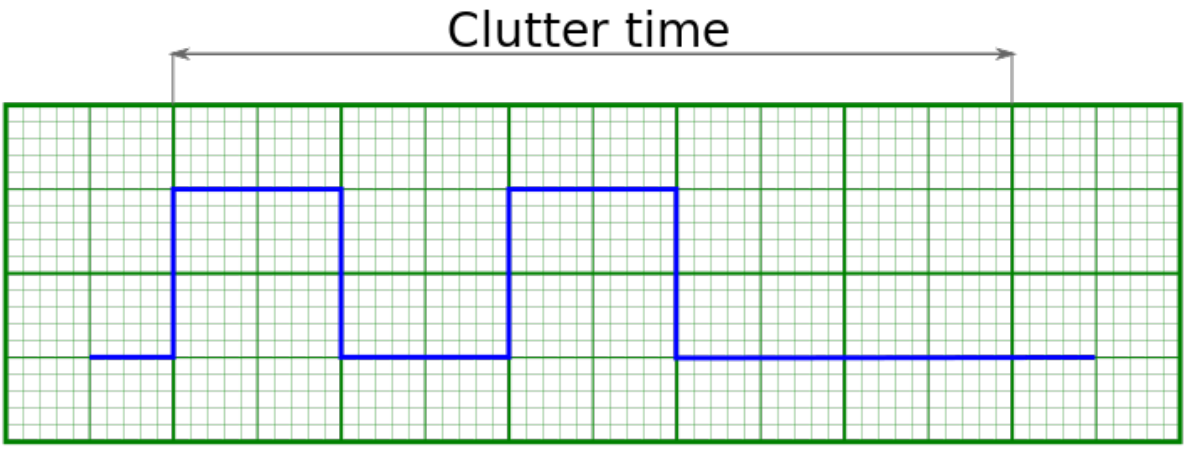


Рис. 4.34: Движение не начнётся так как входные импульсы короче времени подавления дребезга

Предупреждение: Если во время исполнения сдвига пришел еще один входной синхроимпульс, то смещение будет произведено на удвоенное значение, если два - на утроенное и т.д.

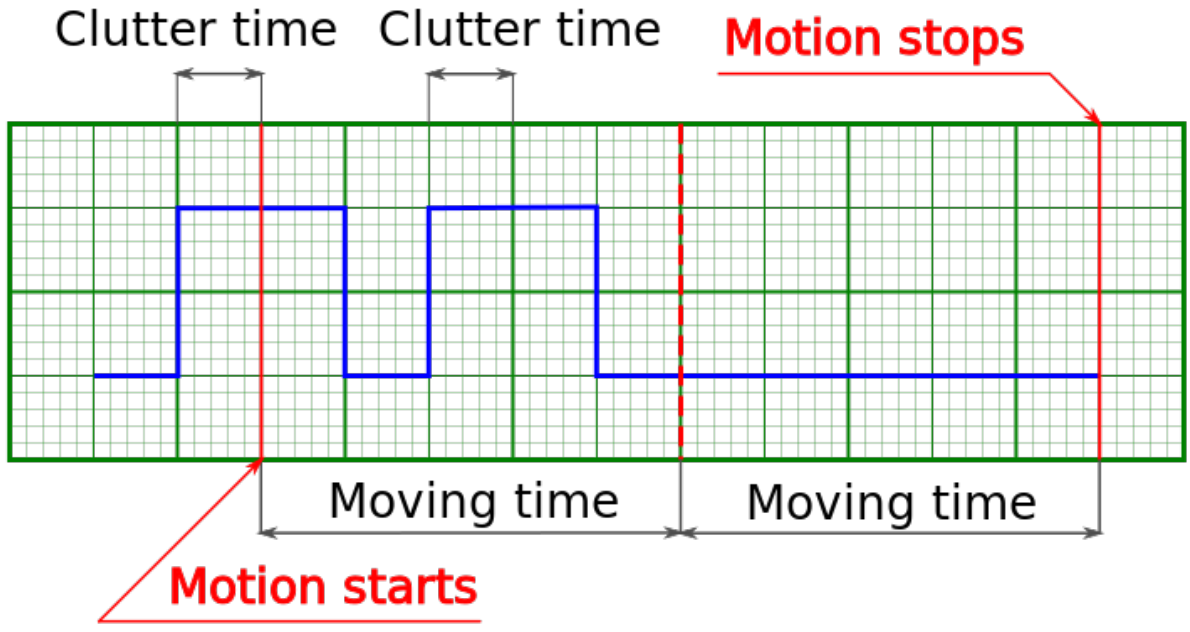


Рис. 4.35: Движение произойдёт один раз на двойное расстояние так как второй импльс синхронизации сработал до окончания первого движения

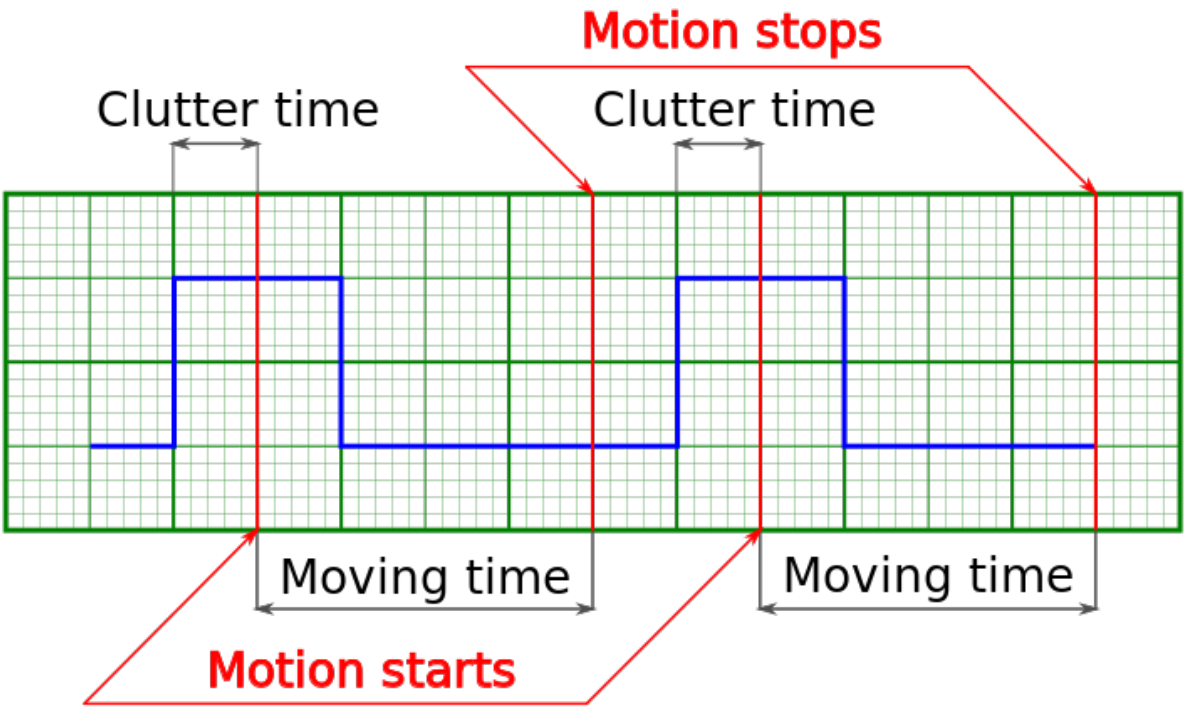


Рис. 4.36: Движение произойдёт два раза с двумя стартами и двумя остановками

По умолчанию активным состоянием считается единичное состояние, а сигнал начала движения это нарастающий фронт. Вход синхронизации может быть инвертирован. При этом активным будет считаться нулевое состояние, а сигналом начала движения спадающий фронт.

Примечание: Инвертирование входа синхронизации приводит к изменению понятий активного и неактивного состояний, проявляющейся, например, в *статусе контроллера*. Однако программное инвертирование само по себе не может являться сигналом начала движения, даже если при этом произошёл переход в активное состояние.

4.5.5.4 Выход синхронизации

Выходная синхронизация используется для управления внешними устройствами, привязанными к определенным событиям движения. Выходной синхроимпульс может подаваться при начале движения и/или при завершении движения, и/или каждый раз при смещении позиционера на заданное расстояние. Настройка ImpulseTime определяет длительность импульса синхронизации (может быть указана в микросекундах или смещении). Выход синхронизации может быть переведен в режим управляемого цифрового выхода. В этом режиме программным способом можно устанавливать единичный или нулевой логический уровень на выводе.

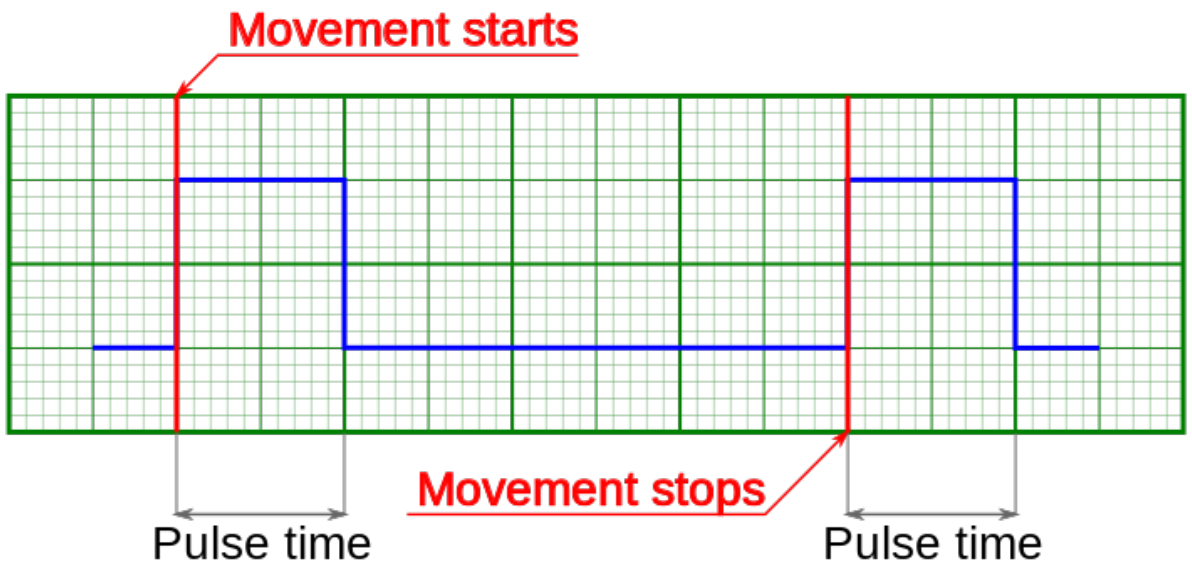


Рис. 4.37: Выходные импульсы синхронизации при генерации их на старте и на остановке движения (импульс фиксированной длительности)

Примечание: Если длительность синхроимпульса выражена в единицах смещения, например 10 шагов шагового двигателя, и поставлен режим «подавать синхроимпульс при завершении движения», то логический уровень на выходе синхронизации будет подан по завершению движения, но снят будет только после 10-ти шагов следующего движения.

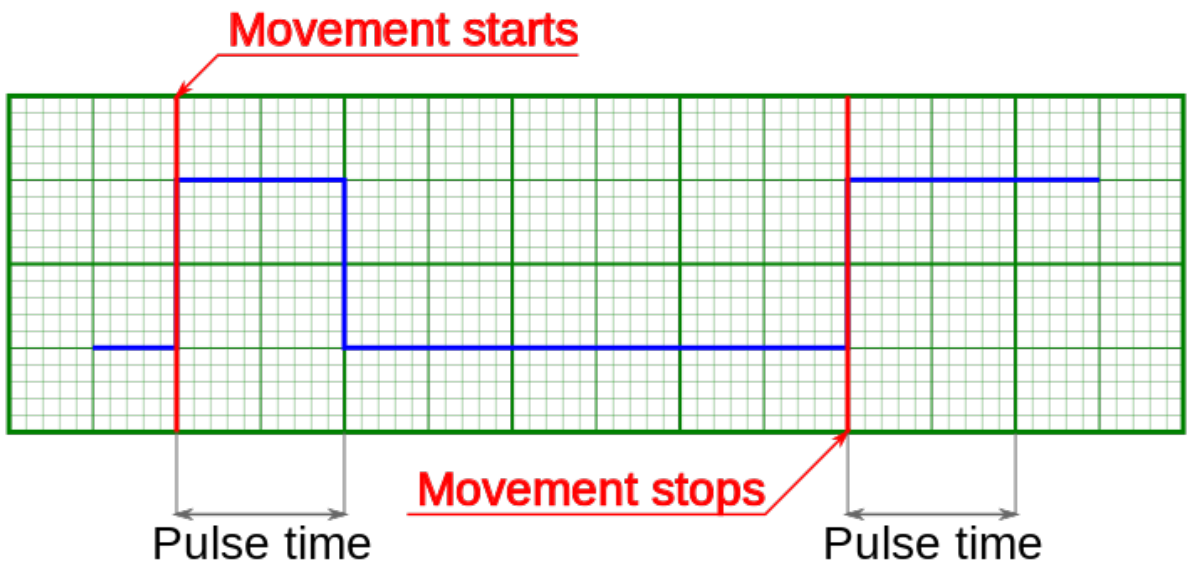


Рис. 4.38: Выходные импульсы синхронизации при генерации их на старте и на остановке движения (импульс измеряется в единицах смещения)

Примечание: Если вы хотите перенастроить синхровыход и не уверены, что знаете в каком состоянии он находится, переведите его в режим выхода общего назначения и установите желаемый логический уровень.

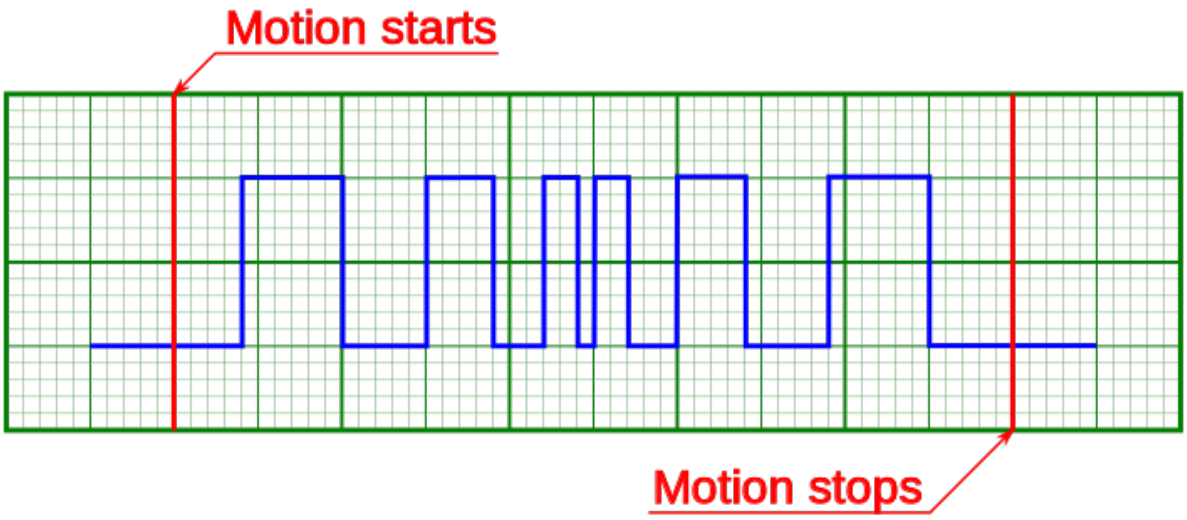


Рис. 4.39: Выходные импульсы синхронизации при движении с ускорением и генерации импульсов каждое смещение на заданное расстояние (импульс измеряется в единицах смещения)

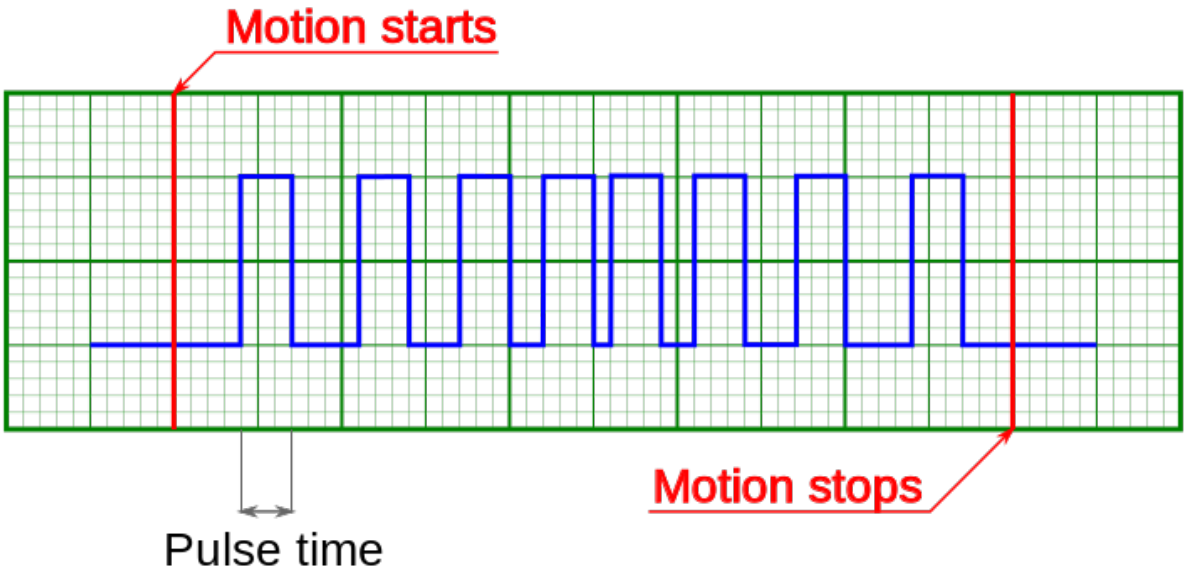


Рис. 4.40: Выходные импульсы синхронизации при движении с ускорением и генерации импульсов каждое смещение на заданное расстояние (импульс измеряется в микросекундах)

Примечание: Периодическая генерация импульсов работает, как имитатор датчика полного оборота

с передаточным числом. Координаты, в которых происходит генерация импульсов отсчитываются от нуля координат, а не от координаты в момент начала движения. Например, если в настройках включена генерация импульсов каждые 1000 шагов, то импульсы будут генерироваться при переходе через точки 0, 1000, 2000, 3000, и т.д. Генерация импульсов происходит при движении в обоих направлениях. Импульс генерируется в момент, когда частное от деления текущей координаты на период изменяется на единицу. Т.е. генерация импульсов при достижении координаты 1000 при движении от меньшей координаты к большей и при покидании координаты 1000 при движении от большей координаты к меньшей. Так же импульс всегда генерируется при переходе в точку 0 из какой-либо другой координаты (в том числе при обнулении координаты кнопкой ZERO).

Примечание: В случае если импульсы на выходе синхронизации накладываются друг на друга, то они сливаются в один импульс.

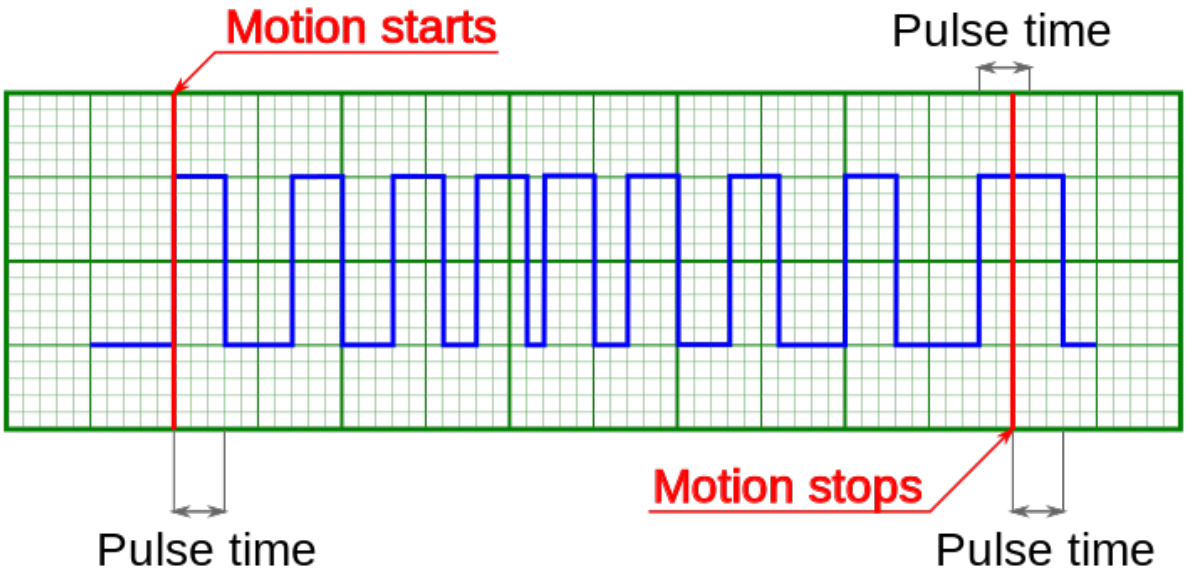


Рис. 4.41: Иллюстрация наложения импульсов синхронизации по старту и по остановке движения, смещение на заданное расстояние (длительность импульса, измеренная в микросекундах)

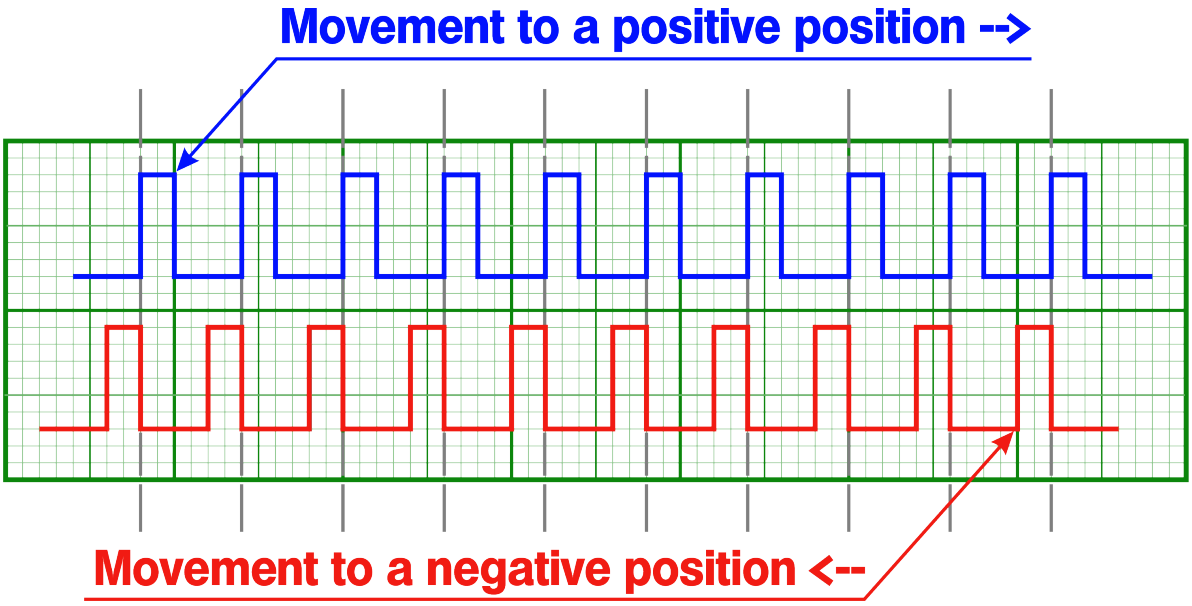
Начиная с нулевой координаты контроллер будет устанавливать виртуальные метки с заданным шагом, соответствующим значению поля «Every». Синхронизирующий импульс всегда генерируется после прохождения следующей метки. Следовательно, положение импульсов зависит от направления движения:

- При движении в положительном направлении (положение увеличивается, импульсы генерируются в направлении движения), то есть они превышают положение метки
- При движении в отрицательном направлении (положение уменьшается, импульсы генерируются в направлении движения), то есть они меньше положения метки

Пример: Pulse width: 100 Флаг Every: 1000 Контроллер установит виртуальные метки: . . . , -2000, 1000, 0, 1000, 2000, . . .

- При переходе от -1500 до 1500 на выходе будет логическая единица при прохождении следующих координат: [-1000, -900], [0, 100], [1000, 1100]

- При движении в обратном направлении от 1500 до -1500 будет логическая единица при прохождении следующих координат: [1000, 900], [0, -100], [-1000, -1100]



Важно: При коротких перемещениях в пределах длительности импульса вокруг метки состояние выхода может не возвращаться в логический ноль, чтобы не создавать лишние шумы при переключении. Флаг «Every» не был рассчитан на одиночные сдвиги, он создан для генерации импульсов на большие расстояния

Абсолютная погрешность длительности импульса - ±25 мкс. Если установить длительность импульса в 30 мкс, то фактическая длительность импульсов будет меняться примерно от 5 мкс до 55 мкс. Величина погрешности от длительности импульса не зависит. Поэтому относительная погрешность для более длинных импульсов будет значительно меньше.

Настройка параметров синхронизации в XILab описана в разделе *Настройки синхронизации*.

4.5.5.5 Схема подключения

4.5.5.5.1 Плата контроллера

В *плате контроллера* предусмотрены два ТТЛ-канала синхронизации на *разъеме BPC*.

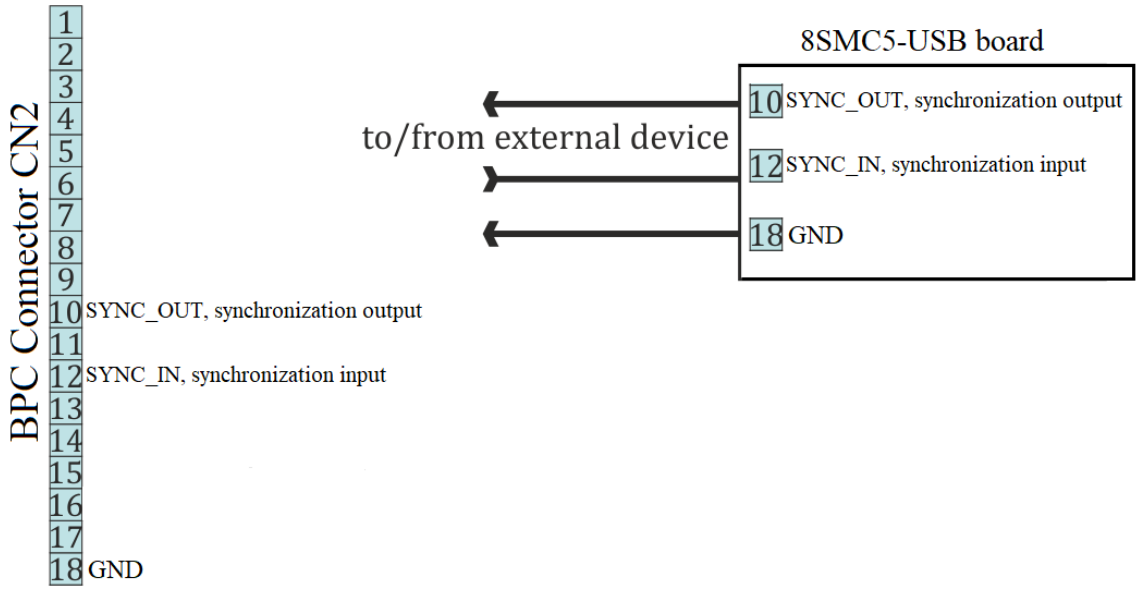


Рис. 4.42: Схема подключения к каналам синхронизации на плате контроллера

4.5.5.5.2 Одноосная и двухосная система

Сигналы синхронизации на *одноосной* и *двухосной* системах выведены на *разъём HDB-26*.

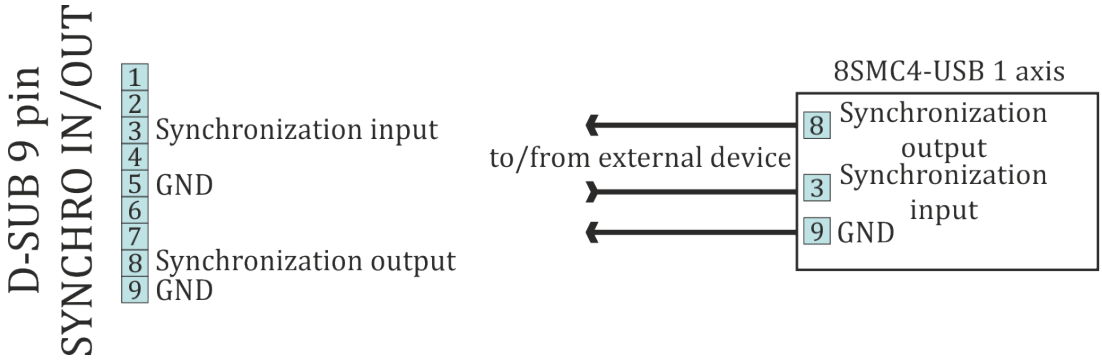


Рис. 4.43: Схема подключения к каналам синхронизации на одноосной системе

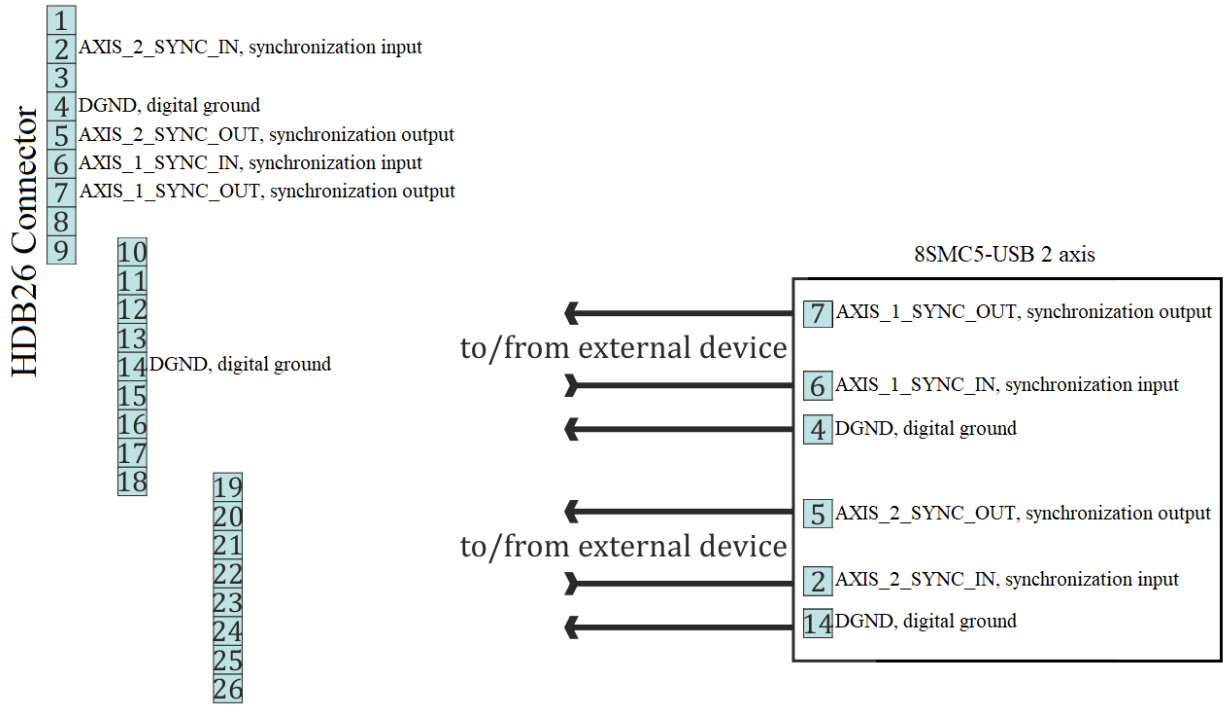


Рис. 4.44: Схема подключения к каналам синхронизации на двухосной системе

4.5.6 Создание многоосных систем

Идентификация осей в составе многоосных систем осуществляется по серийному номеру контроллера. Каждый контроллер имеет свой уникальный серийный номер, который отображается в ПО XILab во вкладке *О контроллере*. Получение серийного номера контроллера возможно с помощью функции **get_serial_number** (см. *Руководство по программированию*).

Многоосные системы на базе данного контроллера строятся с использованием активной соединительной платы на основе USB-хаба или внешнего USB-хаба.

Важно: Мы рекомендуем строить соединительные платы на базе USB-хаба с гальванической развязкой от компьютера и дополнительным источником питания или преобразователем 36 В -> 5 В, так как такая схема обеспечивает повышенную помехозащищенность и гарантированно обеспечивает достаточное электропитание по 5В.

Для правильного функционирования многоосных конфигураций плат необходимо первым действием подключить все контроллеры многоосной конфигурации друг к другу шинами питания и USB сигналов (установка контроллеров на соединительную плату).

Далее в любом порядке выполнить все следующие действия:

- Подключение электропитания к соединительной плате
- Подключение внешних устройств
- Подключение управляющего контроллера по USB

Примечание: Замечание. Требования к дополнительному источнику питания 5 В: выходной ток - не

менее 250 мА на каждую ось, а для полной функциональности - не менее 400 мА на каждую ось.

Схема подключения плат к многоплатной конфигурации подразумевает разводку питания (VBUS, GND) к ножкам 1 и 2 разъёма BPC, а также подвод от USB хаба независимых каналов данных D-/D+ к ножкам 3 и 4 разъёма BPC соответственно.

Далее подключается силовое питание:

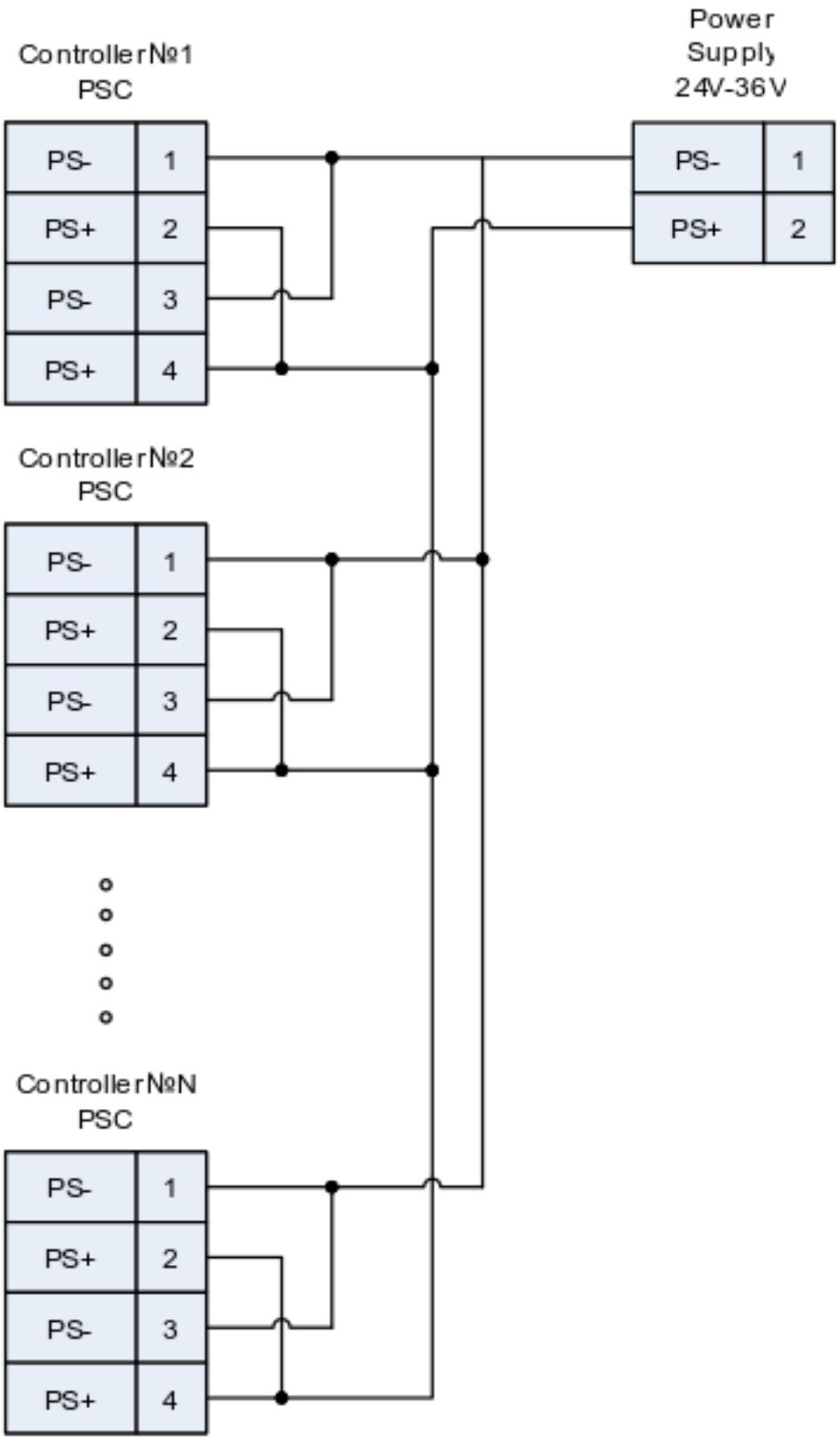


Рисунок 4.5: Схема подключения силового питания контроллеров в многоосной конфигурации

PSC - Power Supply Connector, разъем для силового питания контроллера

BPC - Back Panel Connector, разъем для подключения дополнительных устройств к контроллеру

4.5.7 Цифровой вход-выход общего назначения (EXTIO)

Выход расположен на *разъеме подключения к объединительной плате*. Цифровой вход-выход общего назначения позволяет пользователю сконфигурировать его как вход или выход. По умолчанию активным считается уровень логической единицы (см. таблицу *Параметры вывода*). Однако его можно инвертировать так, что активным будет считаться уровень нуля.

Таблица 4.9: Параметры вывода

Тип	ТТЛ уровень
Логический ноль	0 В
Логическая единица	3.3 В

В режиме входа можно либо просто получать информацию о логическом уровне на линии (см. *Статус контроллера*), либо инициировать следующие действия при переходе в активное состояние:

- Выполнить *Команда STOP* (быстрая остановка).
- Выполнить *Команда PWOF* (отключение питания обмоток).
- Выполнить *Команда MOVR* (смещение на заданное расстояние с последними использованными настройками).
- Выполнить *Команда HOME* (автоматическая калибровка позиции).
- Войти в состояние *ALARM* (отключение силовых мостов и ожидание переинициализации).

Не имеет значения, каким образом состояние входа становится активным (после изменения настройки инвертирования состояний или же при смене уровня напряжения). Контроллер использует программное подавление дребезга контакта на входе: действие по сигналу инициируется, только если активное состояние на входе кнопки длилось более 3-х миллисекунд.

Предупреждение: Если при включении контроллера или его перезагрузке на входе присутствует уровень напряжения, который считается активным, то контроллер воспримет это как сигнал для инициирования какого-либо из действий.

Примечание: Цифровой вход имеет слабую подтяжку к земле.

В режиме вывода можно устанавливать активный или неактивный логический уровень или состояния на выбор:

- EXTIO_SETUP_MODE_OUT_MOVING – Активное состояние пока мотор находится в движении.
- EXTIO_SETUP_MODE_OUT_ALARM – Активное состояние пока мотор находится в состоянии Alarm.
- EXTIO_SETUP_MODE_OUT_MOTOR_ON – Активное состояние пока питание подано на обмотки мотора.
- EXTIO_SETUP_MODE_OUT_MOTOR_FOUND - Активное состояние пока мотор подключен.

Таблица 4.10: Технические характеристики вывода

Тип логики	TTL 3.3 В
Частота обновления	1 кГц
Номинальный ток	5 мА

4.5.7.1 Схема подключения

4.5.7.1.1 Плата контроллера

Вывод расположен на разъеме *подключения к объединительной плате*

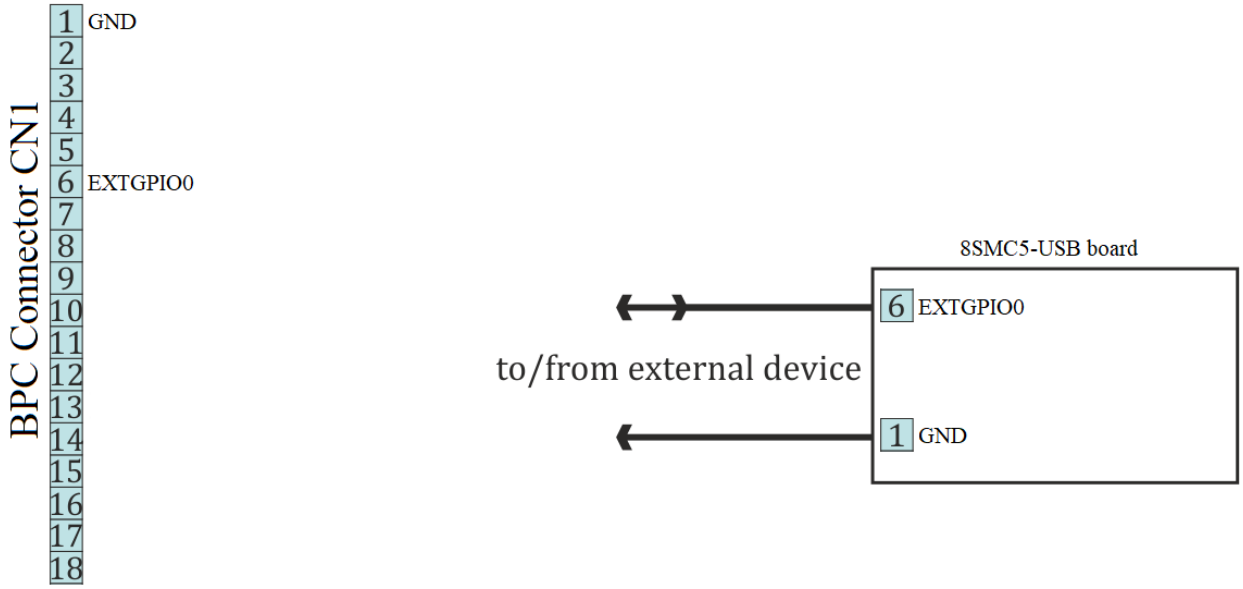


Рис. 4.46: Схема подключения к цифровому входу выходу на плате контроллера

4.5.7.1.2 Одноосная и двухосная системы

Среди двух коробочных версий, цифровой вход-выход есть только у одной из них - двухосной. Соответствующие контакты для каждой оси выведены на *HDB-26 разъём*.

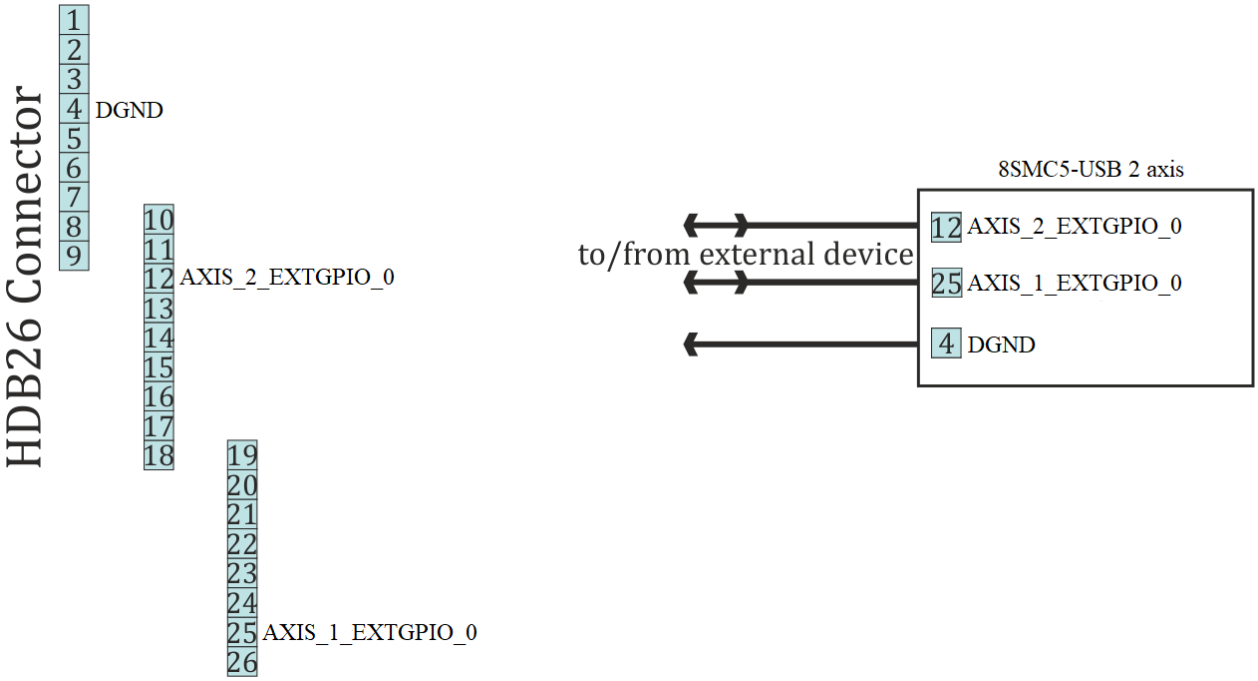


Рис. 4.47: Схема подключения к цифровому входу-выходу в двухосной системе

4.5.8 Аналоговый вход общего назначения

Аналоговый вход общего назначения можно использовать для собственных нужд. Например, для измерения каких-либо внешних сигналов. Полученное значение с аналогового входа можно считать *командой GETC* или посмотреть в *графиках XiLab*.

Для данного контроллера диапазон аналогового входа от 0 до 10000 условных отсчетов. Аналоговый вход расположен на *разъёме подключения к объединительной плате*.

Важно: Напряжение снимаемое с аналогового входа не должно выходить за пределы от 0 до 3.3 В. При превышении напряжения питания возможны ошибки в работе аналогового входа и работе других систем контроллера! Это может привести к выходу из строя как контроллера, так и подключенного к нему мотора.

Таблица 4.11: Параметры входа.

Напряжение сигнала	0-3.3 В
Частота считывания	1 кГц

4.5.8.1 Схема подключения

4.5.8.1.1 Плата контроллера

Контакт аналогового входа на *плате контроллера* расположен на *разъеме BPC*.

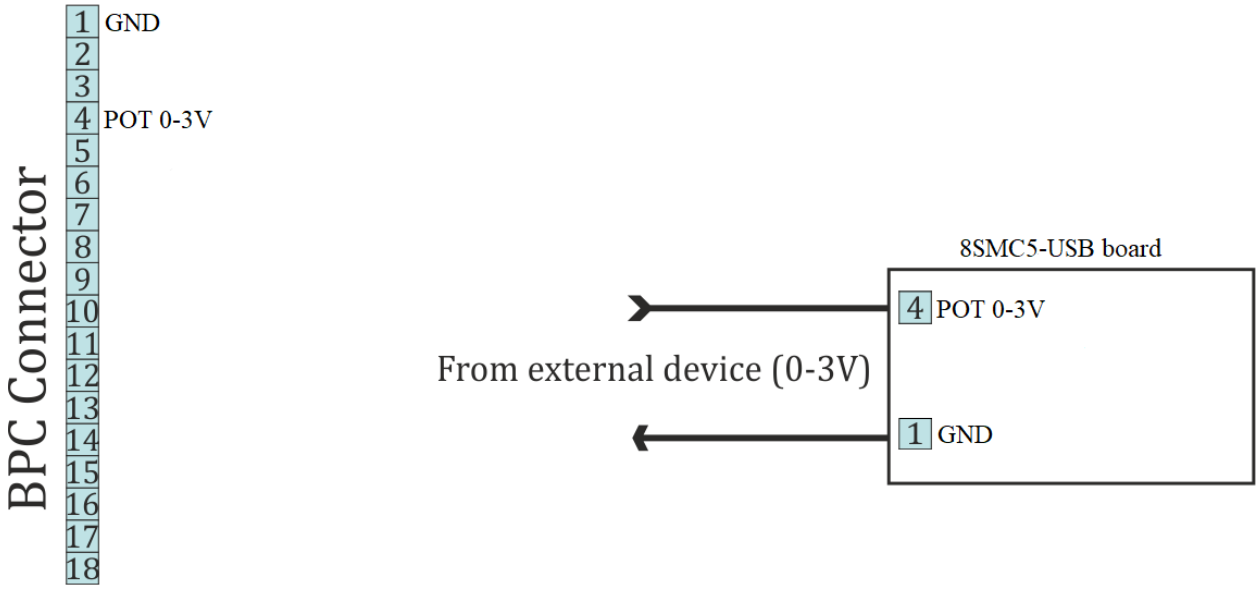


Рис. 4.48: Схема подключения к аналоговому выводу на плате контроллера

4.5.8.1.2 Одноосная и двухосная система

Аналоговый вход есть только в *двухосной системе* и расположен на *HDB-26 разъёме*.

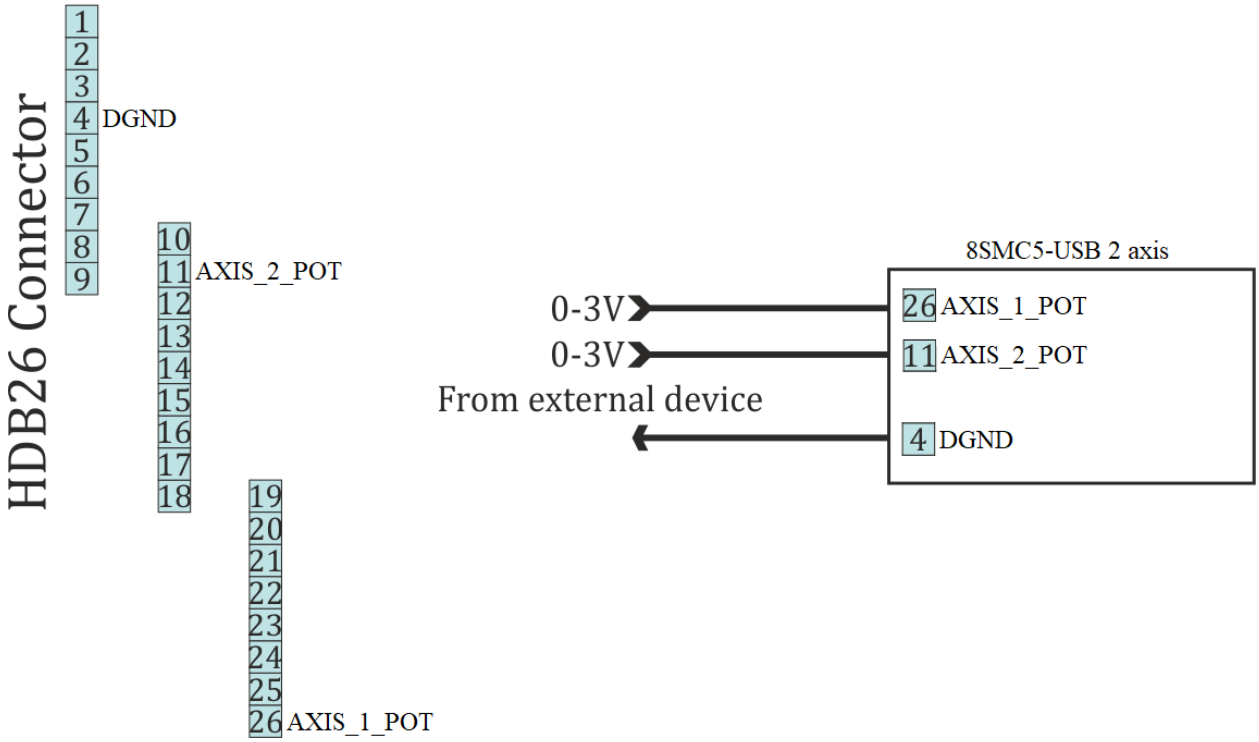


Рис. 4.49: Схема подключения к аналоговому выводу в двухосной системе

4.5.9 Интерфейс управления внешним драйвером

Интерфейс позволяет управлять любым внешним драйвером с помощью трех стандартных сигналов: enable, direction, clock. Этот режим удобен, когда силового функционала контроллера недостаточно и, при этом, хочется иметь такие его возможности, как концевики, датчик оборотов, контроль позиции, управление скриптами, создание многоосных систем, управление джойстиком или кнопками, использование магнитного тормоза, и т. п. Например, для создания многоосной системы с одной мощной подъёмной осью, которая управляется внешним контроллером, и двумя менее мощными горизонтальными осями, можно использовать XiLab с трехосным интерфейсом, скриптами, а также синхронизировать движение всех трёх осей. То есть использование внешнего драйвера заменяет только силовую часть контроллера.

Важно: Режим External driver работает только для **шаговых двигателей** и только **в режиме «None»** (xilab settings/Stepper motor tab/Feedback None) В любых других режимах или с любым другим типом двигателя режим External driver не будет работать!

Сигнал Clock определяет количество сдвигов в направлении Direction (логическая единица - вправо, логический ноль - влево). Сдвиг это минимальный шаг, при текущих настройках деления шага. Для деления шага 1/32 будет подано 32 импульса на один шаг. Не забудьте настроить внешний драйвер так, чтобы он использовал то же деление шага.

Предупреждение: Частота сигнала clock в данном контроллере ограничена 78 кГц. Поэтому для достижения необходимой скорости, требуется уменьшать деление шага. Например, если необходима скорость вращения 4000 шагов в секунду, то необходимо использовать деление шага не точнее 1/8.

Таблица 4.12: Параметры выходов управления внешним драйвером

Тип	ТТЛ
Уровень логического нуля	0 В
Уровень логической единицы	3.3 В

4.5.9.1 Схема подключения

Предупреждение: Выводы для управления внешним драйвером малозащищены и могут быть повреждены при неправильном использовании. Обеспечение правильного подключения и необходимых электрических защит лежит на инженере, конструирующем систему связи драйверов.

4.5.9.1.1 Плата контроллера

Для подключения внешнего драйвера используется три вывода на *разъеме BPC* .

Предупреждение: Вывод 13 является входом-выходом общего назначения (см. *Цифровой вход-выход общего назначения (EXTIO)*), но он теряет функциональность, когда включен режим управления внешним драйвером.

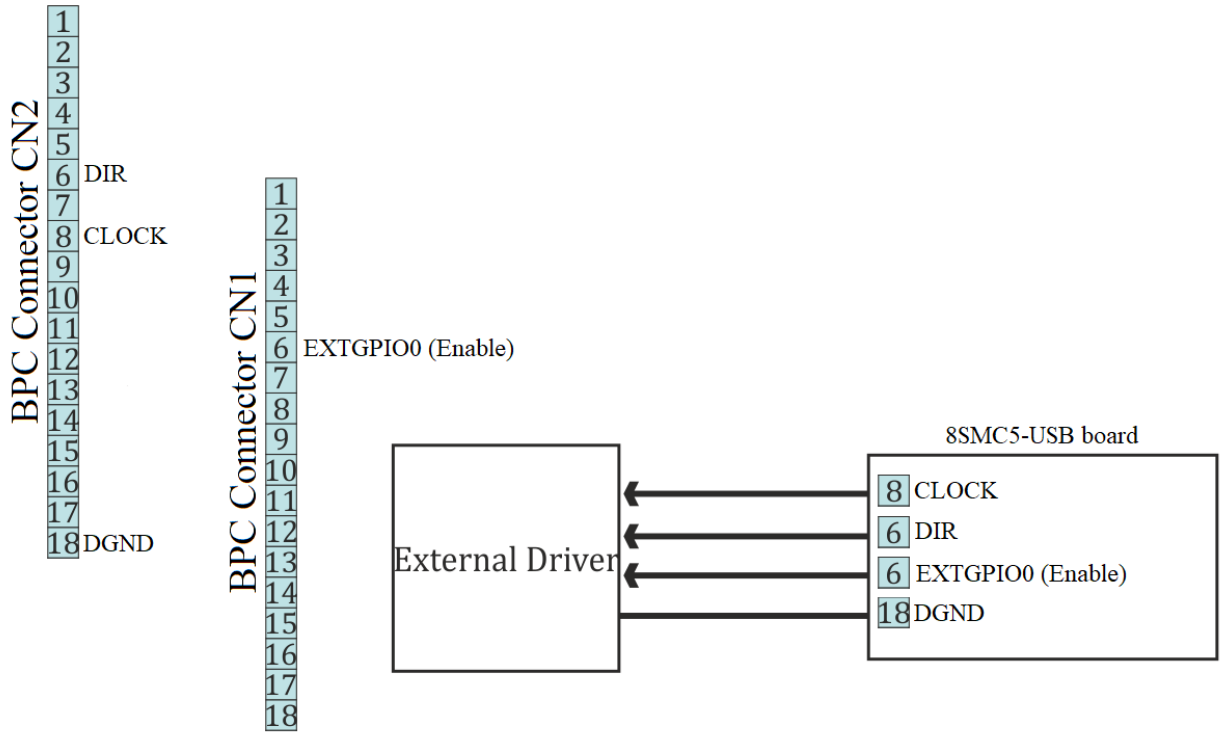
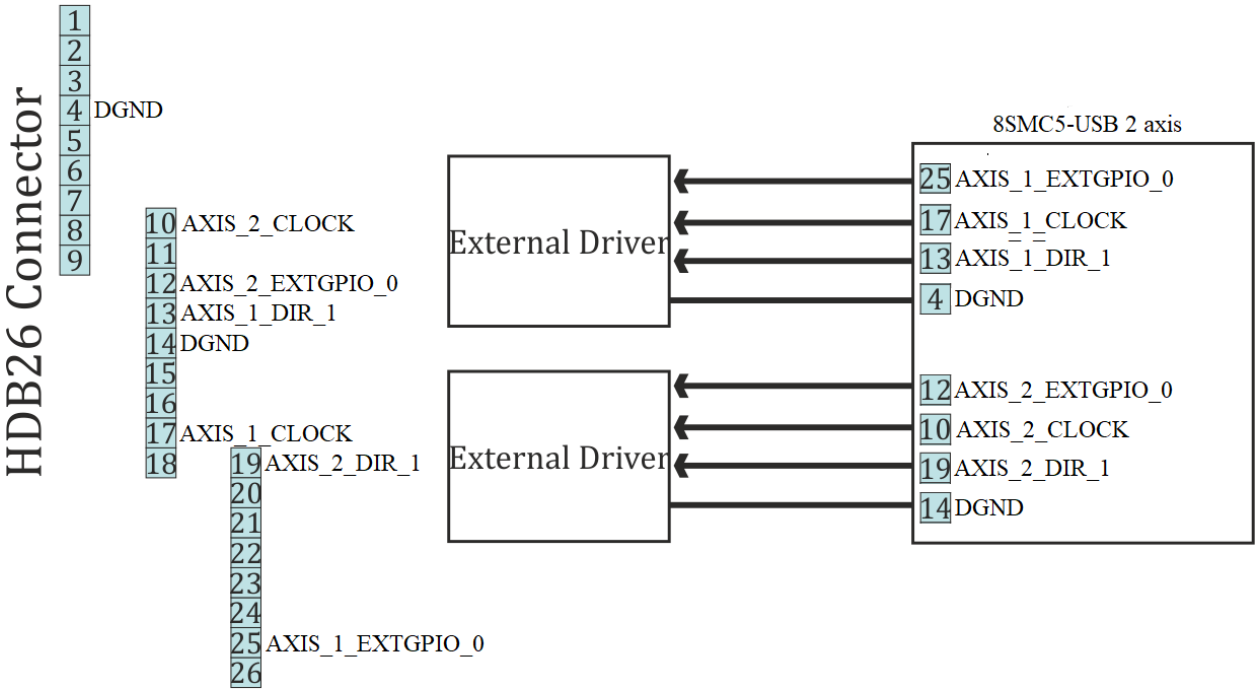


Рис. 4.50: Схема подключения к внешнему драйверу

4.5.9.1.2 Одноосная и двухосная система

Интерфейс управления внешним драйвером есть только в *двухосной системе* и выведен на разъём *HDB-26*.

Предупреждение: Выводы 12 и 25 являются входами-выходами общего назначения (см. *Цифровой вход-выход общего назначения (EXTIO)*), но они теряет свою функциональность, когда включен режим управления внешним драйвером.



4.5.10 Последовательный порт

Контроллер допускает управление по последовательному порту UART с логикой TTL 3.3 В. Выводы UART расположены на *разъеме BPC*. За счёт высокой распространённости UART и переходников на него с USB, Bluetooth, Ethernet и других стандартных интерфейсов, контроллером можно управлять беспроводным образом (Bluetooth) или по интернету (Ethernet). Протокол передачи данных по UART такой же как и по USB. То есть потребуется только включить в XiLab или в libximc опрос нестандартных последовательных портов и устройство обнаружится если задержка ответа с него не превышает 2 секунды. Можно также управлять контроллером с помощью другого самостоятельно запрограммированного микроконтроллера, хотя это потребует поддержки *протокола общения*.

UART поддерживает следующие настройки.

Скорость передачи	9600-921600 бит/с
Количество бит в передаче	8
Четность	включена или выключена
Тип четности	четное, нечетное, единица, ноль
Количество стоповых бит	1 или 2

Примечание: Чтобы установить связь с контроллером по UART нужно предварительно соединиться с ним по USB или Ethernet, установить необходимые настройки скорости, четности, количества стоповых бит, а затем сохранить их в энергонезависимую память контроллера. Стандартными для прилагаемого программного обеспечения являются настройки, описанные в *протоколе общения*. Если связь не устанавливается, то попробуйте использовать их.

Таблица 4.13: Параметры входа и выхода

Тип логики	TTL 3.3 V
Максимальная скорость передачи данных	921 кбит
Номинальный ток выхода	5 мА

4.5.10.1 Схема подключения последовательного порта

Предупреждение: Высокая скорость передачи данных невозможна по длинному кабелю (**кабель должен быть до 5 метров**) в условиях электромагнитных наводок. Если случаются ошибки в передаче данных, то используйте фильтрующие RC цепи и снизьте скорость, чтобы характерное время цепи было как минимум в 4 раза меньше времени передачи одного бита. Характерное время RC цепи подбирается индивидуально.

4.5.10.1.1 Подключение платы контроллера

В *плате контроллера* выводы UART расположены на *разъеме BPC*.

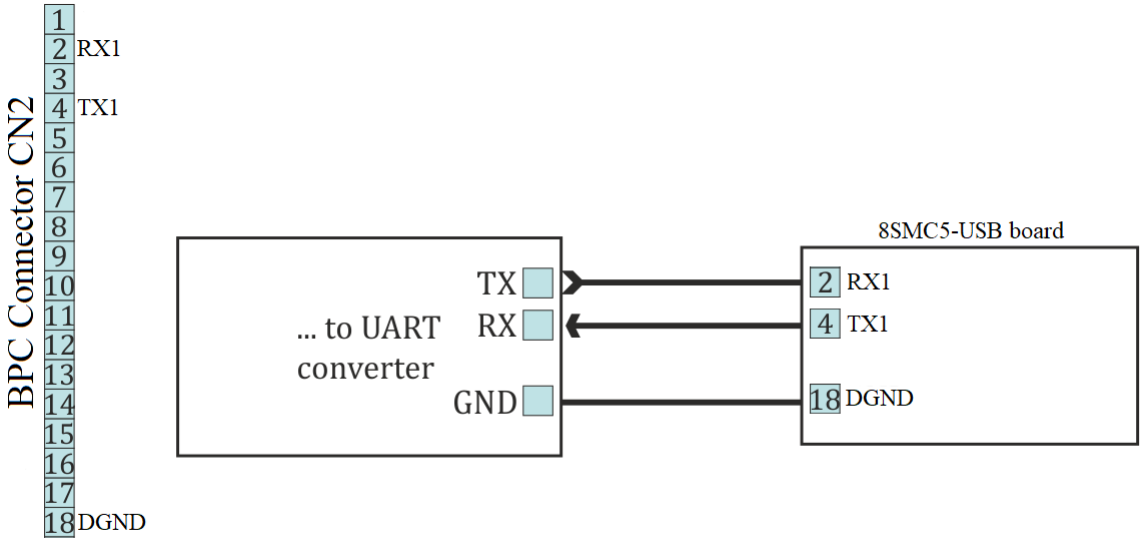


Рис. 4.51: Рекомендуемая схема подключения для платы контроллера

4.5.10.1.2 Одноосная и двухосная система

UART есть у *двухосной* и *одноосной* системы. Соответствующие контакты для каждой оси выведены на *HDB-26 разъём*.

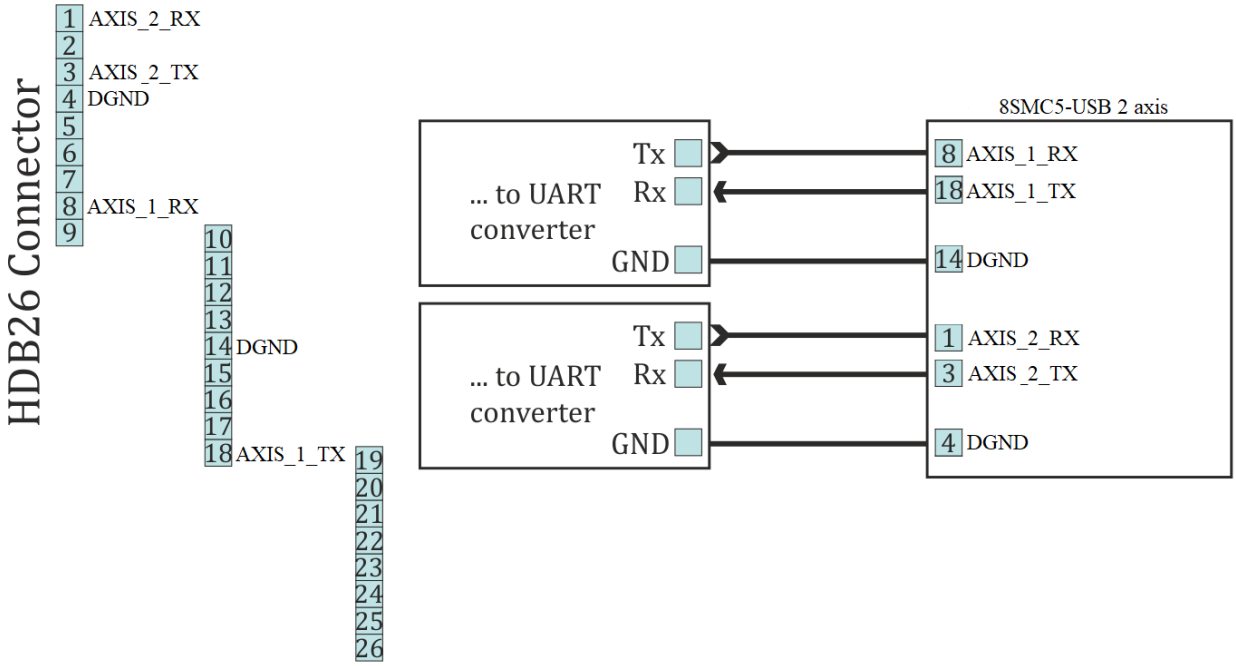


Рис. 4.52: Рекомендуемая схема подключения для двухосной системы

4.5.11 Хранение позиции во FRAM-памяти контроллера

Контроллер имеет функцию автоматического запоминания позиции, которая позволяет просто отключать ему питание после остановки и при следующем включении питания, контроллер будет удерживать то же положение двигателя, значение положения и инкрементального счётчика энкодера. Эта функция работает, если в момент, когда контроллер обесточен, не происходит вращения оси мотора.

Примечание: Для работы этой функции необходимо подождать хотя бы 0.5 секунды после остановки вращения до выключения питания контроллера. Обесточивание контроллера во время вращения также приведёт к сохранению позиции, но она будет лишь примерной и потребуется новая *калибровка*.

4.5.12 Опознавание позиционеров Standa

В новейших позиционерах фирмы Standa (список конкретных устройств уточняйте у производителя) предусмотрено хранение настроечных и информационных параметров позиционера во встроенной в подвижку памяти. Эта память заранее прошита при изготовлении подвижки верными значениями, что позволяет не заботиться об оптимальных для конкретного позиционера настройках и начать работу сразу после сборки системы. В памяти есть также и пользовательское поле имени подвижки, которое пользователь может установить сам (см. вкладку XiLab *Название позиционера*).

При подключении такого позиционера к контроллеру (более подробно об электрической схеме подключения написано в разделе *Пример подключения простого мотора* и *Разъём подключения позиционера*) осуществляется автоматическая загрузка информационных параметров подвижки, см. раздел *Характеристики позиционера*, в память контроллера. Если был установлен флаг EEPROM_PRECEDENCE (преимущество настроек считанных из памяти подвижки перед настройками во flash-памяти контроллера, см. *О контроллере*), то дополнительно считываются и устанавливаются все настройки системы, кроме настроек UART и имени контроллера.

При установленном флаге EEPROM_PRECEDENCE не нужно проверять и/или устанавливать на-

стройки позиционера (к примеру, полярность и расположение концевиков, рабочий ток, параметры энкодера и магнитного тормоза). Все это будет сделано автоматически при подключении позиционера, оборудованного встроенной памятью. Однако при установленном флаге EEPROM_PRECEDENCE загрузка настроек из памяти позиционера будет происходить каждый раз при подключении подвижки, оборудованной памятью, либо при включении питания контроллера, к которому подключена такая подвижка. Поэтому если требуется изменить какие-то настройки, то необходимо снять данный флаг, изменить необходимые настройки, сохранить их во внутреннюю память контроллера.

Примечание: Простое правило для использования этого флага: На ранних этапах работы этот флаг должен быть включен для удобства автоматических настроек. Со временем, когда захочется самостоятельно менять настройки, этот флаг стоит выключить, не забыв сохранить настройки во FRAM.

Примечание: При отключении позиционера оборудованного памятью не происходит никаких изменений настроек.

4.5.12.1 Для разработчиков

Данные подвижки хранятся в микросхеме DS28EC20, соединяемой по интерфейсу 1-wire.

При опознавании подвижки на микросхему EEPROM периодически посылается сигнал сброса. При получении от микросхемы ответного сигнала, контроллер считывает все данные подвижки в собственную оперативную память, автоматически настраивается на подвижку и выставляет бит STATE_EEPROM_CONNECTED в статусной структуре. В XILab это отображается индикатором EEPR в главном окне. Далее проверка наличия памяти производится регулярно. В случае потери связи с EEPROM (отсутствию ответа на сигнал сброса) индикатор EEPR в XiLab гаснет.

4.5.12.2 Схема подключения для проверки внешней памяти

Выводы для соединения с микросхемой на всех системах (*плата контроллера*, *одноосная* и *двухосная* в корпусе) расположены на *разъёме D-SUB*.

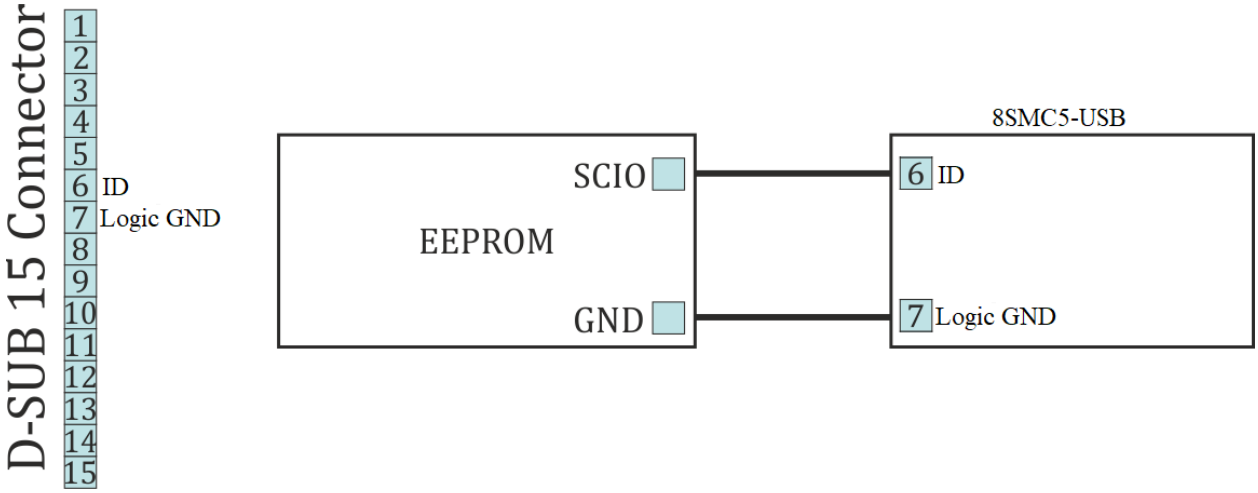


Рис. 4.53: Схема подключения для проверки внешней памяти

4.6 Второстепенные функции

4.6.1 Установка нулевой позиции

Контроллер поддерживает установку нулевой позиции. Эту функцию стоит использовать для позиционеров маркированных репером, чтобы позиция по реперу соответствовала программной. Также эта функция удобна, когда существует одна избранная физическая позиция на диапазоне перемещения.

Для установки нулевой позиции используется *специальная команда*. При этом обнуляются счётчики шагов, микрошагов, энкодера. Установка нулевой позиции происходит одновременно для всех счётчиков позиции и не может привести к их рассогласованию. Влияние на текущую команду движения не оказывается. Если контроллер отрабатывал движение в некоторую физическую позицию в момент, когда текущая позиция была обнулена, то движение завершится в прежней физической позиции. Например, при движении к позиции 1000 в момент прохода 200 была послана команда обнуления позиции. Тогда счётчик позиции уменьшится на 200 и движение завершится в координате 800.

Примечание: Установка нулевой позиции при работе в режиме смещения (см. *Смещение на заданное расстояние*) не изменит физической позиции, к которой осуществлялось движение. Следующее смещение будет происходить к той же физической позиции, что и без использования установки нулевой позиции.

4.6.2 Установка пользовательской позиции

Если необходимо установить позицию или счётчик энкодера не в ноль, а в некоторую пользовательскую позицию, то для этого существует команда SPOS. В этой команде передаются значения новых счётчиков позиции, микрошагов для шаговых двигателей и счётчика энкодера, если он является второстепенным датчиком положения. Если необходимо установить лишь одну из позиций, нужно воспользоваться флагами игнорирования части полей команды.

Отличие этой команды от обнуления позиции, в том, что не происходит обнуление последней позиции, к которой происходило смещение, нет отличия в поведении команды в момент движения и при остановке. Если команду применить в момент движения к позиции, то движение завершится в той же физической точке, в которой оно завершилось бы и без смены позиции командой SPOS.

4.6.3 Статус контроллера

Контроллер отслеживает свой статус и способен передать его в статусной структуре команды *GETS*. Статус контроллера включает в себя информацию о совершаемом движении, его результате, состоянии питания, энкодера, обмоток двигателя, цифровых входов и выходов, числовую информацию о позиции и питающем напряжении и токах, а также флаги ошибок.

4.6.3.1 Статус движения

MoveSts содержит:

- Флаг движения, который устанавливается когда контроллер меняет позицию двигателя.
- «Флаг достижения требуемой скорости», который устанавливается если скорость равна той, с которой контроллер должен отрабатывать текущее движение.
- Флаг антилюфта, который устанавливается при подавлении люфта во время заключительной стадии движения (см. *Компенсация люфта*).

MvCmdSts содержит информацию о выполняемой команде. Все движения мотора вызываются командами движения к целевой позиции *MOVE*, сдвига относительно последней целевой позиции *MOVR*,

движения вправо *RIGT* или влево *LEFT*, плавной *SSTP* или резкой *STOP* остановки, калибровки домашней позиции *HOME* или принудительного подавления люфта *LOFT*. Управление кнопками, джойстиком, импульсами синхронизации и т. п. тоже приводится к этим командам. Например, джойстик вызывает команды движения вправо и влево при отклонении или команду плавной остановки в центральном положении (см. *Управление с помощью джойстика*). В переменной MvCmdSts приведена текущая команда движения или последняя выполненная команда, а также статус команды: выпоняется она или уже выполнена. Если команда выполнена, то еще один бит показывает результат её выполнения (успешный или неуспешный). Неуспешное выполнение означает, что мы не попали к той позиции, к которой пытались двигаться или не смогли отработать люфт. Причиной может быть неожиданная остановка по концевикам, попадание в состояние Alarm. Изначальное состояние этого поля показывает неизвестную команду и статус успешного выполнения.

4.6.3.2 Статус питания мотора

PWRSts содержит информацию о питающем напряжении. Обмотки могут быть:

- Отключены (в этом случае на них не подаётся никакого напряжения).
- Запитаны сниженным током относительно номинального тока (например при использовании функции снижения тока в обмотках при остановке).
- Запитаны номинальным током.
- Запитаны недостаточным напряжением, чтобы обеспечить установленный номинальный ток.

Последний статус часто появляется при высоких скоростях вращения, ведь чем выше скорость переключения шагов, тем выше должно быть питающее напряжение, чтобы обеспечить нарастание тока в индуктивности обмоток двигетеля. Недостаточное питание не означает, что двигатель не будет вращаться, а означает, что двигатель может больше шуметь и падает крутящий момент. (См. *Управление питанием мотора*).

4.6.3.3 Статус энкодера

EncSts содержит информацию о подключенном энкодере если включен режим управления без обратной связи (например для шаговых двигателей). Состояния энкодера могут быть:

- Не подключен
- Неизвестное состояние, когда недостаточно данных чтобы определить состояние энкодера.
- Подключен и исправен.
- Подключен и реверсирован (тогда нужно включить в настройках реверс энкодера).
- Подключен и неисправен.

Последнее состояние реализуется, когда на входы энкодера поступают сигналы переключения, но они не соответствуют движению ротора двигателя. Смена состояний проходит после набора достаточной статистики. Поэтому обнаружение происходит немгновенно. Также невозможно точно определить статус энкодера без движения. (См. *Работа с энкодерами*).

4.6.3.4 Статус обмоток двигателя

WindSts содержит информацию о состоянии обмоток. Показывается состояние для каждой из двух обмоток отдельно. Они могут быть:

- Отключены от контроллера
- Быть подключены
- Замкнуты накоротко

- Их состояние можут быть неизвестно.

Замкнутым накоротко считается слишком маленькое сопротивление и индуктивность в обмотке. Отключенными обмотками считается нагрузка с слишком высоким сопротивлением.

4.6.3.5 Статус положения.

В статусной структуре выводятся все данные о положении и скорости позиционера. Для этого используются поля основной позиции (CurPosition, uStep), второстепенной позиции (EncPosition), скорости (CurSpeed, uCurSpeed). Основное положение отсчитывается в шагах шагового двигателя и микрошагах, если используется управление без обратной связи. При использовании режима ведущего энкодера

хранится счётчик положения по энкодеру, а в uStep записан 0. Поле второстепенной позиции это поле энкодера если используется управление шаговым двигателем без обратной связи, счетчик шагов, если подключен шаговый двигатель в режиме ведущего энкодера, или 0, если подключен DC двигатель. Скорость выводится всегда для основного датчика позиции и измеряется в тех же единицах, что и установленная скорость движения.

4.6.3.6 Статус питания контроллера и температура.

В статусной структуре выводятся:

- Ток потребления контроллера (в мА).
- Напряжение на силовой части (в десятках мВ).
- Температура микропроцессора (в десятых долях градуса Цельсия).

4.6.3.7 Статусные флаги

Флаги делятся на ошибки команд управления, ошибки превышения критических параметров, общие ошибки и флаги состояния.

Примечание: Многие флаги не снимаются сами, пока их принудительно не снять командой *STOP*.

Ошибки команд протокола:

- errc - Неопознанная команда протокола. Ошибка возникать не должна если используется софт, совместимый с используемой в контроллере версией протокола. Флаг не снимается самостоятельно.
- errd - Код проверки целостности данных команды не сошёлся. Ошибка возникает при сбоях передачи данных. Флаг не снимается самостоятельно.
- errv - Не удалось применить одно или несколько переданных в команде значений. Возникает когда команда была принята и успешно распознана, но передаваемые в ней данные были некорректны, выходили за допустимый диапазон. Также эта ошибка может означать, что требуюмую операцию не удалось выполнить из-за аппаратного сбоя. Например, эта ошибка возникнет при установке режима деления шага, не входящего в список поддерживаемых или при установке нулевого количества шагов на оборот двигателя. Флаг не снимается самостоятельно. Ошибки превышения критических параметров:
- Флаг, что сейчас контроллер находится в режиме Alarm.
- Флаг, который говорит о том, что сейчас силовой драйвер сигнализирует о перегреве. Флаг снимается сам в зависимости от *настроек критических параметров*.
- Флаг, который говорит о том, что температура микропроцессора вышла за допустимый диапазон. Флаг снимается сам в зависимости от *настроек критических параметров*.

- Флаг, который говорит о том, что напряжение питания превысило допустимое значение. Флаг снимается сам в зависимости от *настроек критических параметров*.
- Флаг, который говорит о том, что напряжение питания оказалось ниже допустимого значения. Флаг снимается сам в зависимости от *настроек критических параметров*.
- Флаг, который говорит о том, что потребляемый ток из блока питания превысил допустимое значение. Флаг снимается сам в зависимости от *настроек критических параметров*.
- Флаг, который говорит о том, что напряжение USB превысило допустимое значение. Флаг снимается сам в зависимости от *настроек критических параметров*. Устарело.
- Флаг, который говорит о том, что напряжение USB оказалось ниже допустимого значения. Флаг снимается сам в зависимости от *настроек критических параметров*. Устарело.
- Флаг, который говорит о том, что ток потребления по шине питания USB превысил допустимое значение. Флаг снимается сам в зависимости от *настроек критических параметров*.
- Флаг, что концевики перепутаны местами. Флаг не снимается самостоятельно.

Общие флаги ошибок:

- Флаг, что система контроля позиции обнаружила рассогласование позиции по счётчику шагов и датчику положения. Флаг не снимается самостоятельно (если не используется автоматическая корректировка позиции).

Флаги состояния:

- Наличие подключенной сейчас подвижки, оснащённой памятью EEPROM.
- Наличие внешнего питания. Иначе питание внутреннее. Установлен всегда.

4.6.3.8 Статус цифровых сигналов.

Контроллер выводит состояние входных и выходных цифровых сигналов в виде флагов активного состояния или в виде текущего логического уровня. Активное состояние соответствует единице или нулю, в зависимости от настроек конкретного блока, например от настройки инвертирования. Флаги бывают:

- Состояние правого концевика (1, если концевик активен).
- Состояние левого концевика (1, если концевик активен).
- Состояние правой кнопки (1, если кнопка нажата).
- Состояние левой кнопки (1, если кнопка нажата).
- 1, если ножка EXTIO работает как выход. Иначе - как вход.
- Состояние ножки EXTIO (1, если активное состояние на входе или на выходе).
- Состояние датчика Холла C (1, если на входе логическая единица).
- Состояние датчика Холла C (1, если на входе логическая единица).
- Состояние датчика Холла C (1, если на входе логическая единица).
- Состояние магнитного тормоза (1, если на тормоз подано питание).
- Состояние датчика полного оборота (1, если датчик активен).
- Состояние входной ножки синхронизации (1, если ножка синхронизации в активном состоянии).
- Состояние выходной ножки синхронизации (1, если ножка синхронизации в активном состоянии).
- Состояние на входе канала энкодера B (1, если на входе логическая единица).

- Состояние на входе канала энкодера B (1, если на входе логическая единица).

4.6.4 Автовосстановление соединения

Примечание: Режим переподключения шины USB/Ethernet никаким образом не влияет на основные характеристики контроллера (к примеру, движение или удержание необходимого тока в обмотках).

4.6.4.1 Автовосстановление USB соединения

Данный блок предназначен для перезагрузки USB шины в случае потери связи (к примеру, это может возникнуть в случае удара статическим разрядом или отключения шины USB без отключения питания). Включение/выключение данного блока определяется флагом *USB_BREAK_RECONNECT* (см. *Критические параметры*). Если блок включен, то он отслеживает потерю связи по шине USB. В случае потери связи по шине USB через 500 мс выполняется программное переподключение к шине USB со стороны контроллера, после чего выполняется проверка состояния шины USB. Если в течение определенного времени не происходит восстановление связи (т.е. обмена даными), то выполняется повторное переподключение. Таким образом в случае не восстановления связи по USB, контроллер будет постоянно переподключаться к шине USB до тех пор, пока не произойдет восстановления связи по USB или время между переподключениями не превысит 1 минуты. В итоге в случае отключения шины USB без отключения питания контроллера (к примеру, в случае управления двигателем от джойстика или кнопок) в течение примерно 5 минут контроллер будет находиться в режиме переподключения шины USB.

Чтобы избежать синхронного переподключения к шине USB как со стороны контроллера, так и со стороны компьютера, время между переподключениями меняется по экспоненциальному закону (см. *Задержка между переподключениями USB*).

Таблица 4.14: Задержка между переподключениями USB

Номер перезагрузки	время ожидания, мс
0 (после потери связи)	500
1	483
2	622
3	802
4	1034
5	1333
6	1718

Визуально определить состояние блока перезагрузки USB можно по частоте мигания светодиода. В случае перехода в режим перезагрузки светодиод начнет мигать с частотой 10Гц (см. *Индикация режима работы*).

Предупреждение: В силу особенностей строения данного программного блока, а также спецификации шины USB, блок не гарантирует 100% восстановление связи с компьютером после удара статическим разрядом.

Со стороны компьютера Xilab также производит попытки восстановления соединения с контроллером, если оно по каким-либо причинам было потеряно. При потере соединения, то есть если библиотека libximc вернула кода ошибки «result_nodevice», сначала происходит ожидание 1000мс. Затем, если платформа на которой запущен Xilab принадлежит семейству Windows, средствами WINAPI опрашивается наличие устройства с соответствующим именем COM-порта в системе. Если такой порт присутствует, но библиотека libximc не может открыть его более двух раз, то вызывается функция

ximc_fix_usbser_sys, которая производит ресет драйвера usbser.sys (исправление ошибки драйвера). На платформе Linux или MacOS Xilab просто пытается повторно открыть устройство каждые 1000 мс. После успешного открытия в устройство посылаются команды чтения серийного номера, версии прошивки и некоторых настроек, необходимых для отображения интерфейса.

Библиотека libximc считает устройство потерянным (result_nodevice) когда системные функции ReadFile/WriteFile (на Windows) или read/write (на Linux/Mac) возвращают ошибку при чтении или записи данных в соответствующий USB-COM порт.

4.6.4.2 Автовосстановление Ethernet соединения

Автоматическое восстановление Ethernet соединения устроено проще по сравнению с USB блоком. Время между повторными подключениями также изменяется экспоненциально.

Таблица 4.15: Задержка между переподключениями Ethernet

Номер перезагрузки	время ожидания
0 (после потери связи)	1 сек
1	2 сек
2	4 сек
3	8 сек
4	16 сек
5	32 сек
6	1 мин

После шести попыток повторного подключения запрос будет отправляться раз в минуту.

4.7 Совместимость с различным ПО

4.7.1 MicroManager

4.7.1.1 Подготовка к работе

Важно: Данное руководство для MicroManager было разработано и протестировано на версии 2.0-gamma. Руководство для более ранних версий можно найти в интернете.

- Скачайте и установите MicroManager. Это достаточно простая процедура, и на данном этапе проблем возникнуть не должно.
- Скачайте дистрибутив библиотеки libximc-2.7.6

Предупреждение: MicroManager **будет работать только** с версией библиотеки libximc-2.7.6. Он **не будет работать** с любой другой версией libximc!

- Скопируйте следующие файлы .dll из распакованной директории *ximc-2.7.6/ximc/winX/* в директорию MicroManager: *libximc.dll*, *xiwrapper.dll*, *bindy.dll*. **Битность скопированных библиотек должна совпадать с битностью установленного Микро-менеджера.**
- Установите VC++ 2013 Redistributable Package из распакованного файла *ximc-2.7.6/ximc/winX/vcredist.exe* **Битность «vcredist» должна соответствовать битности вашей ОС.**
- Подсоедините источник внешнего питания к *контроллерам* и настройте соответствующее напряжение для моторов на вашей подвижке. Включите источник питания.

- Вам понадобится XILab, чтобы проверить наличие *контроллеров* в ОС и настроить их. Версия XILab зависит от версии прошивки контроллера. Вы можете загрузить XiLab и обновить прошивку контроллеров по этой ссылке:
- Подсоедините контроллеры к компьютеру через USB. XILab должен обнаруживать их. Нажмите *Load setting from file...* в окне Settings... и выберите соответствующий профиль для вашей подвижки. Нажмите *Save settings to flash*. Это нужно, чтобы сохранить профиль в XiLab. Для получения дополнительной информации пройдите по ссылке: *Руководство по программе XILab*.

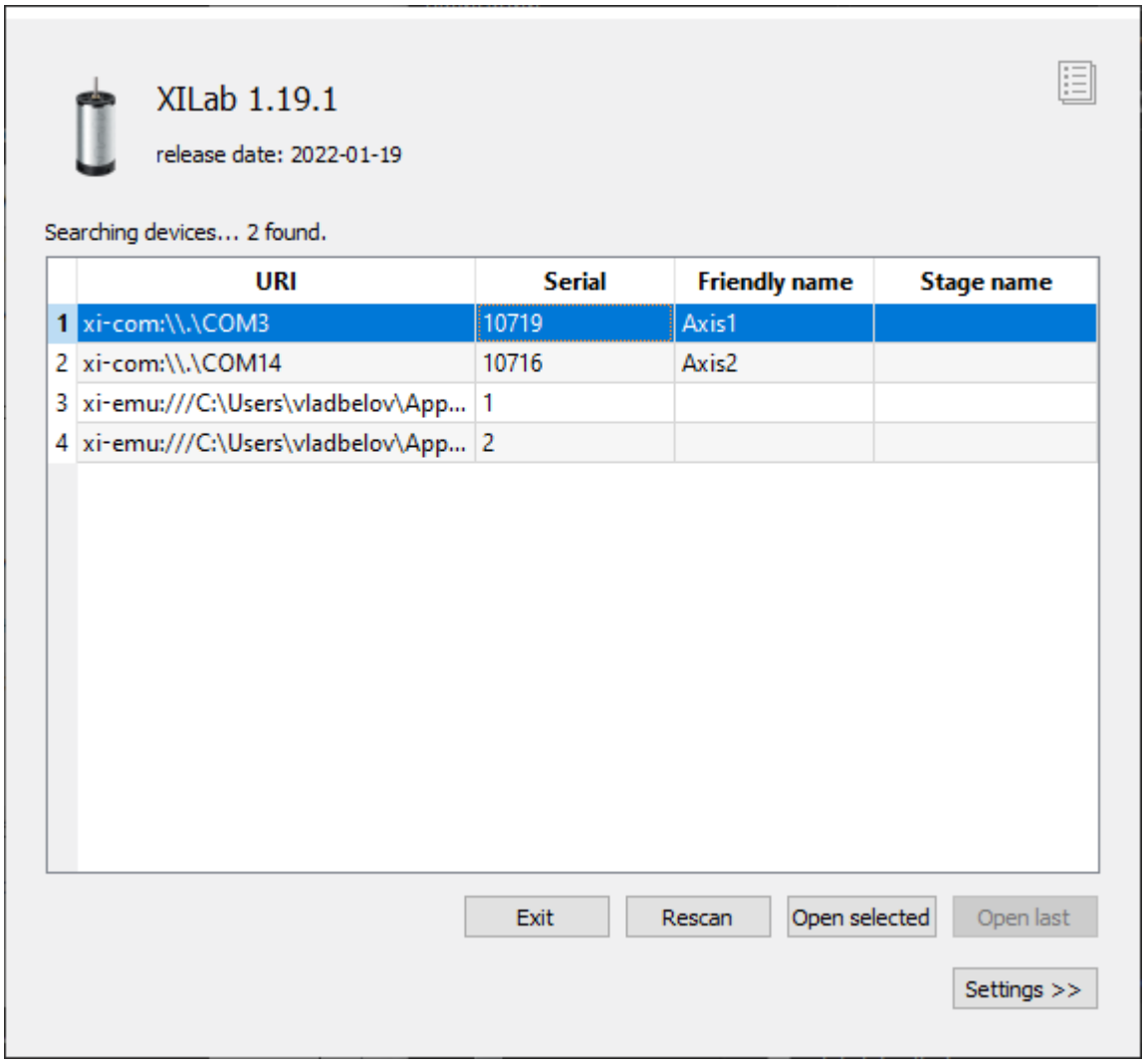


Рис. 4.54: Пример обнаружения контроллера в XILab

4.7.1.2 Начало работы с MicroManager

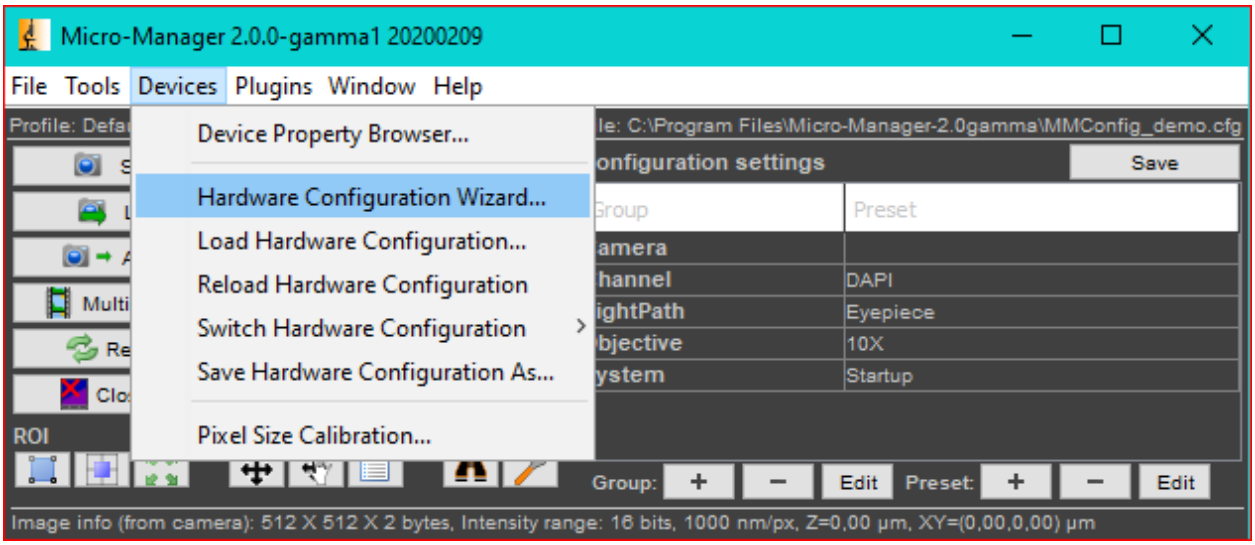
4.7.1.2.1 Запуск MicroManager

Примечание: При установке MicroManager в **C:\Program Files** доступ к файлам конфигурации ограничен и требуется обладать правами администратора. Для запуска приложения от имени администратора нажмите на него правой кнопкой мыши и выберите *Запуск от имени администратора*.

- Запустите MicroManager через ярлык на рабочем столе или *ImageJ.exe* из корневой директории программы. При первом запуске ПО MicroManager приветствует Вас и предлагает некоторую информацию о себе.
- Следующее окно содержит выпадающий список с конфигурационными файлами. Нажмите *None*.

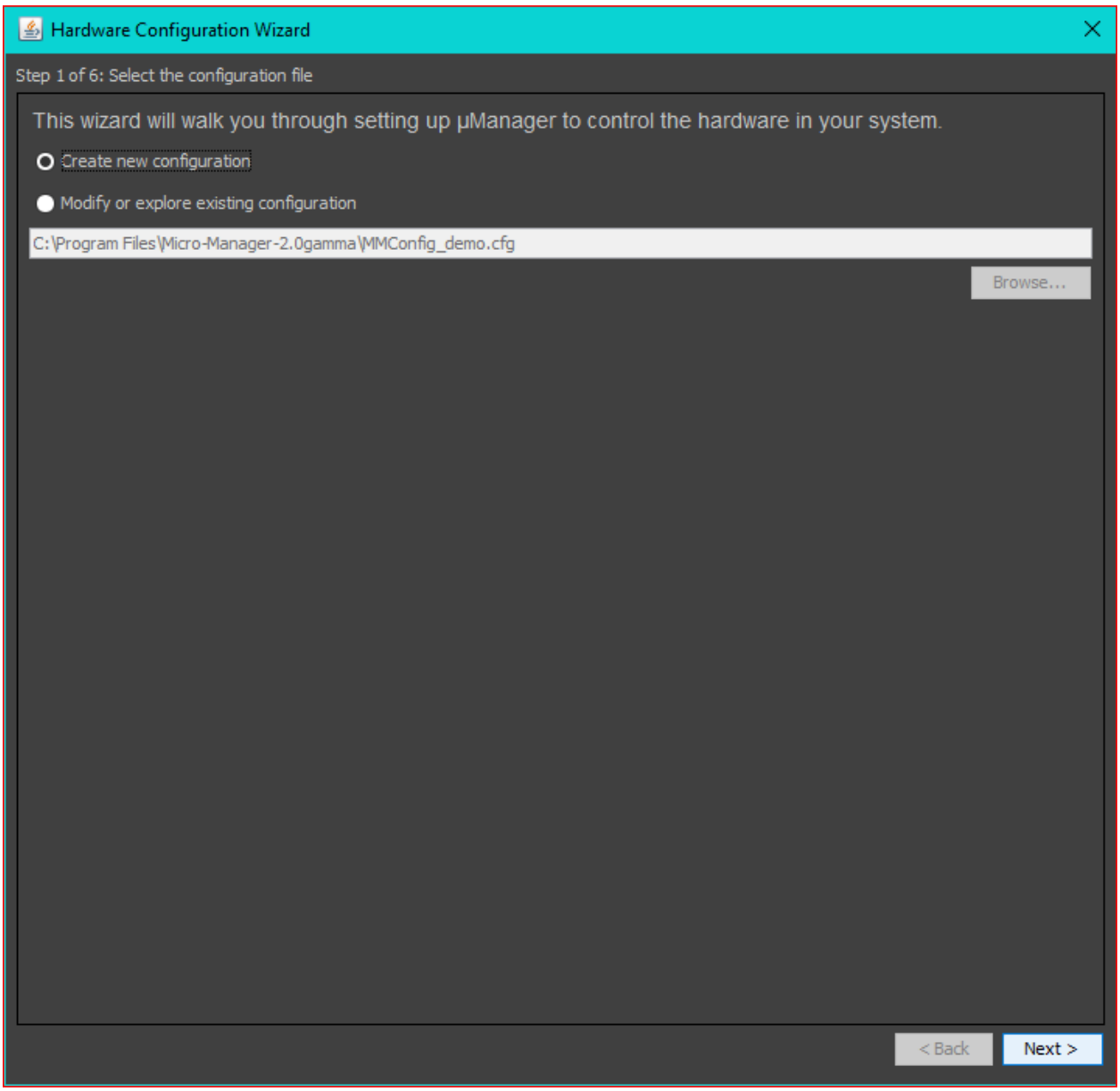
4.7.1.2.2 Настройка оборудования

- В главном окне программы выберите *Devices* → *Hardware configuration wizard*.

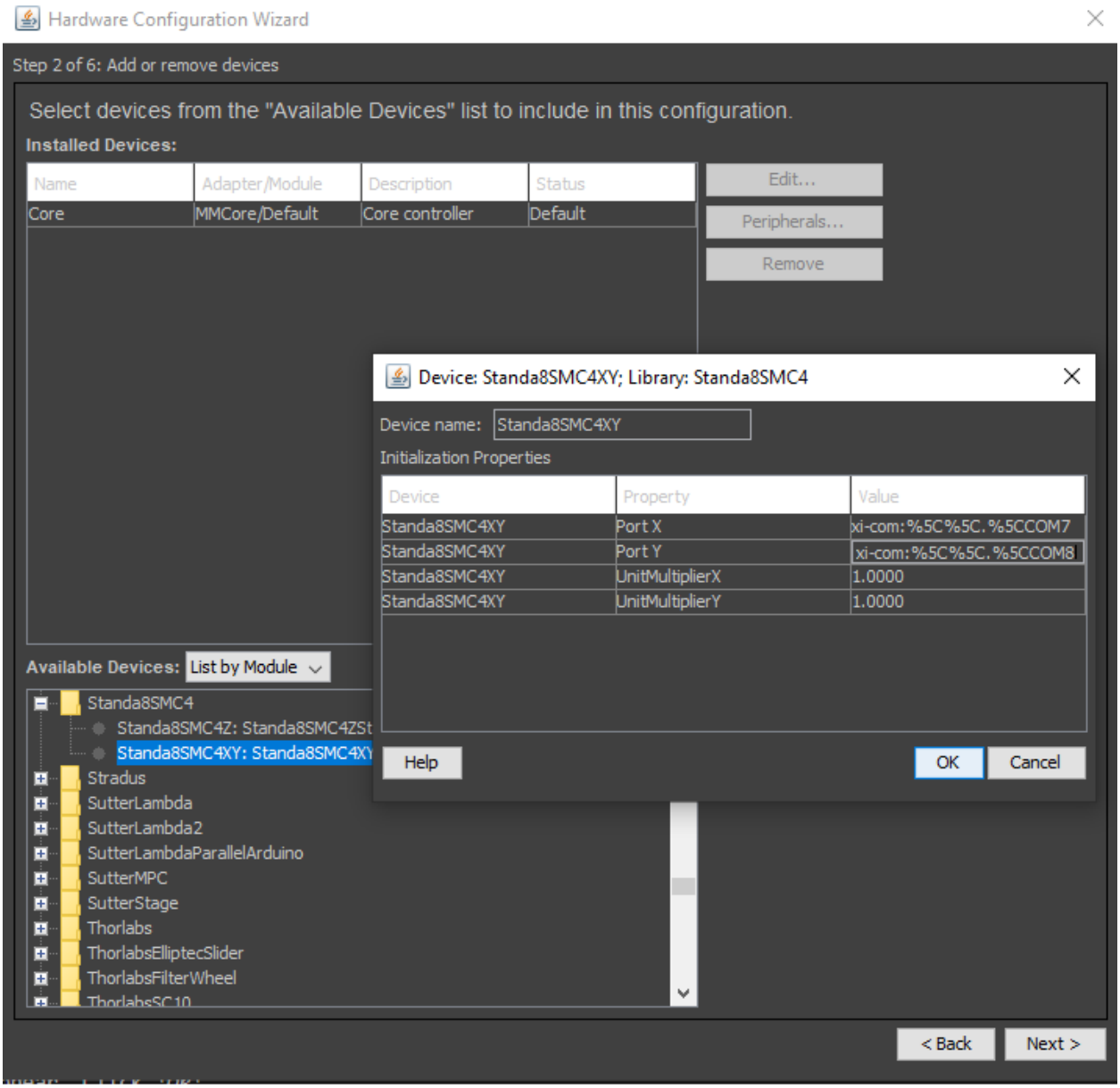


Процесс настройки состоит из 6 шагов:

1. Для начала, выберите «Create new configuration» и нажмите «Next».



2. Во-вторых, MicroManager попросит Вас выбрать оборудование для работы. Если вы хотите работать с *8SMC4/5-USB* контроллерами, выберите папку *Standa8SMC4* в нижнем окне с доступными устройствами. В ней содержится 2 драйвера: *Standa8SMC4Z* для работы с 1 осью и *Standa8SMCXY* для работы с 2 осями. При выборе одной из них появится следующее диалоговое окно.



3. В колонке *Value* необходимо вписать номер COM порта для 8SMC4-USB устройства в следующем формате:

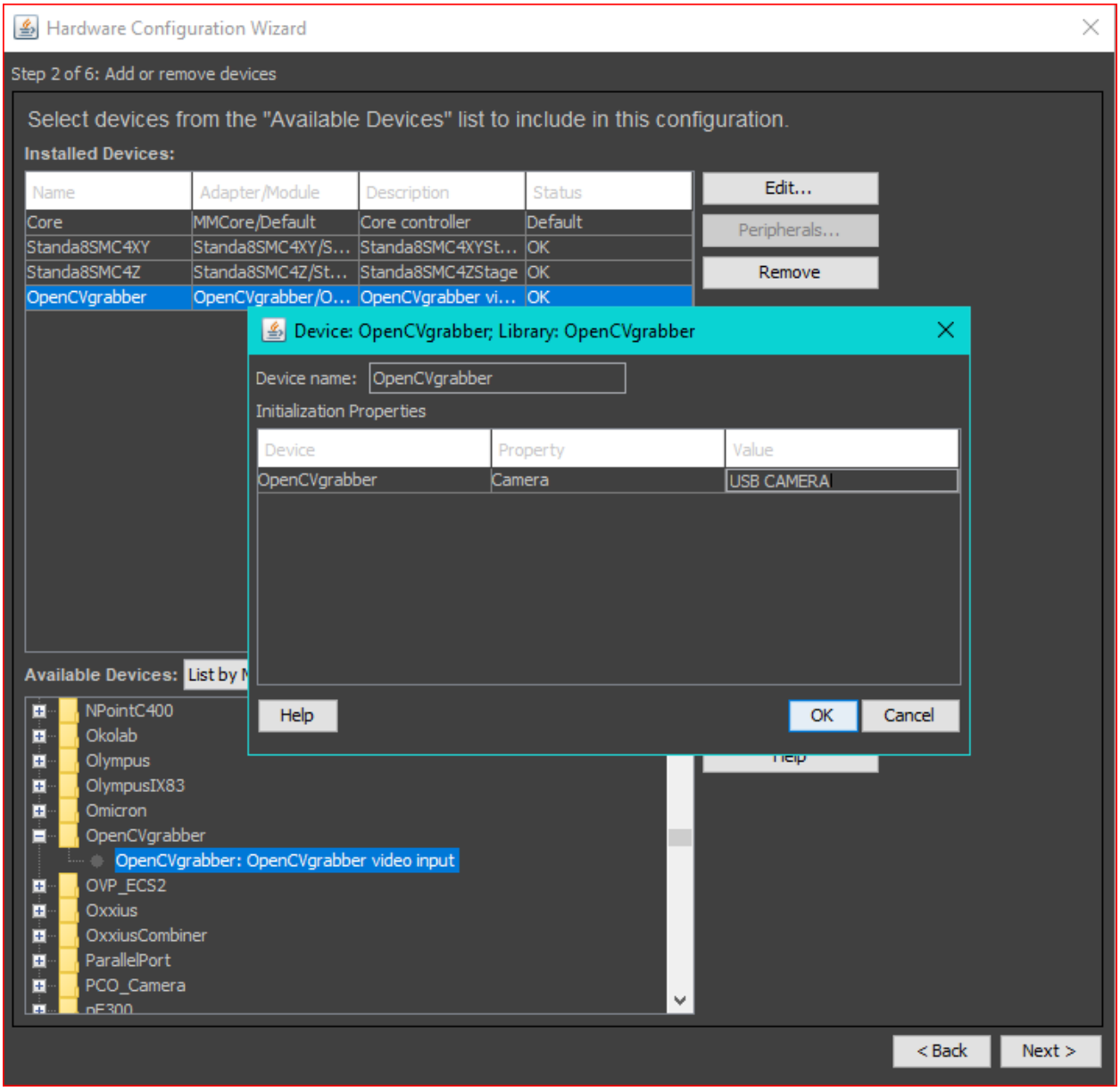
```
xi-com:%5C%5C.%5CCOMn
```

где *n* - номер соответствующего COM порта (смотрите скриншот выше). Найти информацию о созданном COM порте можно в XILab или *Device Manager* → *Ports*. Нажмите *Ok*. В полях *Unit Multiplier X/Y* оставьте значения по умолчанию. Это поле позволяет вам настроить скорость вашего позиционера.

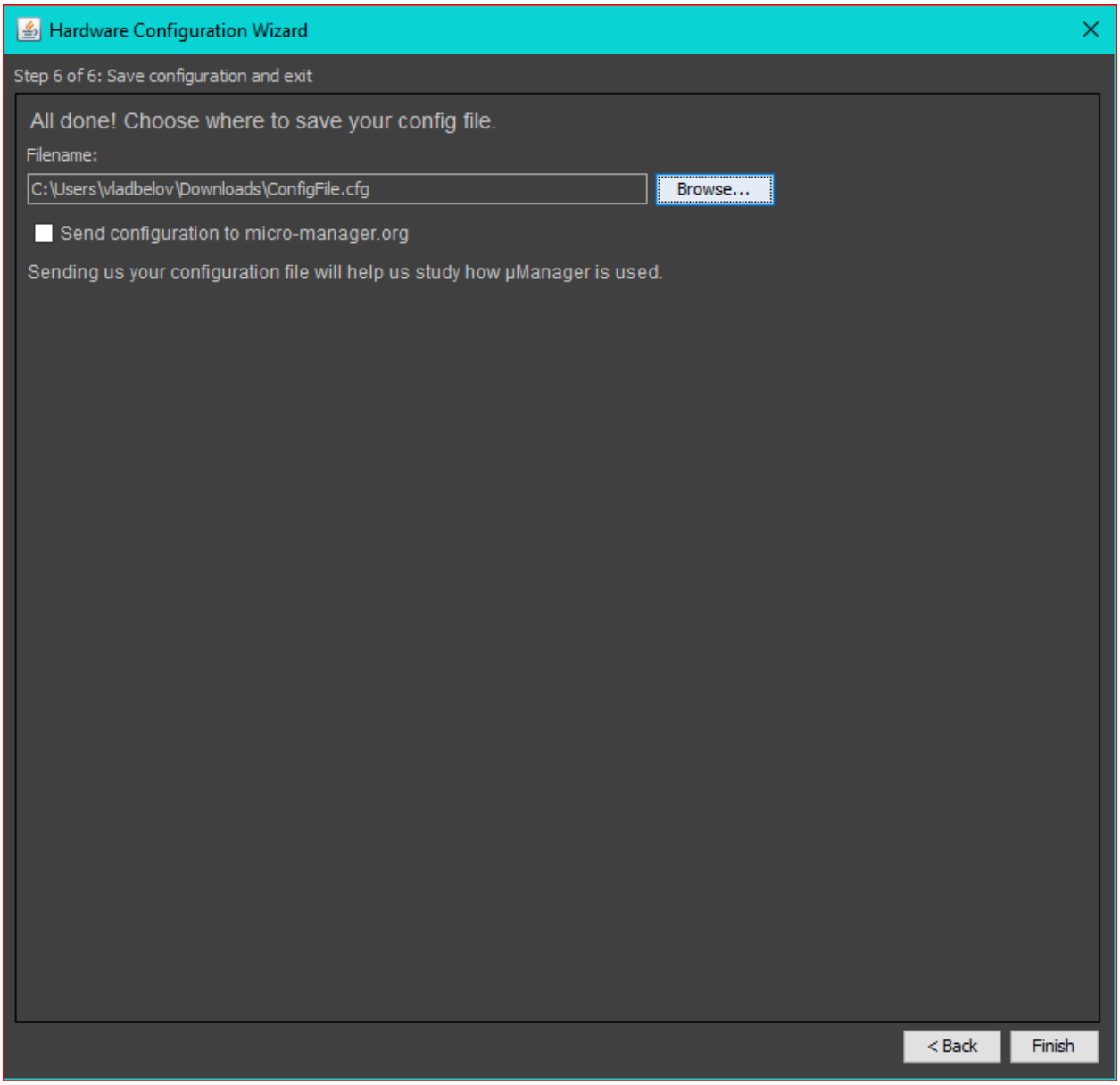
4. Для *Standa8SMC4Z* (ось Z) Вам также надо указать номер COM порта и значение Unit Multiplier. Ось Z обычно используется для фокусировки.

5. Таким же образом в список устройств добавляется камера. Для этого необходимо знать точное название драйвера для вашей камеры. Для работы с USB камерой используется драйвер

OpenCVGrabber. Выберите папку OpenCVGrabber среди Available Devices. Появится следующее окно. Нажмите *Ok*.



6. Далее нажмите *Next* в последующих шагах, выберите, где сохранить файл конфигурации. На последнем шаге нажмите *Finish*.



Конфигурация устройств для MicroManager завершена.

4.7.1.2.3 Работа с контроллерами

В предыдущих пунктах была проведена настройка MicroManager. Ниже приведены шаги для проверки корректной работы.

1. Во-первых, настройте размер пикселя в *Devices* → *Pixel Size Calibration...*. Вам необходимо указать значение размера пискеля и отметить параметры на которые MicroManager должен опираться при работе. Нажмите *New*, чтобы задать новый размер пикселя. В появившемся окне отметьте разрешение вашей камеры (при использовании драйвера *OpenCVGrabber* * - *это параметр *OpenCVGrabber* - Resolution) и укажите размер пикселя (в um). Для расчёта размера пикселя вам нужно знать размер изображения из микроскопа, его разрешение и смещение (исходя из шага винта) для вашей подвижки.

2. Во-вторых, выберите *Devices* -> *Stage Position List...*. Нажмите *Set Origin*, чтобы задать ноль

начала координат вашей системы. Затем в главном окне нажмите *Live*. Выберите *Scrolling Tool* (похоже на руку) в главном окне. Курсор должен быть похож на руку. Нажмите в открытое окно, в котором транслируется видео с камеры. Теперь вы можете контролировать перемещения подвижки с клавиатуры (стрелки «вверх», «вниз», «влево», «вправо» для работы с осями XY и клавиши U,J для работы с осью Z).

3. Также, вы можете перемещать изображение мышью. Нажмите *Tools* → *Mouse Moves Stage* (Use Hand Tool). Теперь нажмите на окно с видео и потяните в желаемом направлении. Подвижка начнёт двигаться в том же направлении. Двойной щелчок мыши по некоторой точке изображения позволит центрироваться на этой точке.

Примечание: Например, вы задаете размер пикселя равным 1um. И *UnitMultiplierX/Y* равно 1.0000. Из таблицы 8MTF однопиксельный сдвиг соответствует 12.5 мкм. Если мы установим *UnitMultiplierX/Y* на 12,5, то движение в одном пикселе будет соответствовать 1 мкм.

Примечание: Вы можете экспериментально уменьшить *UnitMultiplier*, если необходимо уменьшить скорость подвижки и увеличить его, если нужна большая скорость.

4.7.2 TANGO

Примечание: Адаптер **8SMC4-USB-Eth** был переименован в **8Eth1**[1] . Все доступные инструкции в нашем руководстве актуальны и применимы к обоим устройствам.

4.7.2.1 Обзор



Рис. 4.55: Наши контроллеры поддерживают TANGO!

TANGO - свободная объектно-ориентированная система, предназначенная для управления ускорителями, экспериментальными установками а также различным промышленным и научным оборудованием и программным обеспечением. Система TANGO активно разрабатывается сообществом энтузиастов и профессиональных инженеров.

TANGO основывается на концепции устройств. Устройства реализуются объектно-ориентированными и ориентированными подходами к архитектуре программного обеспечения. Модель устройства в TANGO реализует команды/методы, атрибуты/поля данных и свойства для настройки устройств. В TANGO все объекты управления - это устройства и аппаратный доступ, управляемые в процессе, называемом Device Server.

Сервер устройства реализует классы, обеспечивающие доступ к оборудованию. В процессе работы сервер устройства создает экземпляры устройств, отображающие логические сущности компонент оборудования. Клиент взаимодействует с устройствами, используя протоколы CORBA и zeromq.

[1] Артикул был изменен в 2020 г., ранее устройство имело артикул 8SMC4-USB-Eth.

Контроллер Standa 8SMC5-USB поддерживает работу с TANGO 8.1 при помощи специально разработанного сервера устройств. Предполагается два основных сценария:

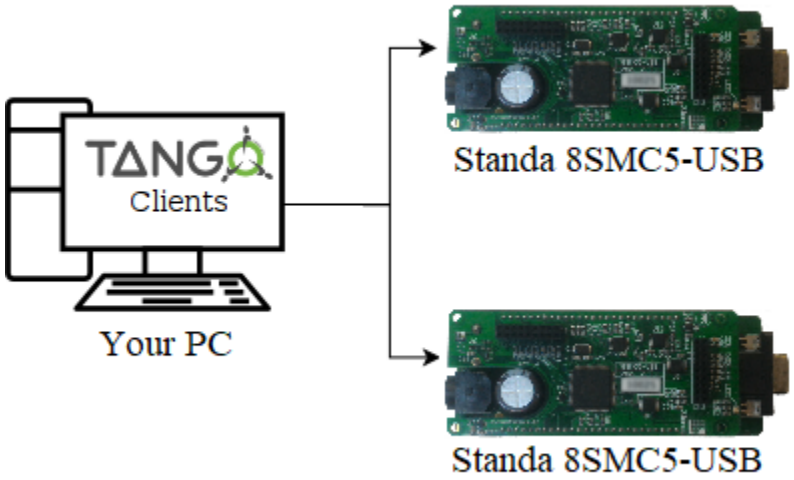


Рис. 4.56: Использование интерфейса TANGO через сервер устройств, запущенный напрямую на вашем ПК

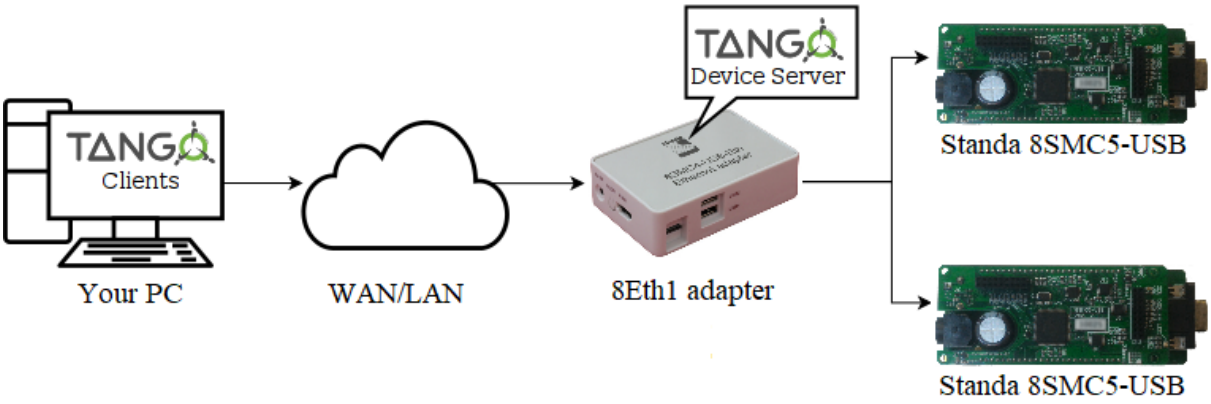


Рис. 4.57: Использование интерфейса TANGO через адаптер *Standa 8Eth1*

4.7.2.2 Конфигурация и запуск сервера устройства, объявление устройства

Предупреждение: Сервер устройств для **Standa 8SMC5-USB** разрабатывался на основе библиотек TANGO версии 8.1. Любая версия инфраструктуры TANGO старше 7 (включительно) также должна быть совместима с данной реализацией, однако тестирование не проводилось.

Для того, чтобы приступить к работе с новым TANGO-совместимым устройством, в первую очередь необходимо зарегистрировать его в БД-Сервере TANGO. Наиболее распространённый и простой способ - использовать для этого Jive:

Важно: В видео по настройке TANGO совместимого устройства **показаны те же действия, что в и в инструкции ниже**

---

https://youtu.be/o6sUHoe7Vjw

Видео-руководство по настройке TANGO

1. Подключите Jive к вашему локальному БД-Серверу TANGO (*Edit -> Change Tango Host*).
2. Запутсите помощник регистрации сервера устройств (*Tools -> Server Wizard*).



Рис. 4.58: Интерфейс помощник регистрации сервера устройства Jive

3. Введите «Standa8smc4usb» в поле *Server Name*. В поле *Instance name* необходимо ввести любое удобное для вас буквенно-цифровое имя, идентифицирующее конкретный экземпляр сервера устройств TANGO (например «TestDevice»).
4. Нажмите *next* - это приведёт к открытию окна *Start the server*. На этом шаге необходимо запустить сам сервера устройств (Jive при этом закрывать не нужно).
    - Если у вас есть *Standa 8Eth1*, и вы собираетесь использовать его встроенную поддержку TANGO, то для её активации и настройки вам необходимо использовать *интерфейс администрирования*.

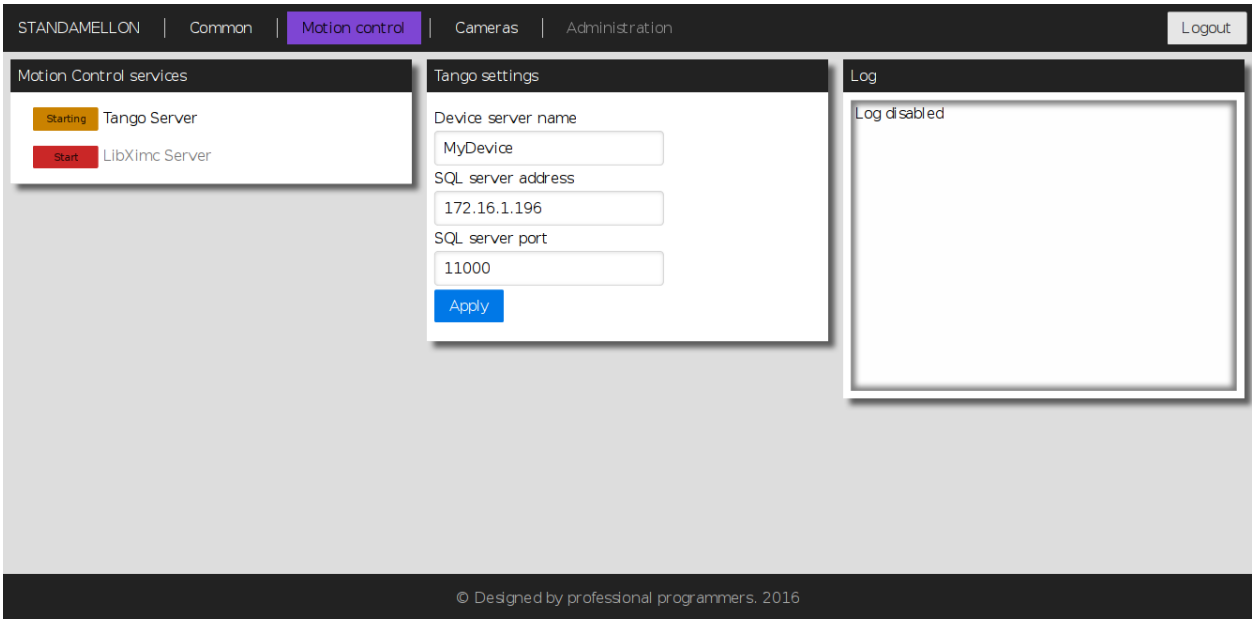


Рис. 4.59: Страница управления сервисом TANGO в *интерфейсе администрирования*

- Если же вы собираетесь использовать сервер устройств TANGO напрямую со своего компьютера, то вам необходимо самостоятельно скомпилировать его под вашу платформу из исходных кодов. В результате вы получите исполняемый файл *Standa8smc4usb*, которому при запуске необходимо передать в качестве аргумента ранее указанное в «Jive» имя экземпляра (например, «TestDevice»). Если сервер был успешно запущен, то в стандартном потоке вывода вы увидете надпись *Ready to accept requests*.

Примечание: Имя экземпляра сервера устройств, которое вы указали в Jive, должно совпадать с тем, что вы указываете при запуске самого экземпляра сервера устройств (как в случае использования исполняемого файла, так и при запуске через *интерфейс администрирования*).

Примечание: Перед запуском сервера устройств «Standa8smc4usb» убедитесь в том, что переменная среды «TANGO_HOST» содержит в себе адрес и порт вашего локального БД-Сервера Tango (например, «192.168.0.172:11000»).

---

5. Перключитесь обратно на Jive, в котором по-прежнему должно быть открыто окно *Start the server*. Нажмите кнопку *next*.
6. В открывшемся окне *Class Selection* выберите класс *Standa8scm4usb* и нажмите кнопку *Declare device*.
7. В следующем окне в поле *Device name* введите соответсвующее спецификации TANGO имя конкретного устройства (например «a/b/c»), после чего нажмите *next*.
8. Следующий шаг требует ввода серийного номера экспортируемого в TANGO контроллера в поле *SerialNumber*. Узнать серийный номер контроллера можно либо в разделе *Common в интерфейсе администрирования*, если вы используете Ethernet-адаптер *Standa 8Eth1*, или через программу *XiLab* (*Settings -> About Device -> Serial number*).

Примечание: XiLab также может быть использован для определения серийных номеров контроллеров, подключенных к Ethernet-адаптеру *Standa 8Eth1*, но для этого на адаптере при помощи интерфейса администрирования должен быть включен сервис «LibXimc server». Более детально об этих функциях XiLab можно узнать из *документации*.

9. Далее потребуется ввод двух параметров: *CalibrationRatio* и *CalibrationUnits*. Они служат для отображения шагов в пользовательские единицы и должны трактоваться как «1 шаг = *CalibrationRatio CalibrationUnits*». Если вам не требуется пересчёт в какие-либо единицы, то задайте «1.0» и пустую строку соответственно.
10. Непосредственно процедура регистрации устройства (т.е. предыдущие 3 шага) должна быть повторена для каждого из подключенных контроллеров, которые требуется использовать через интерфейс TANGO.

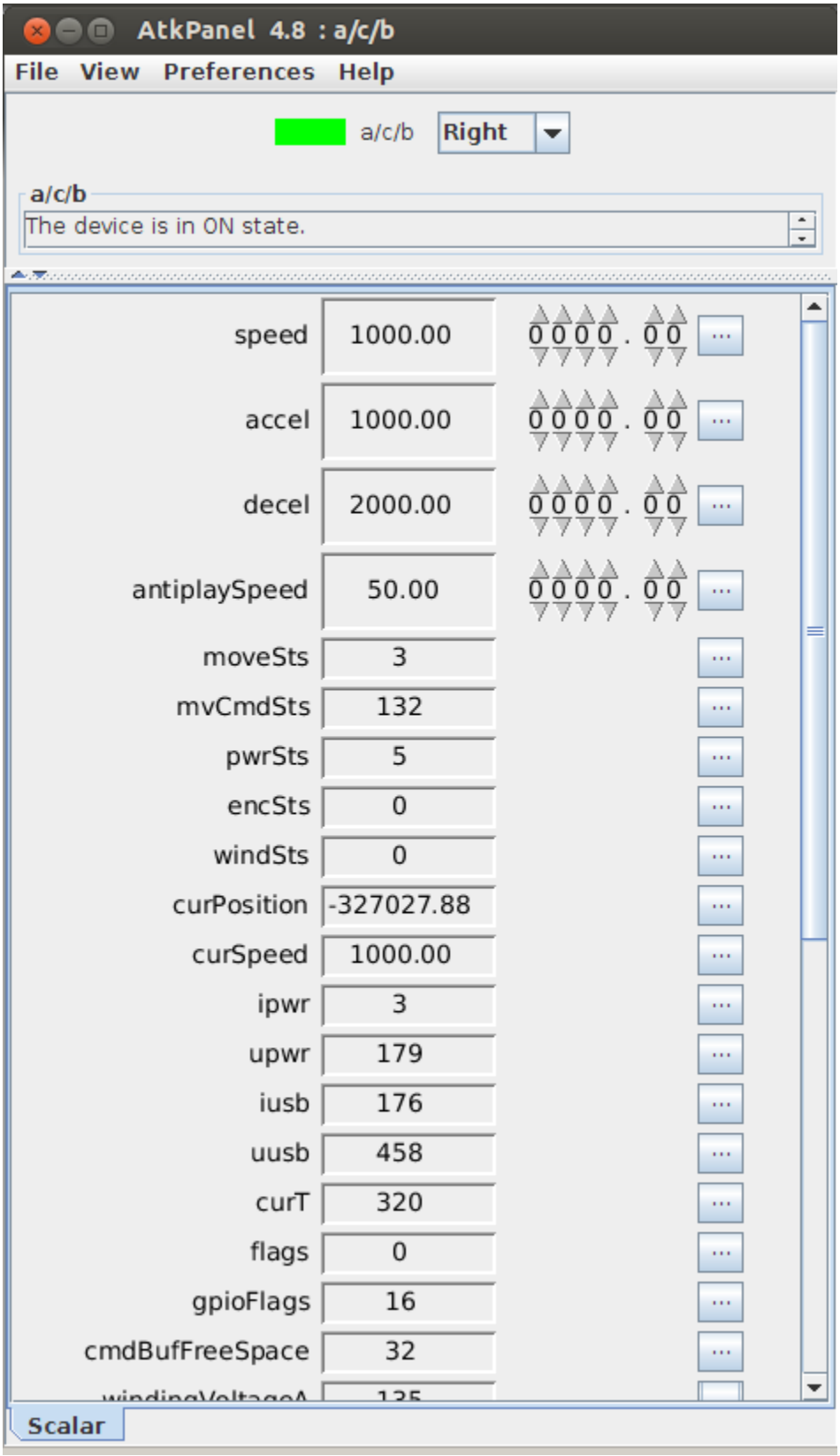


Рис. 4.60: Пример содержимого стандартной ATK-панели для экспортированного в TANGO контроллера

После вышеописанных шагов любой из экспортированных контроллеров становится доступным для непосредственного управления и настройки при помощи TANGO, а значит готов к немедленной интеграции с вашей существующей инфраструктурой без каких-либо дополнительных трудозатрат! Для получения более подробной информации обратитесь к официальной документации TANGO.

Глава 5

Руководство по программе XILab

5.1 О программе XILab

XILab представляет собой удобный графический интерфейс пользователя для управления позиционерами, диагностики моторов и настройки двигателей, управляемых данными контроллерами. XILab позволяет быстро настраиваться на подключенный позиционер с помощью загрузки подготовленных заранее конфигурационных файлов. Управление можно автоматизировать с помощью скриптового языка, что может использоваться непосредственно или ускорит процесс разработки собственной программы управления. XiLab поддерживает многоосевой режим и многомерные скрипты управления. Предусмотрена возможность выводить данные о состоянии контроллера и двигателя на графики и сохранять их в файл, экспортировать данные в табличном виде для обработки внешними программами. Программное обеспечение совместимо с операционными системами Windows (XP SP3, Vista, 7, 8, 10, 11), Linux, MacOS Sierra и новее для intel и Apple Silicon (с использованием Rosetta 2). В зависимости от операционной системы вашего компьютера, вид некоторых окон может отличаться.

Краткое руководство по установке программы приведено *здесь*. В данной главе приведено подробное руководство по работе в ПО XILab.

5.2 Основные окна программы XILab

5.2.1 Стартовое окно программы XILab

При запуске XILab открывается окно поиска контроллеров. XILab с помощью библиотеки libximc опрашивает контроллеры, подключенные к системе, и выводит список найденных и успешно опознанных контроллеров на экран.

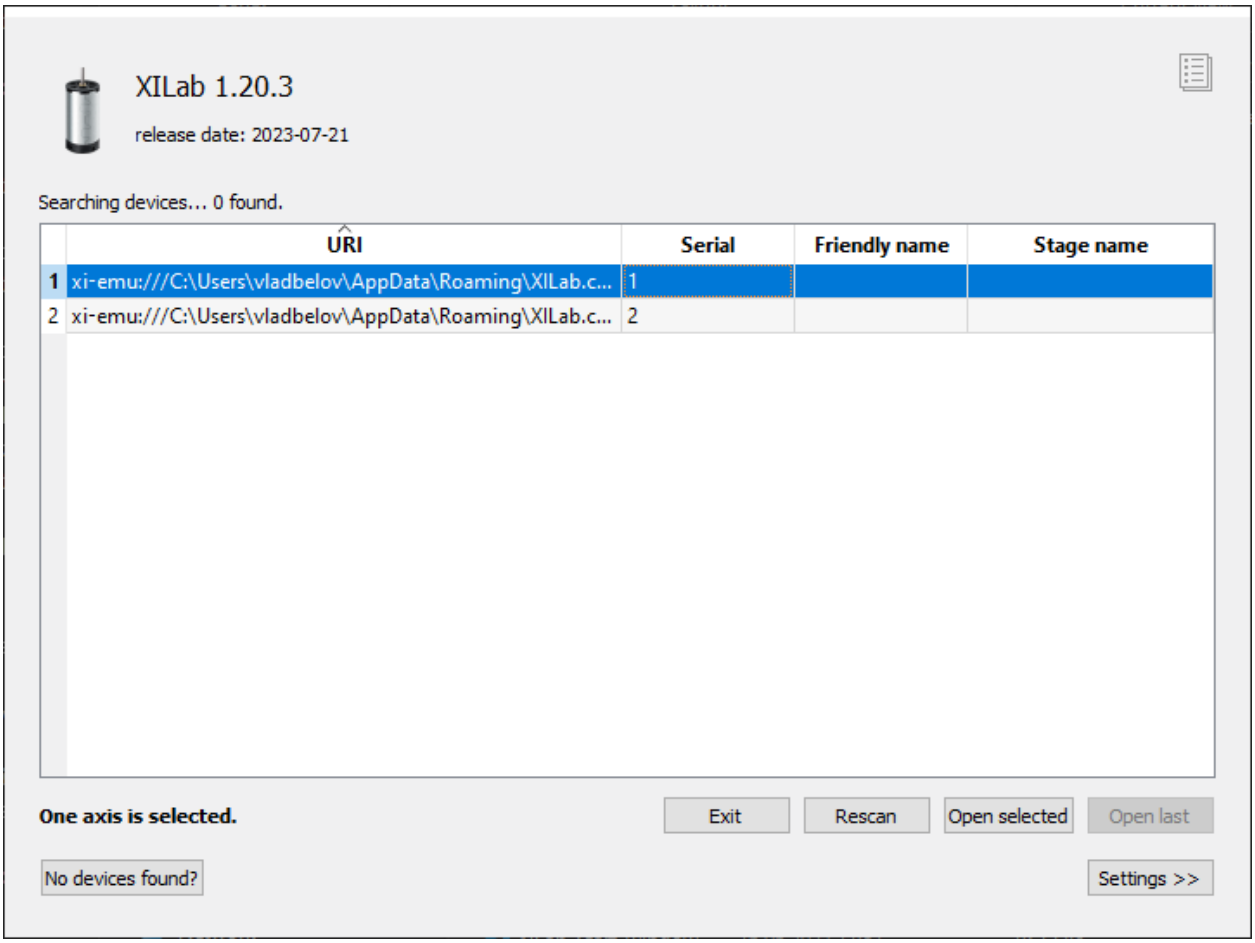


Рис. 5.1: Стартовое окно XILab, найдено 2 виртуальных контроллеров

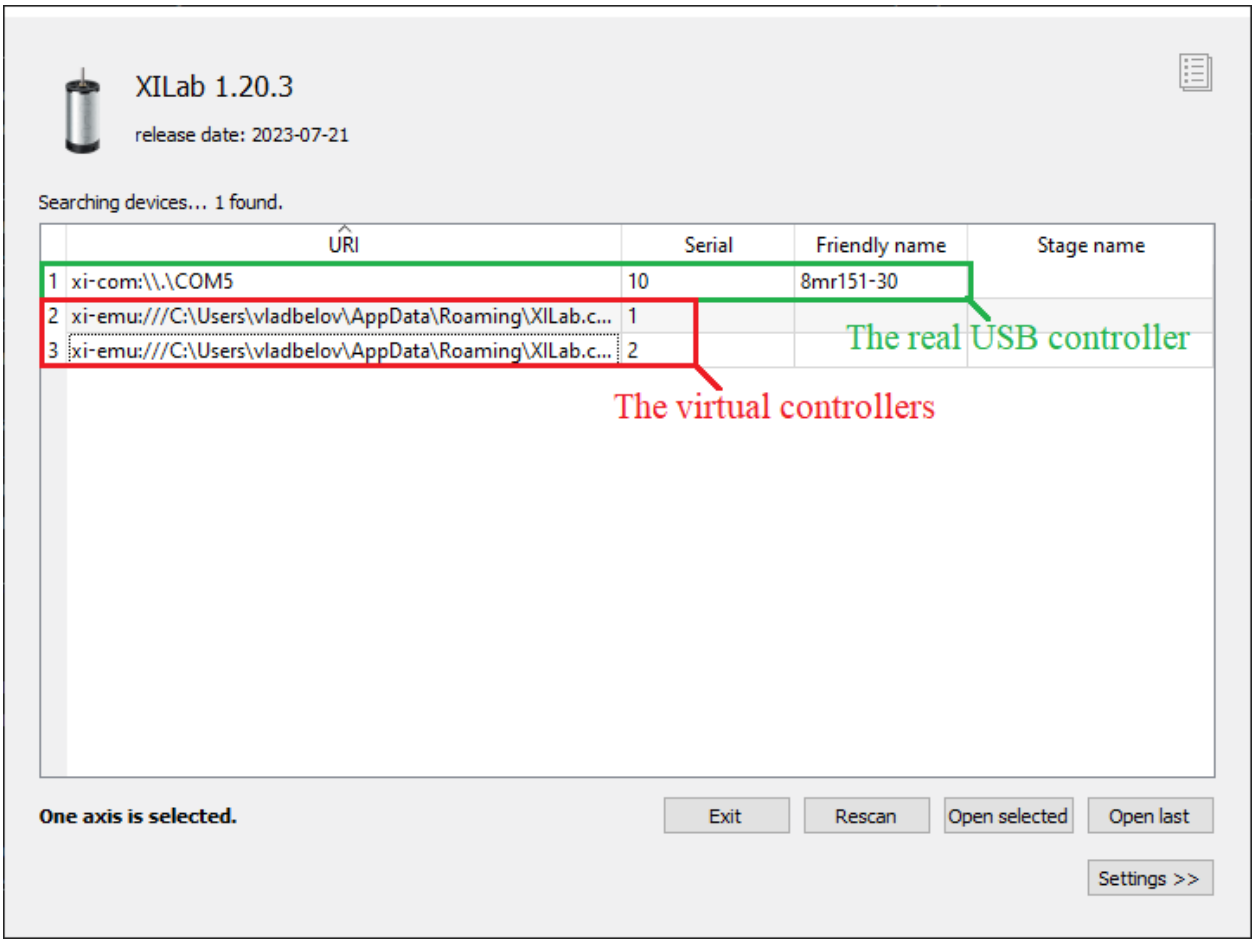


Рис. 5.2: Стартовое окно XILab, показаны 1 USB контроллер и 2 виртуальных контроллера

Список найденных контроллеров выводится в стартовое окно. Контроллеры могут быть подключены как по USB так и по Ethernet (поддерживаются tcp, udp и xinet). Можно выбрать один или несколько контроллеров и открыть их с помощью кнопки *Open selected*. Если выбран один контроллер, то будет открыто *главное окно программы XILab в режиме управления одной осью*, если выбрано более одного, то будет открыто *главное окно программы XILab в режиме управления несколькими осями*. Повторный поиск осуществляется нажатием *Rescan*, а выход - нажатием *Exit*. Если активна кнопка *Open last*, то это означает, что найдены все контроллеры, которые были открыты при предыдущем запуске XILab, и нажатие *Open last* откроет эту сохраненную конфигурацию.

XILab может работать с виртуальными контроллерами XIMC, которые поддерживают протокол ответов реального контроллера. Виртуальный контроллер может быть полезен для ознакомления с интерфейсом XILab, в случае если к системе не подключены реальные контроллеры.

По кнопке *Settings* открывается вкладка с настройками обнаружения устройств XIMC.

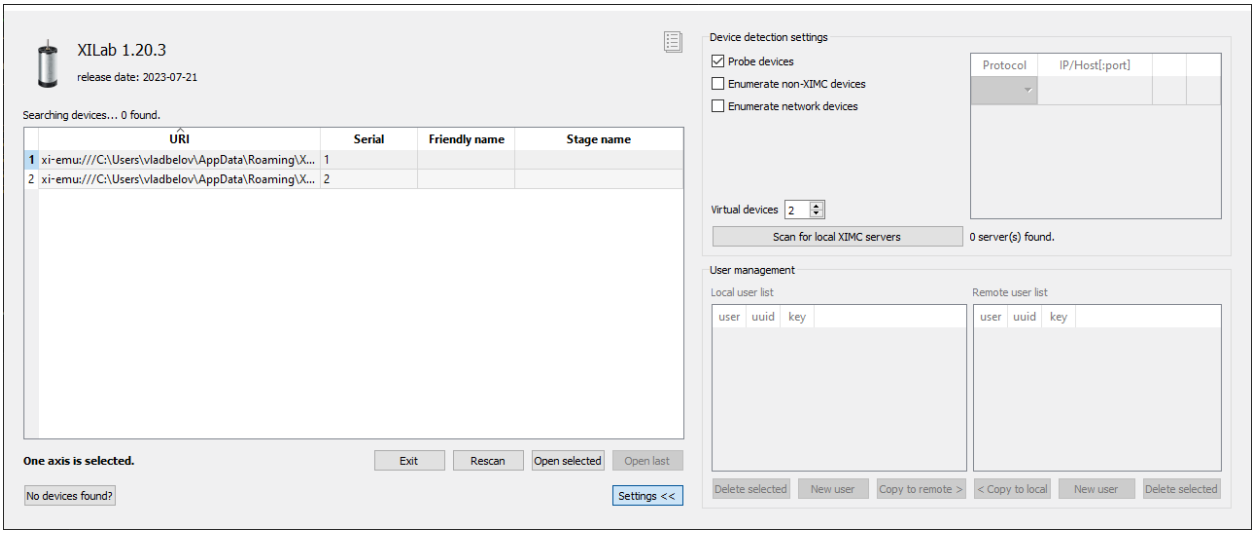


Рис. 5.3: Стартовое окно XILab, вкладка настроек

Блок **Device detection settings** содержит настройки обнаружения устройств XIMC.

Probe devices - при включенной опции XiLab пытается идентифицировать контроллеры, посылая в них при открытии команды GETI и GSER.

Enumerate non-XIMC devices - при включенной опции опрашивает все устройства типа COM-порт в системе. При отключенной опции опрашивает только устройства, имена которых соответствуют маске устройств XIMC («XIMC Motor Controller» в Windows, /dev/ximc/* и /dev/ttyACM* на Linux/Mac).

Enumerate network devices - при включенной опции опрашивает сетевые устройства. Список адресов доменных имен и/или IP-адресов, на которых производится поиск устройств, находится ниже. Записи в списке можно добавлять как вручную, так и автоматически, нажав на кнопку *Scan for local XIMC servers*. Обратите внимание, что в случае наличия нескольких XIMC-серверов с устройствами в локальной сети будет найден случайный из них и для нахождения всех серверов потребуется несколько попыток автоматического поиска.

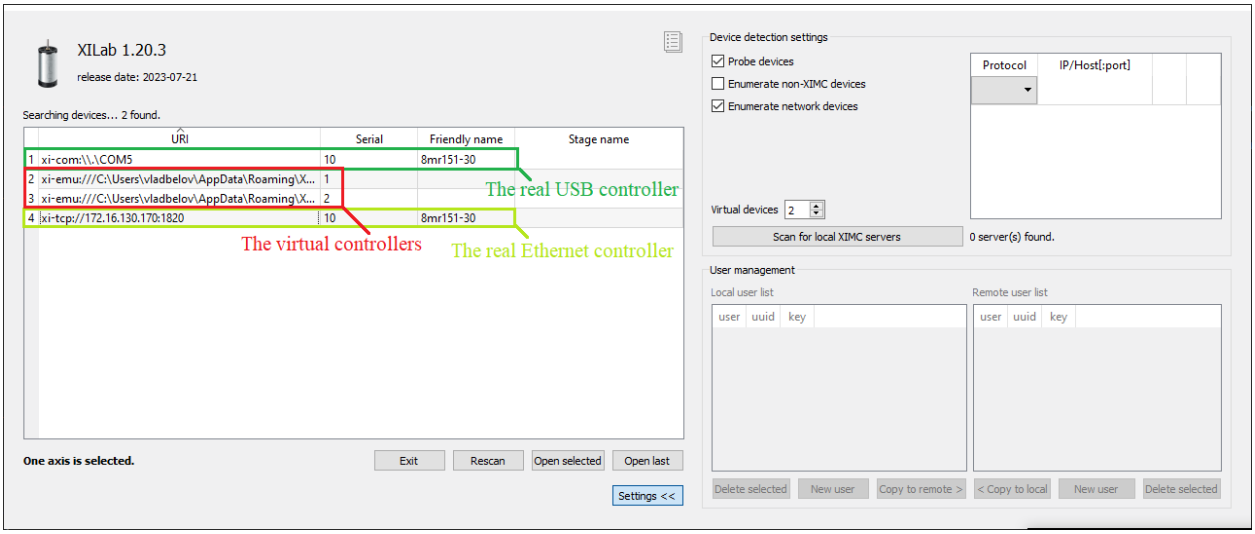


Рис. 5.4: Стартовое окно XILab, показаны 1 USB контроллер, 1 Ethernet контроллер и 2 виртуальных контроллера

Предупреждение: При одновременно включенных опциях *Probe devices* и *Enumerate non-XIMC devices* XILab при старте посылает данные во все COM-порты. При наличии в системе множества Bluetooth COM-портов из-за особенностей работы Bluetooth опрос будет происходить последовательно с затратами от единиц до десятков секунд на одну попытку соединения.

В поле *Virtual devices* указано количество виртуальных контроллеров, которые будут выведены в список доступных для открытия при следующем нажатии *Rescan* или следующем старте XILab.

Примечание: Так как библиотека libximc открывает устройства XIMC в режиме эксклюзивного доступа, при запуске последующих копий программы XILab будут найдены и доступны для выбора только свободные контроллеры. **Не относится к контроллерам, открытым по Ethernet (протоколы tcp/udp)**

Панель **User management** обеспечивает возможность редактирования списка управления доступом для локальных и удаленных серверов. Эта функция позволяет конечному пользователю выборочно предоставлять разрешения для подключения и управления удаленными устройствами 8SMC4/5-USB. Чтобы предоставить разрешение, необходимо создать одного и того же пользователя с одним и тем же паролем локально и удаленно. Удаление пользователя (локально или удаленно) отменяет разрешение. По умолчанию во всех 8Eth1 адаптерах, библиотеках SDK и XILab предварительно установлен пользователь root.

Примечание: Для использования этой функции необходим Standa 8Eth1 адаптер с поддержкой ACL. За дополнительной информацией обращайтесь в нашу службу технической поддержки.

5.2.2 Главное окно программы XILab в режиме управления одной осью

- *Блок управления движением двигателя*
  - *Движение без точного задания конечного положения*
  - *Движение в заданную точку*
  - *Текущая позиция для команд движения*
- *Блок управления аттенюатором*
- *Состояние контроллера и мотора*
  - *Электропитание контроллера*
  - *Состояние мотора*
  - *Состояние программы*
- *Группа кнопок для управления программой*
- *Статусная строка*

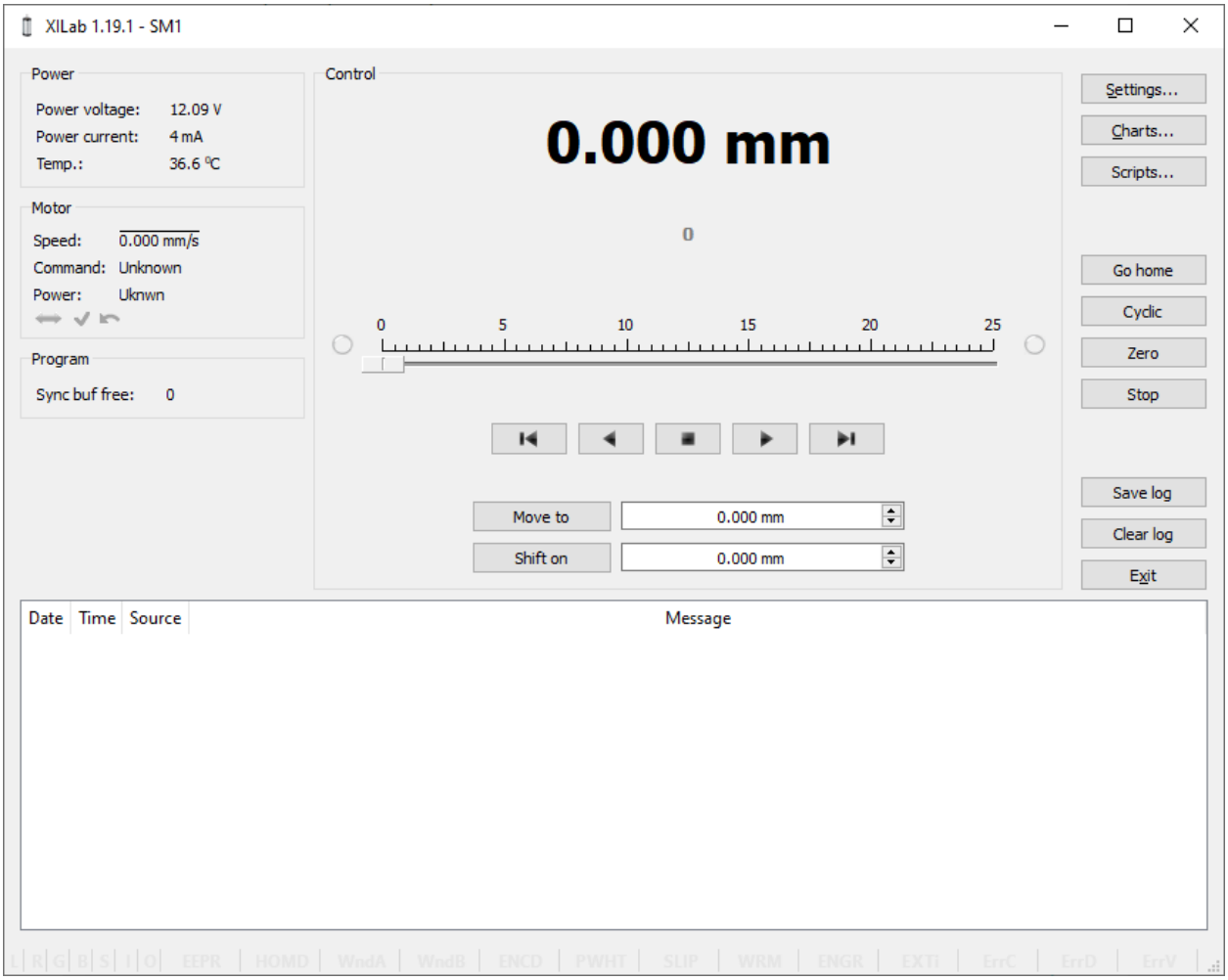


Рис. 5.5: Главное окно программы XILab в режиме двигателя

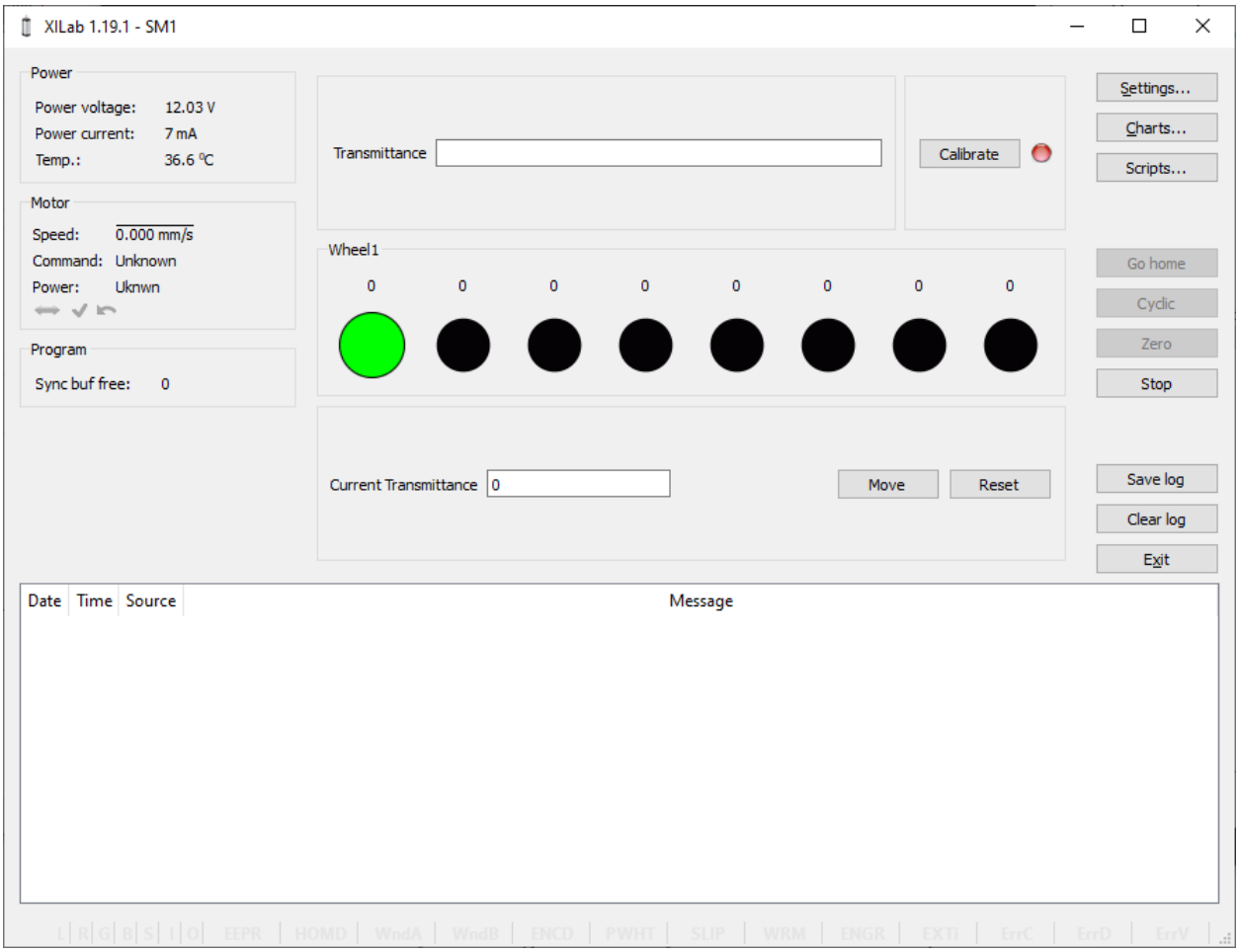


Рис. 5.6: Главное окно программы XILab в режиме аттенюатора

В левой части окна в группах параметров **Power** и **Motor** находятся данные о состоянии контроллера и мотора в настоящий момент. В центральной части окна расположен блок **Control**, содержащий индикаторы текущей позиции и элементы управления движением мотора. Блок **Control** в зависимости от настроек может принимать вид блока управления движением произвольного двигателя или блока управления аттенюатором. Справа расположена группа кнопок для управления программой в целом. Внизу расположен *лог*, при минимальном размере окна он скрыт. Под логом находится статусная строка. Рассмотрим эти группы более подробно.

5.2.2.1 Блок управления движением двигателя

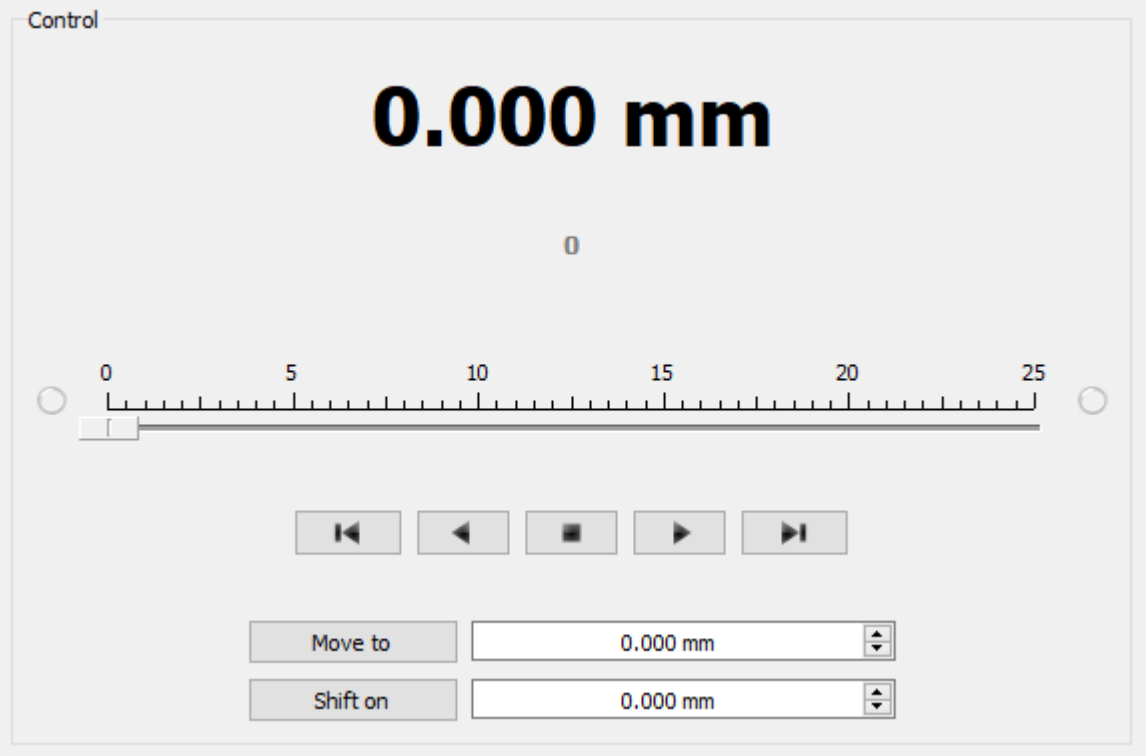


Рис. 5.7: Блок Control

В центральной части блока расположен индикатор текущей позиции, под ним, если энкодер включен, располагается индикатор позиции по энкодеру. В режиме ведущего энкодера, см. раздел *Работа с энкодерами* главный и второстепенный индикаторы меняются местами.

Ниже расположен блок **Control**, содержащий элементы управления движением мотора. Рассмотрим их более подробно:

5.2.2.1.1 Движение без точного задания конечного положения



Рис. 5.8: Кнопки управления движением

- Кнопки *Влево*, *Стоп* и *Вправо* запускают движение влево без указания конечной позиции, *останавливают с замедлением* начатое движение и запускают движение вправо без указания конечной позиции, соответственно.
- Кнопка *Влево до границы* заставит мотор вращаться до левой границы слайдера. *Вправо до границы*, соответственно, до правой границы слайдера.
- Нажатие и удержание кнопок клавиатуры *Вправо*, *Влево* при нахождении фокуса ввода в блоке слайдера начинает движение в направлении увеличения или уменьшения координаты. При отпускании кнопки движение прекращается, как будто была нажата кнопка *Стоп* на главном окне.

5.2.2.1.2 Движение в заданную точку

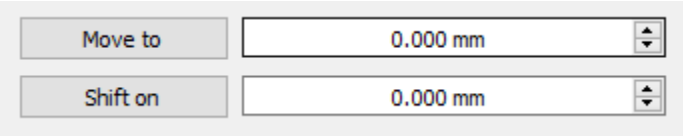


Рис. 5.9: Управление движением в заданную точку

- Кнопка *Move to* запускает процесс перемещения в заданную позицию.
- Кнопка *Shift on* запускает процесс смещения на заданное расстояние от текущей позиции.

5.2.2.1.3 Текущая позиция для команд движения

Команды *Move to* и *Shift on* использую текущую позицию для расчета движения. Текущая позиция изменяется следующими командами:

Move to <величина>
Текущая позиция = <величина>

Shift on <смещение>
Текущая позиция = текущая позиция + <смещение>

Zero (при условии отсутствия движения в момент посылки команды)
Текущая позиция = 0

Команды *Stop*, *Влево*, *Вправо*, *Влево до границы* и *Вправо до границы* не изменяют текущую позицию.

5.2.2.2 Блок управления аттенюатором

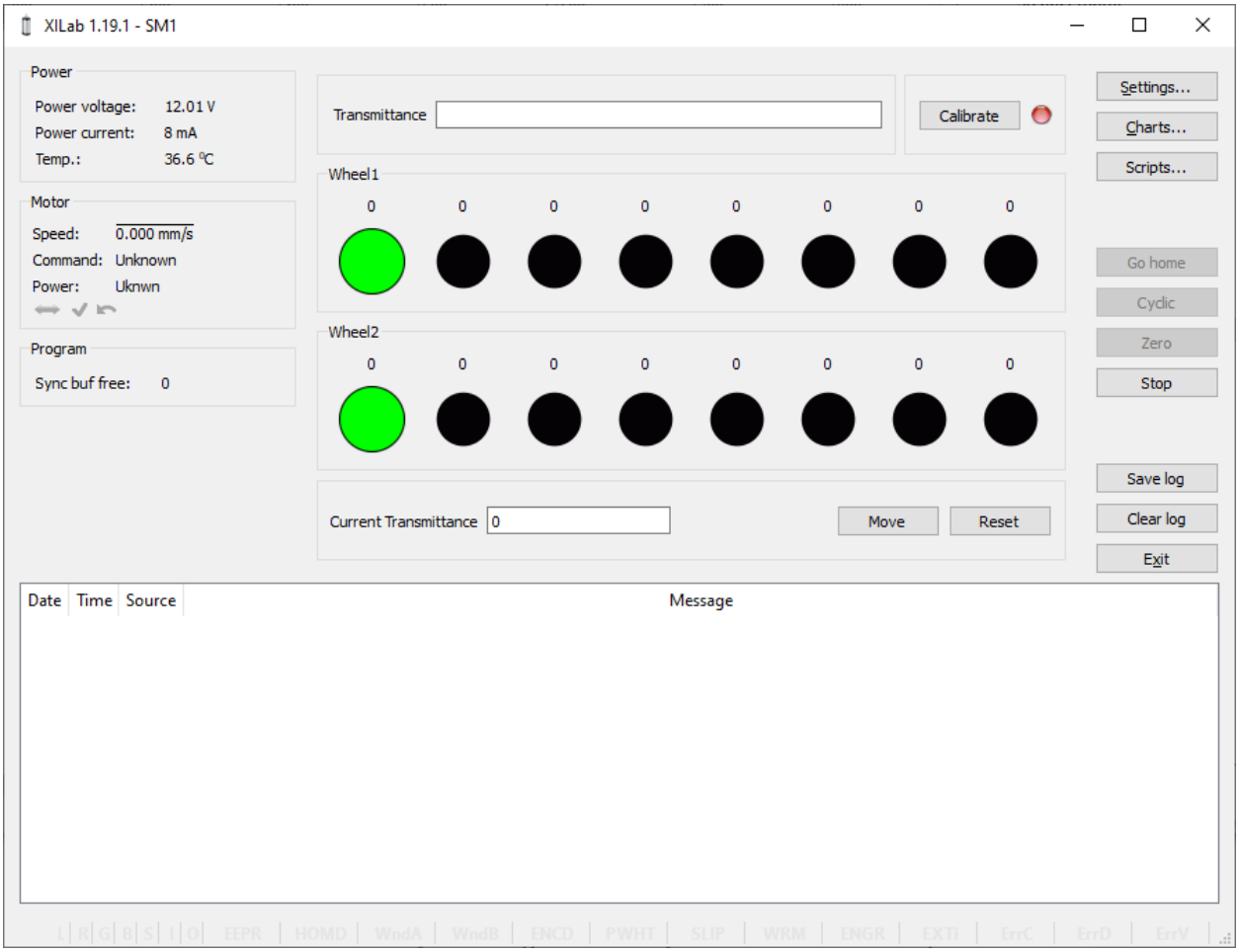


Рис. 5.10: Блок управления аттенюатором

В верхней части блока расположено окно *Transmittance* и кнопка *Calibrate*. Окно *Transmittance* предназначено для выбора желаемого коэффициента пропускания. Кнопка *Calibrate* предназначена для ручного поиска начальной позиции аттенюатора и при нажатии запускает сначала движение на один оборот аттенюатора с текущими настройками для однозначного определения относительного положения дисков аттенюатора, а затем функцию *Автокалибровка домашней позиции*. Нажатие *Calibrate* не является необходимым для движения - в случае если аттенюатор не производил калибровку или калибровка была сброшена, к примеру нажатием Cancel в процессе движения, при следующем движении калибровка будет сделана автоматически.

Аттенюатор может работать с одним или двумя дисками (у каждого диска имеется 8 фильтров), поэтому ниже будет находиться одно или два поля, соответствующие одному или двум дискам. Далее располагается окно *Current Transmittance*, в котором показывается коэффициент пропускания (который группируется из коэффициентов пропускания имеющихся фильтров), наиболее близкий к желаемому.

При нажатии на кнопку *Move* происходит движение к тем фильтрам, которые соответствуют *Current Transmittance*, причем данные фильтры подсвечиваются зеленым цветом, т.е делаются активными.

Нажатие на кнопку *Reset* делает все фильтры неактивными (становятся серого цвета).

5.2.2.3 Состояние контроллера и мотора



5.2.2.3.1 Электропитание контроллера

Группа параметров **Power** содержит индикаторы:

- *Power voltage* - напряжение на силовой части.
- *Power current* - ток потребления силовой части.
- *Temp.* - температура процессора контроллера.

Изменение цвета индикатора *Power voltage* на красный показывает выход за рамки диапазона допустимых значений напряжения источника питания относительно разрешенного. В этом случае контроллер переходит в состояние *Alarm*. Вы можете изменить этот параметр в разделе *Настройка предельных параметров контроллера*.

Появление горизонтальной черты над индикатором *Power voltage* означает, что напряжение питания контроллера превышает максимальное напряжение двигателя, Вы можете изменить этот параметр в разделе *Настройка кинематики движения (DC мотор)*.

Изменение цвета индикатора *Power current* на красный показывает превышение тока, потребляемого контроллером от источника питания, относительно разрешенного. В этом случае контроллер переходит в состояние *Alarm*. Вы можете изменить этот параметр в разделе *Настройка предельных параметров контроллера*.

Появление горизонтальной черты над индикатором *Power current* означает, что ток, потребляемый контроллером, превышает максимальный ток двигателя, этот параметр можно изменить в разделе *Настройка кинематики движения (DC мотор)*.

Изменение цвета индикатора *Temp* на красный показывает превышение температуры на плате контроллера относительно разрешенной. В этом случае контроллер переходит в состояние *Alarm*. Параметр можно изменить в разделе *Настройка предельных параметров контроллера*.

Важно: Выход из состояния *Alarm* возможен после прекращения событий, вызвавших *Alarm*, при условии, что флаг *Sticky Alarm* не установлен. Если флаг *Sticky Alarm* установлен , используйте кнопку «STOP» , чтобы выйти из состояния тревоги.

5.2.2.3.2 Состояние мотора

Группа параметров **Motor** содержит индикаторы:

- *Speed* - скорость вращения мотора.
- *Command* - последняя выполняемая (жирный шрифт) или выполненная (обычный шрифт) *команда контроллера*. *Команда контроллера* отображается черным цветом, если флаг ошибки движения MVCMD_ERROR не установлен, в противном случае красным. Может быть одним из следующих вариантов:
  - *Move to position* - перемещение в заданную позицию
  - *Shift on offset* - смещение на заданное расстояние
  - *Move left* - движение влево
  - *Move right* - движение вправо
  - *Stop* - остановка
  - *Homing* - нахождение начальной позиции
  - *Loft* - компенсация люфта
  - *Soft stop* - плавная остановка
  - *Unknown* - неизвестная команда (возможно сразу после включения контроллера)
- *Power* - состояние питания шагового двигателя. Может быть одним из следующих вариантов:
  - *Off* - обмотки мотора разомкнуты и не управляются драйвером„
  - *Short* - обмотки замкнуты накоротко через драйвер,
  - *Norm* - обмотки запитаны номинальным током,
  - *Reduc* - обмотки намеренно запитаны уменьшенным током от рабочего для снижения потребляемой мощности,
  - *Max* - обмотки запитаны максимально доступным током, который может выдать схема при данном напряжении питания.

Примечание: GPIO флаг можно использовать для обнаружения подключенного двигателя

Появление горизонтальной черты над индикатором Speed означает, что достигнута максимальная скорость движения, установленная в поле Max nominal speed настроек мотора *Настройка кинематики движения (DC мотор)*.

5.2.2.3.3 Состояние программы

Группа параметров **Program** содержит индикаторы:

- *Sync buf free* - количество свободных ячеек в буфере команд контроллера (на данный момент эта функция отключена).

5.2.2.4 Группа кнопок для управления программой

- Кнопка *Settings...* открывает настройки контроллера, см. раздел *Настройки программы*.
- Кнопка *Chart...* открывает окно с графиками, см. раздел *Графики*.
- Кнопка *Scripting...* открывает окно работы со скриптами, см. раздел *Скрипты*.

- Кнопка *Go home* осуществляет поиск начальной позиции, см. раздел *Настройка исходного положения*.
- Кнопка *Cyclic* включает режим циклического движения, см. раздел *Настройка режима циклического движения*.

Примечание: *Команда Cyclic* является составной командой: при вызове *Cyclic* в XiLab на уровне контроллера производится выполнение последовательности из команд *Move to*.

- Кнопка *Zero* обнуляет текущую позицию мотора и значение энкодера.
- Кнопка *Stop* посылает команду *немедленной остановки*, сбрасывает состояние Alarm, очищает буфер команд для синхронного движения и останавливает выполнение скрипта, если он запущен.
- Кнопка *Save log* сохраняет содержимое лога в файл в формате CSV (открывается диалог выбора файла для записи).
- Кнопка *Clear log* очищает содержимое лога.
- Кнопка *Exit* осуществляет корректное завершение работы, см. раздел *Корректное завершение работы*.

5.2.2.5 Статусная строка

В статусной строке находятся индикаторы текущего состояния контроллера. Слева направо это блок 7 флагов,

- **L** - Левая кнопка нажата
- **R** - Правая кнопка нажата
- **G** - Вход/выход GPIO активен
- **B** - Магнитный тормоз запитан
- **S** - Датчик оборотов активен
- **I** - Вход синхронизации активен
- **O** - Выход синхронизации активен

и отдельные индикаторы (флаги)

- **EEPR** - Загорается зеленым цветом, когда подвижка оснащена микросхемой памяти EEPROM.|br| Встроенный профиль подвижки загружается из микросхемы памяти EEPROM, при условии, что установлен флаг EEPROM_PRECEDENCE (XILab Settings->About device tab), что позволяет подключать различные подвижки к контроллеру с автоматической настройкой.
- **HOMD** - Загорается после успешного выполнения команды home (), что означает, что шкала относительного положения откалибрована с помощью аппаратного датчика абсолютного положения, такого как концевой выключатель.

- **WndA/WndB** имеет 1 из 4 состояний: - Обмотка A/B отключена. - Состояние обмотки A/B неизвестно. - Обмотка A/B закорочена. - Обмотка A/B подключена и работает исправно.

Важно: Статус определяется использованием статистических данных во время перемещения, что отнимает занимает время и делает этот статус довольно бесполезным в обычных приложениях. Поэтому в данный момент эта функция отключена.

- **ENCD** - Состояние энкодера имеет 1 из 5 состояний: - Энкодера отсутствует. - Состояние энкодера неизвестно. - Энкодер подключен и неисправен. - Энкодер подключен и работает, но считает в другом направлении. - Энкодер подключен и работает исправно.
- **PWHT** - Перегрев силового драйвера. Управление двигателем отключено до восстановления рабочей температуры драйвера. Этого не должно происходить в коробочных версиях контроллера. Это может произойти в версии контроллера с «голой» платой и с пользовательским радиатором. Решение: используйте другой радиатор.
- **SLIP** - Обнаружено проскальзывание мотора. Флаг устанавливается, когда положение энкодера и положение шага слишком далеки друг от друга. Вы можете отключить флаг «Position control» или увеличить ошибку в поле «threshold» на вкладке «XILab Settings->Position control», чтобы предотвратить возникновение этой ошибки.

- **WRM** - Загорается при существенной разнице между сопротивлениями обмоток шагового двигателя. Обычно это происходит с поврежденным шаговым двигателем у которого полностью или частично закорочены обмотки. Вы можете диагностировать проблему согласно инструкциям *в нашем руководстве.*

Важно: WRM алгоритм изначально не был рассчитан на использование для подвижек с ременной передачей (например, 8MRB450-350 или 8MRB240-152) из-за того что ремень может растягиваться и вибрировать. Вибрация, как правило, происходит на высоких скоростях, что и сбивает работу WRM алгоритма. Для подвижек ременной передачи это нормальное поведение

- **ENGR** - Загорается красным при возникновении ошибки управления мотором. Отказ алгоритма управления мотором означает, что он не может определить правильное решение с помощью полученных данных обратной связи. Единичный отказ может быть вызван механической проблемой. Повторяющийся сбой может быть вызван неправильной настройкой мотора.
- **EXTI** - Ошибка вызвана игнорированием внешнего сигнала EXTIO, этой ошибкой можно управлять в настройках (XILab Settings->EXTIO tab)
- **ErrC** - обнаружена ошибка команды. Полученная команда отсутствует в списке известных команд контроллера. Наиболее вероятная причина - устаревшая прошивка, которую можно обновить в настройках XILab ->вкладка About device ->кнопка Autoupdate.
- **ErrD** - обнаружена ошибка целостности данных. Данные внутри команды и ее код CRC не соответствуют, поэтому данные не могут считаться действительными. Эта ошибка может быть вызвана EMI в интерфейсе UART/RS232.
- **ErrV** - обнаружена ошибка значения. Значения в команде не могут быть применены без корректировки, потому что они выходят за допустимый диапазон. Вместо исходных значений использовались исправленные.
- **Ctbl** - Статус загрузки корректирующей таблицы. Флаг загорается зеленым, когда для позиционера была загружена и применена корректирующая таблица.

5.2.3 Главное окно программы XILab в режиме управления несколькими осями

Важно: XILab позволяет одновременно открывать до 32 осей, но в многоосевом интерфейсе будут отображаться только первые 3 (оси X/Y/Z). Однако возможно работать и с другими открытыми осями (теми, которые визуально не отображаются в XILab), для этого нужно использовать скриптовый

язык XILab. Для визуальной работы с большим количеством осей (более трех) потребуется запустить несколько окон XILab. Например у вас есть 5-ти осевой контроллер, тогда в первом окне XILab вы можете открыть первые три оси, а во втором окне оставшиеся две.

Если вам нужно открыть более 3 осей в одном окне и визуально контролировать их, вы можете использовать неподдерживаемые примеры, написанные на языках *C* и *Python*

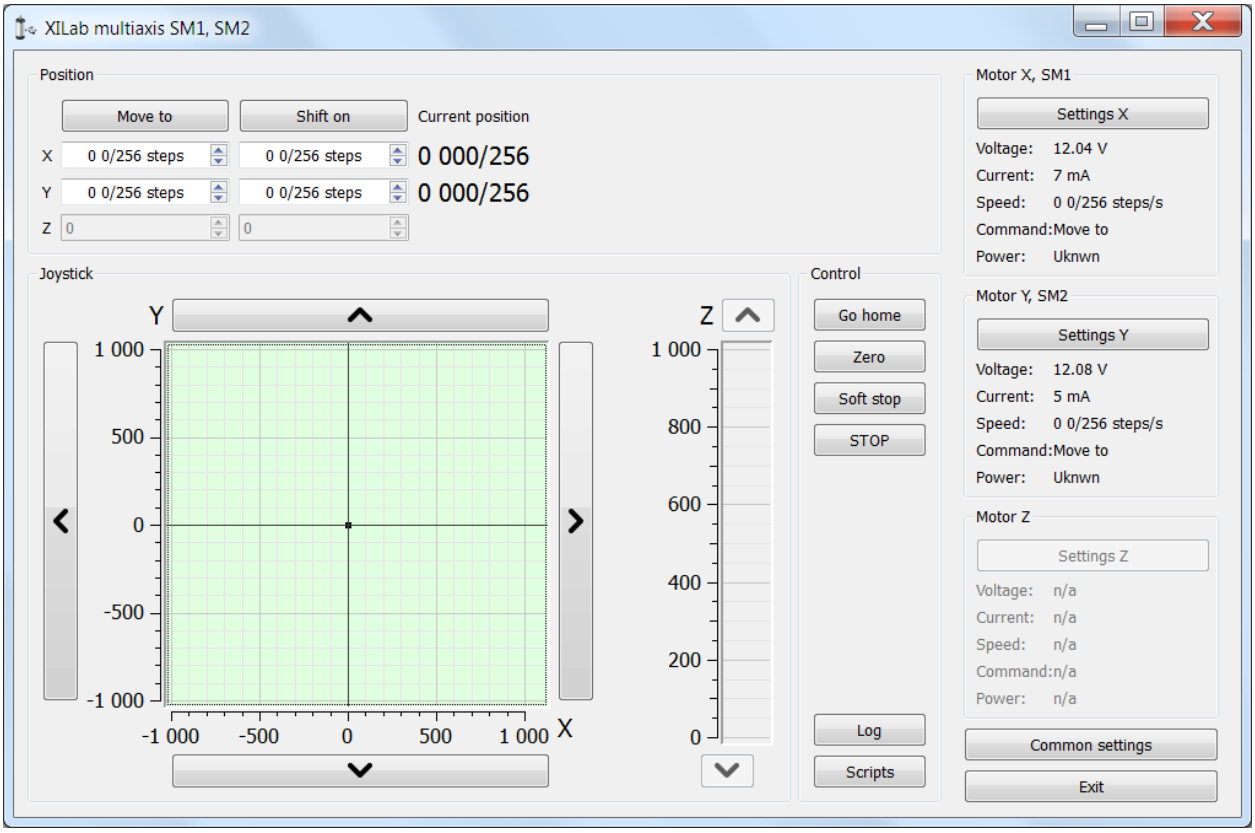


Рис. 5.11: Главное окно программы XILab

В левой верхней части окна в группе параметров **Position** находятся индикаторы текущей позиции. В левой нижней части окна расположены блоки **Joystick** и **Control**, представляющие собой графический элемент управления движением по нескольким осям и блок кнопок соответственно. В правой верхней части в блоках **Motor** находятся данные о состоянии контроллеров и подключенных к ним моторов в настоящий момент. В правой нижней части окна расположена группа кнопок для управления программой в целом. Рассмотрим эти группы более подробно.

5.2.3.1 Блок позиции и движения

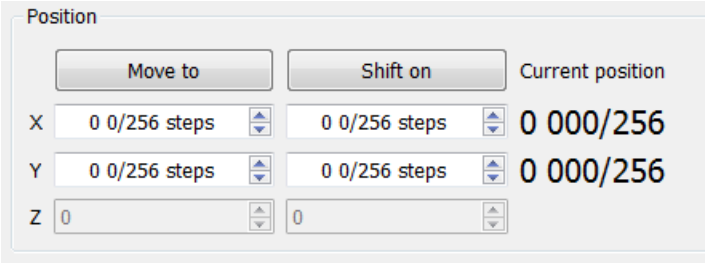


Рис. 5.12: Блок позиции и движения

В столбце *Current position* расположены индикаторы текущей позиции в шагах или калиброванных единицах (см. далее) для осей X, Y и Z сверху вниз. Кнопка *Move to* осуществляет перемещение в координату, задаваемую элементами управления в этом столбце, а кнопка *Shift on* осуществляет смещение на задаваемое расстояние от текущей позиции. Если какой-то из контроллеров отсутствует или временно отключен, то соответствующая ему строка становится недоступной.

5.2.3.2 Блок виртуального джойстика

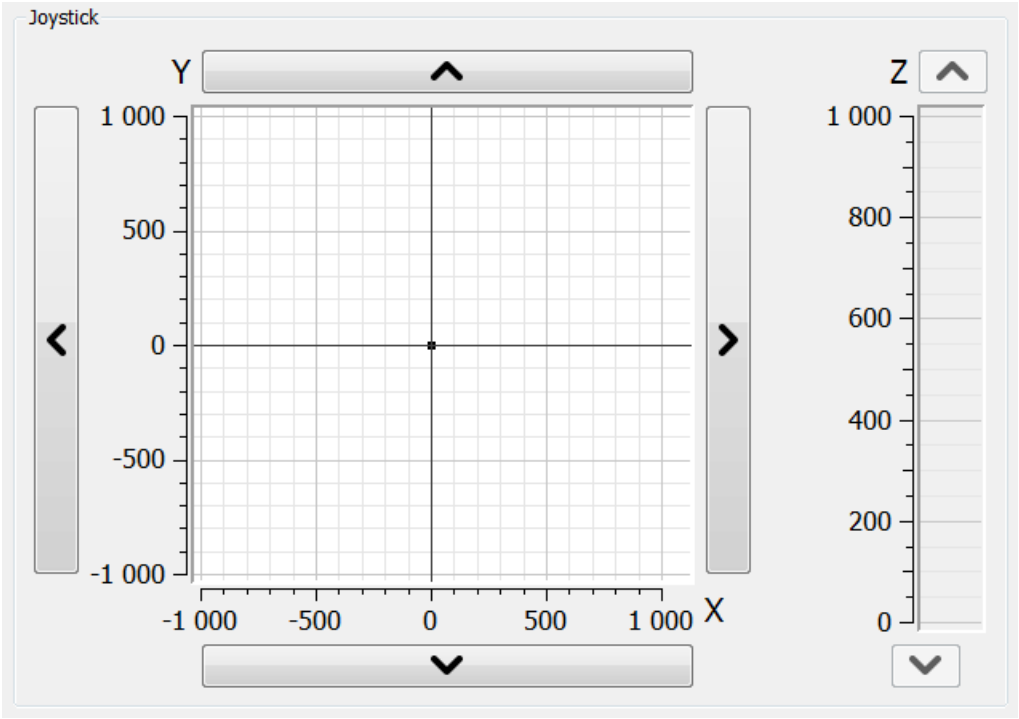


Рис. 5.13: Блок виртуального джойстика

В этом блоке текущая координата контроллеров визуализируется точкой с двумя линиями на плоскости для осей X-Y и линией для оси Z.

Здесь также возможно управлять движением контроллеров несколькими способами:

- При клике мышкой где-либо на плоскости X-Y или в столбце Z соответствующий контроллер или контроллеры начинают движение в выбранную координату со своими настройками движения.

- При нажатии и удержании мышкой экранных кнопок со стрелками вверх, вниз, вправо и влево соответствующая ось начинает движение в выбранном направлении. Движение *прекращается с замедлением* при отпускании кнопки (посылается команда SSTP).
- При нажатии и удержании кнопок клавиатуры вправо, влево, вверх, вниз, PageUp, PageDown при нахождении фокуса ввода в блоке джойстика ось X, Y или Z соответственно начинает движение в направлении увеличения или уменьшения координаты. Движение *прекращается с замедлением* при отпускании кнопки (посылается команда SSTP). Наличие фокуса ввода в блоке джойстика можно отследить по изменению цвета его фона с белого на светло-зеленый.

Масштаб осей задается в блоке «Slider settings» вкладки *General motor* в окне Settings индивидуально для каждого контроллера. Если включена опция *пользовательских единиц*, то координата по соответствующей оси отсчитывается в этих единицах. В случае если считанное из контроллера положение по какой-либо оси выходит за диапазон оси то соответствующий индикатор не отображается.

5.2.3.3 Блок управления



Рис. 5.14: Блок управления

Кнопка *Go home* осуществляет поиск начальной позиции независимо для каждого контроллера, см. раздел *Настройка исходного положения*.

Кнопка *Zero* обнуляет текущую позицию мотора и значение энкодера для каждого контроллера.

Кнопка *Soft stop* выполняет команду плавной остановки для каждого контроллера.

Примечание: Кнопка *STOP* посылает команду *немедленной остановки* каждому контроллеру, сбрасывает их состояния Alarm, очищает их буферы команд для синхронного движения и останавливает выполнение скрипта, если он запущен.

Кнопка *Log* открывает окно, отображающее лог программы. Сюда попадает диагностическая информация (ошибки, предупреждения, информационные сообщения) от самой программы XiLab, от библиотеки libximc, а также от исполняемых скриптов.

Кнопка *Scripts* открывает окно работы со скриптами, см. раздел *Скрипты*.

5.2.3.4 Блок индикаторов состояния контроллеров и моторов

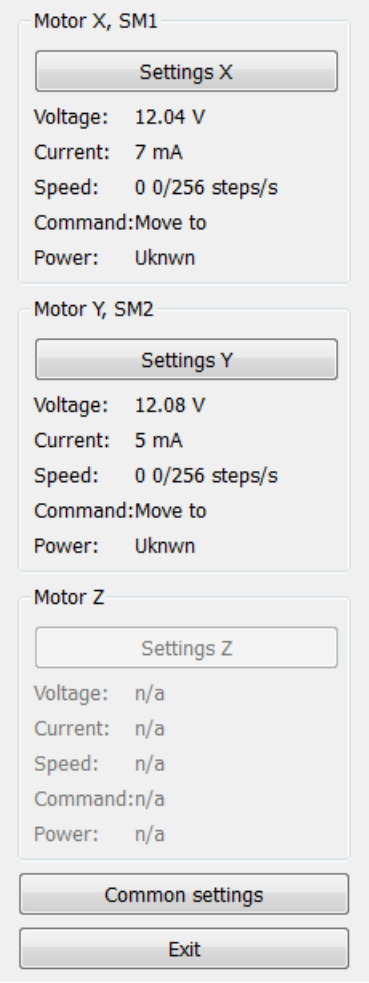


Рис. 5.15: Блок индикаторов состояния контроллеров и моторов

Здесь расположены три блока индикаторов состояния контроллеров и моторов для осей X, Y и Z. В заголовке блока расположен серийный номер соответствующего контроллера. Каждый блок содержит индикаторы:

- Voltage - напряжение на силовой части мотора.
- Current - ток потребления силовой части мотора.
- Speed - текущая скорость мотора.
- Command - последняя выполненная или выполняемая команда контроллера.
- Power - состояние питания мотора.

Кнопка Settings X,Y,Z открывает настройки контроллера соответствующего этой оси, см. раздел *Настройки программы*.

Снизу от блоков индикаторов состояния расположены две кнопки: *Common settings* и *Exit*.

Кнопка *Common settings* открывает диалог с общими настройками, включающими в себя настройки логирования и соответствие серийных номеров контроллеров отображаемым осям X, Y и Z.

Кнопка *Exit* осуществляет корректное завершение работы, см. раздел *Корректное завершение работы*.

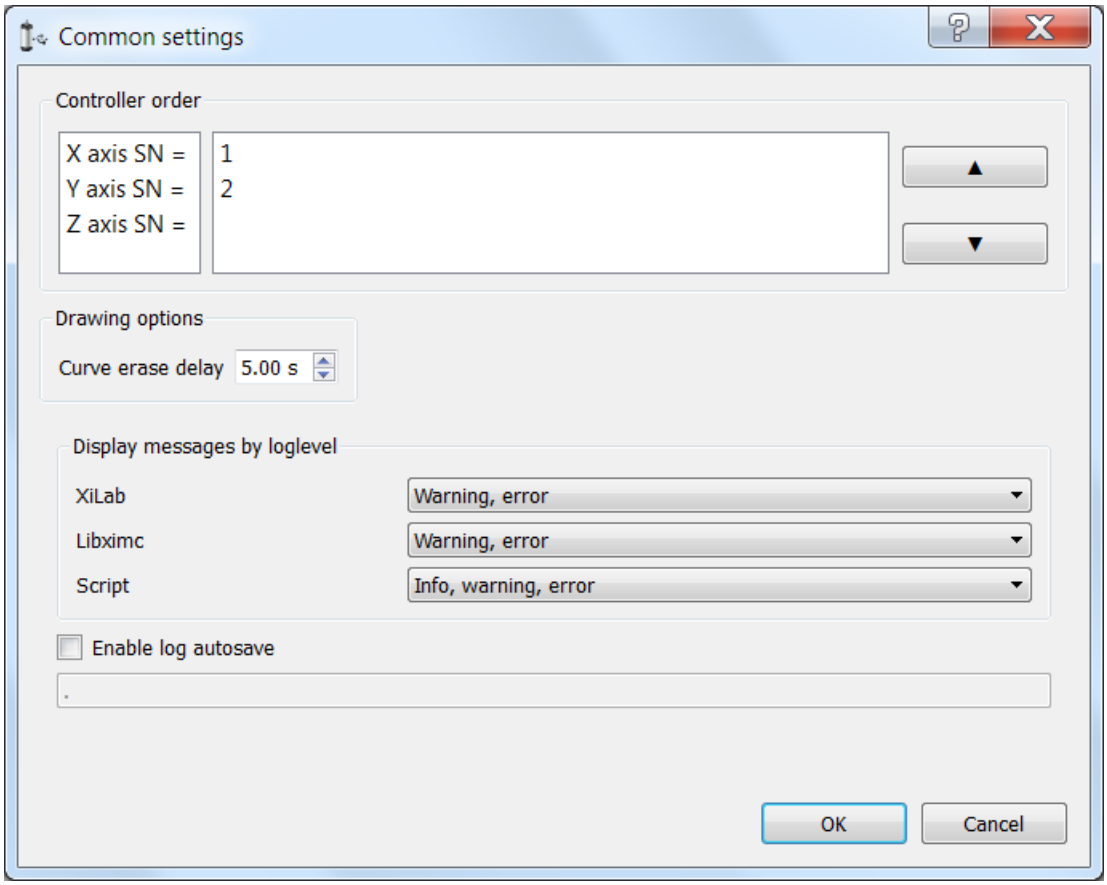


Рис. 5.16: Диалоговое окно общих настроек многоосного интерфейса

В окне общих настроек расположен блок управления порядком отображения осей в интерфейсе, блок настроек отрисовки кривой движения и блок настроек логирования.

Порядок осей можно менять, выбрав серийный номер желаемой оси и нажимая кнопки «вверх» и «вниз», которые расположены справа.

Первая ось в списке, то есть ось с серийным номером в строке справа от надписи «X axis SN = » будет идентифицироваться как «ось X», вторая как «ось Y», третья, если она присутствует, как «ось Z».

Если вы введете значение N, большее нуля секунд в элементе управления «Удержание стирания» в «Параметры рисования», то последние N секунд движения контроллера оси X и Y будут отображаться как траектория в 2D-пространстве, наложенном на виртуальное окно джойстика.

Блок настроек логирования полностью аналогичен *одноосной версии*.

В нем можно настроить уровень подробности логгирования: не выводить ничего (None), выводить только ошибки (Error), ошибки и предупреждения (Error, Warning), ошибки, предупреждения и информа-

ционные сообщения (Error, Warning, Info) для каждого из источников: программа XILab, библиотека libximc и модуль скриптов Scripts.

При включенном автосохранении запись в файл производится каждые 5 секунд. Имя файла: «xilab_log_ГГГГ.ММ.ДД.csv», где ГГГГ, ММ и ДД - текущие год, месяц и день соответственно. Формат сохраненных данных: CSV.

5.2.4 Настройки программы

Кнопка Settings из *главного окна* программы открывает окно настроек.

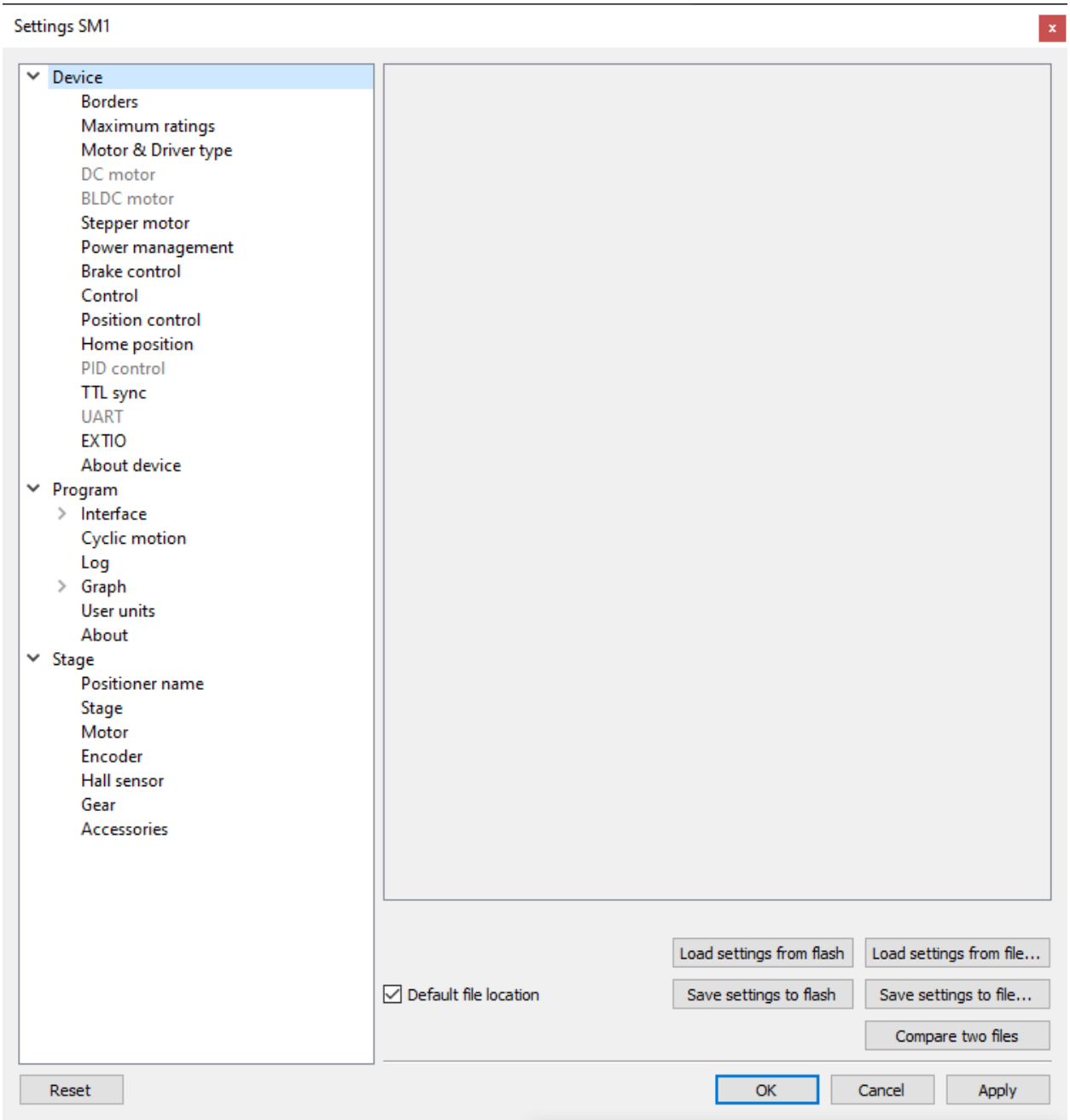


Рис. 5.17: Основное окно настроек XILab

Параметры приложения делятся на три группы: настройки контроллера - группа «Device», настройки приложения XILab - группа «Program», характеристики позиционера - группа «Stage».

В первой группе *Device* находятся параметры, значения которых могут храниться непосредственно в устройстве (во флеш памяти или в ОЗУ контроллера).

Вторая группа *Program* содержит настройки программы XILab, которые не записываются в контроллер, а служат для управления работой самого интерфейса XILab.

Третья группа *Stage* содержит информацию о параметрах позиционера, считанную из ROM-микросхемы памяти позиционера.

Важно: Информация на вкладке *«Stage»* временно не используется

Описание кнопок **Load setting from flash** и **Save settings to flash** находится в разделе *Хранение параметров во flash-памяти контроллера*.

Все настройки программы из первой и второй группы настроек могут быть записаны во внешний файл при нажатии на кнопку **Save settings to file** .

При нажатии в XILab на кнопку **Load setting from file...** настройки программы загружаются в окно Settings.

При нажатии на кнопку **Compare two files** открывается диалог выбора двух файлов, а затем сравниваются все их настройки и выводится список различий. Отсутствующие в одном из файлов ключи помечаются в таблице как «<NO KEY>».

Кнопка **OK** закрывает окно Settings с сохранением всех измененных настроек в контроллер, кнопка **Cancel** закрывает окно без сохранения, кнопка **Apply** сохраняет настройки без закрытия окна.

Кнопка **Reset** сбрасывает все изменения настроек, сделанные после последнего нажатия **Apply**, или после открытия окна Settings, если кнопка **Apply** не нажималась.

5.2.5 Графики

Кнопка **Chart** из *главного окна* программы открывает окно для работы с графиками.

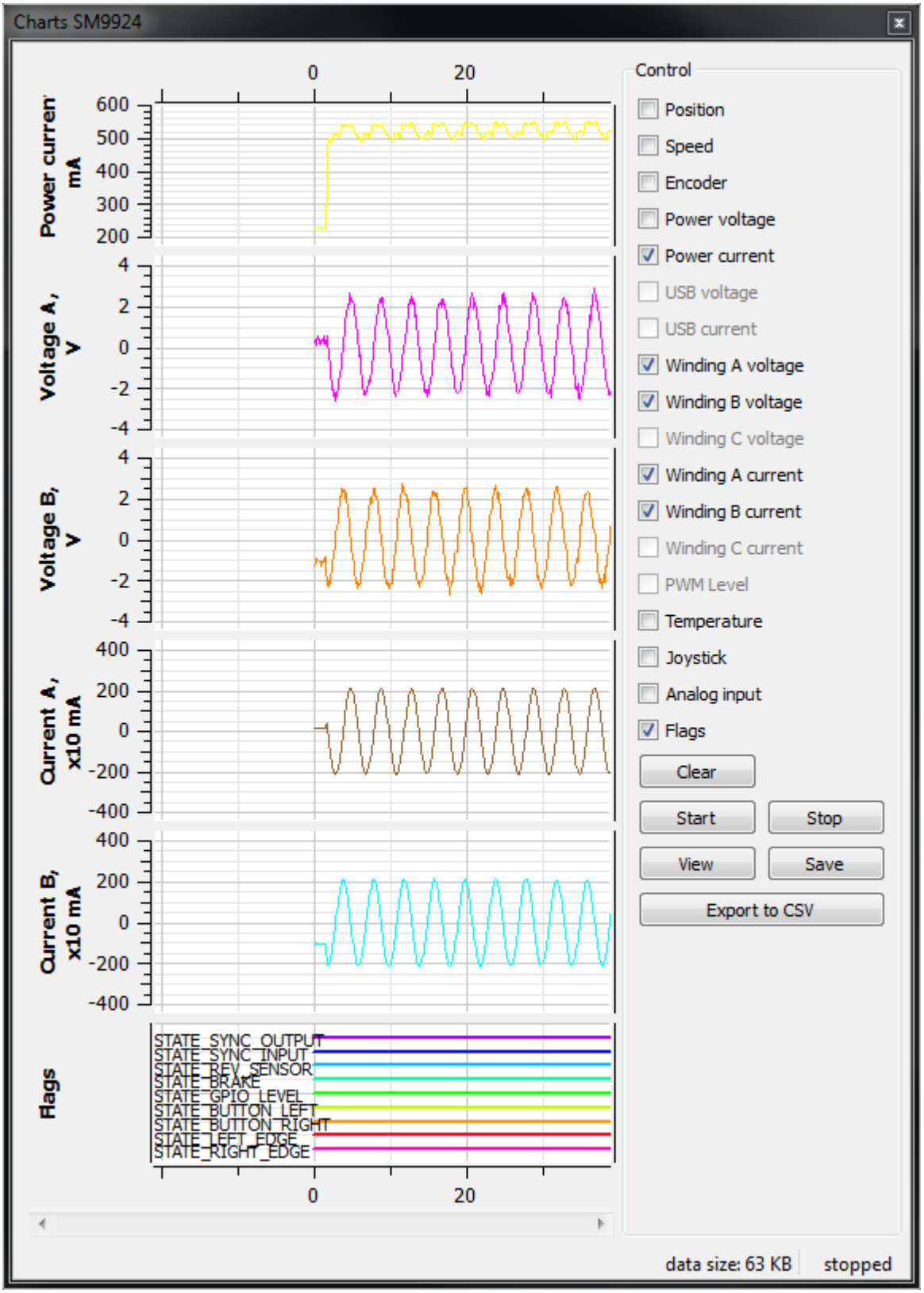


Рис. 5.18: Окно графиков программы XILab

В левой части окна расположены графики величин, в правой части окна расположен блок Control, содержащий элементы управления графиками. Вверху блока Control расположены флаги включения различных графиков, внизу блока Control расположена группа кнопок для управления сохраненными данными графиков.

5.2.5.1 Отображаемые на графиках величины

- *Position* - первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь. В случае работы с DC-мотором в этом поле находится текущая позиция по данным с энкодера, в случае работы с ШД (шаговым двигателем) в этом поле содержится значение текущей позиции в шагах;
- *Speed* - текущая скорость;
- *Encoder* - позиция по второстепенному датчику положения;
- *Power voltage* - напряжение силовой части;
- *Power current* - ток потребления силовой части;
- *USB voltage* - напряжение на USB (устарело);
- *USB current* - ток потребления по USB (устарело);
- *Winding A current* - в случае ШД, ток в обмотке A; в случае бесщеточного, ток в первой обмотке; в случае DC в единственной;
- *Winding B current* - в случае ШД, ток в обмотке B; в случае бесщеточного, ток в второй обмотке; в случае DC не используется;
- *Winding C current* - в случае бесщеточного, ток в третьей обмотке; в случае ШД и DC не используется;
- *Winding A voltage* - в случае ШД, напряжение на обмотке A; в случае бесщеточного, напряжение на первой обмотке; в случае DC на единственной;
- *Winding B voltage* - в случае ШД, напряжение на обмотке B; в случае бесщеточного, напряжение на второй обмотке; в случае DC не используется;
- *Winding C voltage* - в случае бесщеточного, напряжение на третьей обмотке; в случае ШД и DC не используется;
- *PWM level* - коэффициент заполнения ШИМ (только для двигателей постоянного тока);
- *Temperature* - температура процессора контроллера;
- *Joystick* - значение входного сигнала от джойстика;
- *Analog input* - значение аналогового входа для пользовательских задач;
- *Flags* - состояние флагов контроллера.

5.2.5.2 Функции кнопок

- *Clear* - очищает сохраненные данные и окно графиков;
- *Start* - начинает запись данных и отображение графиков. Если включена опция «Break data update when motor stopped» в **Program -> Graph**, то запись данных и автопрокрутка графиков при остановленном двигателе происходить не будет;
- *Stop* - останавливает считывание данных;
- *Save* - сохраняет данные графиков в файл;
- *Load* - загружает данные графиков из ранее сохраненного файла;

- *Export to CSV* - экспортирует данные графиков в файл формата CSV

5.2.5.3 Ограничение значения

Если установлено ограничение для скорости, напряжения силовой части или тока потребления силовой части, это ограничение отображается на графике пунктирной линией:

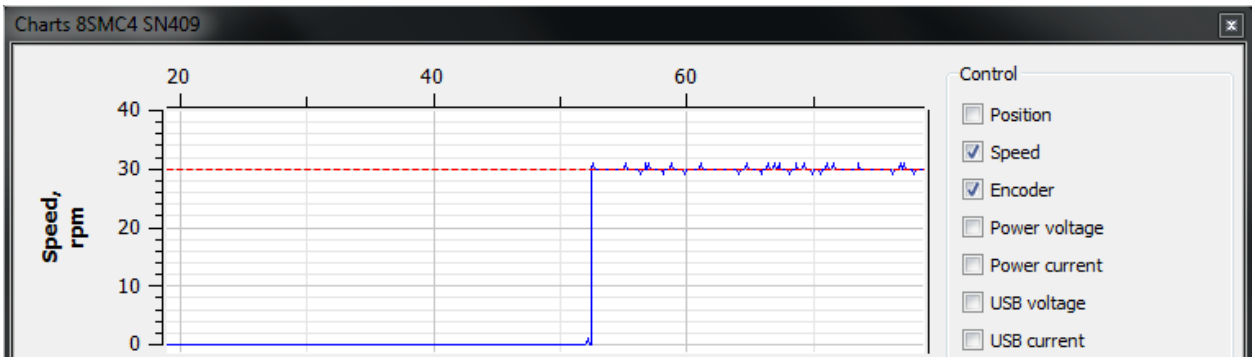


Рис. 5.19: График скорости в программе XiLab с ограничением скорости

5.2.6 Скрипты

Кнопка «Script» из *главного окна* программы открывает окно для работы со скриптами.

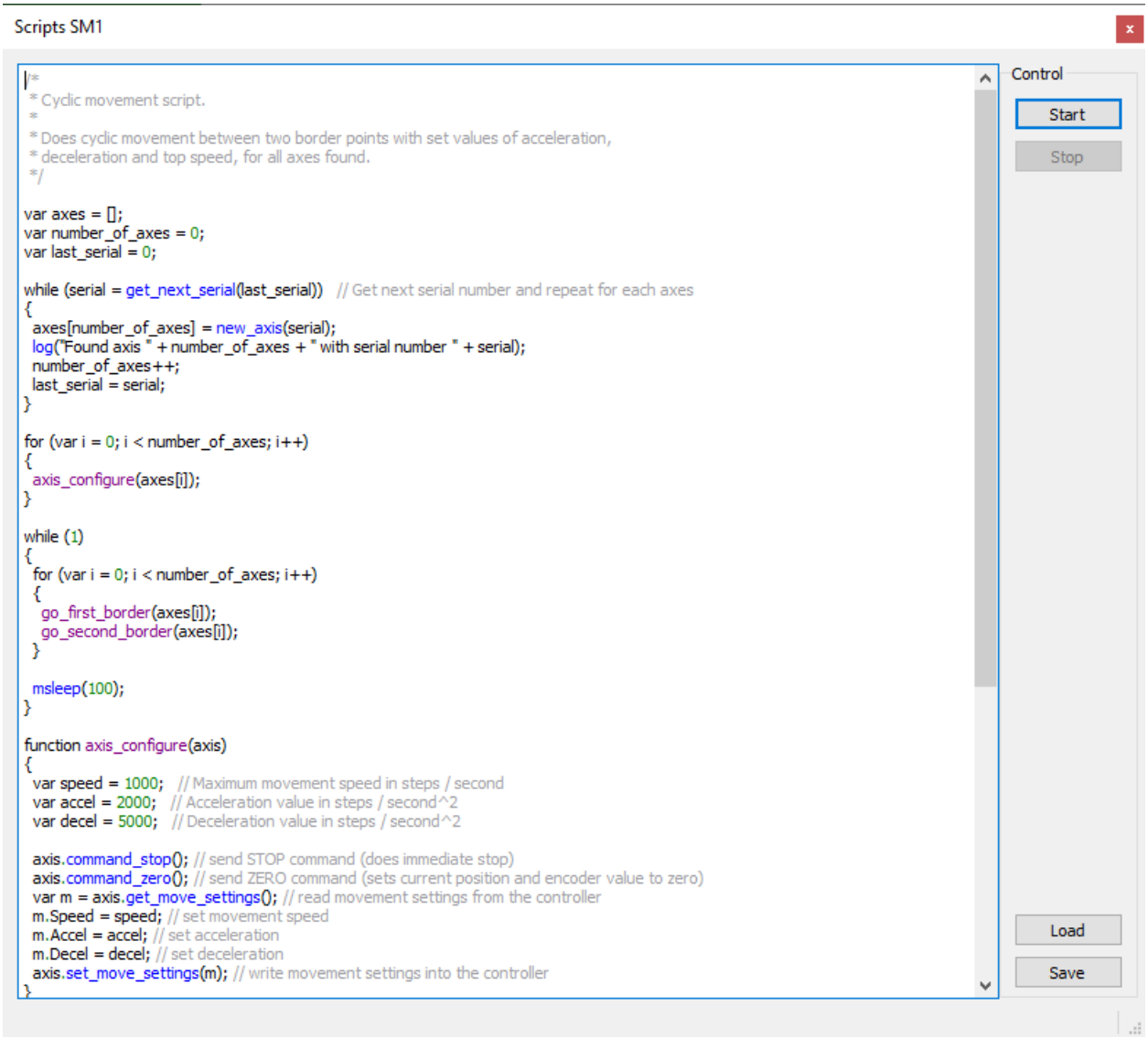


Рис. 5.20: Окно скриптов XILab

В левой части окна расположено поле для редактирования текста скрипта, в правой части окна расположен блок Control, содержащий элементы управления скриптами.

5.2.6.1 Функции кнопок

- *Start* - запускает выполнение скрипта. Неактивна, если скрипт уже выполняется. Сразу после нажатия кнопки и до начала интерпретации скрипта происходит автоматическое сохранение скрипта во временный файл (см. ниже).
- *Stop* - останавливает выполнение скрипта. Неактивна, если скрипт в данный момент не выполняется.
- *Save* - вызывает диалог выбора файла куда будет сохранен текущий скрипт, отображаемый в окне. Неактивна во время выполнения скрипта.
- *Load* - вызывает диалог выбора файла для загрузки в окно скриптов. Неактивна во время выполнения скрипта. Внимание, в случае загрузки все несохраненные изменения текста в окне будут

потеряны!

В момент старта программы в окно «Scripting» загружается последнее сохраненное содержимое текста в окне. Автосохранение происходит перед каждым стартом скрипта, а также перед выходом из XILab, файл автосохранения находится в директории пользовательских настроек программы и имеет имя «scratch.txt».

Примечание: Выполнение скрипта останавливается при нажатии кнопки «Stop» в главном окне программы, это сделано для экстренной остановки движения в случае необходимости.

Кнопка *STOP* посылает команду *немедленной остановки* каждому контроллеру, сбрасывает их состояния Alarm, очищает их буферы команд для синхронного движения и останавливает выполнение скрипта, если он запущен.

Описание языка скриптов находится в разделе Программирование.

5.2.7 Лог XiLab

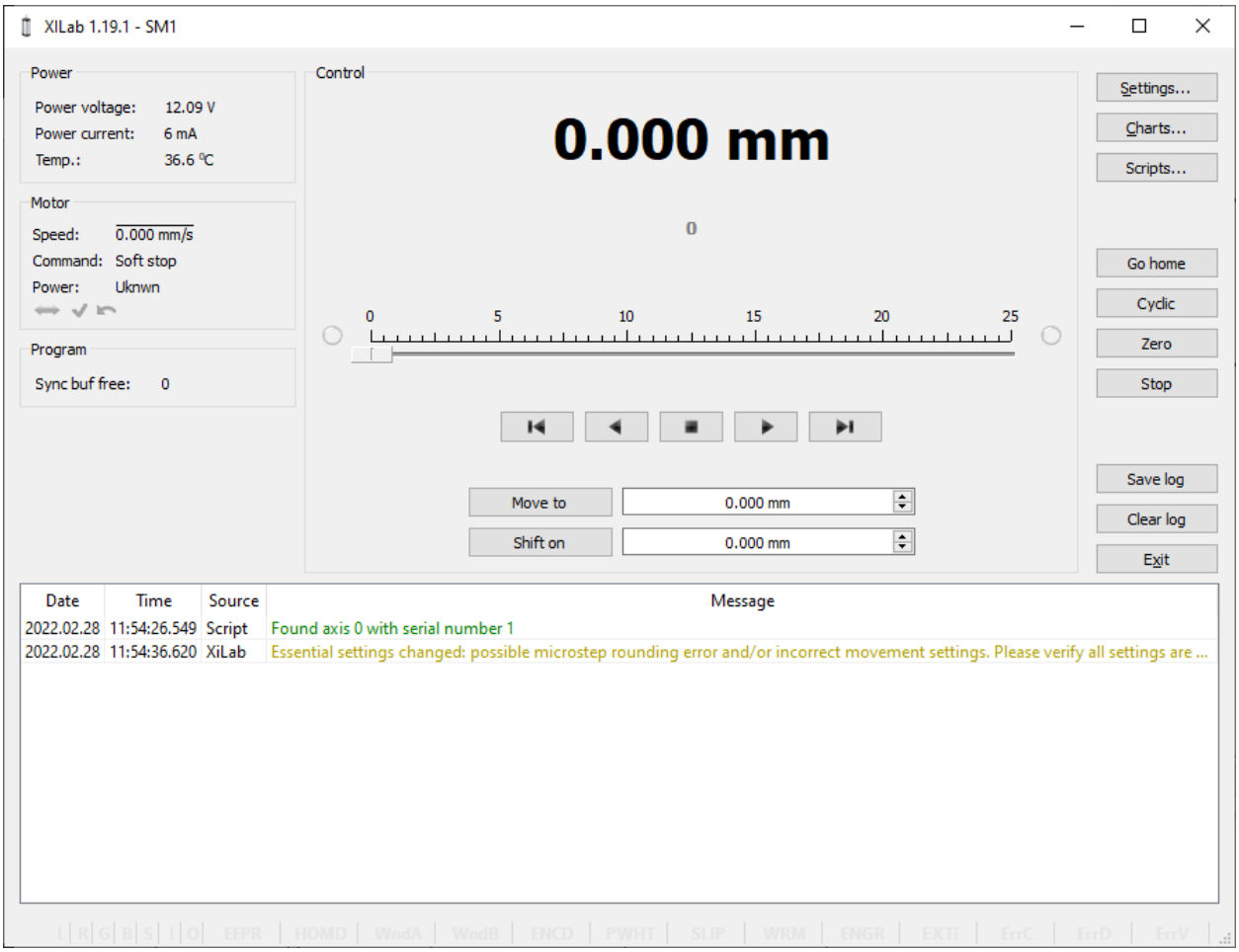


Рис. 5.21: Окно лога программы XILab

В нижней части главного окна XILab расположен лог, в который записываются сообщения от библиотеки libximc. В него также выводятся сообщения самого XILab и интерпретатора *скриптов*.

Лог имеет 4 колонки: дата и время возникновения записи, источник и текст сообщения.

Сообщения имеют уровень логгирования, означающий важность сообщения: ошибка, предупреждение и информационное сообщение. Сообщения ошибок выводятся красным цветом, предупреждения желтым, информационные сообщения зеленым.

Настроить тип выводимых сообщений в лог можно на вкладке *Настройка логирования* в окне настроек программы.

5.3 Настройки контроллера

5.3.1 Настройка кинематики движения (Шаговый двигатель)

В окне *Настройки программы* **Device -> Stepper motor**

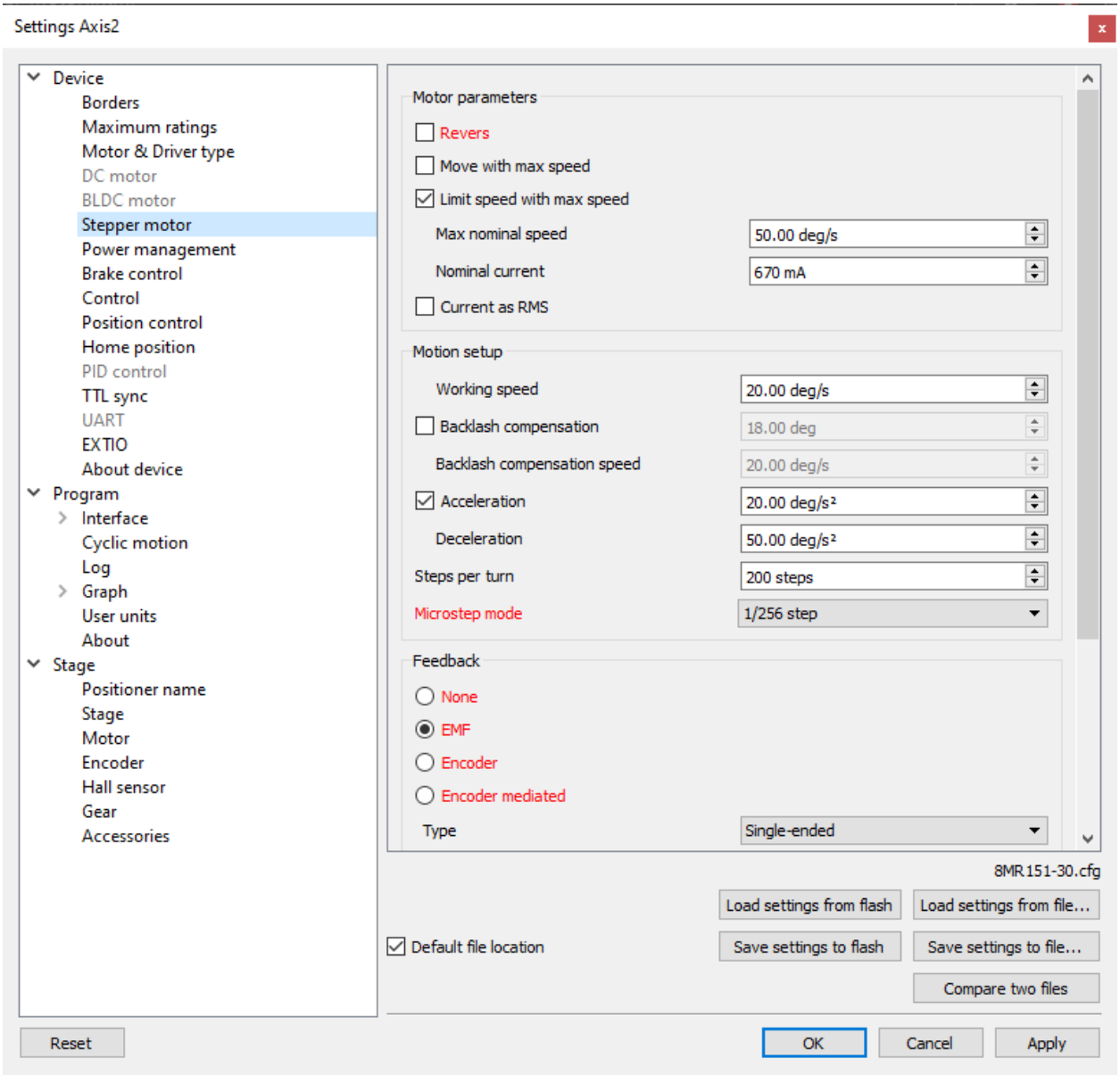


Рис. 5.22: Окно настроек кинематики движения шагового двигателя

5.3.1.1 Motor parameters - настройки, непосредственно связанные с электромотором

Revers - установка этого флага позволяет связать направление вращения мотора с направлением счета текущей позиции. Измените состояние флага, если положительное вращение мотора уменьшает счетчик позиции. Действие этого флага равносильно подключению обмотки мотора в обратной полярности.

Move with max speed - при установленном флаге мотор игнорирует заданную скорость и вращается с максимальной допустимой скоростью.

Limit speed with max speed - при установленном флаге контроллер ограничивает максимальную скорость по количеству шагов в секунду значением поля *Max nominal speed*. Например, если скорость превысила номинальное значение, контроллер будет снижать выходное воздействие, пока значение скорости не вернется в пределы нормы. Однако при этом контроллер останется в рабочем состоянии и будет выполнять текущую задачу.

Max nominal speed - номинальная скорость работы мотора.

Nominal current - номинальный ток через двигатель. Контроллер будет ограничивать ток этим значением.

Current as RMS - при установленном флаге задаваемое значение тока интерпретируется как среднеквадратичное значение тока, если флаг снят, то задаваемое значение тока интерпретируется как максимальная амплитуда тока. Подробнее в разделе *Расчёт номинального тока*.

5.3.1.2 Motion setup - настройки, связанные с кинематикой движения

Working speed - скорость движения.

Backlash compensation - компенсация люфта. Так как механика позиционера не идеальна, существует различие при подходе к заданной точке справа и слева. При включенном режиме компенсации люфта позиционер будет подходить к точке всегда с одной стороны. Установленное значение определяет количество шагов, на которое позиционер будет проходить заданную точку чтобы возвращаться к ней с одной и той же стороны. Если указанное число больше нуля, позиционер будет подходить к точке всегда справа. Если меньше нуля, то всегда слева.

Backlash compensation speed - скорость компенсации люфта. При включенном режиме компенсации люфта *Backlash compensation* позиционер будет подходить к точке справа или слева с установленной скоростью, определяемой количеством шагов в секунду.

Acceleration - включает режим движения с ускорением, числовое значение поля это величина ускорения движения.

Deceleration - величина замедления движения.

Steps per turn - определяет для контроллера количество шагов для совершения мотором одного полного оборота. Параметр устанавливается пользователем.

Microstep mode - режим деления шага. Доступно 9 режимов: от целого шага до 1/256 шага. Описание режимов в разделе *Поддерживаемые типы двигателей*.

5.3.1.3 Настройки обратной связи

В качестве датчика обратной связи для шаговых двигателей может использоваться энкодер. Для шаговых двигателей доступно три режима обратной связи.

None - без обратной связи. Движение осуществляется в шагах.

Encoder - режим задания движения в величинах отсчета энкодера. При включении данной опции доступен выбор типа энкодера: недифференциальный, дифференциальный или автоматическое определение.

Encoder mediated - в этом случае движение осуществляется за несколько итераций с контролем положения по завершению каждой итерации по энкодеру.

Encoder counts per turn - параметр определяет количество импульсов *энкодера* на один полный оборот оси мотора.

Encoder reverse - реверс энкодера.

5.3.2 Настройка диапазона движения и концевых выключателей

В окне *настроек программы* **Device -> Borders**

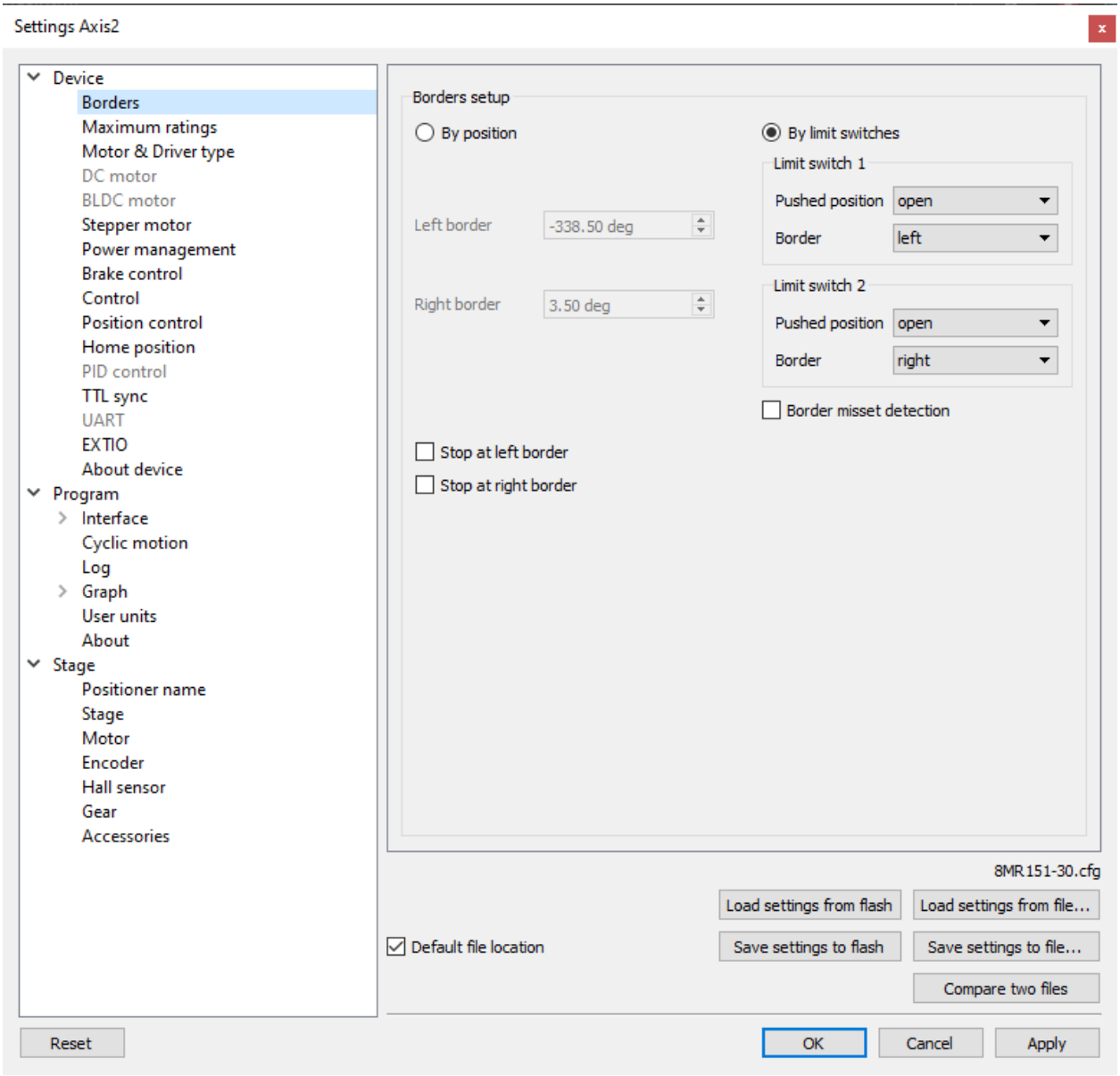


Рис. 5.23: Окно настроек диапазона движения и концевых выключателей

Группа параметров **Borders setup** содержит параметры границ и концевых выключателей. Эти параметры позволяют предотвратить выход позиционера за допустимые физические границы его перемещения или ограничить диапазон перемещения в соответствии с требованиями пользователя. Границы могут определяться либо по положению позиционера (определяемому по внутреннему счетчику шагов контроллера), либо по *концевым выключателям*, установленным в крайних положениях позиционера.

Если аппаратных ограничителей на диапазон движения нет, а позиционер требует такого ограничения, то можно использовать программные концевые выключатели. Для установки границ по виртуальным концевым выключателям необходимо выбрать пункт *By position* и указать значения *Left border* и *Right border*, которые соответствуют левому и правому краю соответственно. Используются поля левой границы и правой границы (значение правой границы должно быть больше левой). В этом режиме левый концевик считается активным если текущая позиция меньше левой границы, а правый - если текущая позиция больше правой границы движения. Срабатывание происходит за время около одной миллисекунды.

Предупреждение: Программное ограничение диапазона работает надежно только, если не происходит непосредственного задания новой позиции командами ZERO или SPOS, нет потери шагов или неисправности энкодера, при его использовании для позиционирования, а также не происходит частой потери питания во время движения. Если возникла одна из таких проблем, то программный диапазон надо перенастроить. Автоматически это можно сделать если есть подходящий опорный датчик с помощью *автоматической калибровки нулевой позиции*.

Для установки границ по концевым выключателям необходимо выбрать пункт *By limit switches* и настроить работу каждого из двух концевых выключателей *Limit switch 1* и *Limit switch 2*.

Pushed position - состояние концевика, когда он достигнут: замкнутое или разомкнутое.

Border - расположение данного концевика: слева или справа рабочего диапазона позиционера.

Для принудительной остановки мотора при достижении границ отметьте *Stop at left border* и/или *Stop at right border*. Тогда контроллер будет игнорировать любые команды, подразумевающие движение в сторону концевика, если соответствующий концевик уже достигнут.

При достижении граничного положения загорается соответствующий индикатор в главном окне программы.

Если флаг *Border misset detection* установлен, мотор останавливается при достижении обоих границ. Эта настройка нужна для предотвращения поломки двигателя при обнаружении потенциально неправильно настроенных концевиков. Обязательно прочитайте подробнее про работу контроллера в этом режиме в разделе *Расположение концевых выключателей на трансляторах*.

5.3.3 Настройка предельных параметров контроллера

В *окне настроек* программы **Device -> Maximum ratings**

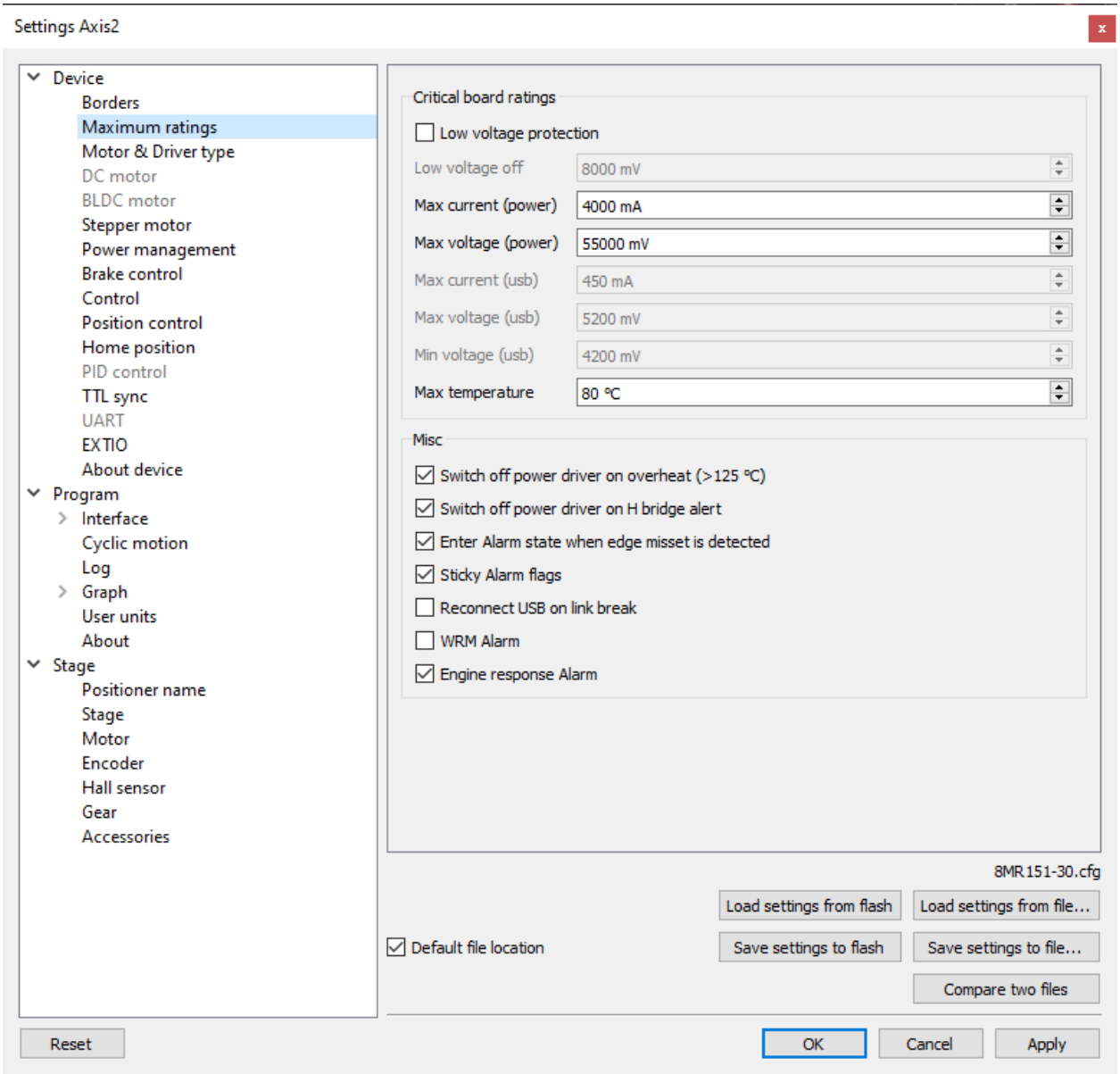


Рис. 5.24: Окно настроек критических параметров контроллера

Critical board ratings - эта группа параметров отвечает за максимальные значения входного тока *Max current (power)* и напряжения *Max voltage (power)* на контроллере, максимальные значения тока *Max current (usb)* и напряжения *Max voltage (usb)* на USB, минимальное значение напряжения *Min voltage (usb)* на USB, а также температуру *Temperature* платы (если измерение температуры производится у данной версии контроллера).

Если значение тока, потребляемого контроллером или величина питающего напряжения, или температура выйдут за пределы установленных здесь значений, контроллер отключает все силовые выходы и переходит в состояние Alarm. При этом в главном окне также будет информация о состоянии *Alarm* (фон окна сменится на красный) и у параметра вышедшего за допустимые границы значение отображается синим или красным цветом (ниже или выше порога соответственно).

При отмеченном флаге *Low voltage protection* включается защита от низкого напряжения питания. *Low voltage off* это то напряжение питания, при котором контроллер переходит в состояние *Alarm*.

В группу **Misc** входят все остальные настройки критических параметров.

Switch off power driver on overheat (>125 °C) - установка данного переключателя обеспечивает состояние *Alarm* при перегреве.

Switch off power driver on H bridge alert - установка данного переключателя обеспечивает состояние *Alarm* при сигнале неполадки в силовом драйвере.

Enter Alarm state when edge misset is detected - при установке этого флага контроллер войдет в состояние *Alarm* при обнаружении достижения неверной границы (срабатывание правого концевика при движении влево, или наоборот).

Sticky Alarm flags - залипание состояния Alarm. При снятом флаге *Sticky Alarm flags* контроллер снимает флаг причины Alarm при её исчезновении (например превышение тока произошло, а дальше обмотки отключились и ток снова снизился). При установленном флаге *Sticky Alarm flags* флаги причины Alarm и сам режим *Alarm* очищаются при посылке команды *Stop*.

Reconnect USB on link break - при установке этого флага будет включен блок перезагрузки USB-соединения со стороны контроллера при поломке связи.

WRM Alarm - при установке этого флага контроллер войдет в состояние *Alarm*, если имеется значительная разница в сопротивлении обмоток шагового двигателя. Флаг реализован для работы в режиме *Feedback None*.

Engine response Alarm - при установке этого флага контроллер войдет в состояние *Alarm* в ответ на действие управления двигателем.

5.3.4 Настройка параметров энергопотребления

В окне *настроек программы* **Device -> Power Management**

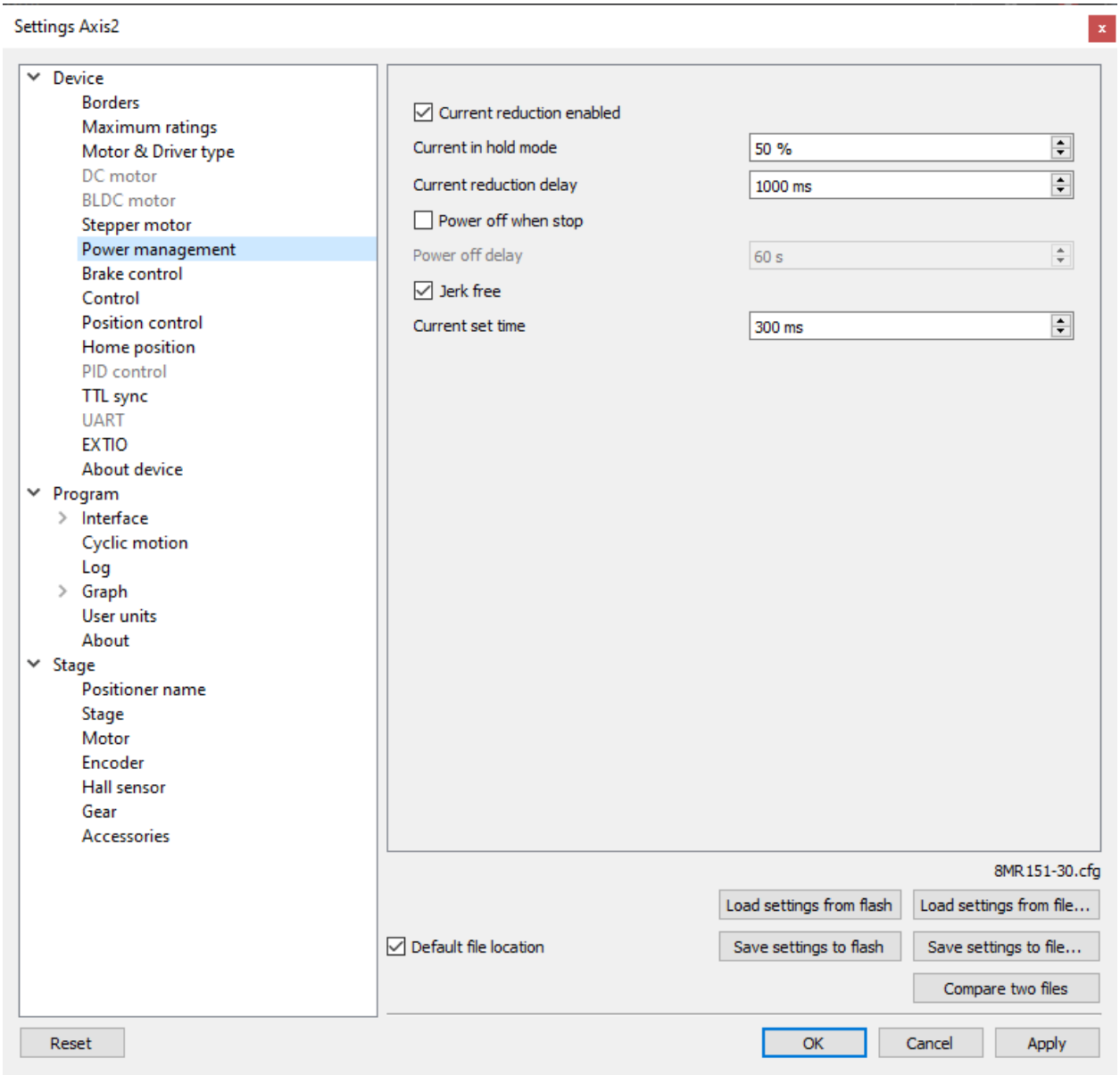


Рис. 5.25: Окно настроек энергопотребления

Current reduction enabled - включение функции снижения энергопотребления.

- *Current in hold mode* - параметр определяет уровень тока в % относительно номинального значения в режиме удержания (hold mode). Диапазон значений: 0 .. 100%.
- *Current reduction delay* - параметр определяет задержку от перехода в состояние STOP до уменьшения тока. Измеряется в мс. Диапазон значений: 0..65535 мс.
- *Power off when stop* - включение функции выключения питания с обмоток мотора при переходе в состояние STOP.
- *Power off delay* - параметр определяет задержку в секундах от перехода в состояние STOP до полного отключения питания мотора. Диапазон значений: 0..65535 с.
- *Jerk free* - включение функции сглаживания тока для устранения вибраций мотора.

- *Current set time* - параметр определяет время установления тока в миллисекундах. Ток не может меняться быстрее, чем на 100% от номинального за это время. Диапазон значений: 0..65535 мс.

Подробное описание этих параметров находится в главе *Управление питанием мотора*.

5.3.5 Настройка исходного положения

В *окне настроек программы* **Device -> Home position**

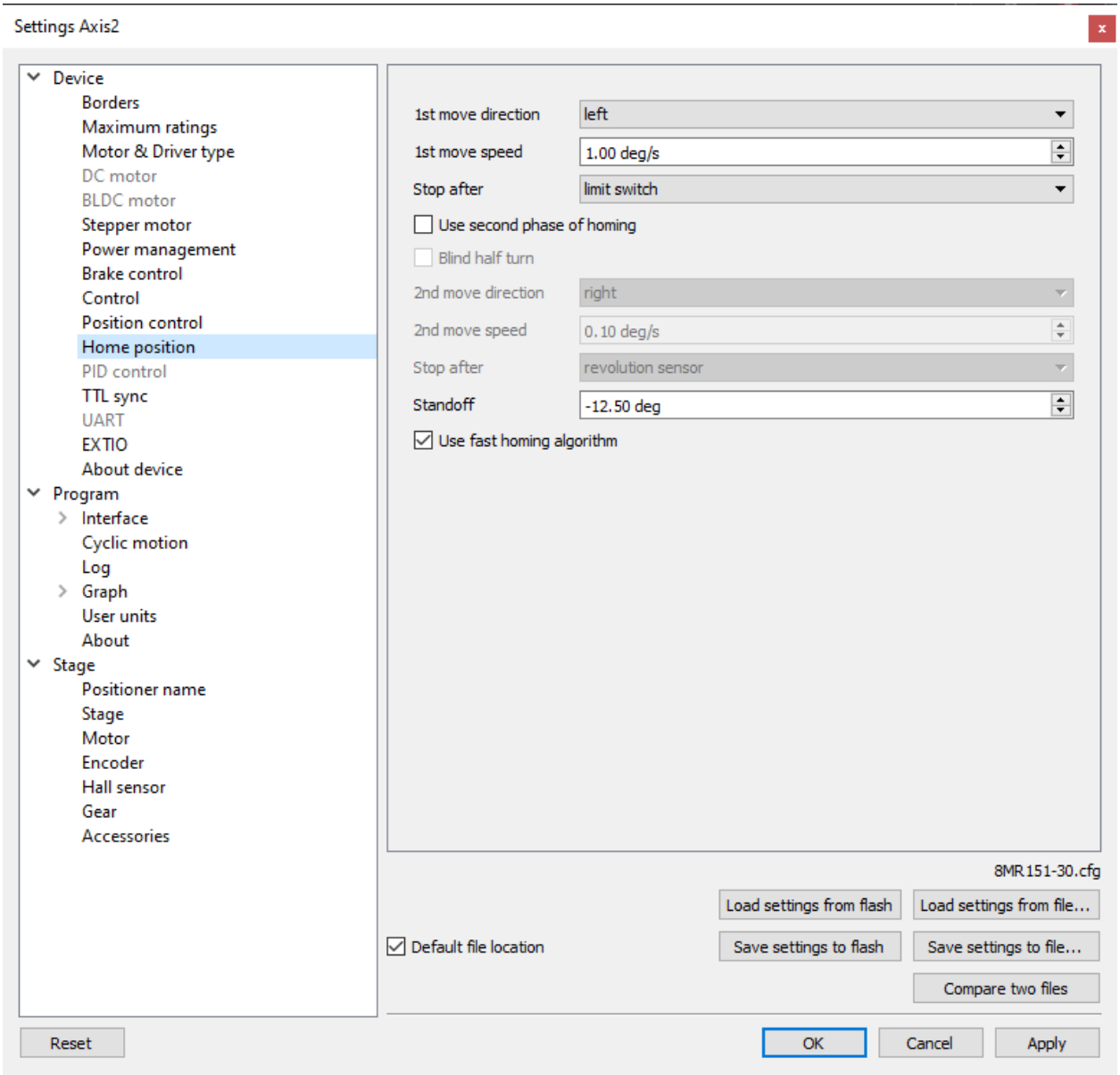


Рис. 5.26: Окно настроек исходного положения

Вкладка **Home position** устанавливает *параметры калибровки исходного положения* позиционера.

- *1st move direction* - задает направление движения мотора для поиска сигнала остановки (вправо или влево) для *стандартного* и *быстрого* алгоритмов автокалибровки.

- *1st move speed* - задает скорость движения для первой фазы *стандартного алгоритма калибровки* и второй фазы *быстрого алгоритма*.
- *Stop after* - задает источник сигнала остановки (*концевой выключатель*, *датчик оборотов* или *внешний импульс синхронизации*).
- *Use second phase of homing* - установка этого флага включает *точную докалибровку* домашней позиции (вторую фазу стандартного алгоритма калибровки).
- *Blind half turn* - при установленном флаге мотор игнорирует сигнал об окончании второй фазы калибровки в течение половины оборота. Это сделано для того, чтобы можно было задать однозначный порядок обнаружения датчиков, по которым происходит окончание первой и второй фазы калибровки, в случае, когда эти датчики расположены достаточно близко друг от друга.
- *2nd move direction* - задает направление движения мотора для поиска сигнала остановки (вправо или влево) для *второй фазы стандартного алгоритма* калибровки.
- *2nd move speed* - задает скорость движения для *второй фазы стандартного алгоритма* калибровки.
- *Stop after* (в блоке настроек для *второй фазы калибровки*) - задает источник сигнала остановки (концевой выключатель, датчик оборотов или внешний импульс синхронизации). Источник сигнала может отличаться от используемого для первой фазы калибровки.
- *Standoff* - задает отступ для финального смещения от реперной точки. Направление смещения задается знаком числового значения отступа (положительный отступ означает смещение вправо, отрицательный - влево).
- *Use fast homing algorithm* - этот флаг включает *быстрый алгоритм автокалибровки* для ускорения процесса поиска исходного положения.

5.3.6 Настройки синхронизации

В окне *настроек программы* **Device -> TTL sync**

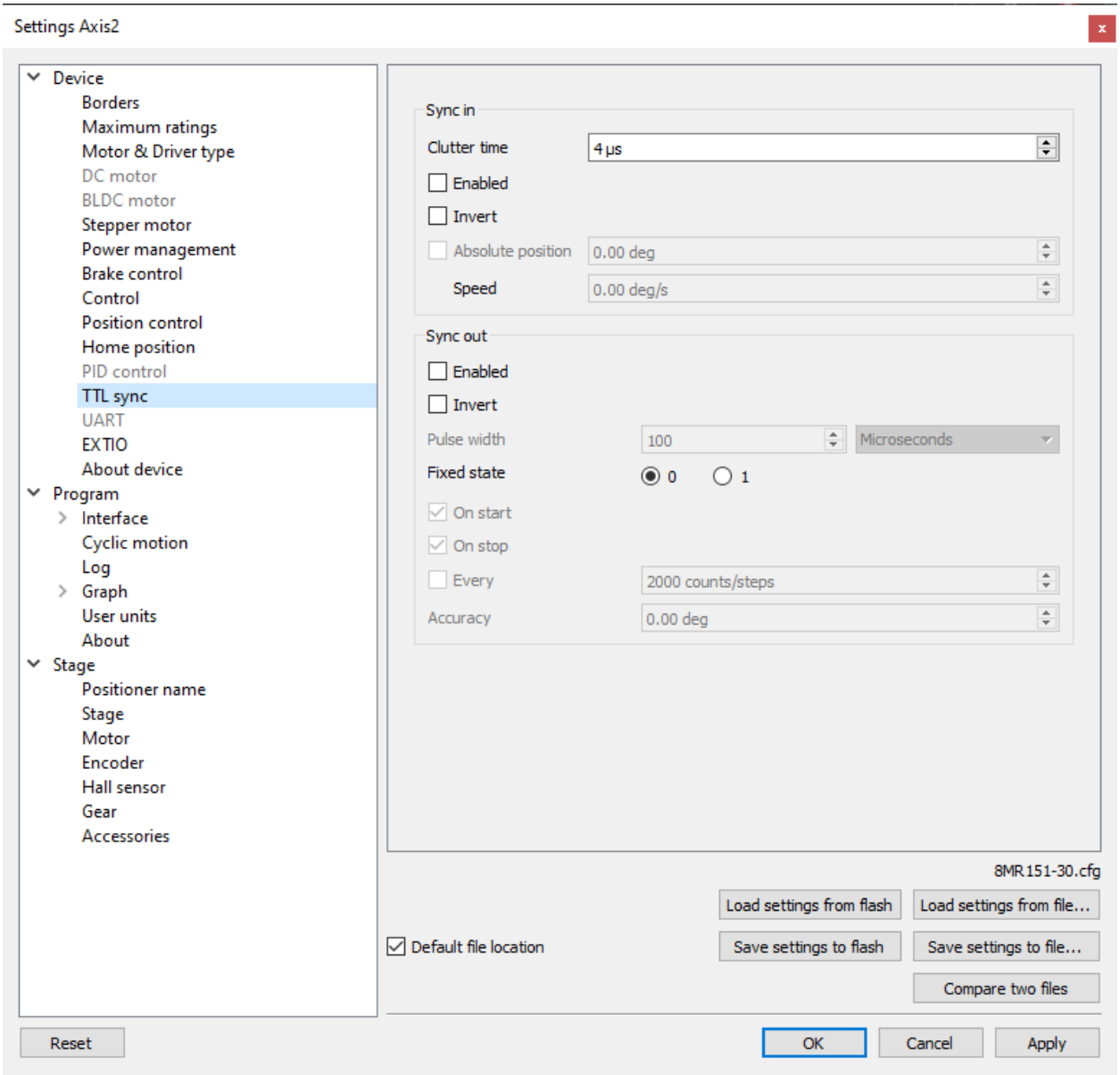


Рис. 5.27: Вкладка «Настройки синхронизации»

Подробно работа синхронизации описана в *разделе ТТЛ-синхронизации*.

Sync in:

- *Clutter time* - настройка минимальной длительности импульса синхронизации (в микросекундах). Определяет минимальную длительность входного синхроимпульса, который может быть зарегистрирован (защита от дребезга).
- *Enabled* - включает работу в режиме входа.
- *Invert* - установленный флаг соответствует срабатыванию по заднему фронту синхроимпульса.
- *Absolute position* - по приходу синхроимпульса при отмеченной настройке смещается в абсолютную координату, задаваемую полем шаг/микрошаг, при снятой настройке осуществляет относительное смещение на задаваемое расстояние.

- *Speed* - определяет скорость с которой производится движение по приходу импульса синхронизации.

Важно: Используя синхронизацию по входному импульсу, чтобы мгновенно начать движение, нужно отключить флаг jerk free, а также рекомендуется отключить флаг power off when stop.

Sync out:

- Выход синхронизации может использоваться как «выходной сигнал общего назначения».
- *Enabled* - если флаг установлен, то синхронизация выхода работает согласно настройкам. При снятом флаге значение выхода фиксировано и равно *Fixed state*.
- *Invert* - если флаг установлен, то нулевой логический уровень является активным.
- *Pulse width* - задает длину выходного импульса в микросекундах или шагах/импульсах энкодера.
- *Fixed state* - устанавливает логический уровень выхода в 0 или 1 соответственно.
- *On start* - синхронизирующий импульс генерируется в начале движения.
- *On stop* - синхронизирующий импульс генерируется при окончании движения.
- *Every* - синхронизирующий импульс генерируется каждые n импульсов энкодера.
- *Accuracy* - окрестность вокруг целевой координаты, попадание в которую считается попаданием в целевую позицию и вызывает генерацию импульса по остановке.

5.3.7 Настройка тормоза

В *окне настроек* программы **Device -> Brake control**

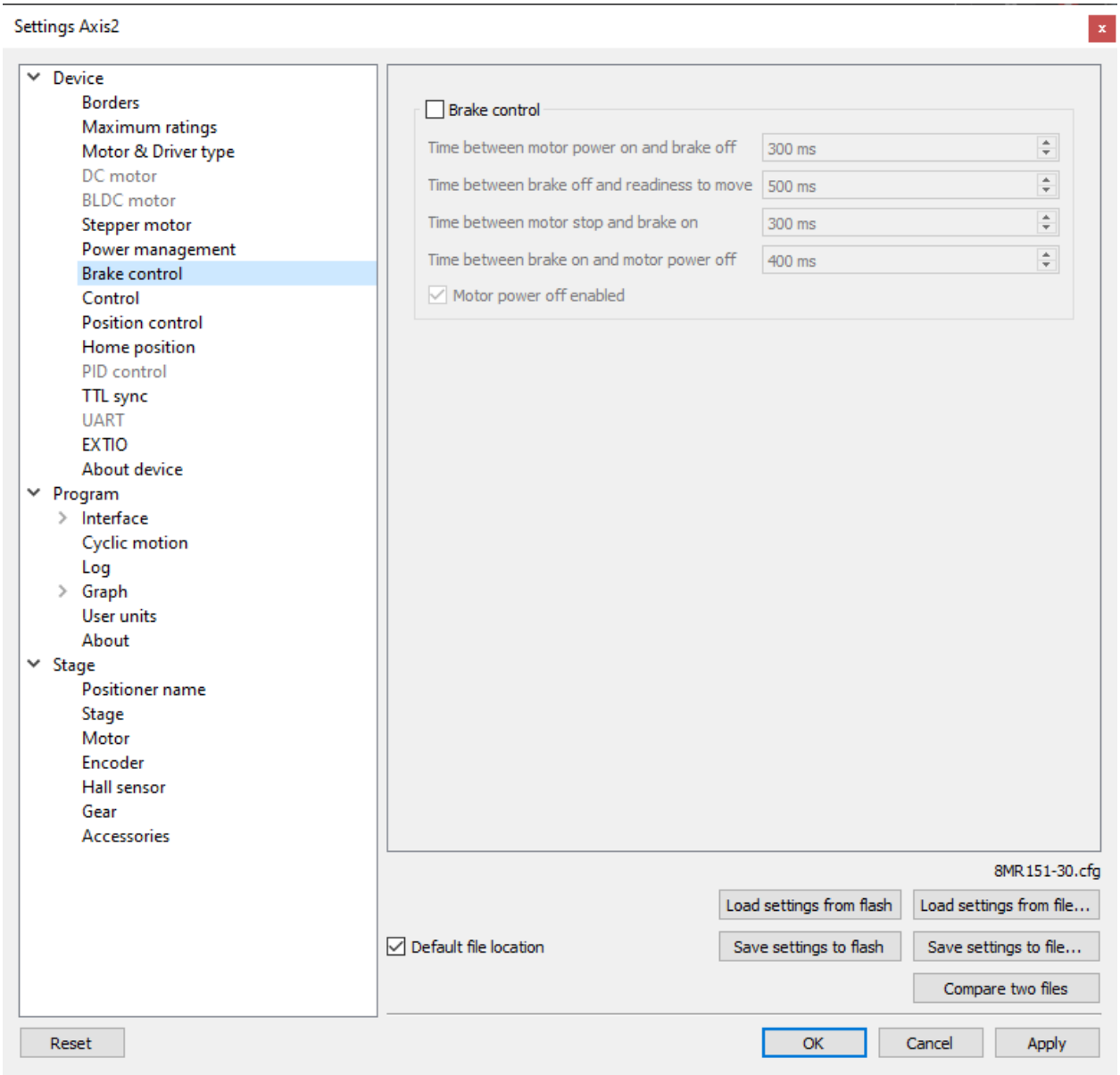


Рис. 5.28: Окно настроек магнитного тормоза

Для включения использования магнитного тормоза необходимо установить флаг *Brake control*.

Параметры:

- *Time between motor power on and brake off* - Время между включением питания мотора и отключением тормоза (мс).
- *Time between brake off and readiness to move* - Время между отключением тормоза и готовностью к движению (мс). Все команды движения начинают выполняться только по истечении этого времени.
- *Time between motor stop and brake on* - Время между остановкой мотора и включением тормоза (мс).
- *Time between brake on and motor power off* - Время между включением тормоза и отключением

питания (мс). Диапазон значений: от 0 до 65535 мс.

- Флаг *Motor power off enabled* означает, что при снятии питания с магнитного тормоза, тормоз отключает питание мотора.

Команды настройки описаны в разделе *Описание протокола обмена*.

5.3.8 Контроль позиции

В окне *настроек программы* **Device -> Position control**

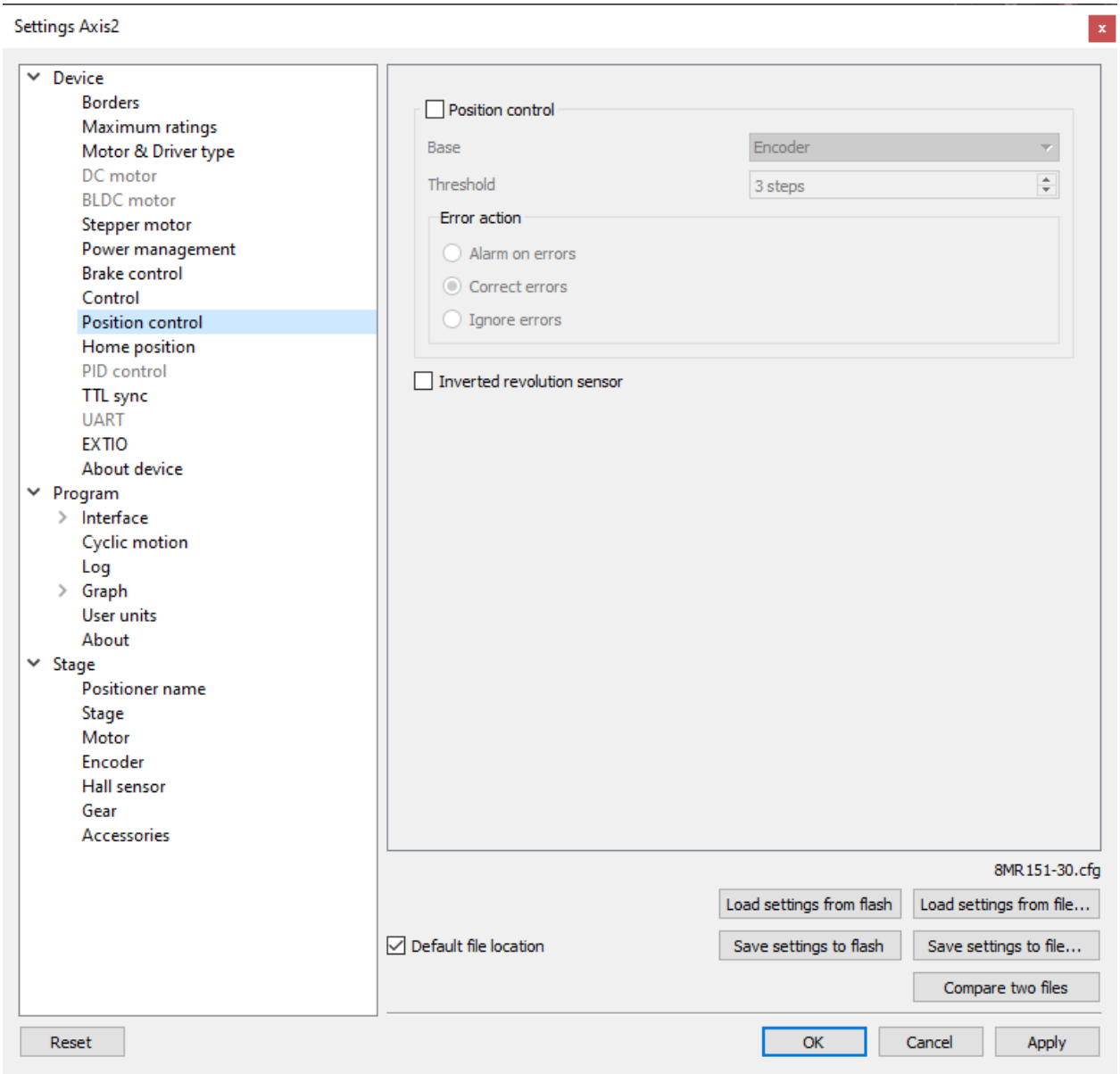


Рис. 5.29: Окно настроек контроля позиции

Для активации контроля позиции необходимо установить флаг параметра *Position control*.

Base - выбор устройства контроля позиции. В выпадающем окне выбирается: энкодер (Encoder) (см. раздел *Работа с энкодерами*) или датчик оборотов (Revolution sensor).

Threshold - определяет количество потерянных шагов (0..255), которое считается ошибочным. Если количество потерь превышает заданное число шагов, то устанавливается флаг рассогласования SLIP. Дальнейшие действия зависят от настройки *Error action*:

Если установлена опция *Alarm on errors*, то контроллер перейдет в состояние *Alarm*.

Если установлена опция *Correct errors*, то контроллер попытается скорректировать ошибку проскальзывания дополнительным движением (см. раздел *Обнаружение потери шагов*).

Если установлена опция *Ignore errors*, то контроллер не будет производить никаких дополнительных действий.

Inverted revolution sensor - при отмеченном флаге датчик оборотов считается сработавшим по уровню 1, при неотмеченном действует обычная логика: 0 - это срабатывание/активация/активное состояние.

Команды настройки описаны в разделе *Описание протокола обмена*.

Важно:

- **Feedback none**: в этом режиме «Position control» полезен и должен использоваться. «Position control» сравнивает позицию по энкодеру/датчику положения и пересчитывает ее в шаги. Если есть расхождения между позициями, в нижней части главного окна XiLab загорится флаг «SLIP». Кроме того, если вкл флаг «Alarm on errors», контроллер перейдет в состояние тревоги.
- **Feedback encoder**: «Position control» не нужно использовать, поскольку положение строго контролируется энкодером.
- **Feedback EMF**: алгоритм не должен использоваться с включенным флагом «Position Control». Для плавности хода в режиме EMF реализовано расхождение между фактическим положением и положением по профилю. Если этот флаг включен, могут быть вызваны ложные срабатывания Alarm.
- **Feedback encoder mediated**: не рекомендуется включать флаг «Position Control. Во время движения алгоритм не отличается от режима «none», но когда двигатель приезжает в позицию реальная позиция сравнивается с желаемой позицией по энкодеру, после чего алгоритм компенсирует расхождение в позициях до момента пока позиция по энкодеру не будет является желаемой.

5.3.9 Настройка внешних управляющих устройств

В окне *настроек программы* **Device -> Control**

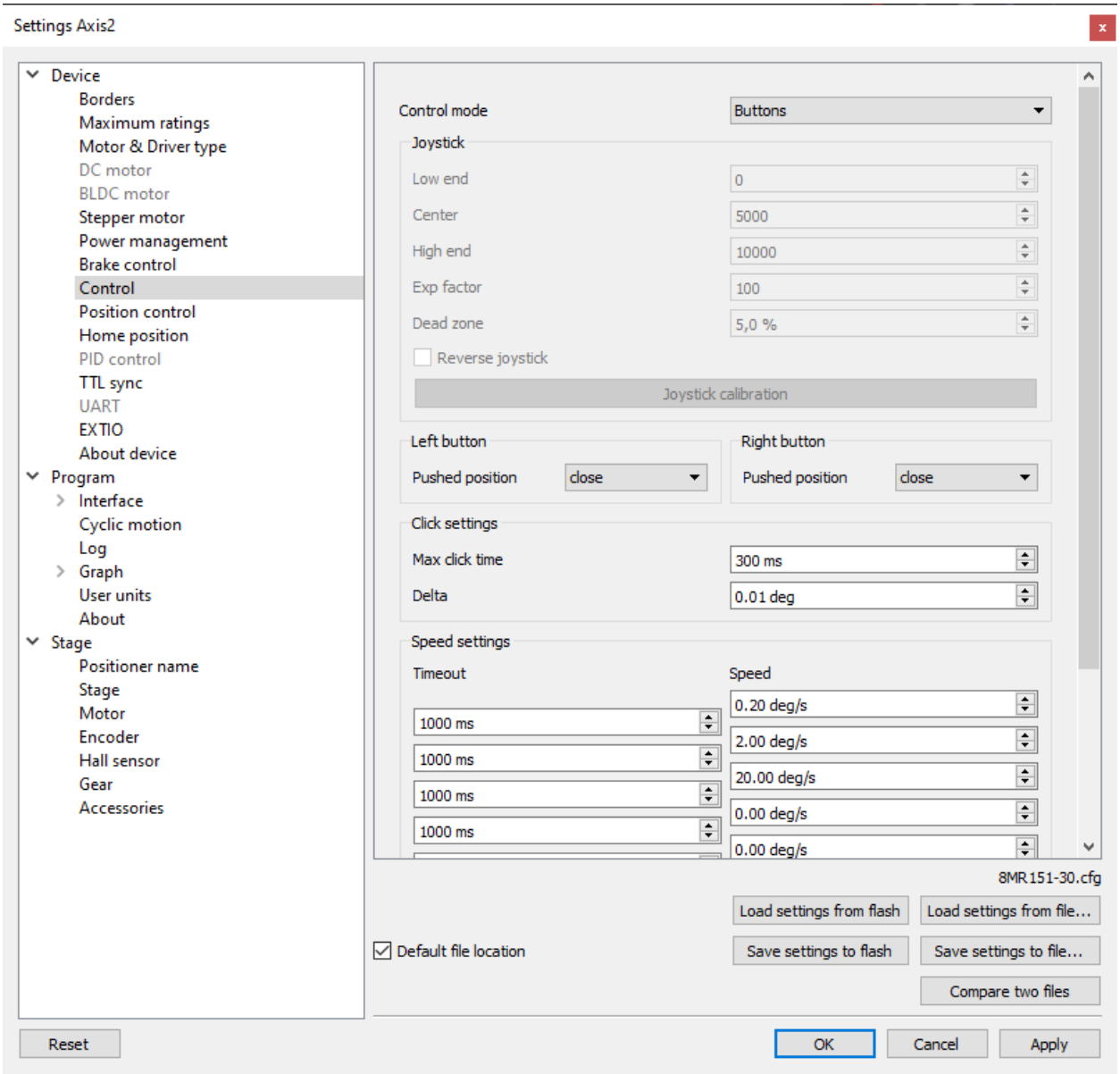


Рис. 5.30: Окно настроек внешних управляющих устройств

Control mode - выбор внешних устройств для управления двигателем.

- *Control disabled* - внешние устройства не используются
- *Joystick* - используется *джойстик*
- *Buttons* - используются *кнопки*

Важно: В режиме управления джойстиком физические и виртуальные кнопки остаются в рабочем состоянии

В блоке **Joystick** содержатся настройки джойстика.

Low end, *Center* и *High end* определяют нижнюю границу, середину и верхнюю границу диапазона

джойстика соответственно. То есть, нормированное значение АЦП джойстика равное или меньше Low end соответствует максимальному отклонению джойстика в сторону меньших значений.

Exp factor - параметр экспоненциальной нелинейности. См. *Управление с помощью джойстика*.

Dead zone - зона нечувствительности к отклонению джойстика от центрального положения. Минимальный шаг изменения: 0.1%, максимальное значение: 25.5%. Отклонению джойстика от положения Center на величину меньшую Dead zone соответствует нулевая скорость.

Reverse joystick - реверс воздействия джойстика. Отклонение джойстика к большим значениям приводит к отрицательной скорости и наоборот.

Кнопка *Joystick calibration* открывает диалог калибровки джойстика.

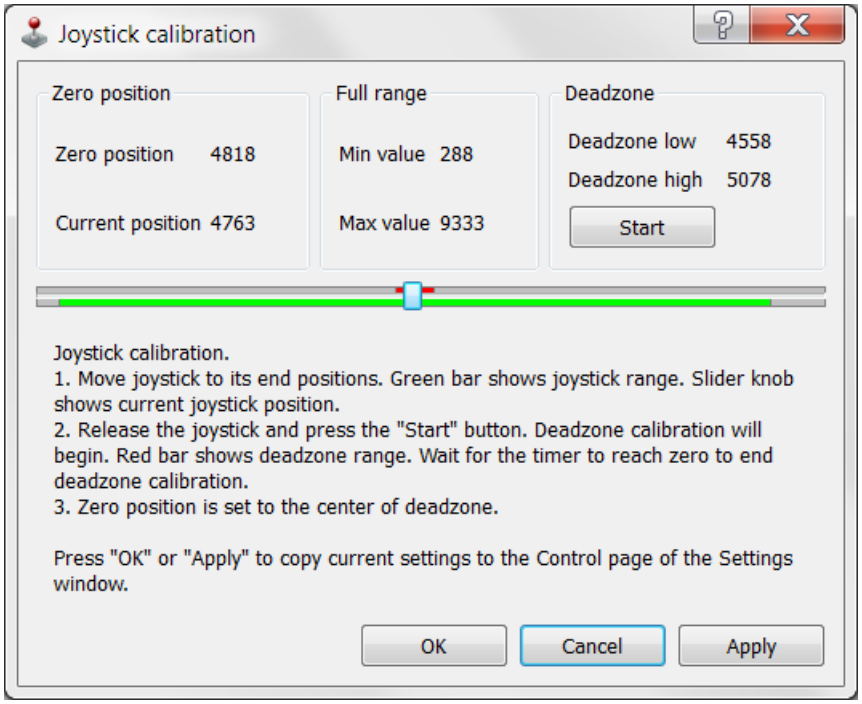


Рис. 5.31: Диалог калибровки джойстика

Калибровка сводится к автоматическому нахождению границ и зоны нечувствительности. Она происходит следующим образом:

Переместите джойстик в крайние положения - это позволит найти границы. Диапазон всех измеренных значений визуализируется зеленой линией.

Отпустите джойстик и нажмите кнопку Start - включится обнаружение зоны нечувствительности. В течение следующих 5 секунд имитируйте случайные воздействия на джойстик, которые не должны быть распознаны как смещение джойстика из нулевого положения. Диапазон зоны нечувствительности визуализируется красной линией.

Нажатие кнопки Apply передаст вычисленные значения в окно настроек, а нажатие OK передаст значения и закроет диалог калибровки.

Блоки **Left button** and **Right button** отвечают за настройку физических кнопок.

Pushed Position - определяет при каком состоянии (нажата кнопка или нет) подается сигнал на движение в контроллер.

- *Open* - отжатая кнопка считается командой движения.

- *Close* - нажатая кнопка считается командой движения.

В блоке **Click settings** настраивается «клик» кнопок. Нажатие кнопки на краткое время интерпретируется как клик.

Max click time - максимальное время клика. До истечения этого времени первая скорость не включается.

Delta - смещение (дельта) позиции. Контроллер смещается на это расстояние при каждом клике.

Блок **Speed settings** содержит настройки таймаутов и скоростей движения.

Timeout [i] - время, по истечении которого скорость переключается со Speed[i] на Speed[i+1]. Если какой-либо из Timeout[i] равен нулю, то переключение на последующие скорости происходить не будет.

Speed[i] - скорость, на которой должен работать мотор после времени Timeout[i-1]. Если какая-либо скорость равна нулю, то переключение на эту и последующие скорости происходить не будет.

Команды настройки описаны в разделе *Описание протокола обмена*.

5.3.10 Настройки цифрового входа-выхода общего назначения

В *окне настроек программы* **Device -> EXTIO**

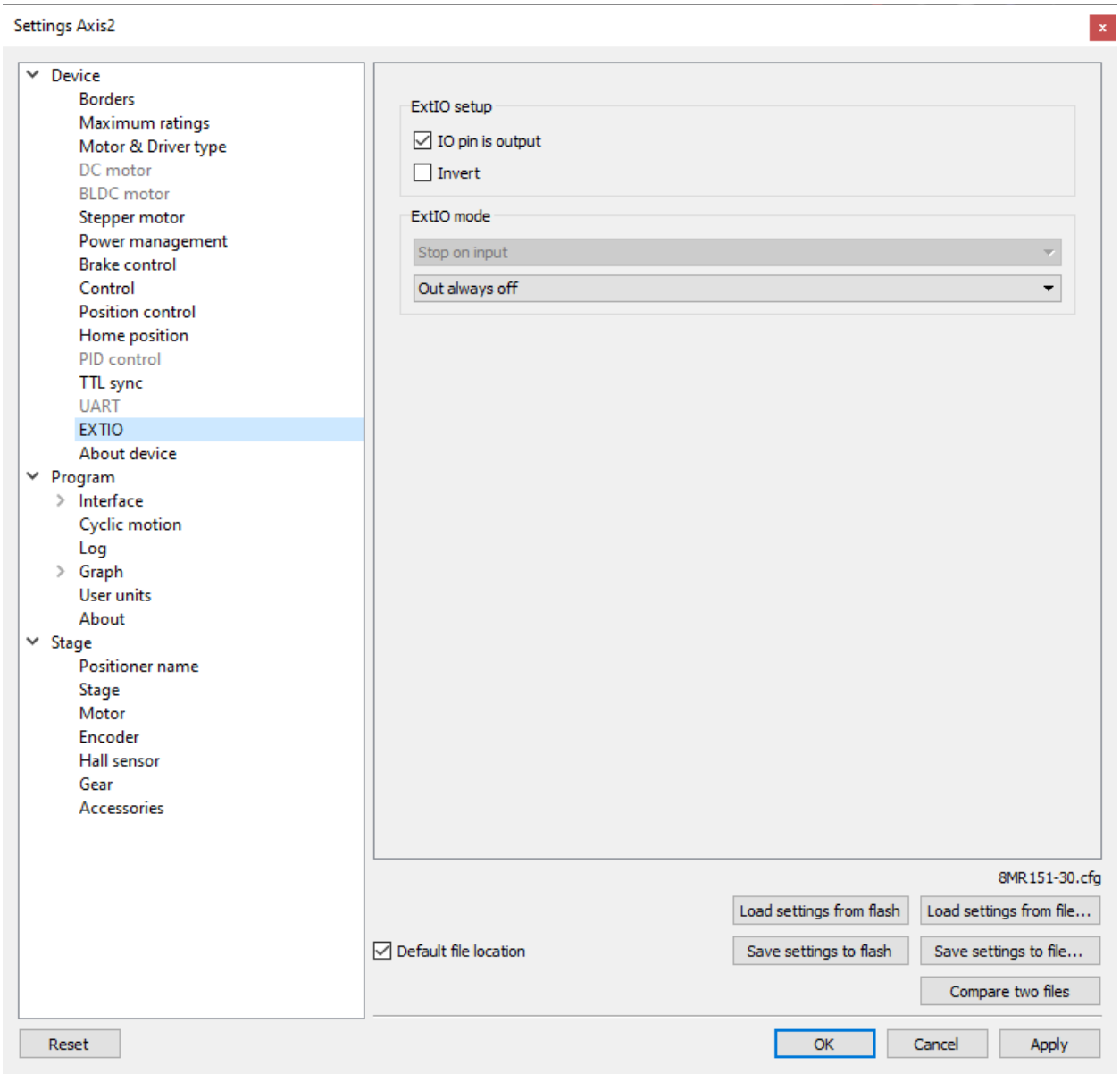


Рис. 5.32: Вкладка «Настройки цифрового входа-выхода общего назначения»

Подробное описание в разделе *Цифровой вход-выход общего назначения*.

ExtIO setup

IO pin is output - если флаг установлен, то ножка ExtIO работает в режиме выхода, иначе в режиме входа.

Invert - если флаг установлен, то переход в нулевой логический уровень приводит к выполнению действия.

ExtIO mode - выбор режима работы

Если ExtIO сконфигурирован в режиме входа, то активен выбор настроек действия контроллера по входному импульсу:

- *Do nothing* - ничего не делать.

- *Stop on input* - выполнить *Команда STOP*.
- *Power off on input* - выполнить *Команда PWOF*.
- *Movr on input* - выполнить *Команда MOVR*.
- *Home on input* - выполнить *Команда HOME*.
- *Alarm on input* - войти в состояние ALARM.

Если ExtIO сконфигурирован в режиме выхода, то активен выбор состояния выхода в зависимости от состояния контроллера:

- *Out always off* - всегда в неактивном состоянии.
- *Out always on* - всегда в активном состоянии.
- *Out active when moving* - в активном состоянии в процессе движения.
- *Out active in Alarm* - в активном состоянии, если контроллер в состоянии Alarm.
- *Out active when motor is on* - в активном состоянии, если запитаны обмотки мотора.
- *Out active when motor is found* - в активном состоянии, если подключен мотор.

5.3.11 Настройка типа двигателя

В окне *настроек программы* **Device -> Motor & Driver type**

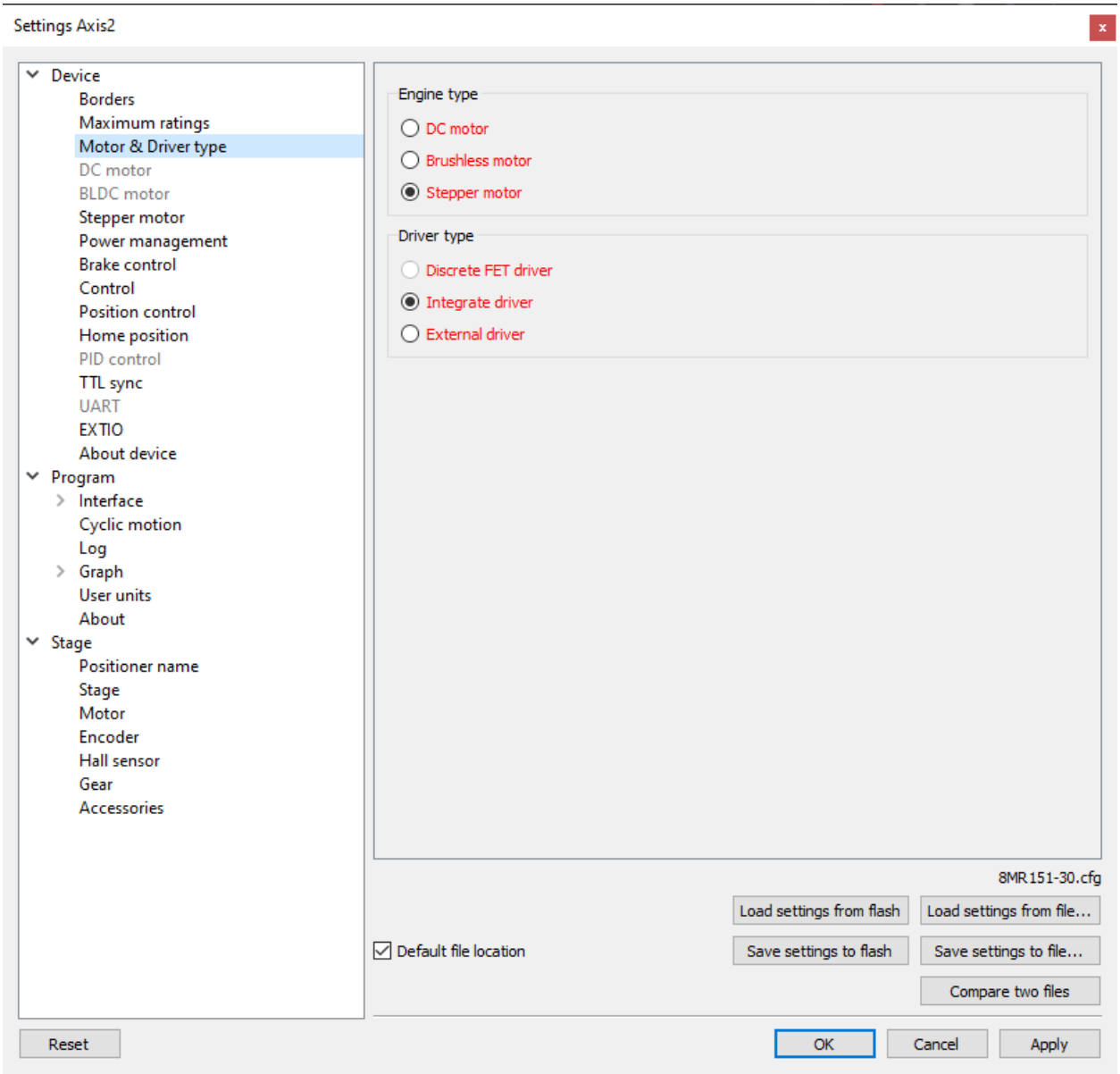


Рис. 5.33: Окно настроек типа двигателя

Индикация типа двигателя - шаговый *Stepper motor* либо двигатель постоянного тока *DC-motor*. Также выбирается силовой драйвер для управления:

- Integrated. Используется в данной модификации контроллера.
- Discrete FET driver. Будет использоваться в будущих версиях.
- *External driver*. Для управления шаговыми двигателями с помощью трёх стандартных сигналов (см. *Интерфейс управления внешним драйвером*)

Предупреждение: Смена типа драйвера или типа мотора это критическая операция, которую не следует выполнять в момент вращения двигателя. Для корректной смены нужно обесточить обмотки текущего двигателя, отключить его, изменить тип двигателя, подключить двигатель другого

типа. То же самое касается и смены типа драйвера с интегрированного на внешний и наоборот.

Примечание: Доступные типы двигателей определяются используемой прошивкой. Доступные драйвера управления определяются типом платы контроллера, за исключением внешнего драйвера.

5.3.12 Настройка кинематики движения (DC мотор)

В окне *настроек программы* **Device -> DC Motor**

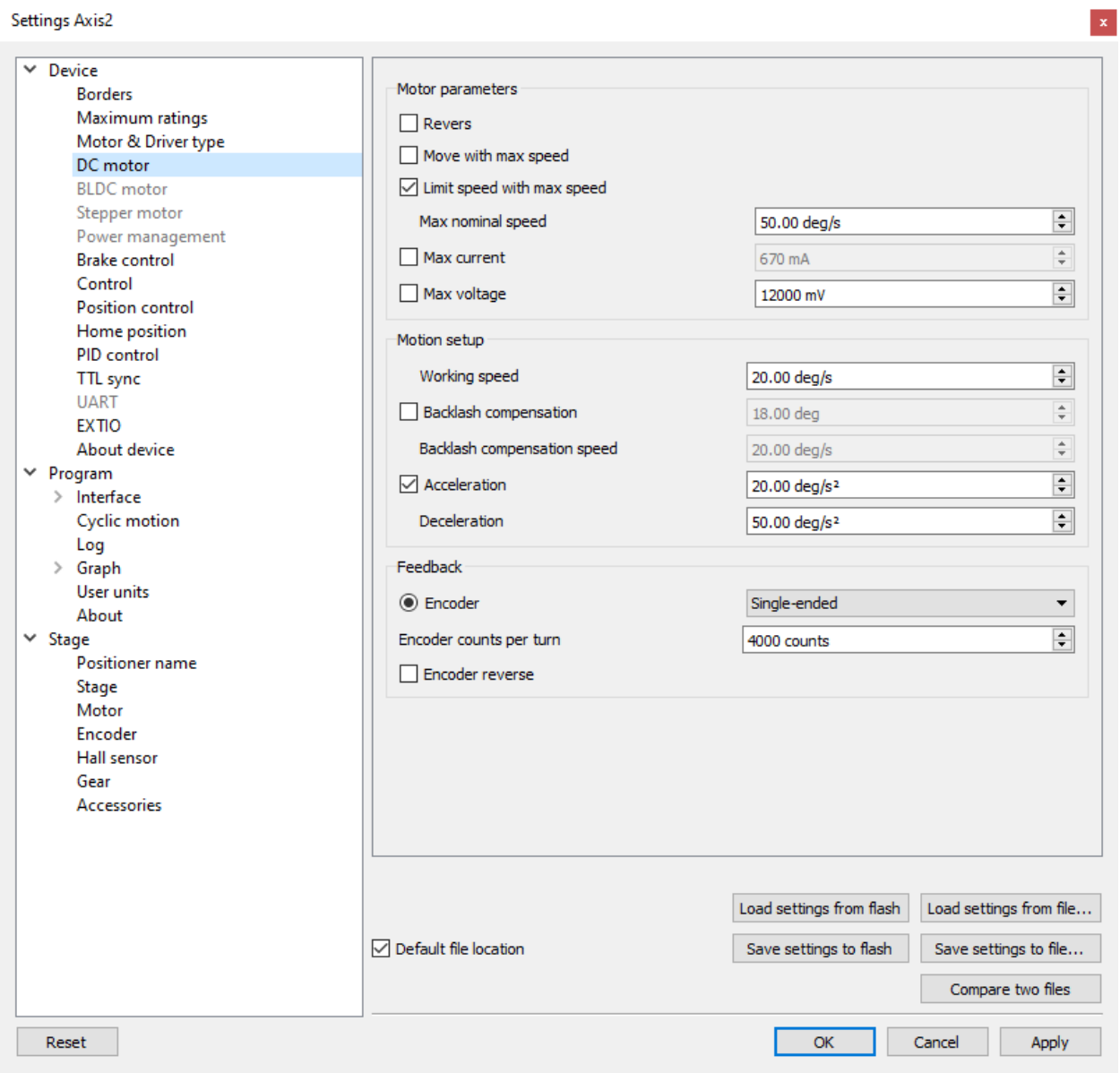


Рис. 5.34: Окно настроек кинематики движения

5.3.12.1 Motor parameters - настройки электромотора

Revers - установка этого флага позволяет связать направление вращения мотора с направлением счета текущей позиции. Измените состояние флага, если положительное вращение мотора уменьшает счетчик позиции. Действие этого флага равносильно подключению обмотки мотора в обратной полярности.

Move with max speed - при установленном флаге мотор игнорирует заданную скорость и вращается с максимальной допустимой скоростью.

Limit speed with max speed - при установленном флаге контроллер ограничивает максимальную скорость по количеству оборотов в секунду значением поля *Max nominal speed*.

Max nominal speed, Max voltage, Max current - номинальные параметры мотора. Если они активны и применимы для данного типа двигателя, то контроллер ограничивает эти параметры в заданных рамках. Например, если скорость и напряжение на моторе превысили номинальные, контроллер будет снижать выходное воздействие, пока оба значения не будут в пределах нормы. Однако при этом контроллер останется в рабочем состоянии, будет выполнять текущую задачу.

5.3.12.2 Motion setup - настройки кинематики движения

Working speed - скорость движения.

Backlash compensation - компенсация люфта. Так как механика позиционера не идеальна, существует различие при подходе к заданной точке справа и слева. При включенном режиме компенсации люфта позиционер будет подходить к точке всегда с одной стороны. Установленное значение определяет количество шагов, на которое позиционер будет проходить заданную точку чтобы возвращаться к ней с одной и той же стороны. Если указанное число больше нуля, позиционер будет подходить к точке всегда справа. Если меньше нуля, то всегда слева.

Backlash compensation speed - скорость компенсации люфта. При включенном режиме компенсации люфта *Backlash compensation* позиционер будет подходить к точке справа или слева с установленной скоростью, определяемой количеством шагов в секунду.

Acceleration - включает режим движения с ускорением, числовое значение поля - величина ускорения движения.

Deceleration - величина замедления движения.

5.3.12.3 Настройки обратной связи

Encoder - использование *энкодера* в качестве датчика обратной связи. При включении данной опции доступен выбор типа энкодера: недифференциальный, дифференциальный или автоматическое определение.

Encoder counts per turn - количество импульсов энкодера на один полный оборот оси мотора.

5.3.13 Настройка контуров PID-регулирования

В окне *настроек программы* **Device -> PID control**

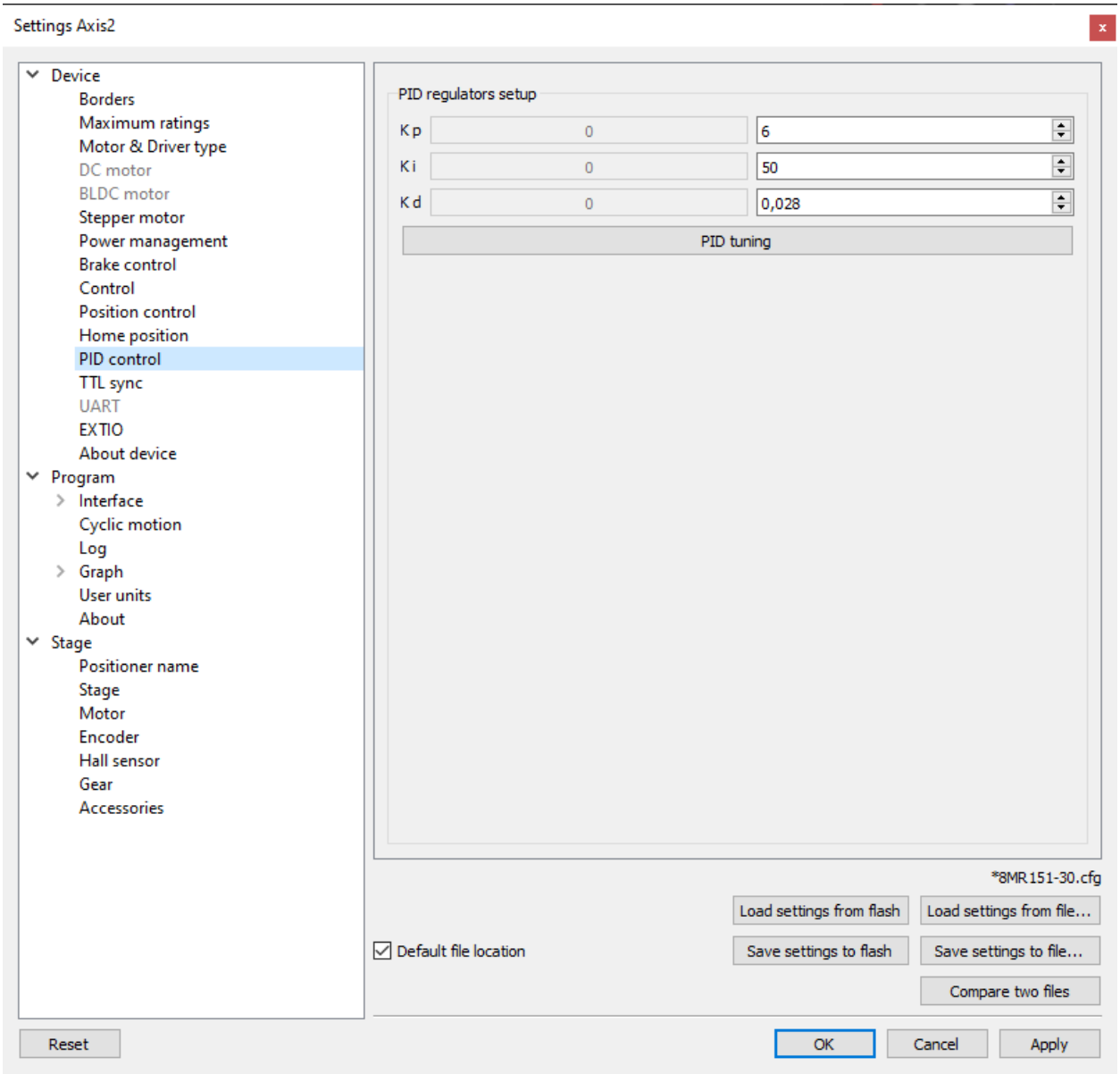


Рис. 5.35: Окно настройки контуров PID-регулирования

В этом разделе вы можете изменить коэффициенты PID-регуляторов. Используется регулятор по напряжению, 3 коэффициента K_p, K_i и K_d могут изменяться в диапазоне 0..65535.

Поля коэффициентов PID с дробной частью используются только для управления BLDC двигателем (поддержка начиная с прошивки 4.1.x). В зависимости от выбранного типа двигателя (DC или BLDC) соответствующие поля становятся активными.

Предупреждение: Не меняйте настройки PID-регуляторов, если вы не уверены, что знаете, что делаете!

Команды настройки описаны в разделе *Описание протокола обмена*. Настройка PID-регуляторов описаны в разделе *PID алгоритм для управления DC двигателем для DC* и PID алгоритм для управления

BLDC двигателем для BLDC.

5.3.14 О контроллере

В окне *настроек программы* **Device -> About device**

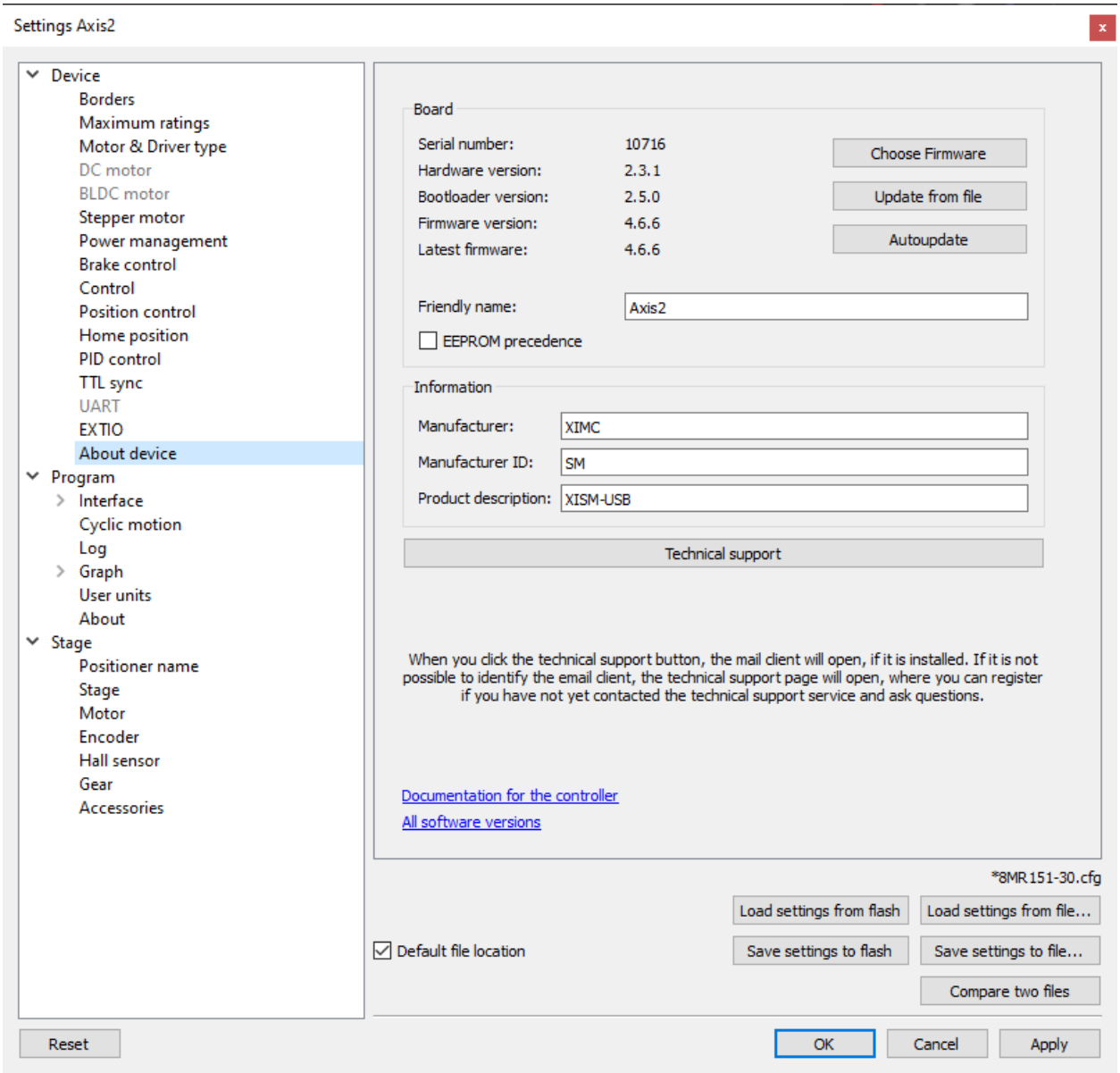


Рис. 5.36: Вкладка «Об устройстве»

В разделе **Board** отображается информация о контроллере:

- *Serial number* - серийный номер устройства.
- *Bootloader version* - версия загрузчика.
- *Firmware version* - версия прошивки.

- *Latest firmware* - последняя доступная версия прошивки для данного контроллера (считывается из интернет при наличии интернет-соединения).
- Кнопка *Update* открывает диалог обновления прошивки.

Кнопка Choose Firmware открывает диалог выбора версии прошивки на которую будет произведено обновление. Список версий и сами файлы прошивок скачиваются автоматически из интернет. Для работы этого типа обновления необходимо наличие активного интернет-подключения. В диалоговом окне выберите необходимую версию прошивки и нажмите «Update firmware».

Кнопка *Choose Firmware* открывает диалог выбора версии прошивки на которую будет произведено обновление. Список версий и сами файлы прошивок скачиваются автоматически из интернет. Для работы этого типа обновления необходимо наличие активного интернет-подключения. В диалоговом окне выберите необходимую версию прошивки и нажмите «Update firmware».

Кнопка *Autoupdate* запускает обновление прошивки из интернет на последнюю доступную версию.

Friendly name - позволяет установить произвольное имя контроллера. Если значение имени непусто, то эта строка будет выводиться в заголовках окон вместо идентификатора и серийного номера устройства. Это имя удобно использовать если к компьютеру подключено несколько контроллеров.

EEPROM precedence - этот флаг действует только для позиционеров с *системой опознавания*. При включенном флаге для контроллера становятся приоритетными настройки из внешней EEPROM-памяти позиционера и они применяются каждую перезагрузку контроллера или каждое подключение позиционера. В ином случае используются настройки из FRAM. Включение настройки окрашивает кнопку «Save settings to flash» в красный цвет - это предупреждает пользователя, что сохраненные во флеш-память контроллера настройки будут перезаписаны при подключении внешней памяти.

В разделе **Information** отображаются данные об устройстве: производитель, идентификатор устройства, тип устройства. Эти данные считываются из внутренней памяти контроллера.

Все эти данные сообщаются программе XILab при подключении устройства.

5.3.15 Настройка кинематики движения (BLDC мотор)

В окне *настроек программы* **Device -> BLDC Motor**

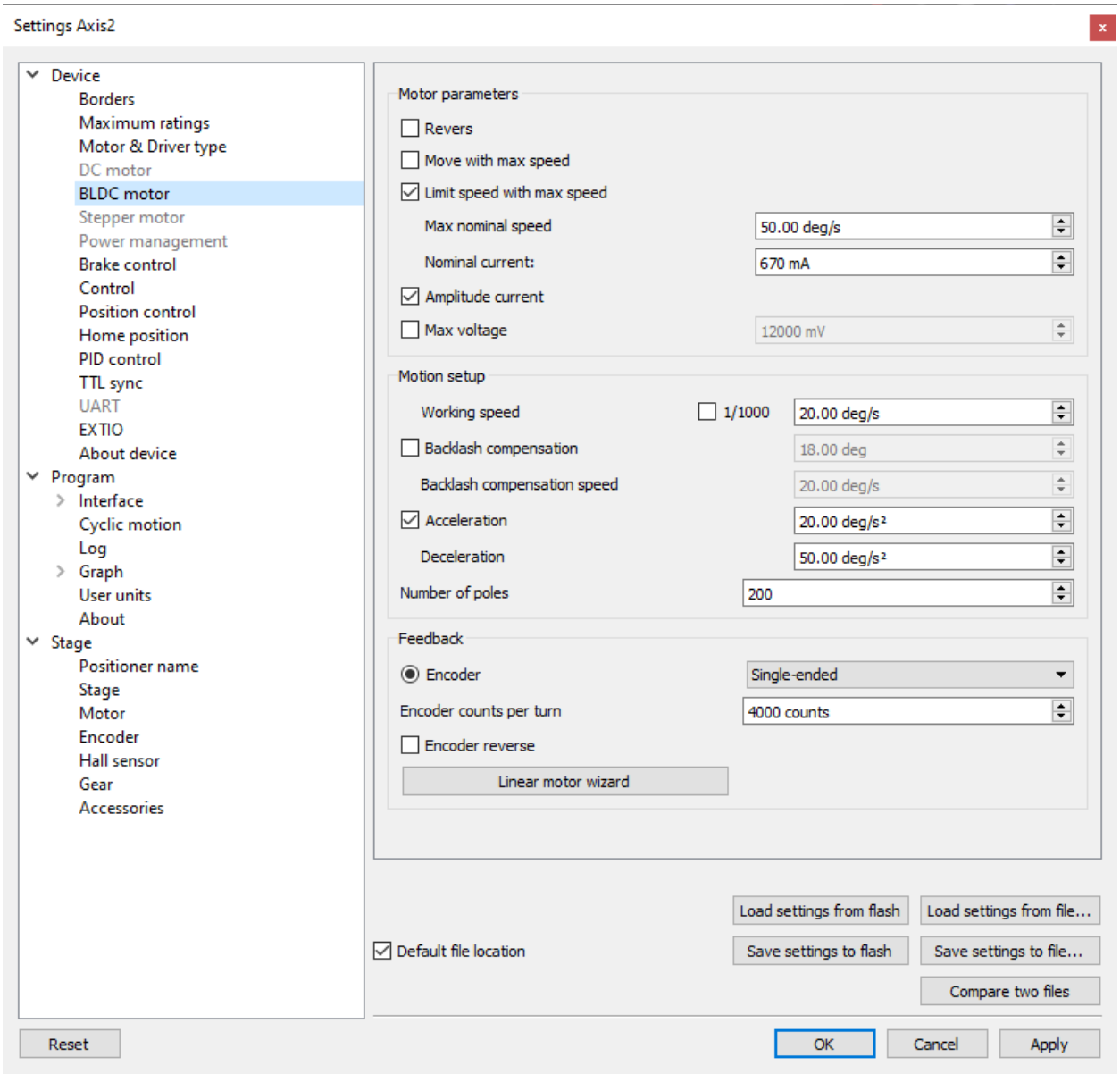


Рис. 5.37: Окно настроек кинематики движения

Важно: Только прошивки 4.1.x (и новее) поддерживают управление BLDC.

5.3.15.1 Motor parameters - настройки электромотора

Revers - установка этого флага позволяет связать направление вращения мотора с направлением счета текущей позиции. Измените состояние флага, если положительное вращение мотора уменьшает счетчик позиции. Действие этого флага равносильно подключению обмотки мотора в обратной полярности.

Move with max speed - при установленном флаге мотор игнорирует заданную скорость и вращается с максимальной допустимой скоростью.

Limit speed with max speed - при установленном флаге контроллер ограничивает максимальную скорость

по количеству оборотов в секунду значением поля *Max nominal speed*.

Max nominal speed, Max voltage - номинальные параметры мотора. Если они активны и применимы для данного двигателя, то контроллер ограничивает эти параметры в заданных рамках. Например, если скорость и напряжение на моторое превысили номинальные, контроллер будет снижать выходное воздействие, пока оба значения не будут в пределах нормы. Однако при этом контроллер останется в рабочем состоянии, будет выполнять текущую задачу.

Важно: «Max voltage» это максимальная разность потенциалов между двумя выводами BLDC мотора. В то же время, напряжение на каждом выводе мотора относительно земли контроллера (именно это напряжение отображается на графике «Windong A Voltage», «Winding B Voltage») может превышать «Max voltage».

Amplitude current - если этот флаг установлен, контроллер интерпретирует настройку тока как максимальный амплитудный ток. Если флаг снят, контроллер интерпретирует введённое значение тока как ток, посчитанный с учётом максимального тепловыделения. Более подробно о номинальном токе и тепловыделении BLDC см. *Расчёт номинального тока*

5.3.15.2 Motion setup - настройки кинематики движения

Working speed - скорость движения.

Флаг 1/1000 - позволяет работать на сверхнизких скоростях. Если флаг установлен, значение из поля «Working speed будет разделено на 1000.

Важно: Флаг для работы со сверхнизкими скоростями был добавлен в xilab 1.17.9 (и новее) и прошивку 4.5.9 (и новее)

Backlash compensation - компенсация люфта. Так как механика позиционера не идеальна, существует различие при подходе к заданной точке справа и слева. При включенном режиме компенсации люфта позиционер будет подходить к точке всегда с одной стороны. Установленное значение определяет количество шагов, на которое позиционер будет проходить заданную точку чтобы возвращаться к ней с одной и той же стороны. Если указанное число больше нуля, позиционер будет подходить к точке всегда справа. Если меньше нуля, то всегда слева.

Backlash compensation speed - скорость компенсации люфта. При включенном режиме компенсации люфта Backlash compensation позиционер будет подходить к точке справа или слева с установленной скоростью, определяемой количеством шагов в секунду.

Acceleration - включает режим движения с ускорением, числовое значение поля - величина ускорения движения.

Deceleration - величина замедления движения.

Number of poles - количество полюсов за оборот.

5.3.15.3 Feedback - настройки обратной связи

Encoder - использование *энкодера* в качестве датчика обратной связи. При включении данной опции доступен выбор типа энкодера: недифференциальный, дифференциальный или автоматическое определение.

Encoder counts per turn - количество импульсов энкодера на один полный оборот оси мотора.

Linear motor wizard - открыть диалог для настройки линейного позиционера.

5.4 Настройки программы XILab

5.4.1 Общие настройки программы XILab

В *окне настроек* программы **Program**

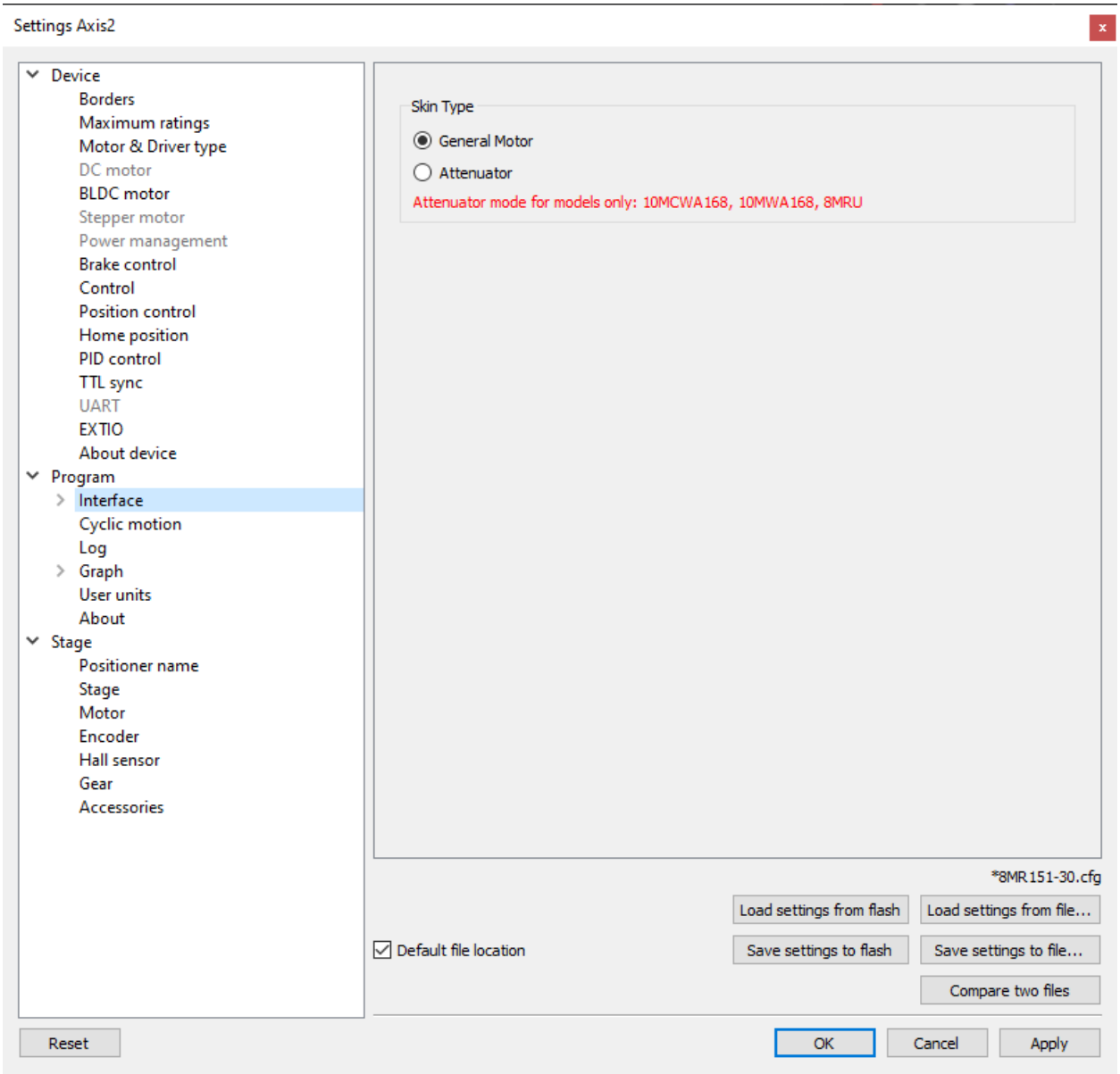


Рис. 5.38: Вкладка общих настроек программы

Данная вкладка отвечает за выбор типа интерфейса Xilab.

Группа **Skin Type** содержит настройку типа интерфейса Xilab. Доступны два варианта интерфейса: «General Motor» и «Attenuator».

General Motor - включает интерфейс мотора с абстрактным позиционером. В этом режиме в главном окне текущая позиция отображается численно и графически слайдером, также доступны элементы

управления движением в произвольную координату. Настройки отображения интерфейса в этом режиме находятся на вкладке *Настройки интерфейса абстрактного позиционера*.

Включает интерфейс аттенюатора. В этом режиме в главном окне текущая позиция отображается графически, также доступны элементы управления движением в позицию с определенной прозрачностью фильтров аттенюатора. Настройки отображения интерфейса в этом режиме находятся на вкладке *Настройки интерфейса аттенюатора*.

5.4.2 Настройки интерфейса абстрактного позиционера

В окне *настроек* программы **Program -> Interface -> General motor**

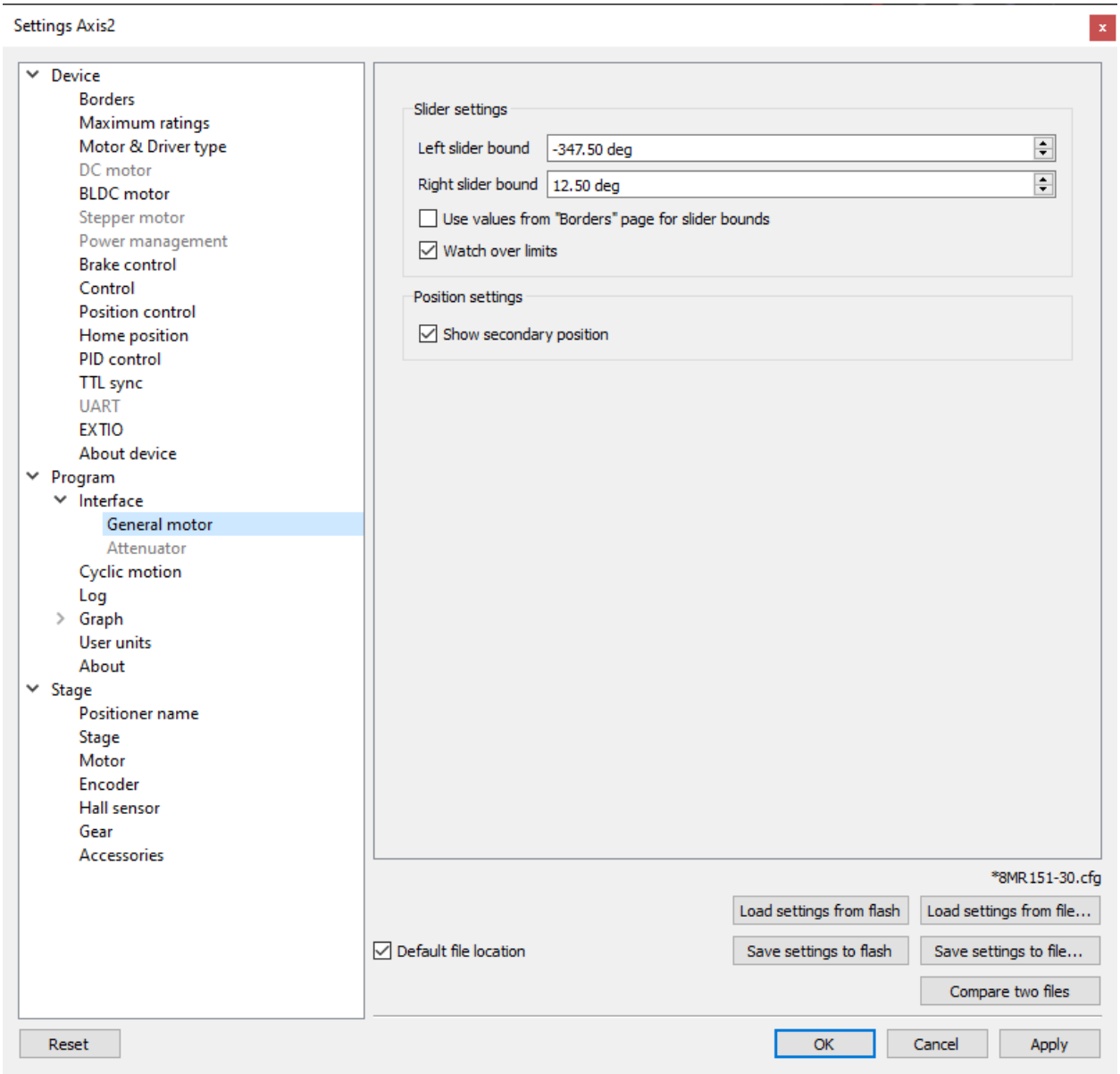


Рис. 5.39: Окно вкладки «Настройки абстрактного позиционера»

В этой вкладке настраиваются параметры отображения слайдера и настройки отображения вторичного

положения для общего моторного устройства. Находится в *главном окне* и визуально представляет текущую позицию относительно границ.



Рис. 5.40: Фрагмент основного окна программы со слайдером

Группа **Slider settings** содержит настройки слайдера:

Left slider bound и *Right slider bound* содержат левую и правую границу слайдера соответственно.

Флажок *Watch over limits* настраивает такое поведение границ слайдера при котором при выходе текущей позиции за диапазон слайдера, шкала начинает смещаться, чтобы отобразить текущую позицию. Общая отображаемая на слайдере дистанция при этом остается неизменной. Не используется по умолчанию. Данный флажок удобен когда вы знаете диапазон перемещения позиционера, но не знаете привязку этого положения к значениям, отображаемым в XILab, например до проведения калибровки. Часто используется совместно с настройками из вкладки *Настройка исходного положения*.

Группа **Position settings** содержит настройки отображения второстепенной позиции.

Флажок *Show secondary position* включает отображение второстепенной позиции по энкодеру в главном окне программы.

5.4.3 Настройки интерфейса аттенюатора

В *окне настроек программы* **Program -> Interface -> Attenuator**

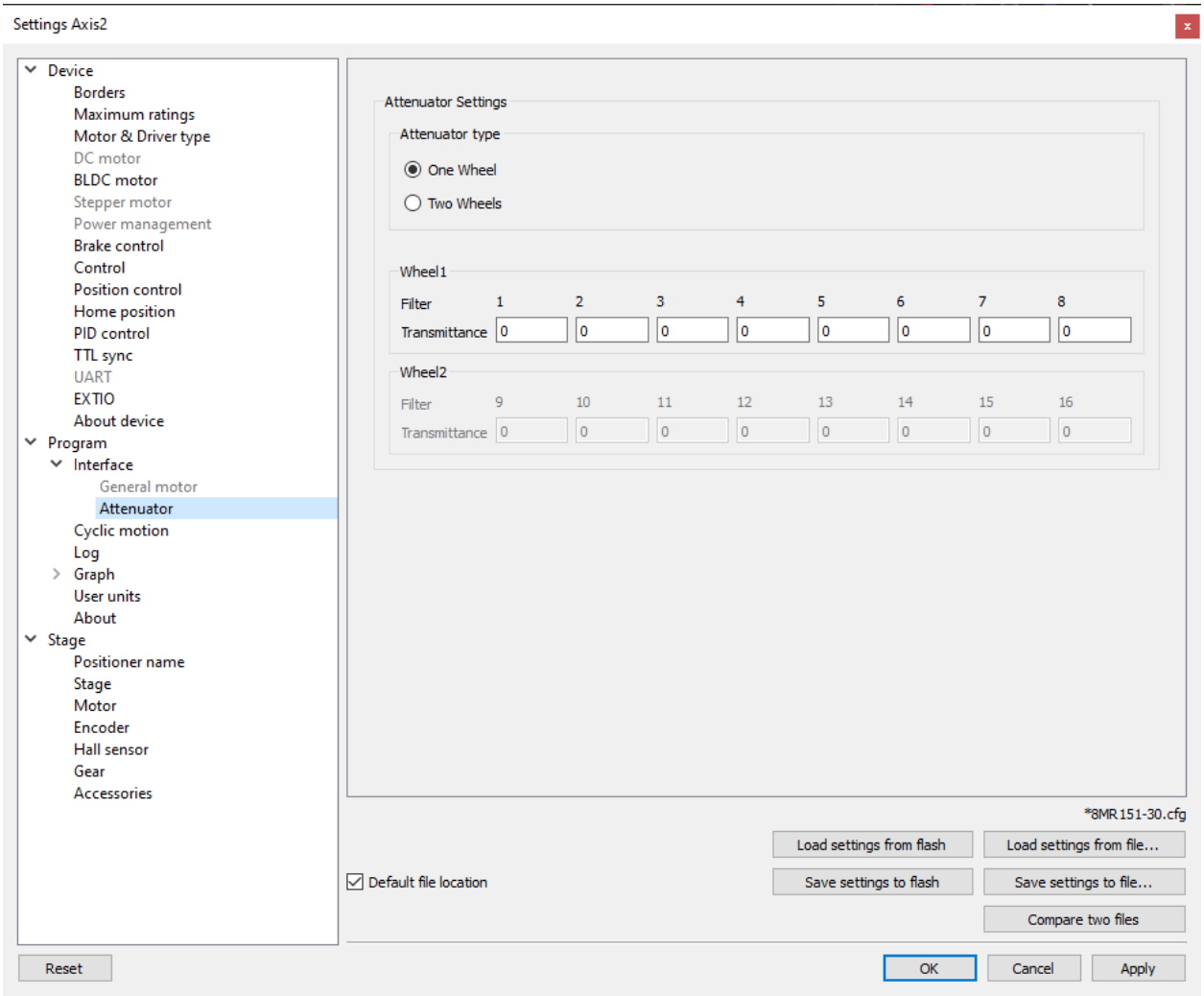


Рис. 5.41: Окно вкладки «Настройки аттенюатора»

Данная вкладка предназначена для настройки интерфейса Xilab при работе с аттенюатором – устройством с несколькими светофильтрами на вращающихся дисках, предназначенным для управления коэффициентом пропускания светового пучка. Вкладка становится активной при выборе опции «Attenuator» в блоке «Skin Type» в окне вкладки «Program».

Группа **Attenuator type** содержит радиокнопки выбора аттенюатора с одним или двумя дисками. В зависимости от выбора становиится доступной один или два группы **Wheel**, в которых необходимо задать коэффициенты пропускания фильтров, расположенных на данном диске, в порядке следования.

5.4.4 Настройка режима циклического движения

В *окне настроек* программы **Program -> Cyclic motion**

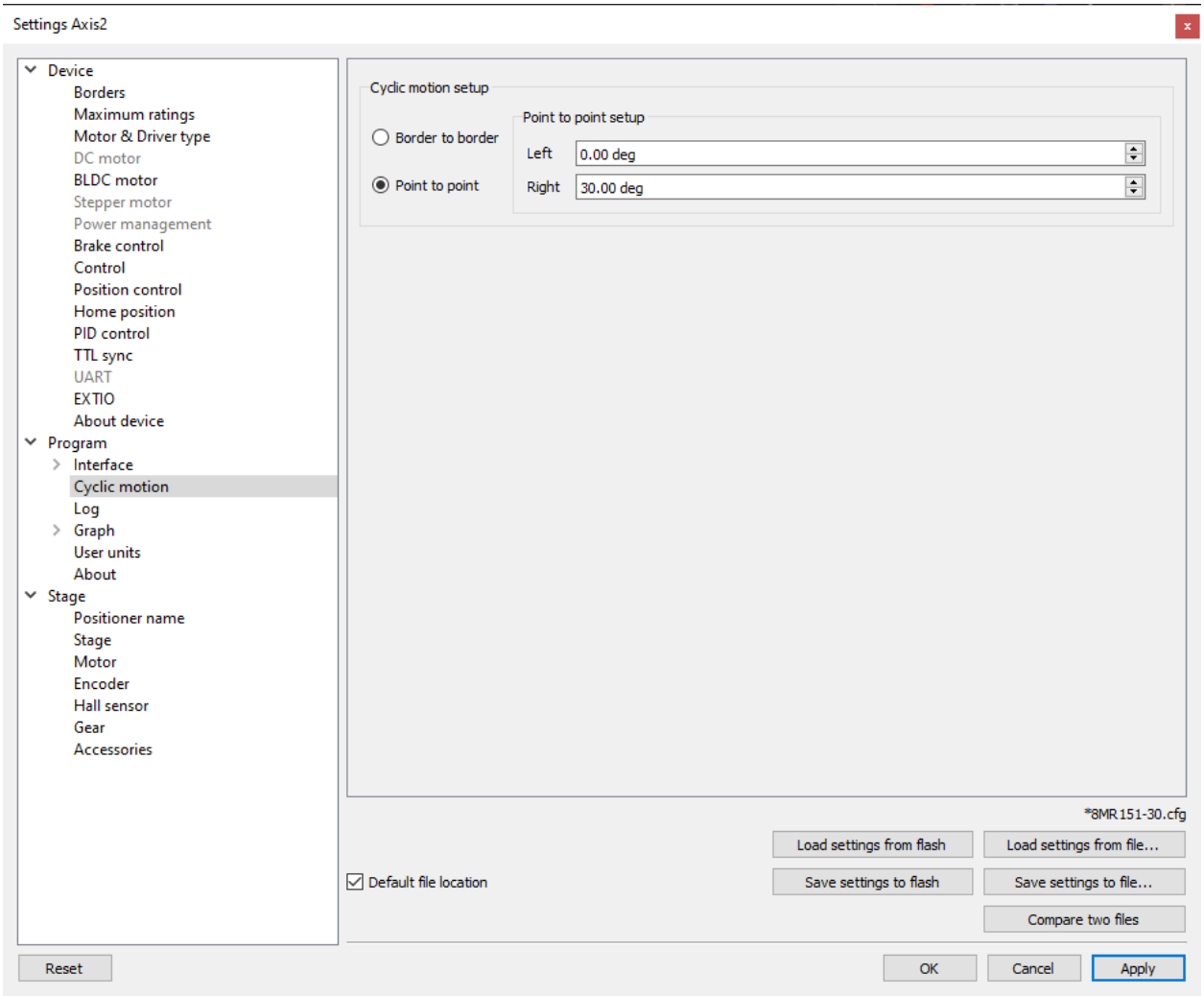


Рис. 5.42: Окно вкладки «Режим циклического движения»

Данная вкладка предназначена для настройки циклического перемещения между двумя заданными положениями. Используется главным образом в демонстрационных целях. Режим включается кнопкой *Cyclic* в *главном окне*, выключается кнопкой *Stop* в *главном окне*.

Настройки режима циклического движения:

Border to border - циклическое перемещение между границами настроенными во вкладке *Настройка диапазона движения и концевых выключателей*. Движение начинается к левой границе.

Point to point - циклическое перемещение между заданными в группе *Point to point setup* точками. Позиционер перемещается в левую точку, останавливается в ней, после этого перемещается в правую точку, останавливается, далее цикл повторяется.

5.4.5 Настройка логирования

В окне *настроек* программы **Program -> Log**

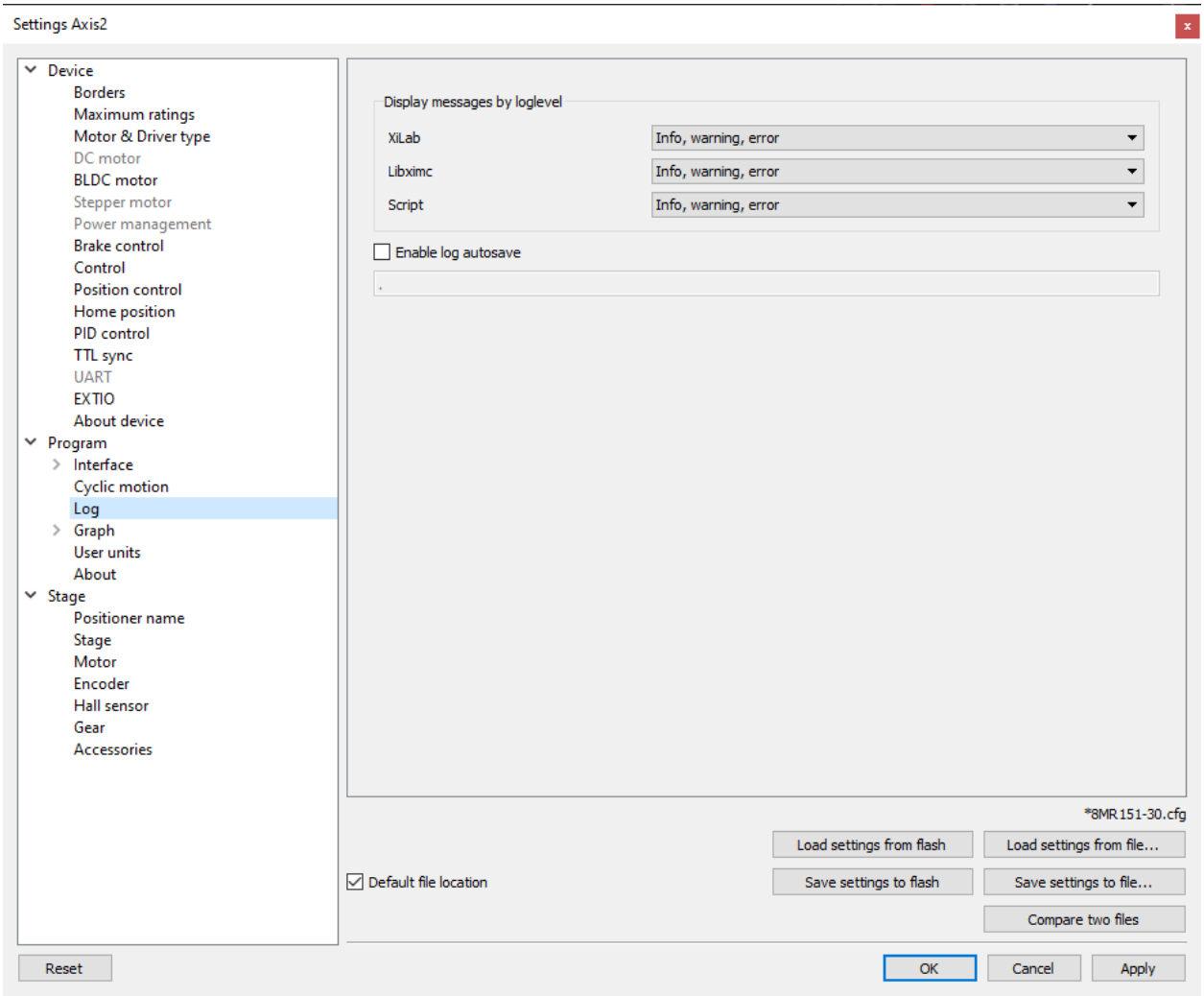


Рис. 5.43: Окно настроек лога программы XILab

Данная вкладка предназначена для настройки уровня подробности логирования.

В блоке *Display messages by loglevel* возможные опции это не выводить ничего (None), выводить только ошибки (Error), ошибки и предупреждения (Error, Warning), ошибки, предупреждения и информационные сообщения (Error, Warning, Info) для каждого из источников: программа XILab, библиотека libximc и модуль скриптов Scripts.

При включении опции *Enable log autosave* включается автосохранение лога в файл, путь к директории хранения файла задается в строке ниже. Запись в файл производится каждые 5 секунд.

Имя файла: «xilab_log_ГГГГ.ММ.ДД.csv», где ГГГГ, ММ и ДД - текущие год, месяц и день соответственно. Формат сохраненных данных: CSV. В файл записываются все сообщения лога независимо от уровня логирования.

5.4.6 Общие настройки отображения графиков

В *окне настроек* программы **Program -> Graph**

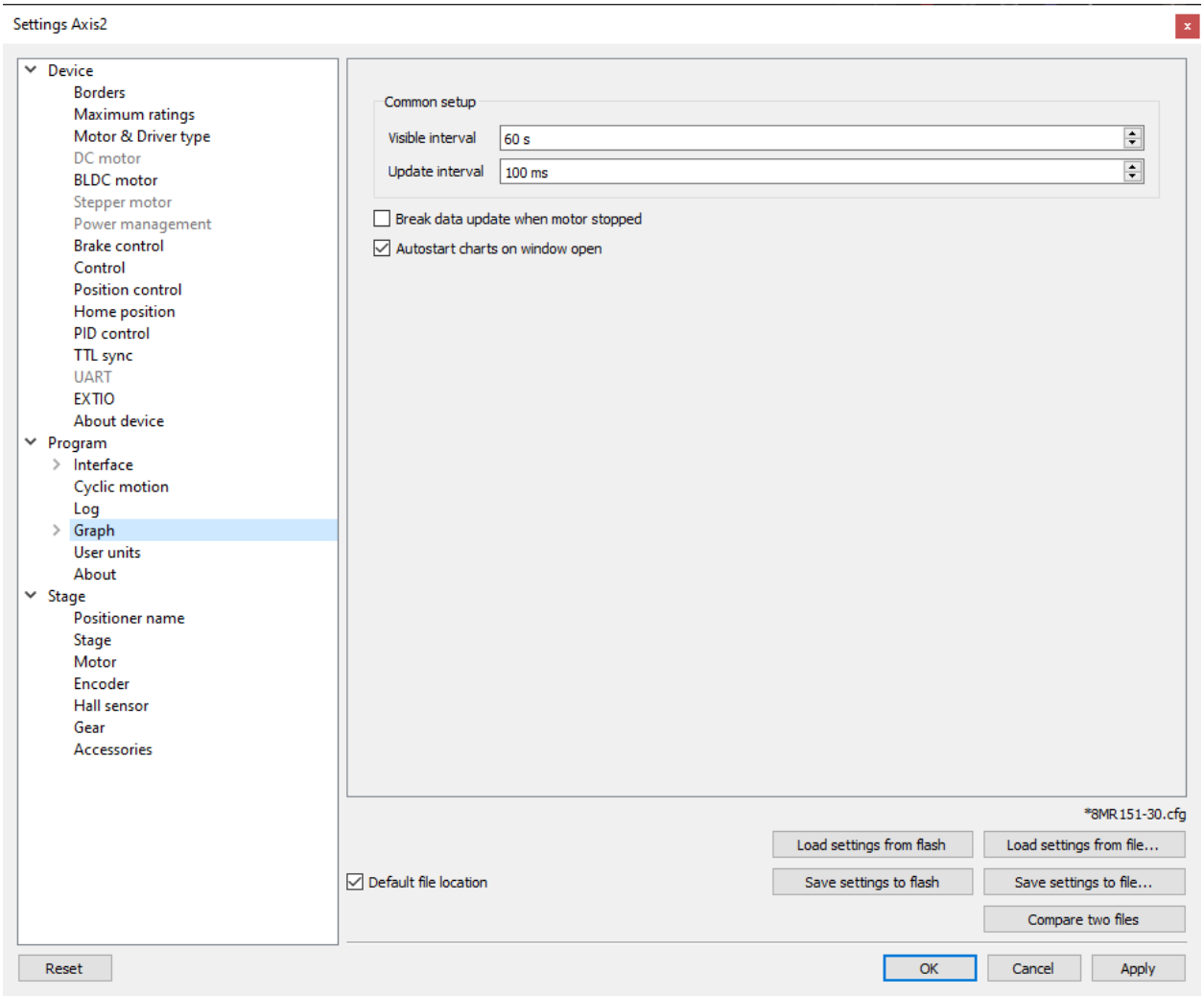


Рис. 5.44: Вкладка общих настроек отображения графиков

Visible interval - интервал времени, отображаемый на графиках по горизонтальной оси.

Update interval - период обновления данных графиков.

Break data update when motor stopped - остановка отрисовки графиков, когда мотор остановлен. Данный флаг позволяет более рационально использовать площадь графика, удаляя области, когда движение мотора отсутствует.

Autostart charts on window open - автоматическое начало отображения данных на графиках при открытии окна. Если вы хотите включать графики вручную, отключите эту опцию.

5.4.7 Индивидуальные настройки отображения графиков

В *окне настроек* программы **Program -> Graph -> ...**

Данный раздел в равной мере относится к индивидуальным настройкам отображения графиков скорости, напряжения, тока, скважности ШИМ, температуры и прочих параметров, отображение которых возможно на графиках в программе XILab.

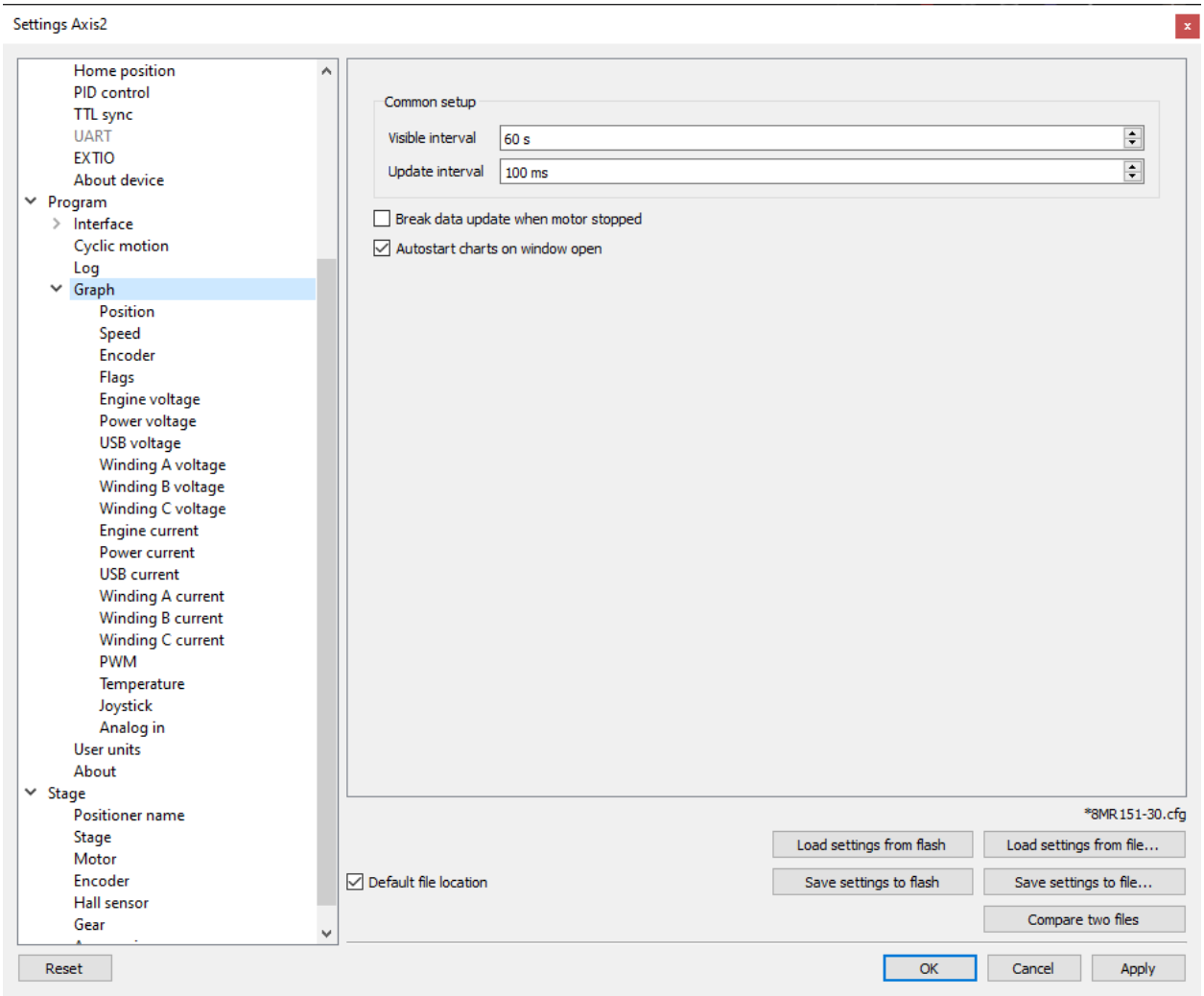


Рис. 5.45: Вкладка индивидуальных настроек на примере графика отображения положения

Настройка отображения графика включает в себя стиль линии и настройки масштабирования графика по вертикальной оси.

Группа параметров **Position curve setup** позволяет менять параметры кривой. Она включают в себя толщину *Line width*, цвет *Color* и тип линии *Line style*.

Группа параметров **Scaling** позволяет менять диапазон отображения кривой по вертикальной оси установкой значений *Scale min* и *Scale max*.

При выбранном флаге *Autoscale* производится автомасштабирование пределов шкалы в соответствии с пределами изменения переменной по оси Y. В этом случае параметры *Scale min* и *Scale max* игнорируются.

Флаг *Antialiasing* включает сглаживание линий графика, позволяющее добиться более качественного отображения, но несколько замедляющее процесс рисования графика.

5.4.8 Настройки отображения пользовательских единиц

В *окне настроек* программы **Program -> User units**

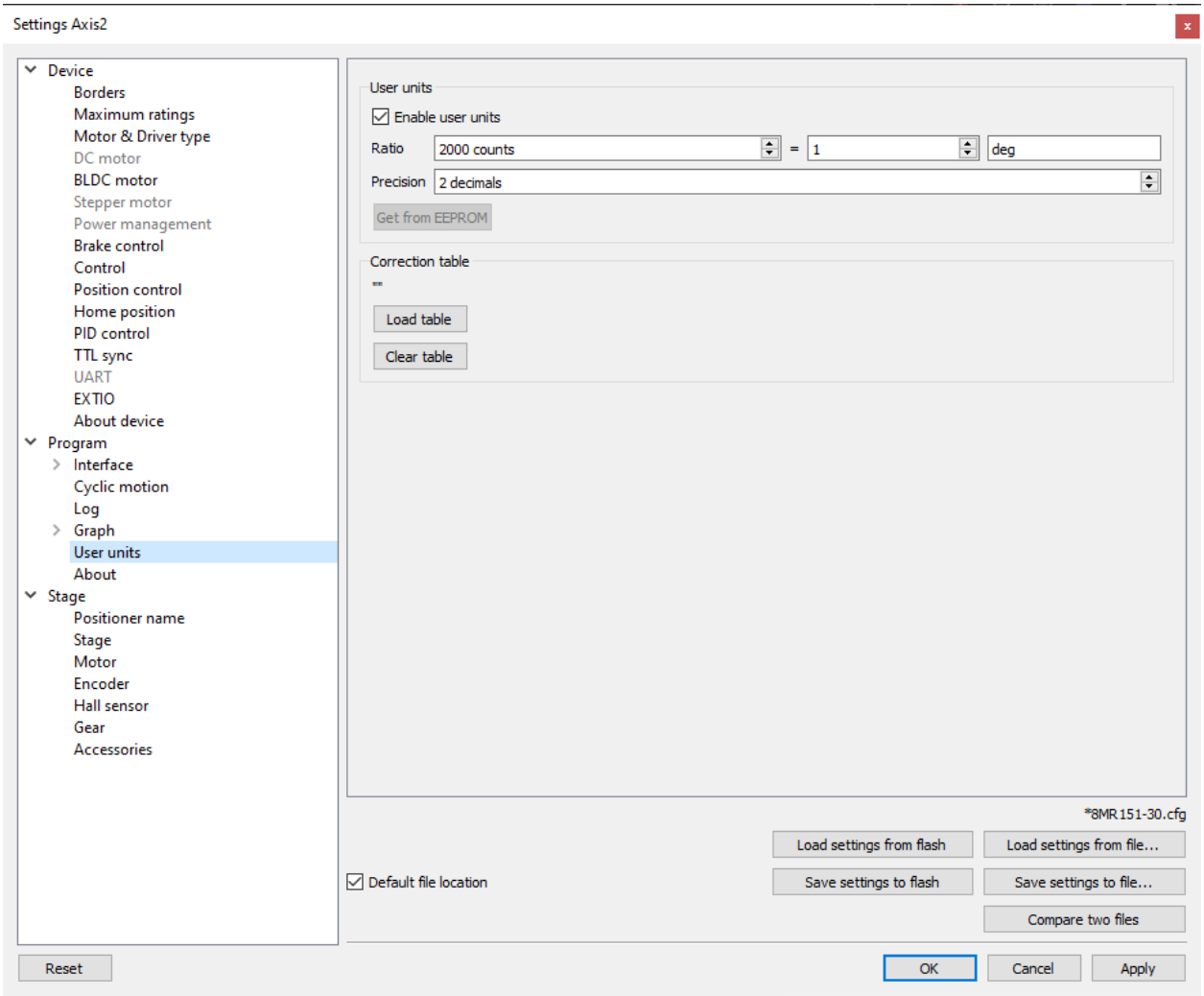


Рис. 5.46: Окно вкладки «Настройки отображения пользовательских единиц»

Данная вкладка предназначена для настройки отображения пользовательских единиц. Используется для удобства задания и считывания координат в привычных пользователю единицах. Эта вкладка также позволяет использовать *таблицу коррекции координат* для пользовательских единиц. Таблица коррекции координат позволяет значительно повысить точность позиционирования при использовании пользовательских единиц измерения.

5.4.8.1 Пользовательские единицы

Enable user units - включает отображение пользовательских единиц вместо шагов/микрошагов (в случае шагового двигателя) или отсчетов энкодера (в случае DC). Пользовательские единицы заменяют собой шаги(отсчеты) только для отображения в главном окне программы и не влияют на настройки вкладок Settings.

Ratio - коэффициент пересчета из шагов или отсчетов в пользовательские единицы, задается как отношение «x шагов = y единиц». Величины x, y, а также отображаемое название единицы вводятся пользователем.

Precision - точность отображения, количество знаков после запятой.

Кнопка *Get from EEPROM* считывает значения пользовательских единиц из памяти EEPROM.

5.4.8.2 Таблица коррекции координат для более точного позиционирования

Некоторые функции для работы с пользовательскими единицами позволяют использовать таблицу коррекции координат для более точного позиционирования.

Load table - загружает *таблицу коррекции координат*. В случае ее успешной загрузки справа от кнопки появится имя файла загруженной таблицы. С этого момента определенные _calb функции, которые выполняют пересчет координат с использованием корректировочной таблицы, будут производить пересчет, вплоть до очистки таблицы с помощью *Clear table*.

Clear table - очищает корректирующую таблицу. Все _calb функции работают в обычном режиме.

5.4.9 О программе

В *окне настроек* программы **Program -> About**

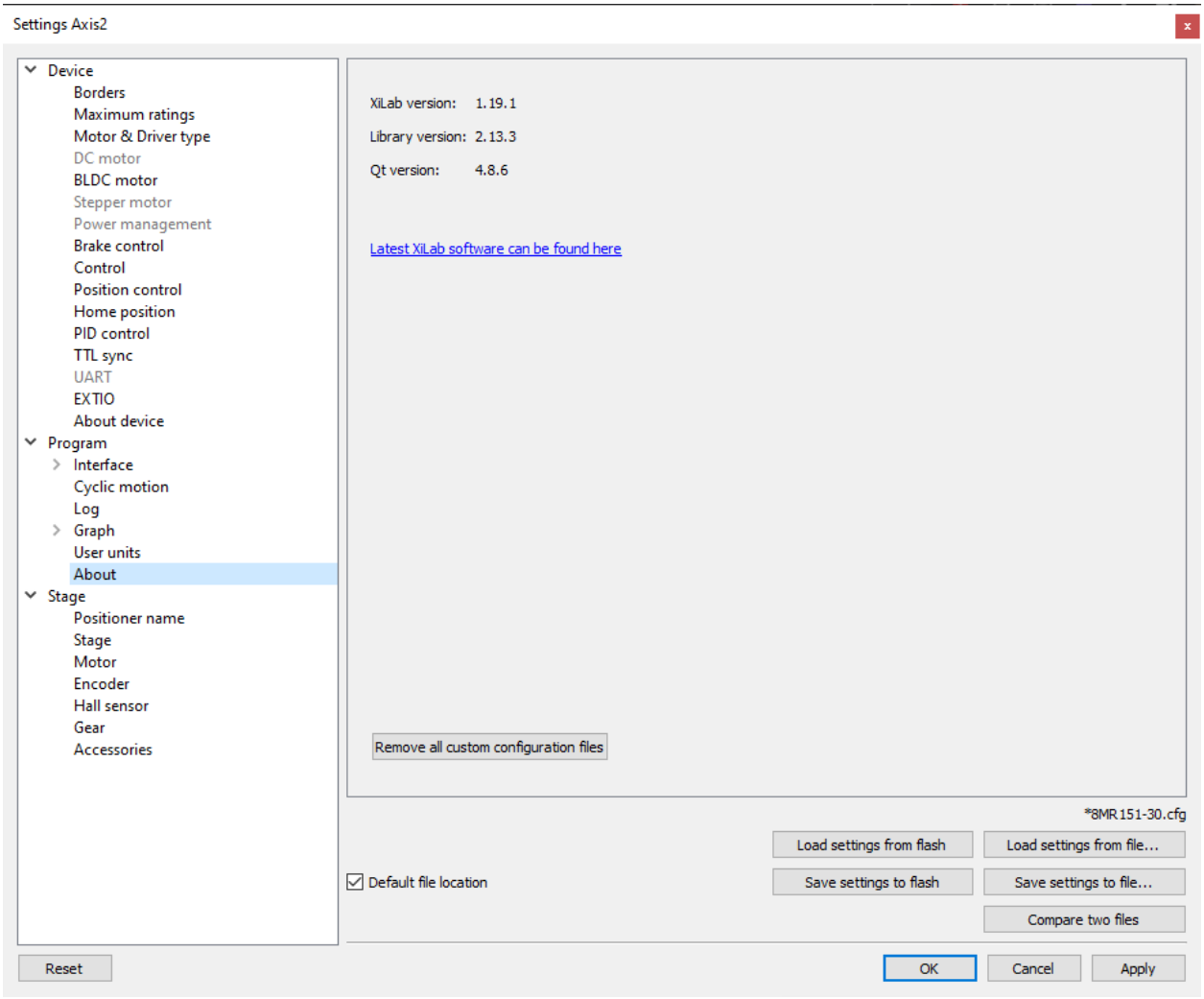


Рис. 5.47: Вкладка «О программе»

В этом разделе отображается версия программы XILab. Также дана ссылка на страницу с последней версией программного обеспечения.

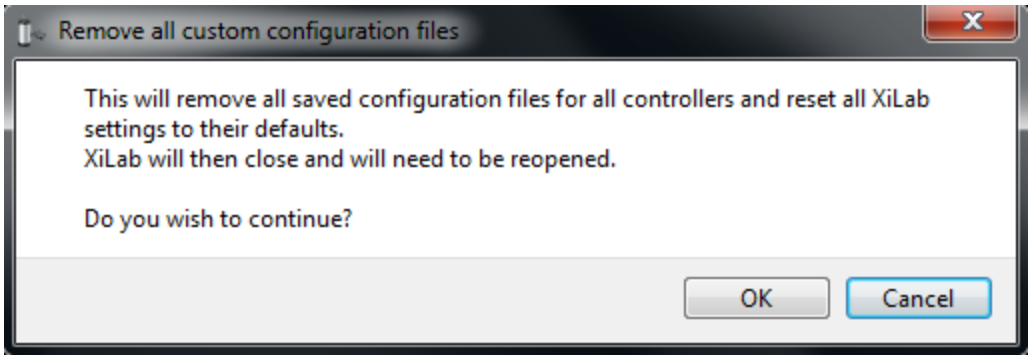


Рис. 5.48: Диалог очистки пользовательских конфигураций

Кнопка «Remove all custom configuration files» отображает диалог с возможностью удаления всех конфигурационных файлов, созданных после инсталляции как результат запусков XiLab. Файлы, которые могут быть удалены, находятся в директории настроек XiLab: файл «settings.ini», хранящий общие настройки программы, файлы «SMnnn.cfg», хранящие индивидуальные настройки контроллеров, файлы «V_nnn», хранящие внутренние состояния виртуальных контроллеров, файл «scratch.txt», хранящий последний запущенный на выполнение скрипт (см. окно *Скрипты*). «nnn» здесь означает любое число. Нажатие OK в диалоге «Remove all custom configuration files» выполнит удаление файлов и закроет XiLab, нажатие Cancel отменит удаление и закроет диалог.

5.5 Характеристики позиционера

5.5.1 Название позиционера

В окне *Настройки программы* **Stage -> Motor**

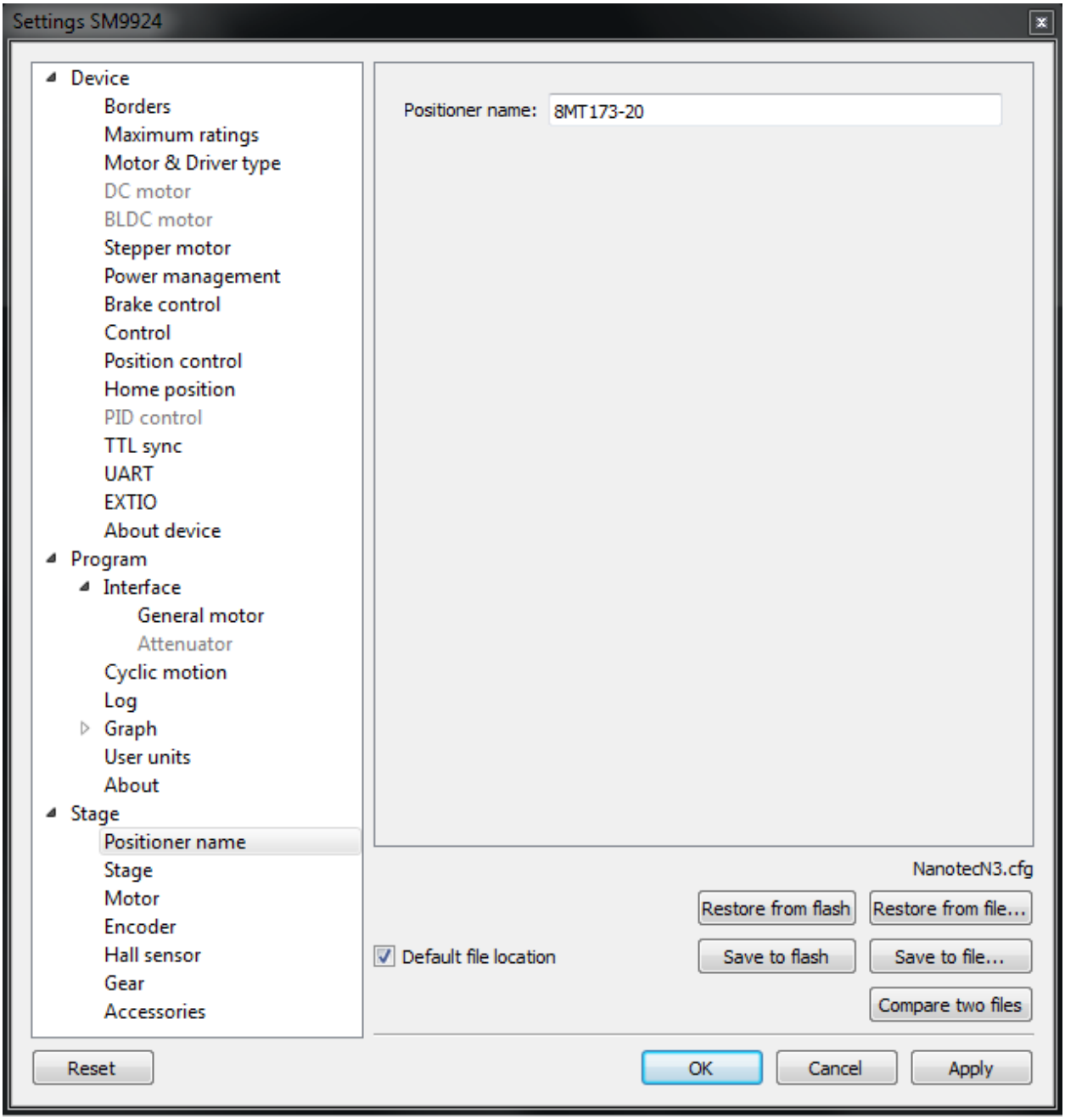


Рис. 5.49: Окно с названием позиционера

Раздел содержит информацию с названием позиционера (задаётся пользователем).

5.5.2 Общие характеристики позиционера

Важно: Информация на вкладке *«Stage -> Stage»* временно не используется

В окне *Настройки программы* **Stage -> Motor**

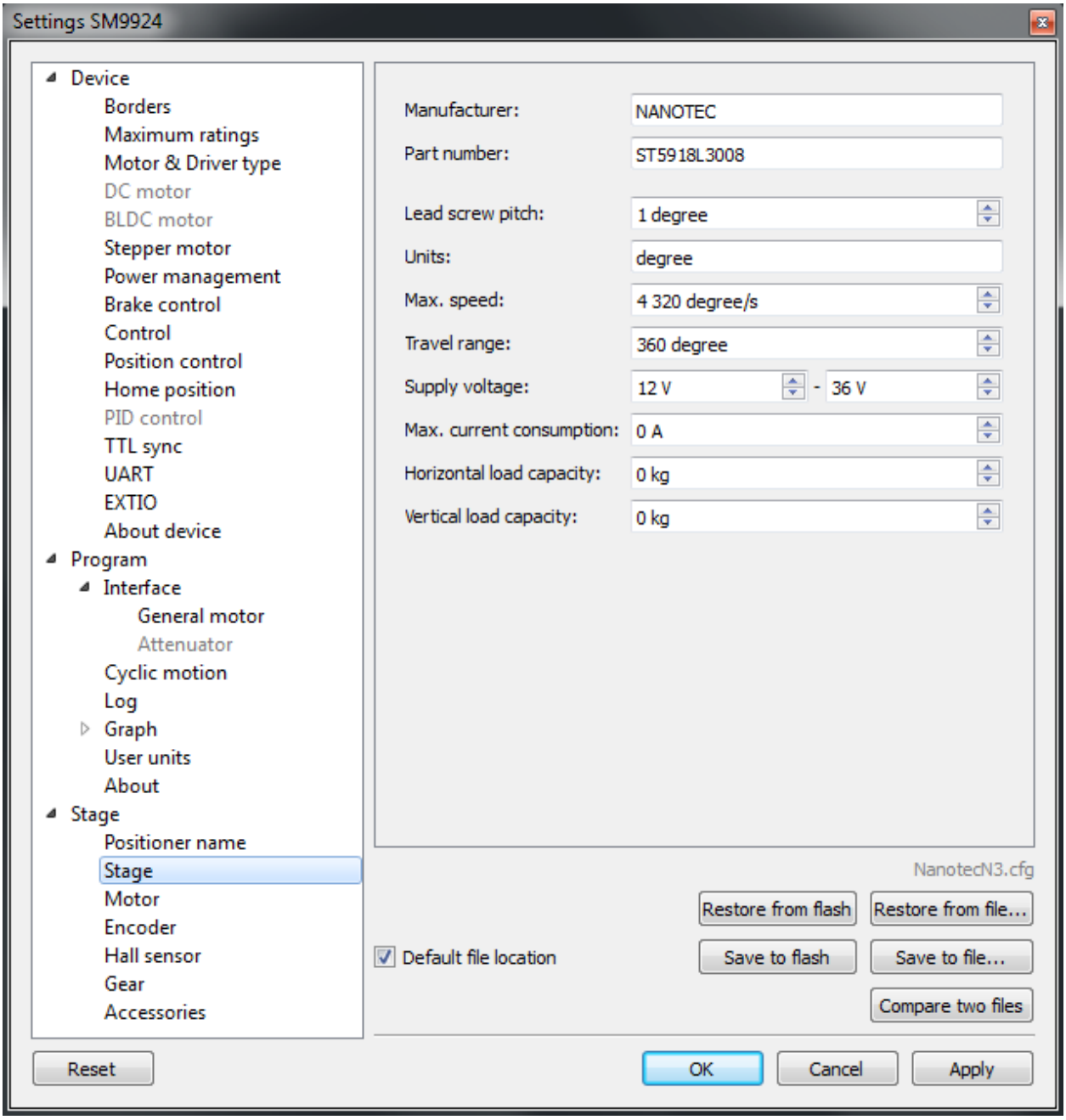


Рис. 5.50: Окно с общими характеристиками позиционера

Группа параметров **Stage** содержит информацию о позиционере:

- *Manufacturer* - наименование производителя.
- *Part number* - номер по каталогу.
- *Lead screw pitch* - шаг резьбы ходового винта.
- *Units* - единицы измерения перемещения в данном позиционере (мм, градусы, шаги).
- *Max. speed* - максимальная скорость.

- *Travel range* - диапазон перемещения.
- *Supply voltage* - диапазон допустимых напряжений питания.
- *Max. current consumption* - максимальное потребление тока.
- *Horizontal load capacity* - максимальная горизонтальная нагрузка на позиционер.
- *Vertical load capacity* - максимальная вертикальная нагрузка на позиционер.

5.5.3 Характеристики двигателя

Важно: Информация на вкладке *«Stage -> Motor»* временно не используется

В окне *Настройки программы* **Stage -> Motor**

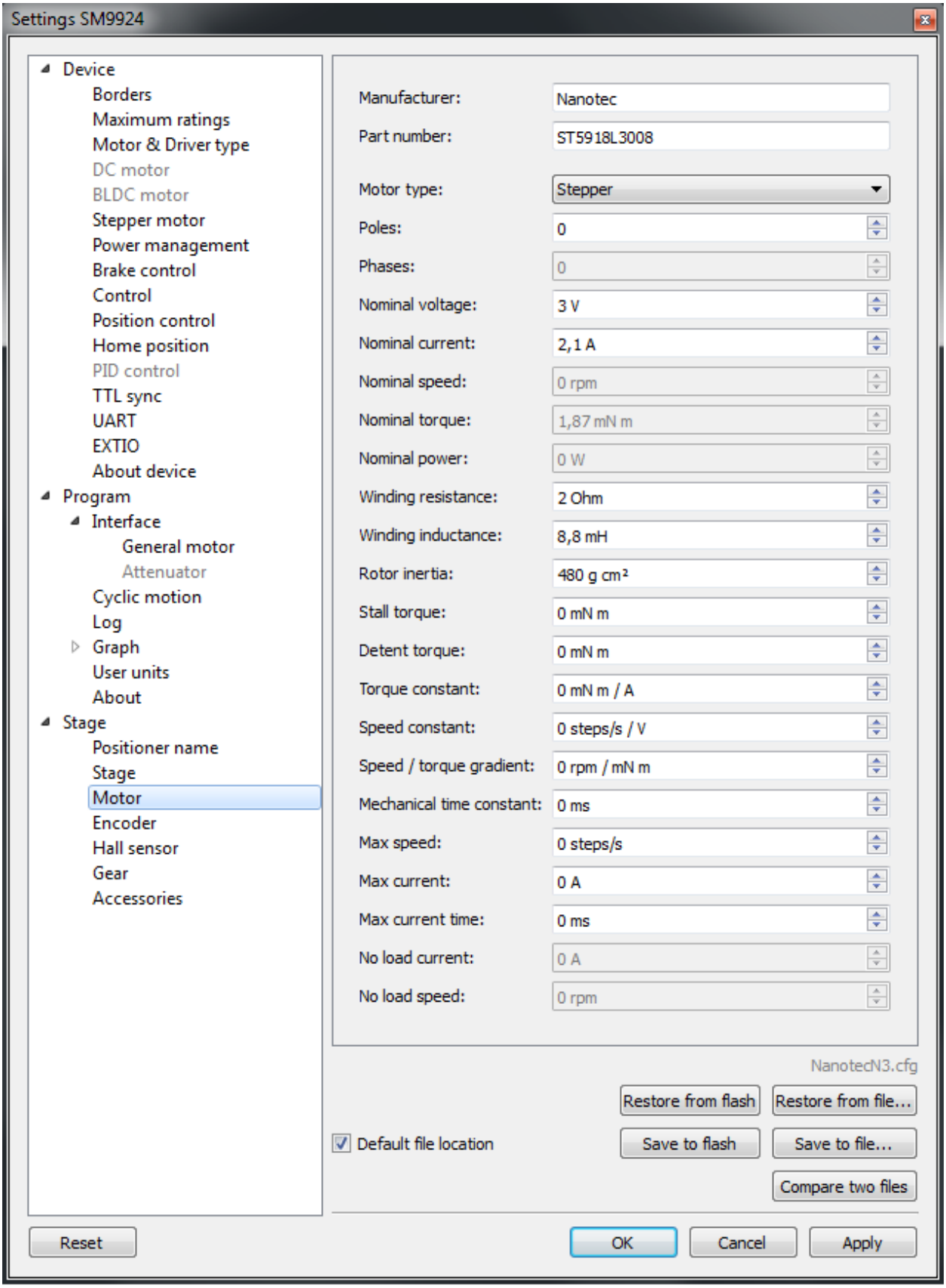


Рис. 5.51: Окно с описанием характеристик двигателя

Раздел содержит информацию о двигателе:

- *Manufacturer* - компания-производитель мотора.
- *Part number* - номер по каталогу.
- *Motor type* - тип мотора (шаговый, DC, BLDC)
- *Poles* - количество пар полюсов у DC или BLDC двигателя или кол-во шагов на оборот для шагового двигателя.
- *Phases* - Кол-во фаз у BLDC двигателя.
- *Nominal voltage* - номинальное напряжение на обмотке.
- *Nominal current* - максимальный постоянный ток в обмотке для DC и BLDC двигателей, номинальный ток в обмотке для шаговых двигателей.
- *Nominal speed* - номинальная скорость.
- *Nominal torque* - номинальный крутящий момент.
- *Nominal power* - номинальная мощность.
- *Winding resistance* - активное сопротивление обмотки.
- *Winding inductance* - индуктивность обмотки.
- *Rotor inertia* - инерция ротора.
- *Stall torque* - крутящий момент при нулевой скорости.
- *Detent torque* - момент удержания позиции с незапитанными обмотками.
- *Torque constant* - константа крутящего момента.
- *Speed constant* - константа скорости.
- *Speed/torque gradient* - константа скорость/момент.
- *Mechanical time constant* - постоянная времени мотора.
- *Max speed* - максимальная разрешённая скорость.
- *Max current* - максимальный ток в обмотке.
- *Max current time* - безопасная длительность максимального тока в обмотке.
- *No load current* - потребляемый без нагрузки ток.
- *No load speed* - скорость на холостом ходу.

5.5.4 Характеристики энкодера

Важно: Информация на вкладке *«Stage -> Encoder»* временно не используется

В окне *Настройки программы* **Stage -> Encoder**

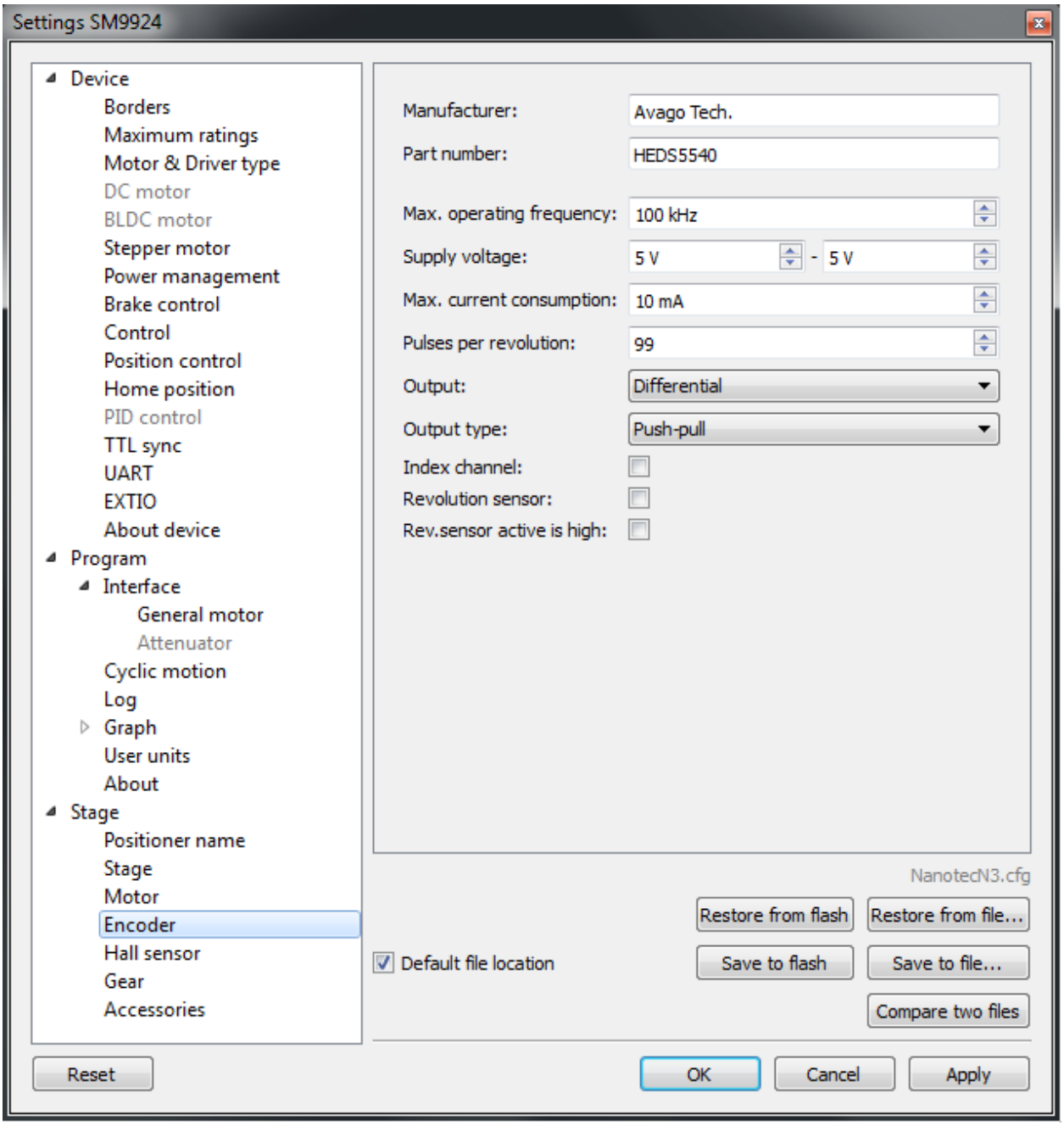


Рис. 5.52: Окно с описанием характеристик энкодера

Раздел содержит информацию *об энкодере*:

- *Manufacturer* - наименование компании-производителя энкодера.
- *Part number* - номер по каталогу.
- *Max. operating frequency* - максимальная рабочая частота.
- *Supply voltage* - диапазон допустимых напряжений питания.
- *Max. current consumption* - максимальное потребление тока.

- *Pulses per revolution* - количество отсчётов на оборот.
- *Output* - дифференциальный выход или несимметричный выход.
- *Output type* - тип выхода (двухтактный выход, или выход с открытым коллектором).
- *Index channel* - наличие дополнительного индексного канала.
- *Revolution sensor* - наличие датчика оборотов.
- *Rev.sensor active is high* - присутствие данного свойства означает что активное состояние датчика оборотов соответствует логической единице, отсутствие - логическому нулю.

5.5.5 Характеристики датчика Холла

Важно: Информация на вкладке *«Stage -> Hall sensor»* временно не используется

В окне *Настройки программы* **Stage -> Hall sensor**

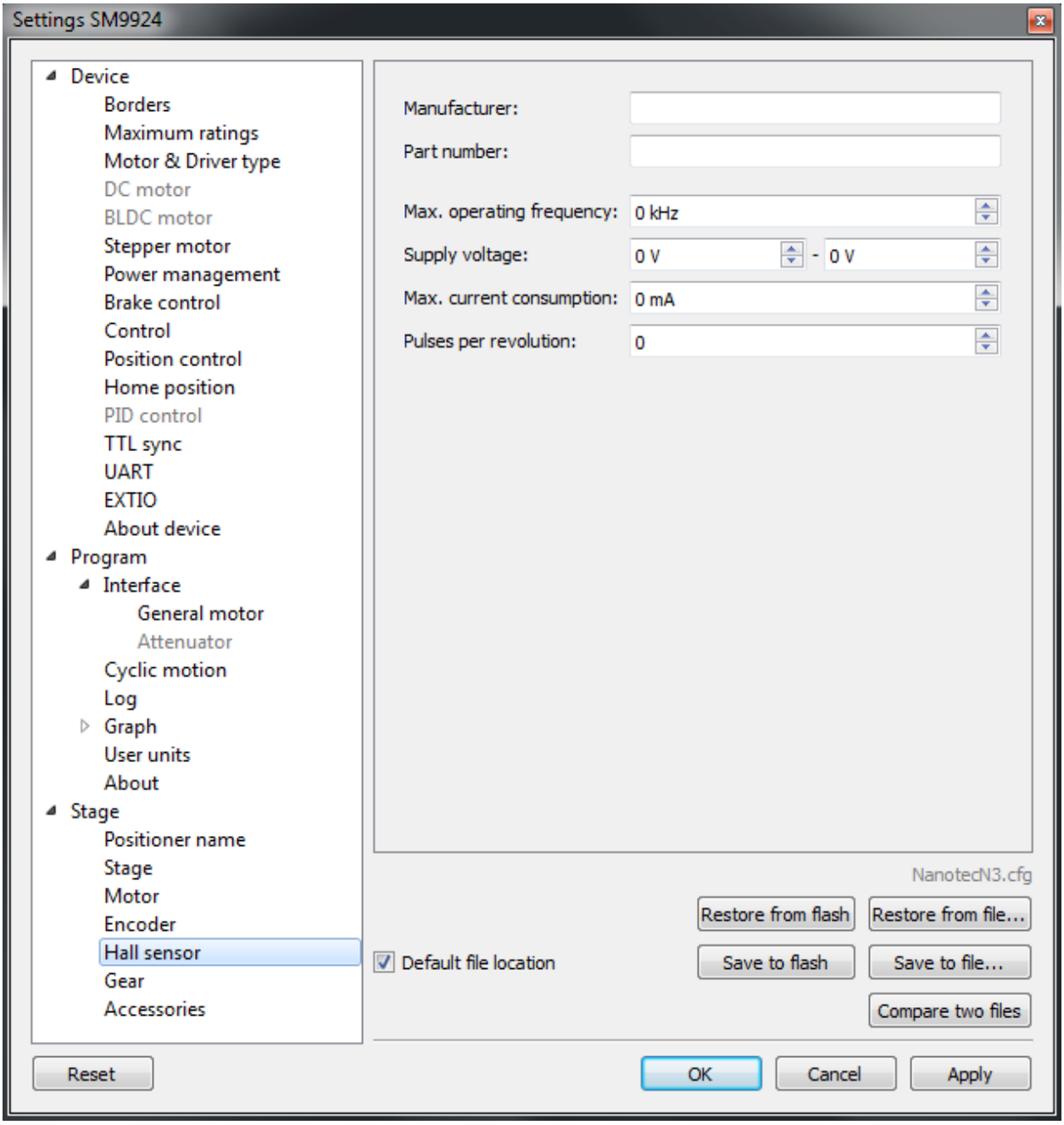


Рис. 5.53: Окно с описанием характеристик датчика Холла

Раздел содержит информацию о датчике Холла:

- *Manufacturer* - наименование компании-производителя датчика.
- *Part number* - номер по каталогу.
- *Max. operating frequency* - максимальная рабочая частота.
- *Supply voltage* - диапазон допустимых напряжений питания.
- *Max. current consumption* - максимальное потребление тока.

- *Pulses per revolution* - количество отсчётов на оборот.

5.5.6 Характеристики редуктора

Важно: Информация на вкладке *«Stage -> Gear»* временно не используется

В окне *Настройки программы* **Stage -> Motor**

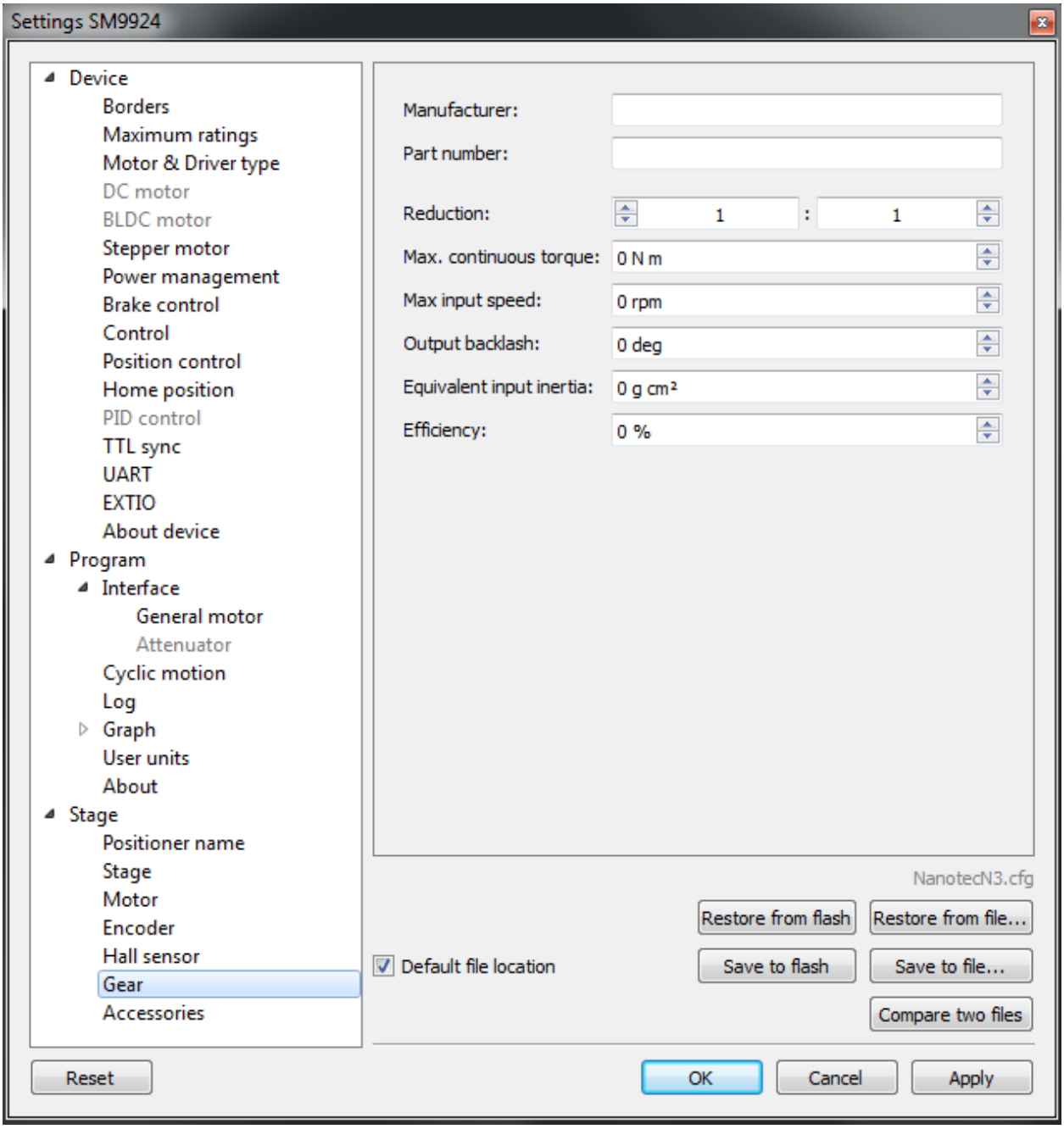


Рис. 5.54: Окно с описанием характеристик редуктора

Раздел содержит информацию о редукторе:

- *Manufacturer* - наименование компании-производителя.
- *Part number* - номер по каталогу.
- *Reduction* - передаточный коэффициент редуктора.
- *Max. continious torgue* - максимальный крутящий момент на входе редуктора.
- *Max. input speed* - максимальная скорость на входе редуктора.
- *Output backlash* - выходной люфт редуктора.
- *Equivalent input inertia* - эквивалентная входная инерция редуктора.
- *Efficiency* - КПД редуктора.

5.5.7 Характеристики вспомогательных устройств

Важно: Информация на вкладке *«Stage -> Accessories»* временно не используется

В окне *Настройки программы* **Stage -> Accessories**

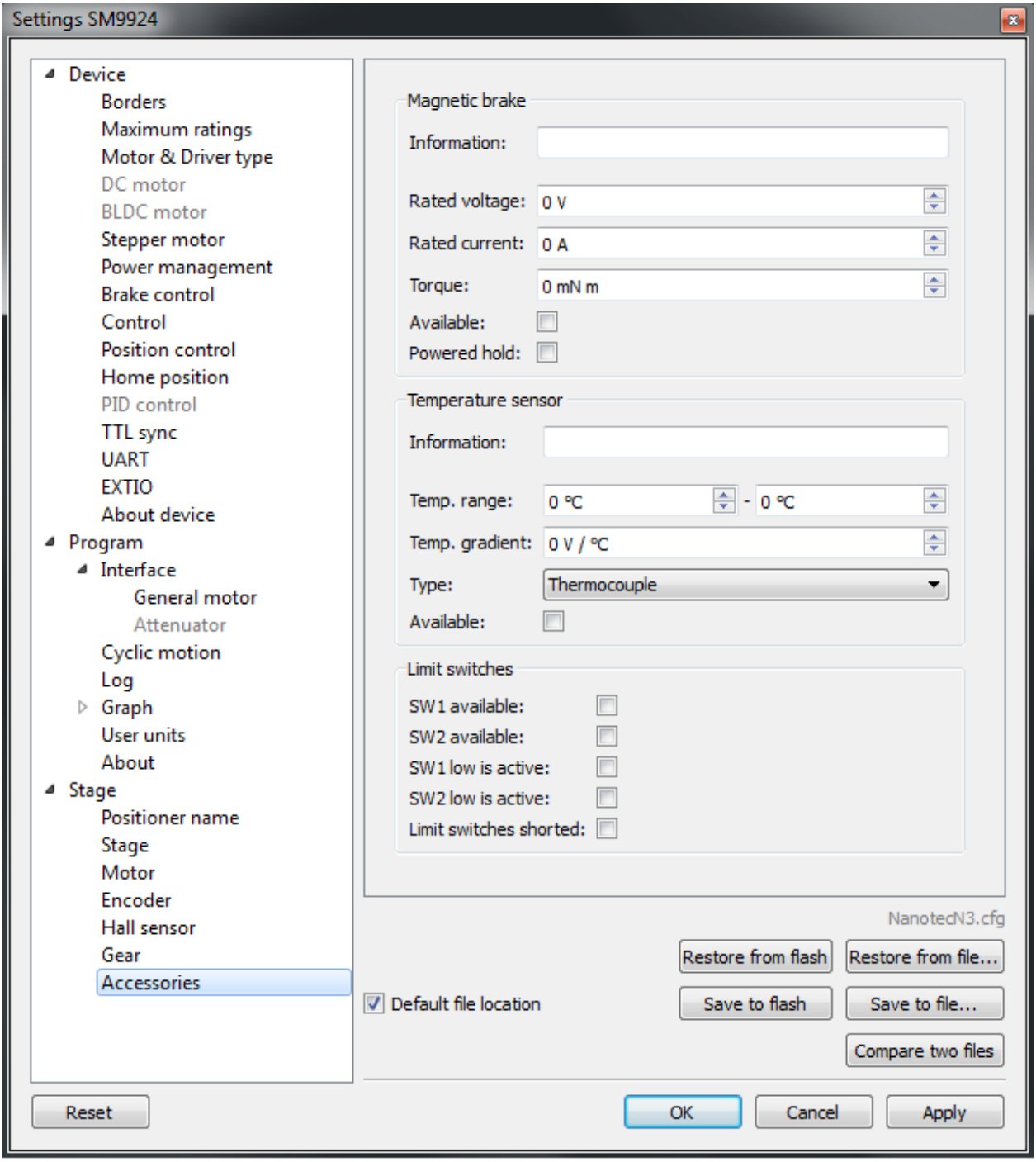


Рис. 5.55: Окно с описанием характеристик вспомогательных устройств

Раздел содержит информацию о вспомогательных устройствах.

Magnetic brake - блок магнитного тормоза:

- *Information* - производитель и номер магнитного тормоза.
- *Rated voltage* - номинальное напряжение для управления магнитным тормозом.

- *Rated current* - номинальный ток для управления магнитным тормозом.
- *Torque* - удерживающий момент.
- *Available* - наличие тормоза.
- *Powered hold* - присутствие данного свойства означает что магнитный тормоз находится в режиме удержания (активен) при подаче питания.

Temperature sensor - блок термодатчика:

- *Information* - производитель и номер температурного датчика.
- *Temp. range* - диапазон измеряемых температур.
- *Temp. gradient* - температурный градиент.
- *Type* - тип датчика (термопара или полупроводниковый температурный датчик).
- *Available* - наличие датчика.

Limit switches - блок концевиков:

- *SW1 available* - присутствие данного свойства означает что концевик, подключенный к ножке SW1, доступен.
- *SW2 available* - присутствие данного свойства означает что концевик, подключенный к ножке SW2, доступен.
- *SW1 low is active* - присутствие данного свойства означает что концевик, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте.
- *SW2 low is active* - присутствие данного свойства означает что концевик, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте.
- *Limit switches shorted* - присутствие данного свойства означает, что концевики закорочены.

5.6 Корректное завершение работы

Корректное завершение работы подразумевает остановку двигателя и сохранение текущей позиции контроллером. Текущая позиция сохраняется автоматически, см. *Хранение позиции во FRAM-памяти контроллера*.

Кнопка *Exit* осуществляет корректное завершение работы и выход из программы. При нажатии на неё программа отдает контроллеру команду плавной остановки, а после завершения остановки команду отключения питания. Если выполнение плавной остановки было прервано каким-либо событием, например подачей команды движения *джойстиком* или *сигналом TTL-синхронизации*, или если при посылке команды плавной остановки или команды отключения питания в контроллер библиотека вернула ошибку, то выход отменяется. В этом случае необходимо проверить *настройки джойстика и кнопок «вправо» и «влево»* и *настройки синхронизации*.

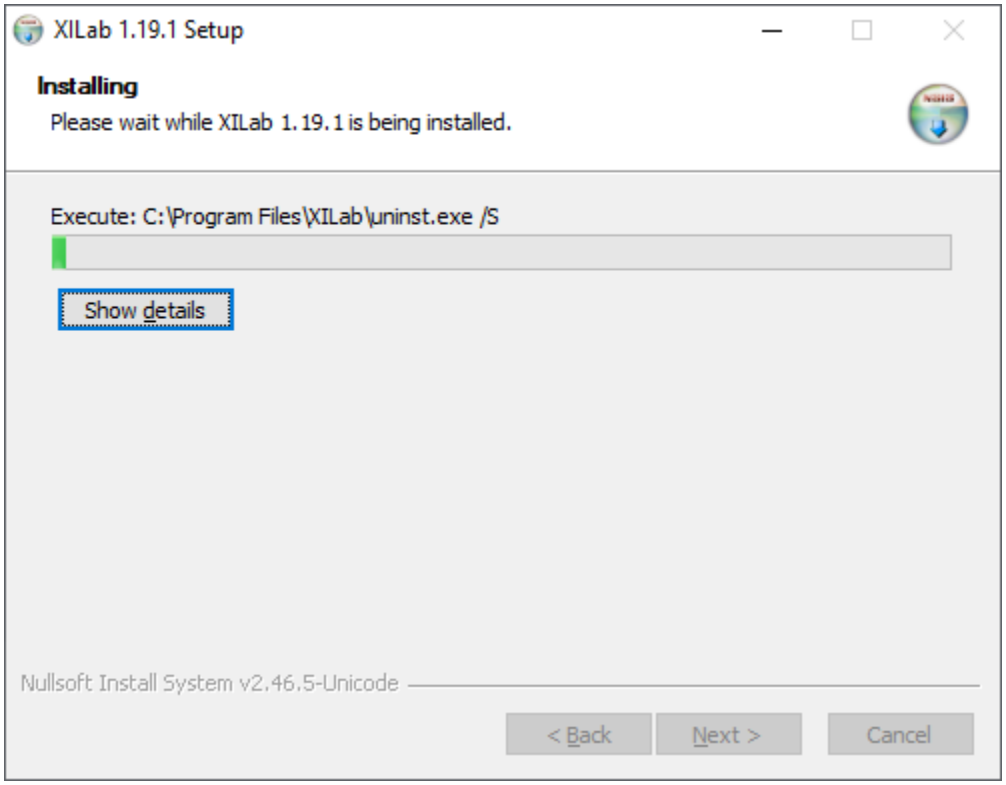
5.7 Установка XILab

5.7.1 Установка под Windows

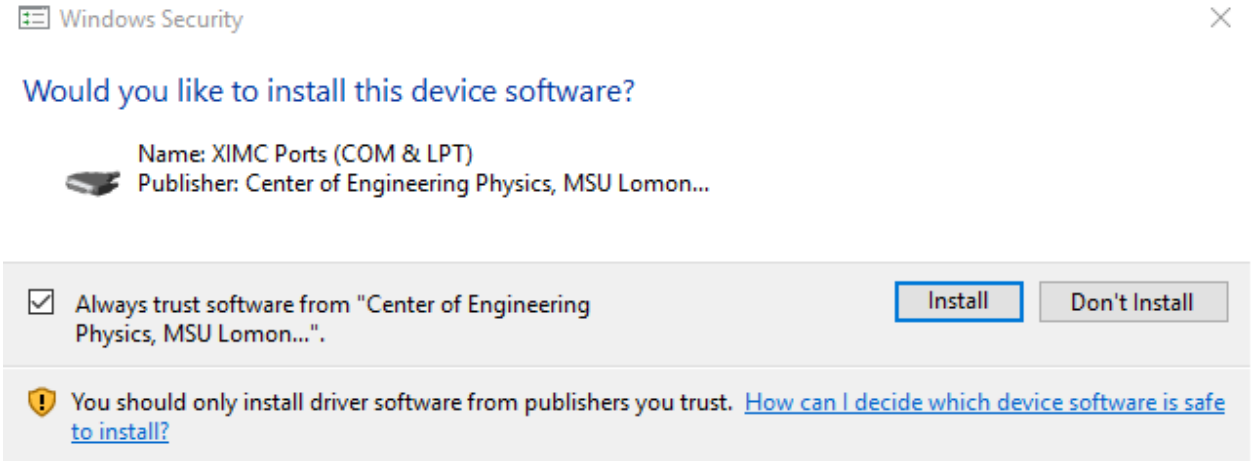
Загрузите файл программы-установщика на свой компьютер. Запустите «xilab-<имя_версии>.exe». Инсталлятор автоматически определяет, запущен ли он на 32-битной или 64-битной системе и устанавливает соответствующую версию XILab.



Запустите программу установки и следуйте инструкциям на экране.



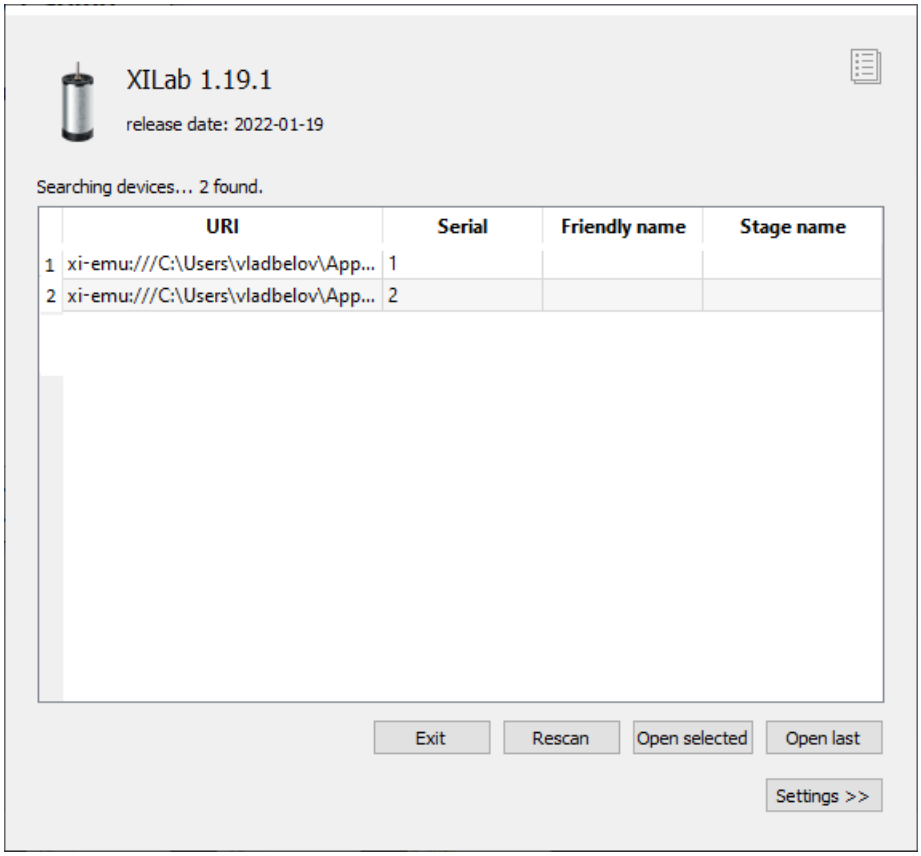
Все необходимое программное обеспечение, пакеты и программы будут установлены автоматически.



Нажмите кнопку *Install*, чтобы установить драйвер контроллеров Ximc.



После установки по умолчанию запустится программа XILab.



Подключите позиционер к контроллеру. Подключите стабилизированный источник питания к контроллеру. Заземлите контроллер. Подключите контроллер к компьютеру с помощью кабеля USB и/или Ethernet. Включите контроллер. LED индикатор на плате контроллера начнет мигать.

Если контроллер подключен через USB-кабель, подождите, пока windows обнаружит новое устройство, после чего нажмите в стартовом окне XILab кнопку *Rescan*. Система автоматически обнаружит подключенный контроллер.

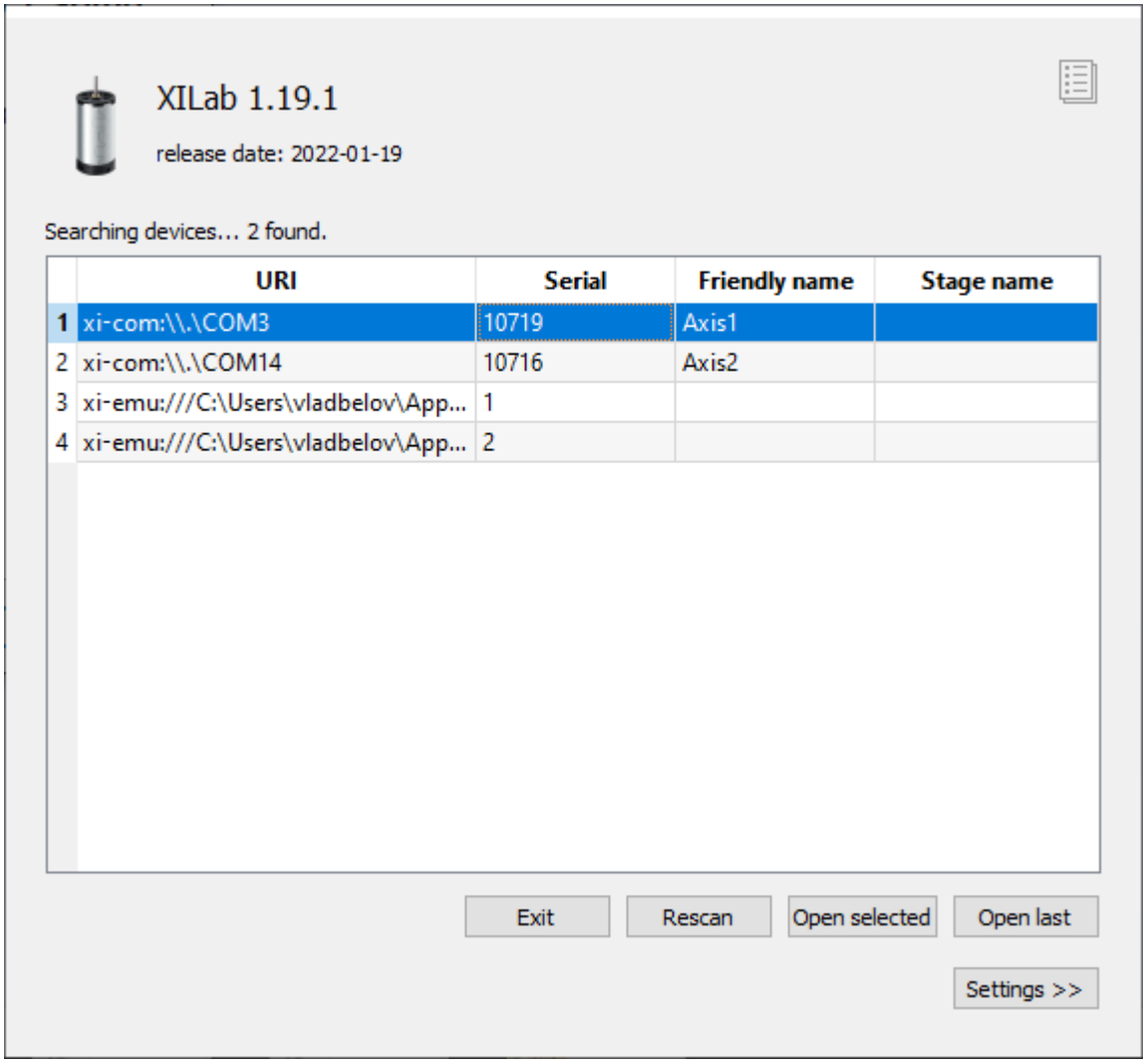


Рис. 5.56: Пример контроллера, подключенного через USB

Если контроллер подключен через Ethernet-кабель, в стартовом окне XILab в разделе *Settings>>* включите флаг *Enumerate network devices*. После этого нажмите кнопку *Rescan*. Система автоматически обнаружит подключенный контроллер. Если ваш контроллер не определяется автоматически в меню *Settings>>* , расположенном в стартовом окне XILab, введите IP-адрес вашего контроллера в поле *IP/Host*. После этого нажмите кнопку *Rescan*.

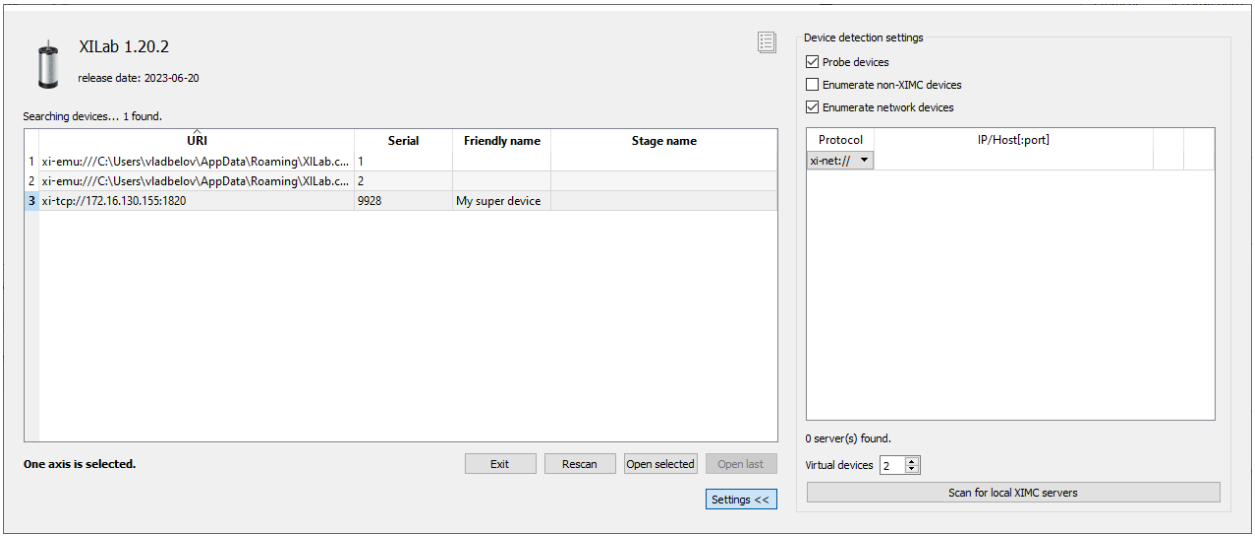


Рис. 5.57: Пример контроллера, подключенного через Ethernet

Чтобы открыть контроллер, дважды щелкните по нему левой кнопкой мыши или нажмите кнопку *Open selected*. Если вы хотите открыть несколько контроллеров одновременно, выберите их с помощью горячих клавиш *Ctrl* или *Shift*, а затем нажмите *Open selected*

Важно: Для работы с контроллером по Ethernet (протоколы tcp/udp) необходим контроллер версии 2.3.6 и выше, версия прошивки 4.7.x, версия XILab 1.20.x, библиотека libximc версии 2.14.x

---

5.7.2 Установка под Linux

Пакет XILab для Linux распространяется в формате AppImage - файл Linux, содержащий приложение и все, что нужно для его запуска (например, библиотеки, значки, шрифты, переводы и т. д.). Чтобы запустить XiLab, просто скачайте приложение, сделайте его исполняемым и запустите. Формат AppImage не требует установки, изменения системных библиотек или системных настроек.

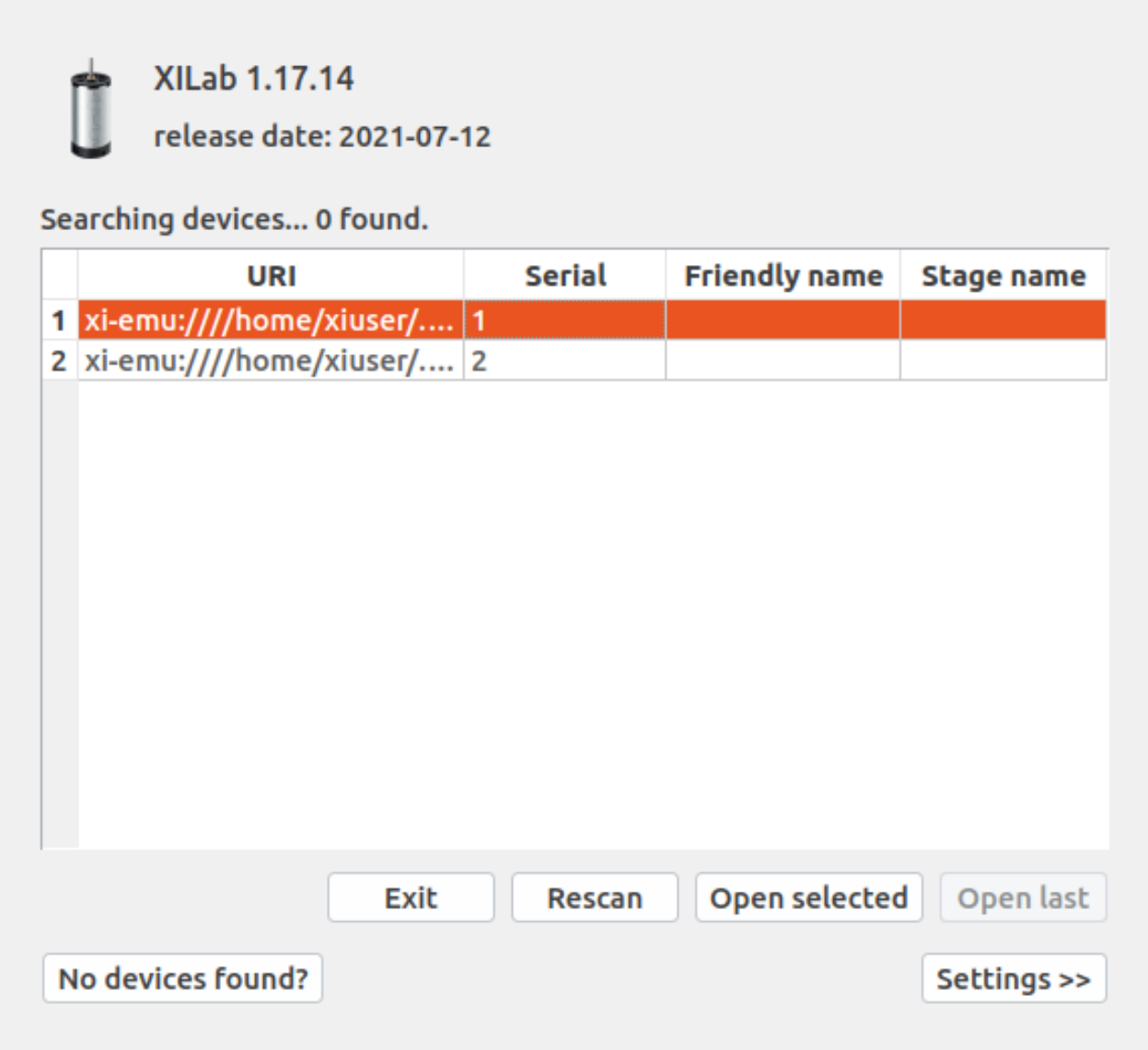
Существует два основных способа сделать файл AppImage исполняемым:

1. Используя графический интерфейс:

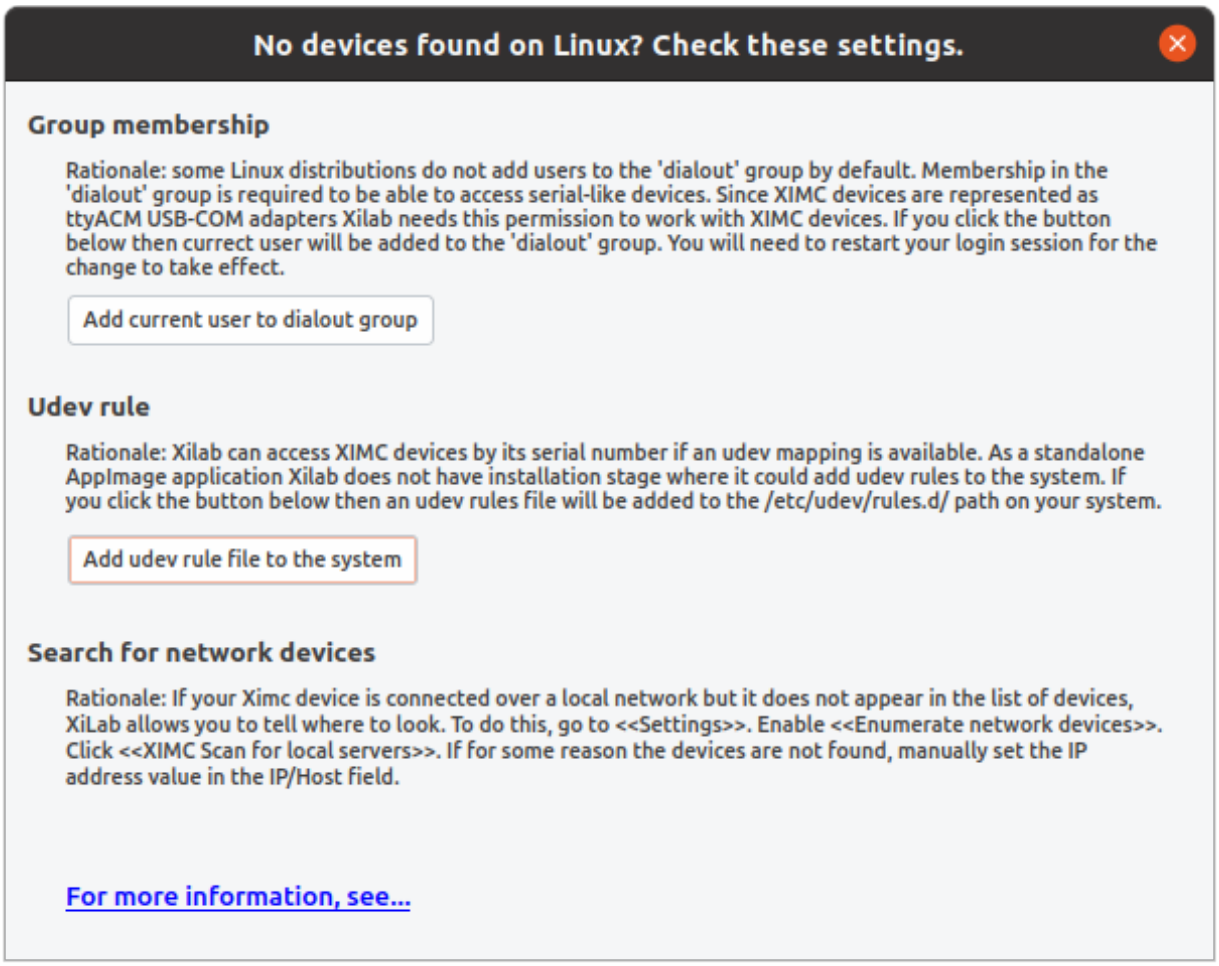
- Откройте диспетчер файлов и перейдите к местоположению файла AppImage;
- Щелкните правой кнопкой мыши на AppImage и нажмите кнопку «Свойства»;
- Перейдите на вкладку «Права» и установите флажок «Разрешить запуск этого файла в качестве программы», если вы используете файловый менеджер на основе Nautilus (Files, Nemo, Caja) или установите флажок «Is executable», если вы используете Dolphin, или измените раскрывающийся список «Execute» список «Anyone», если вы используете PCManFM;
- Закройте диалоговое окно;
- Запустите AppImage двойным щелчком по файлу.

2. С использованием командной строки:

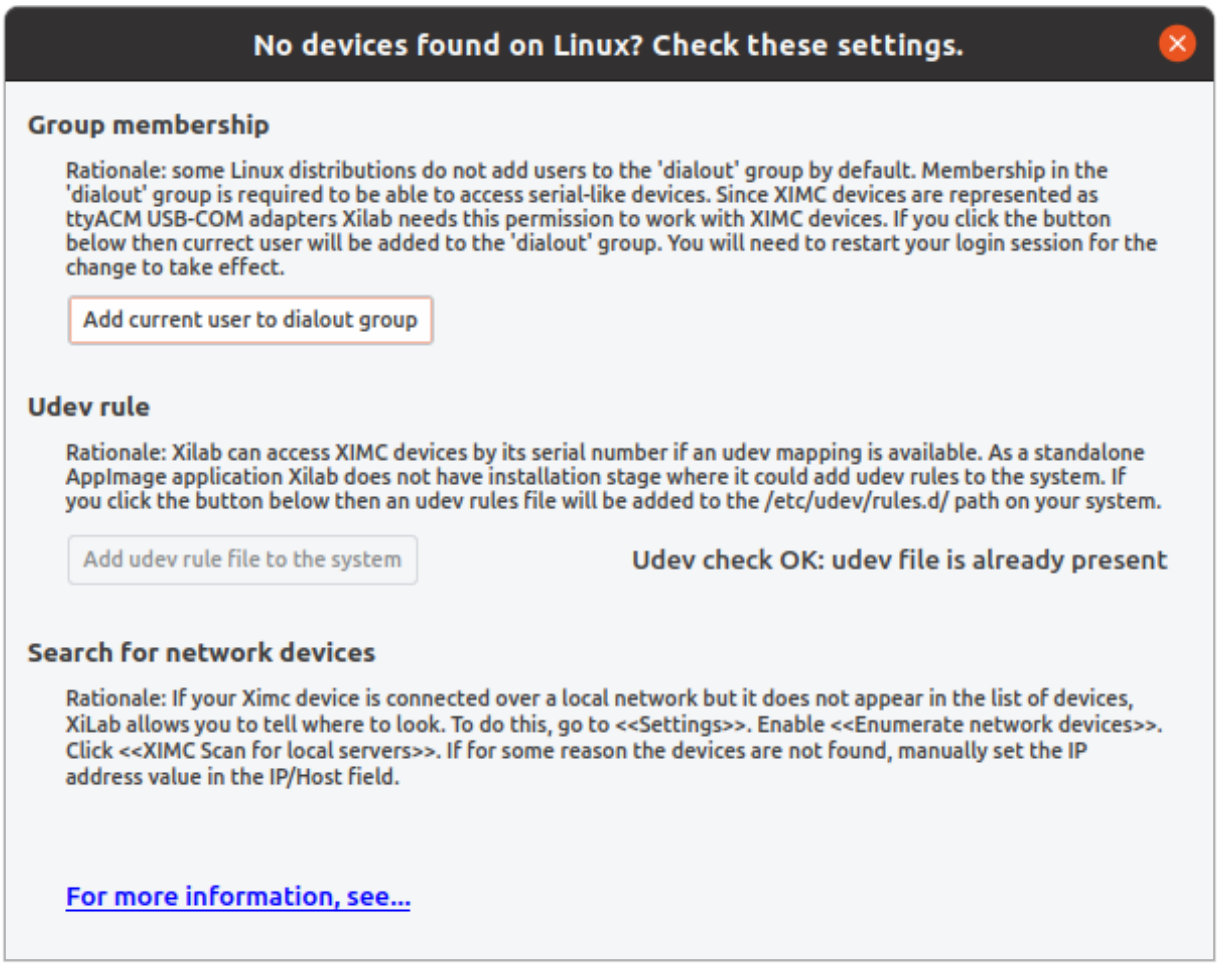
```
chmod a+x xilab-<version>-x86_64.AppImage
./xilab-<version>-x86_64.AppImage
```



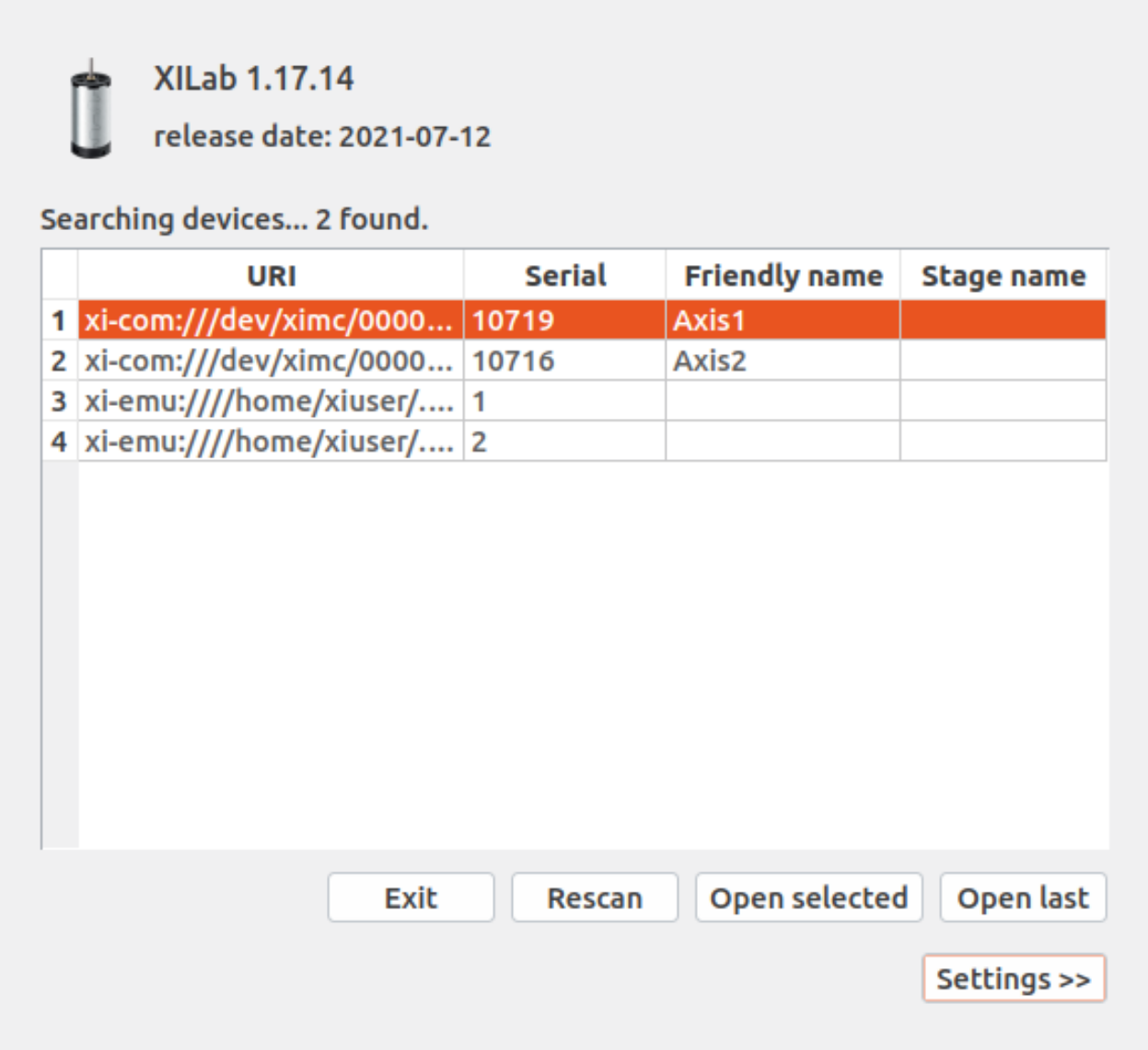
При первом запуске XiLab может не найти контроллеры, подключенные через USB. Для обнаружения XIMC-устройств XiLab требуется список устройств udev. В качестве автономного приложения AppImage, XiLab не имеет этапа установки, который может добавить в систему правила udev. Нажмите кнопку *No devices found?* в стартовом окне XiLab, затем нажмите *Add udev rule file to the system.*



Некоторые дистрибутивы Linux не добавляют пользователей в группу «dialout» по умолчанию. Членство в группе «dialout» необходимо для доступа к последовательным портам. Программе XiLab необходим этот доступ, поскольку XIMC-устройства представляются в системе как ttyACM USB-COM адаптеры. Нажмите *Add current user to the dialout group* и перезагрузите систему для применения изменений.



Важно: Программа XiLab для работы требует наличия X-сервера (графического режима).



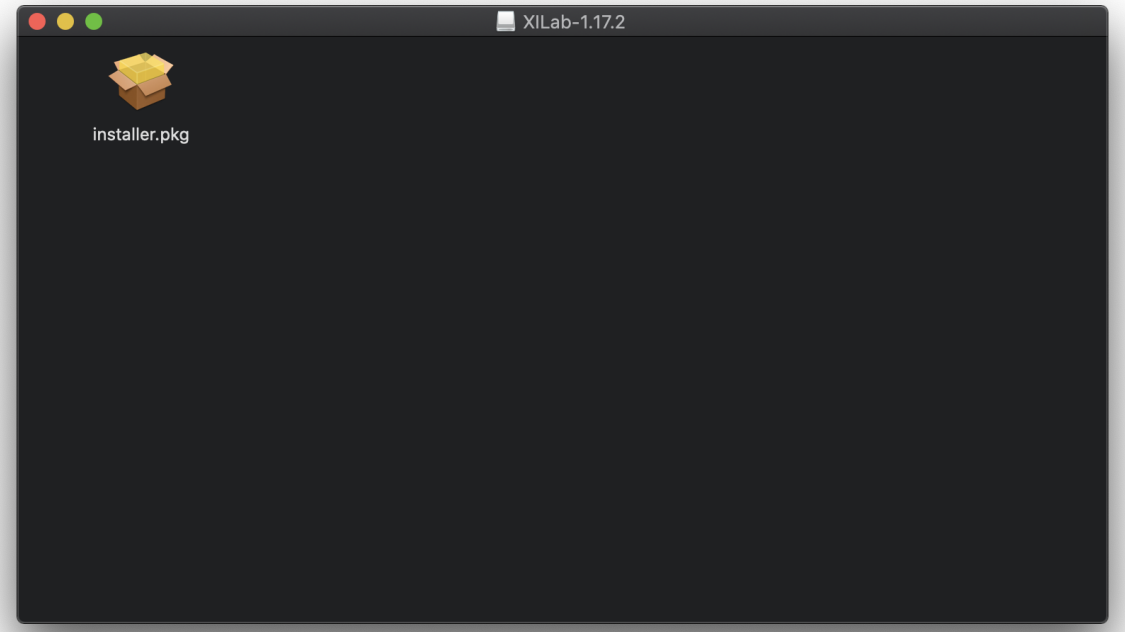
5.7.3 Установка под MacOS



Скопируйте файл с архивом программы установки на компьютер. Архив с программой установки имеет название «xilab-<номер версии>-osx64.tar.gz».



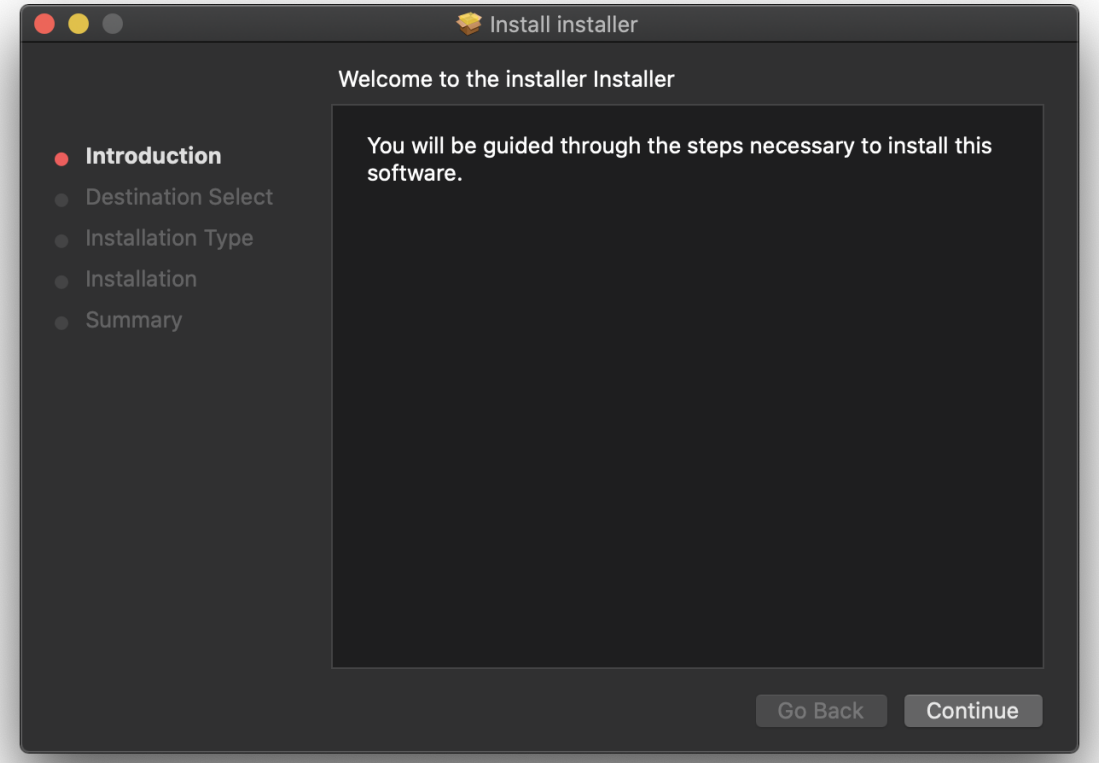
Распакуйте архив щелчком мыши.



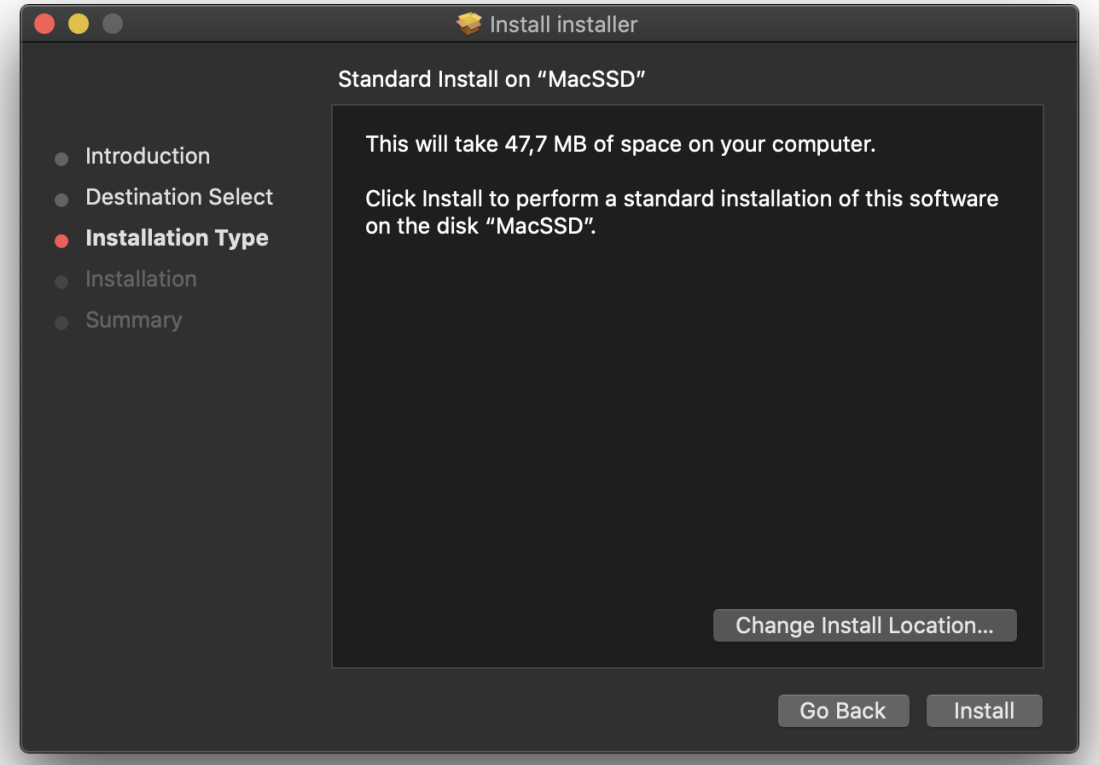
Двойным кликом левой кнопкой мыши кликните на появившийся installer.pkg.



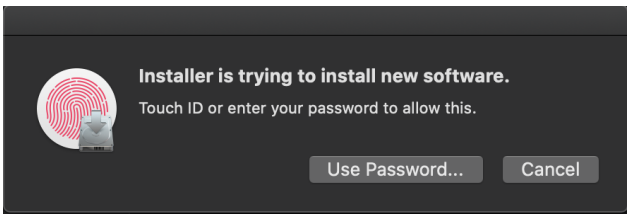
Выберите «Открыть».



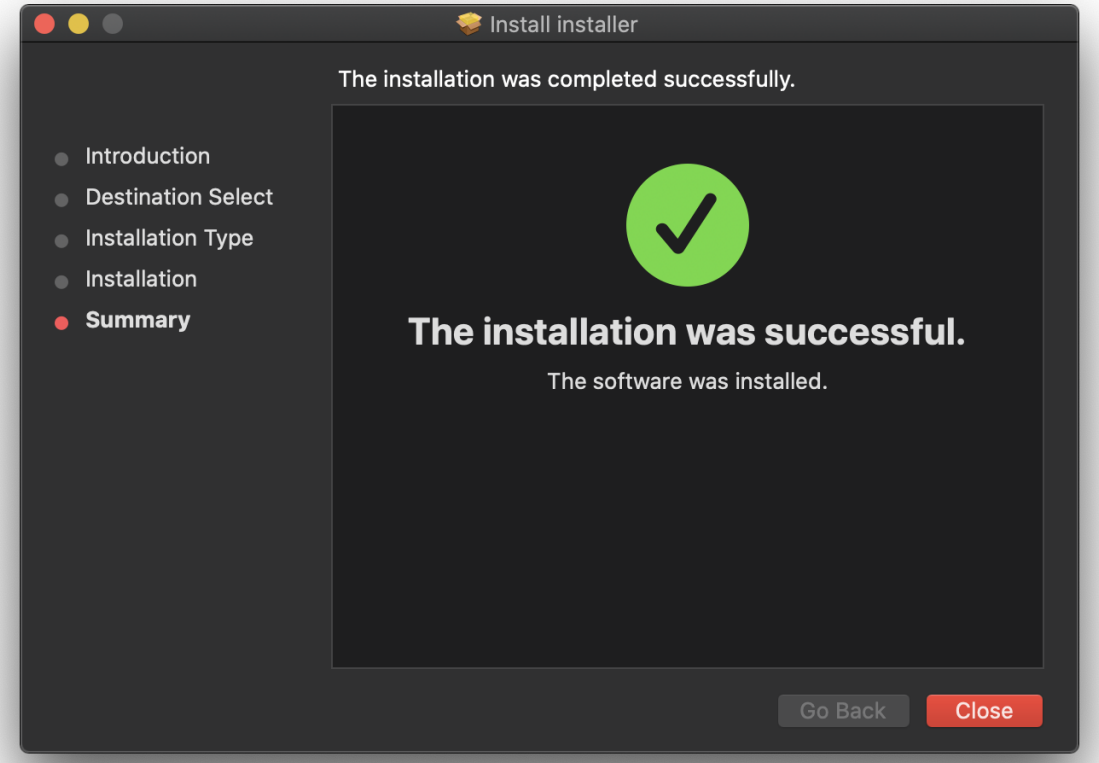
В главном окне установщика выберите «Продолжить».



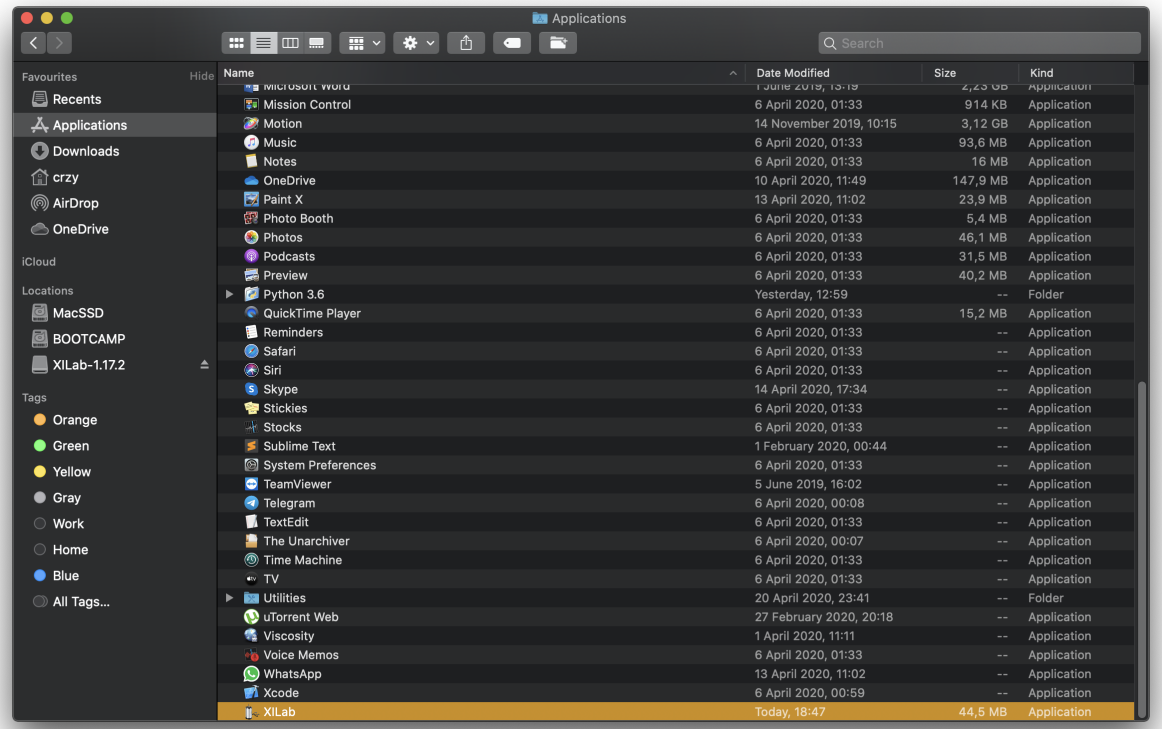
Далее выберите «Установить».



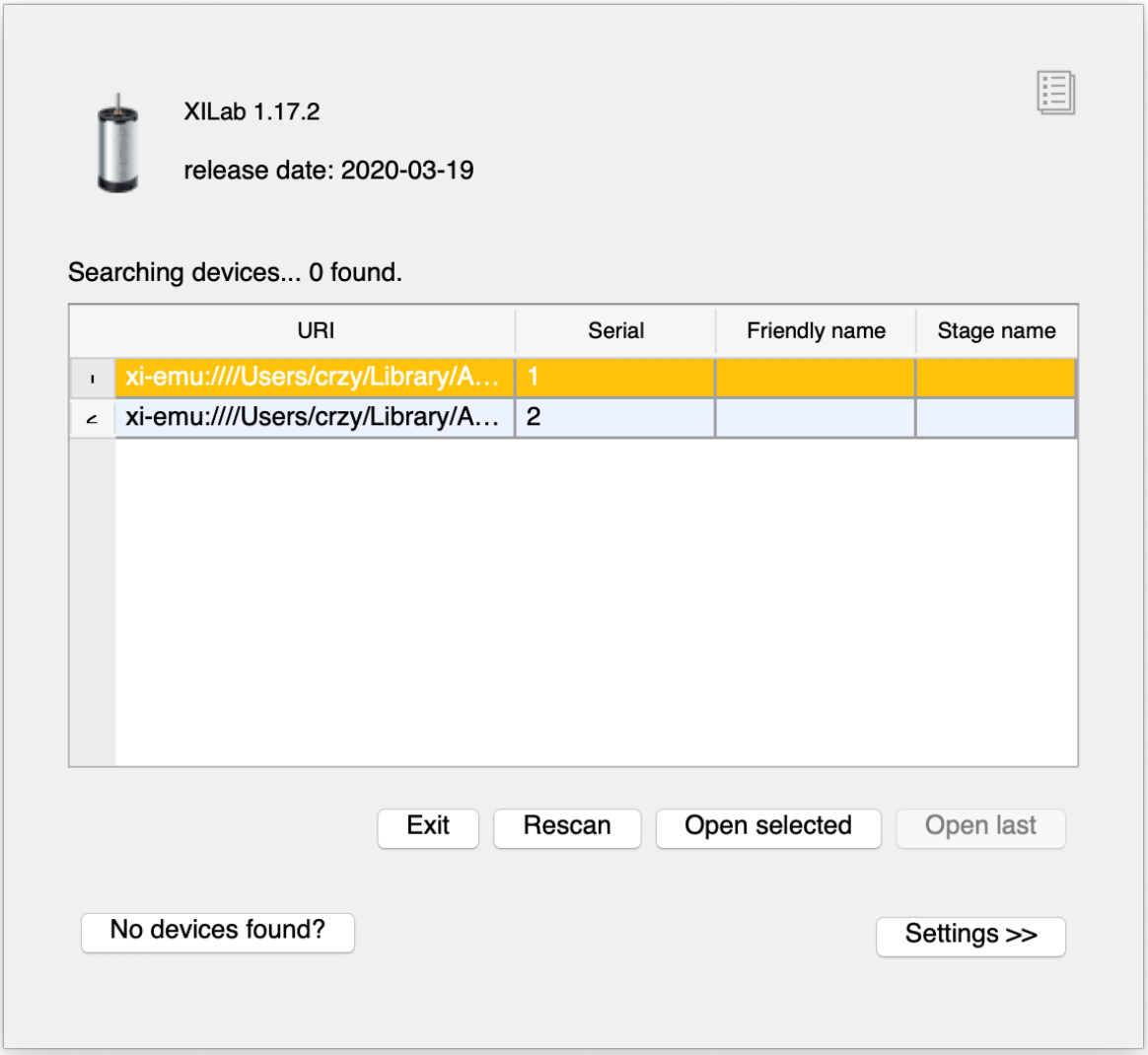
Введите пароль.



Дождитесь успешного завершения установки.



Выберите приложение XILab в разделе «Программы»


XILab 1.17.2
release date: 2020-03-19
Searching devices... 0 found.
URI
Serial
Friendly name
Stage name
xi-emu:////Users/crzy/Library/A…
1
xi-emu:////Users/crzy/Library/A…
2
Exit
Rescan
Open selected
Open last
No devices found?
Settings >>

Запустите.

Глава 6

Программирование

6.1 Руководство по программированию

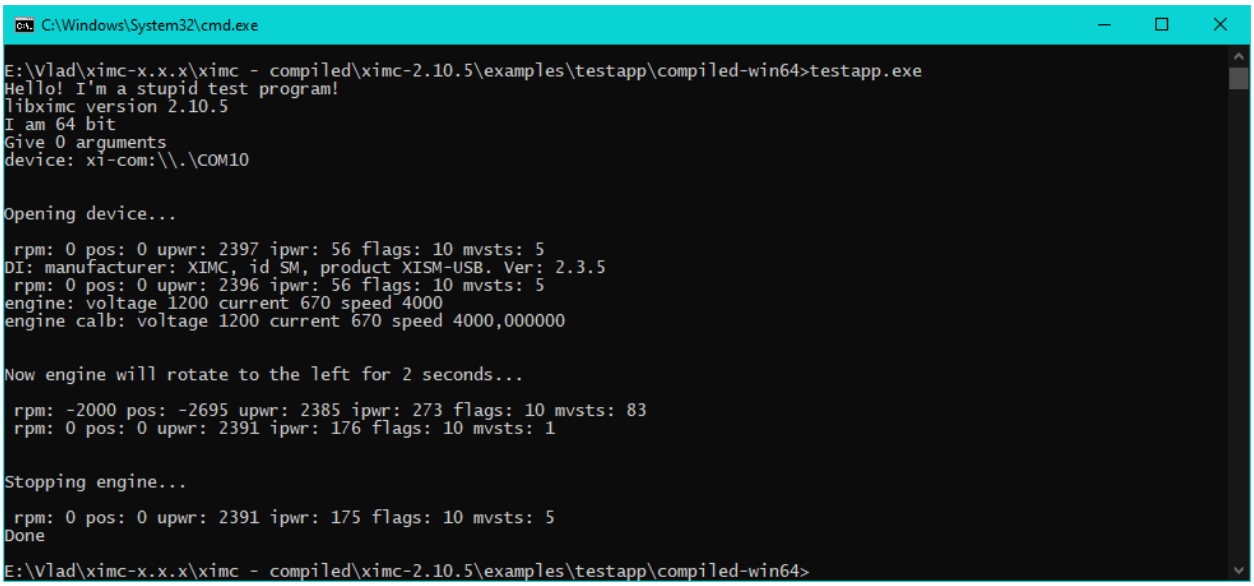
6.1.1 Работа с контроллером в среде Visual Studio

Скачайте пример программы для VisualStudio со страницы Программное обеспечение .

Примечание: Тестовое приложение может быть собрано с помощью *testapp.sln*. Для компиляции необходимо использовать также MS Visual C++. Убедитесь, что Microsoft Visual C++ Redistributable Package 2013 установлен.

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

Распакуйте архив и запустите *«testapp»*.



После запуска программы откроется командная строка. В ней Вы увидите сообщение: «Hello! I'm a stupid test program!»

Программа «testapp» сообщает версию используемой библиотеки, а также сообщает свою битность. Также *«testapp»* указывает, какой порт она удерживает.

После открытия устройства программа считывает поля данных из ссылки на структуру *«status_t»*.

rpm	int CurSpeed	Текущая скорость
pos	float CurPosition	Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь
upwr	int Upwr	Напряжение на силовой части, десятки мВ
ipwr	int Ipwr	Ток потребления силовой части
flags	unsigned int Flags	Флаги состояния
mvsts	unsigned int MvCmdSts	Состояние команды движения

*Функция result_t XIMC_API get_device_information (device_t id, device_information_t *device information)* - Возвращает информацию об устройстве

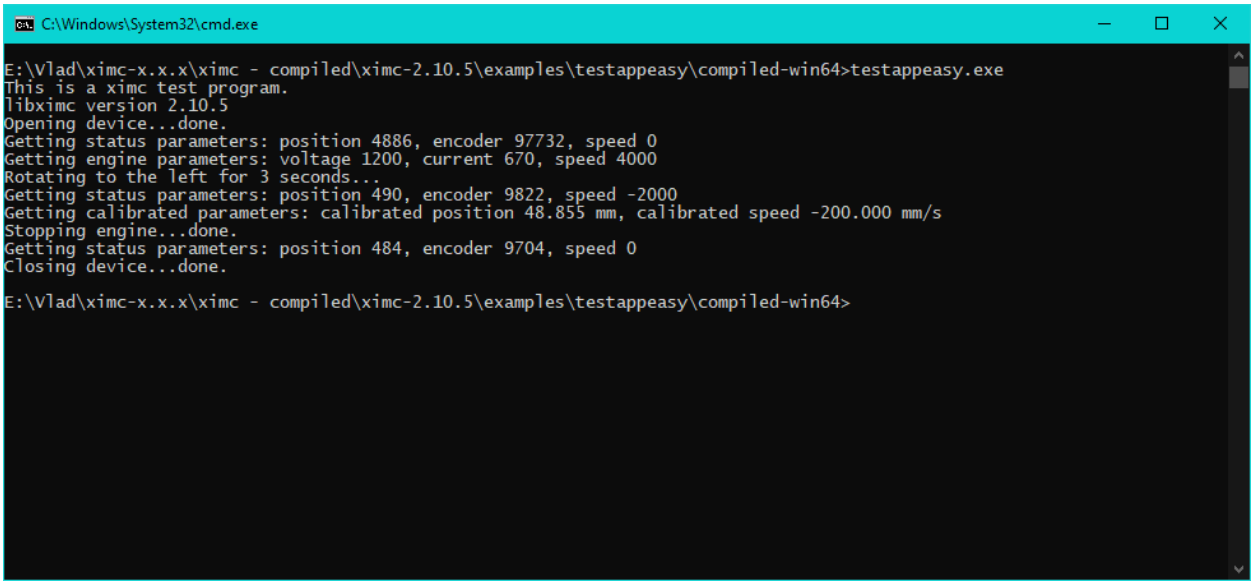
*Функция result_t XIMC_API get_engine_settings (device_t id, engine_settings_t *engine settings)* - Считывает настройки двигателя

Структура engine_settings_calb_t - result_t XIMC_API set_engine_settings_calb (device_t id, const engine_settings_calb_t *engine_settings - calb, const calibration_t *calibration)

После этого программа *testapp* отправляет в контроллер команду движения влево в течение 2 секунд *«command_left»*. После успешного выполнения команды *«command_left»*, вызывается команда остановки *«command_stop»*.

Важно: В конце на устройство отправляется команда *«command_stop»*. *«Close_device»* закрывает указанное устройство.

Программа *«testappeasy»* не так уж сильно отличается от программы *«testapp»*. Откройте проект *examples/testappeasy/testappeasy.sln*, выполните сборку и запустите приложение из среды разработки.



После запуска программы откроется командная строка. В ней Вы увидите сообщение: «This is a ximc test program.»

Программа умеет сообщать версию используемой библиотеки.

С помощью команды *«open_device»* программа *«testappeasy»* открывает устройство в режиме эксклюзивного доступа.

Предупреждение: Библиотека Libximc работает с контроллером в режиме эксклюзивного доступа. Каждый контроллер, открытый билиотекой libximc (XiLab тоже использует эту билиотеку), должен быть закрыт, прежде чем может быть использован другим процессом.

После открытия устройства программа *testappeasy* отправляет в контроллер команду движения влево в течении 3 секунд *«command_left»*. После успешного выполнения команды *«command_left»* вызывается команда остановки *«command_stop»*.

Команда *«calibration.A = 0.1;»* устанавливает калибровочную константу 0.1 (один шаг контроллера равен этому количеству единиц)

Команда *«calibration.MicrostepMode = engine_settings.MicrostepMode;»* - Используется для установки режима микрошагов, используется для правильного преобразования микрошагов в калиброванные единицы измерения.

После программа «testappeasy» считывает состояние калиброванного устройства с устройства.

В конце на устройство отправляется команда *«command_stop»*. *«Close_device»* - закрывает указанное устройство.

6.1.2 Работа с контроллером в среде Delphi

Скачайте пример программы для Delphi со страницы Программное обеспечение.

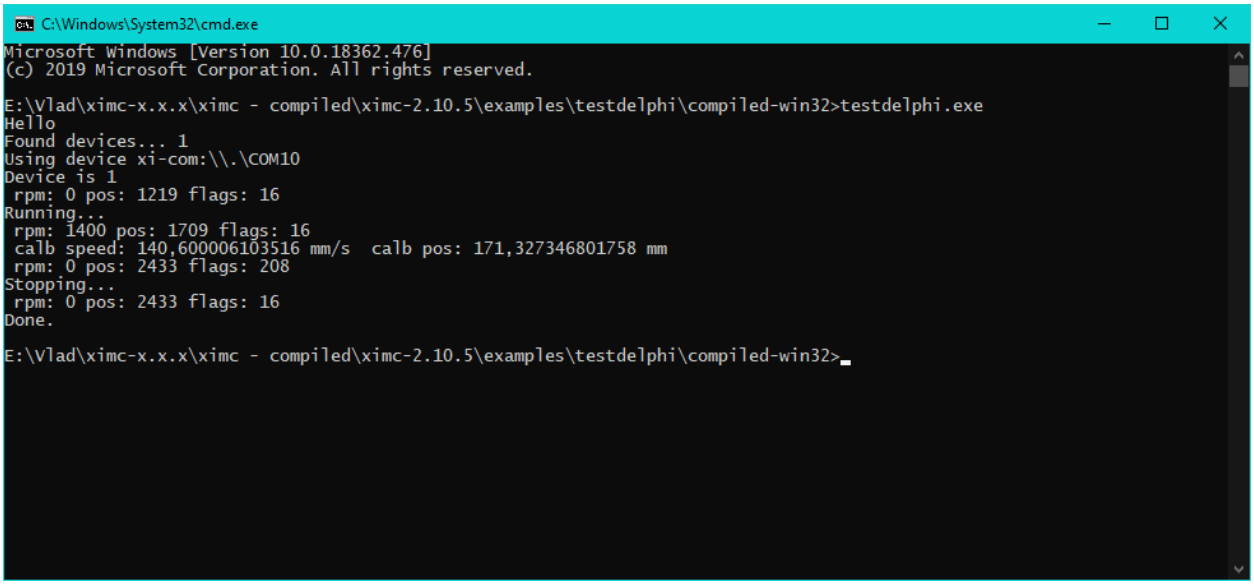
Примечание: Обертка для использования в Delphi libximc.dll предлагается как модуль wrappers/pascal/ximc.pas. Консольное тестовое приложение размещено в директории «testdelphi». **Проверено с Delphi 6 на 32-битной системе**. Просто скомпилируйте, разместите DLL в директории с исполняемым модулем и запустите его.

Предупреждение: Для работы примера скопируйте файлы ximc.pas, libximc.dll, xiwrapper.dll, bindy.dll из папки ximc рядом с *«testdelphi.dpr»* и запустите *«testdelphi.dpr»* с помощью Вашей Delphi IDE.

Чтобы запустить скомпилированный пример из командной строки, скопируйте все файлы (libximc.dll, bindy.dll ... и т.д.) в папку «compiled-win32» и запустите *«testdelphi.exe»*

Откройте проект *examples/testdelphi/testdelphi.dpr*, выполните сборку и запустите приложение из среды разработки.

Распакуйте архив и запустите *«testdelphi.exe»*.



После запуска программы откроется командная строка. В ней Вы увидите сообщение: «Hello»

Демонстрационная программа *«testdelphi»* использует команду **«result_t XIMC_API get_status (device_t id, status_t ,status)»** для вывода текущего состояния устройств. Также *«testdelphi»* указывает, какой порт она удерживает.

С помощью команды *«open_device»* программа *«testdelphi»* открывает устройство в режиме эксклюзивного доступа.

Предупреждение: Библиотека Libximc работает с контроллером в режиме эксклюзивного доступа. Каждый контроллер, открытый билиотекой libximc (XiLab тоже использует эту билиотеку), должен быть закрыт, прежде чем может быть использован другим процессом.

После открытия устройства программа считывает поля данных из ссылки на структуру *«status_t»*.

rpm	int CurSpeed	Текущая скорость
pos	float CurPosition	Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь
flags	unsigned int Flags	Флаги состояния

Функция *result_t XIMC_API get_engine_settings (device_t id, engine_settings_t *engine settings)* считывает настройки двигателя

После этого программа *«testdelphi»* выполняет команду движения вправо *«command_right»*. После успешного выполнения команды *«command_right»*, вызывается команда *«command_stop»*.

Команда *«calibration.A = 0.1;»* - устанавливает калибровочную константу 0.1 (один шаг контроллера равен этому количеству единиц)

Команда *«calibration.MicrostepMode = engine_settings.MicrostepMode;»* - используется для установки режима микрошагов, также команда используется для правильного преобразования микрошагов в калиброванные единицы измерения.

После программа «testdelphi» считывает состояние калиброванного устройства.

Предупреждение: В конце на устройство отправляется команда *«command_stop»*. *«Close_device»* - закрывает указанное устройство.

6.1.3 Работа с контроллером в среде LabView

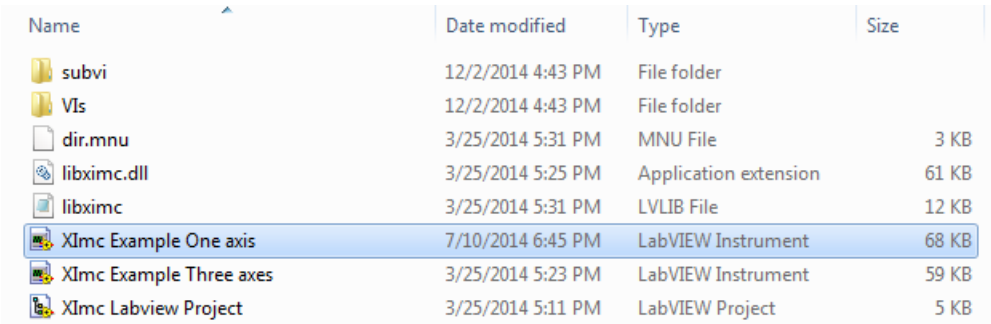
Скачайте пример программы для LabView со страницы Программное обеспечение. Разархивируйте скачанный файл в желаемую директорию и запустите пример «XImc Example One axis».

Предупреждение: Наши LabView примеры работают только для Windows. Битность примеров LabView должна соответствовать битности операционной системы, **а не битности LabView**

Важно: Примеры, приведённые ниже, предназначены для LabVIEW версии 12. Корректная работа примеров с более поздними версиями LabVIEW не гарантируется.

За более старыми версиями, пожалуйста, обращайтесь в техническую поддержку или присылайте письмо на почту: **8smc4@standa.lt**

Разархивируйте скачанный файл в желаемую директорию и запустите пример «Ximc Example One axis».



Предупреждение: «One axis example.vi» это простой в использовании, но довольно сложный пример псевдо-xilab, который не рекомендуется изменять самостоятельно. Обратите свое внимание на пример «ximc simple example.vi», его легко адаптировать для ваших целей.

Откроется окно среды LabView. В нем вы увидите графический интерфейс передней панели программы-примера, он представляет собой упрощенную копию интерфейса *XILab*.

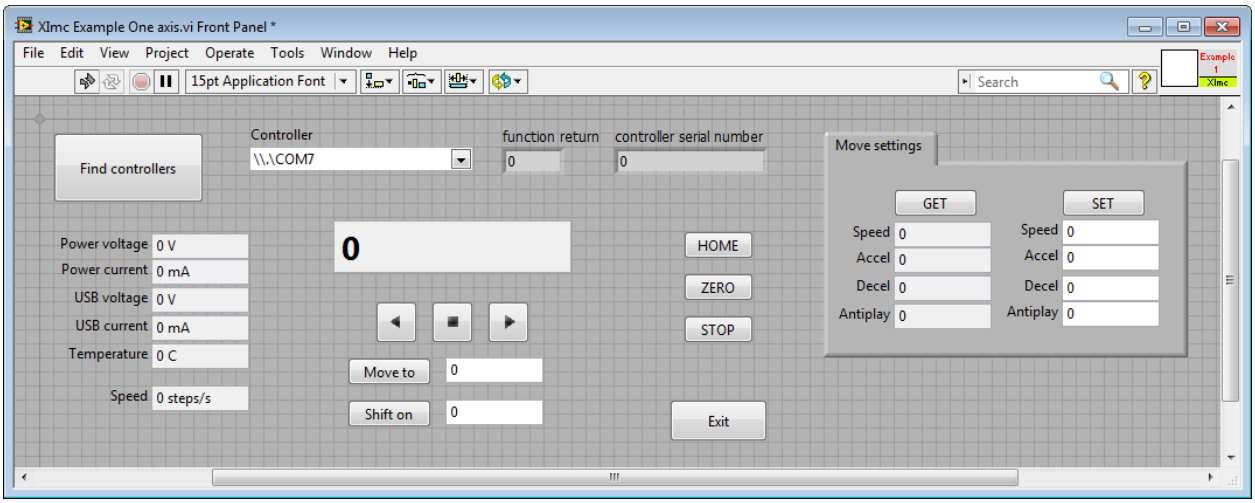
Примечание: LabVIEW работает независимо от XiLab. XiLab – это альтернативный инструмент для взаимодействия с контроллером. Его удобно использовать для конфигурации устройства и тестов. Все функции, доступные через XiLab, могут быть независимо реализованы средствами LabVIEW.

В левой части расположена кнопка «Find controllers» для повторного опроса доступных контроллеров, поле выбора контроллера по имени последовательного порта и информационный блок текущего состояния открытого контроллера (напряжение и ток на силовой части и USB, температура, скорость движения).

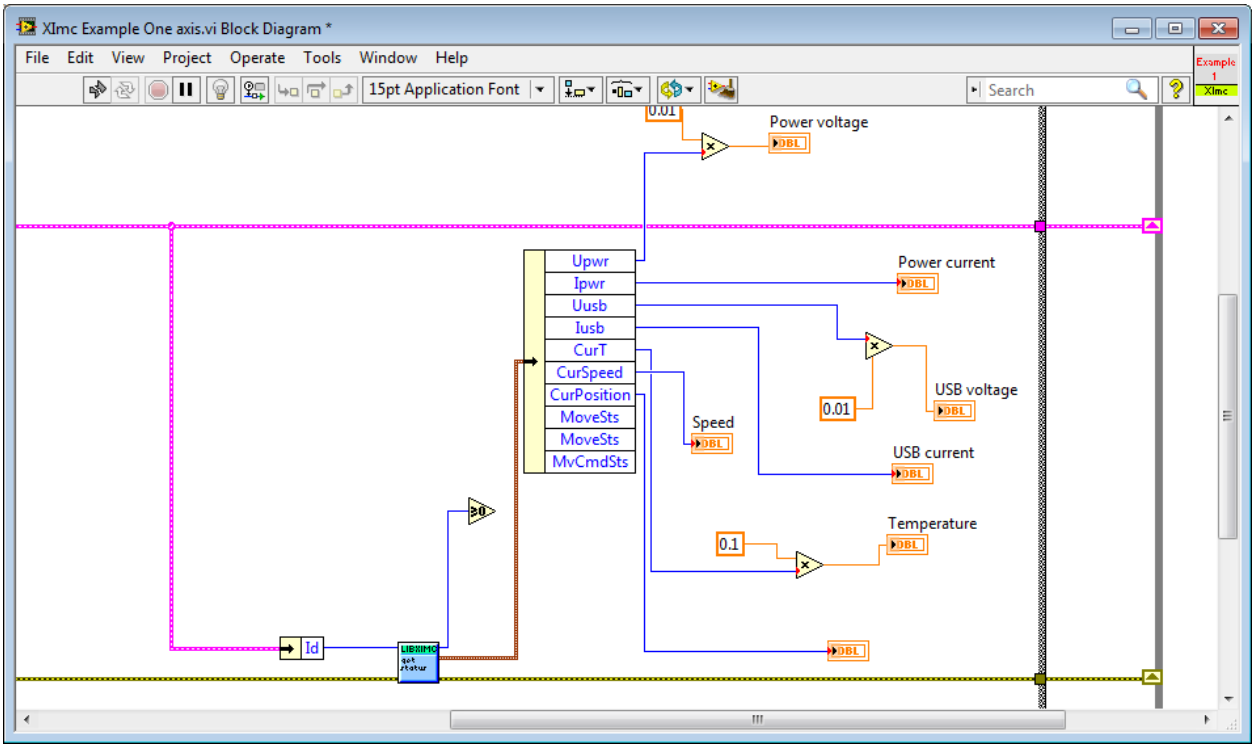
В центральной части расположен блок индикации и управления. В числовое поле выводится текущая координата; доступны кнопки движения влево, вправо, остановки, движения в определенную координату («Move to») и смещения на определенное расстояние («Shift on»).

В центральной части расположен блок индикации и управления. В числовое поле выводится текущая координата; доступны кнопки движения влево, вправо, остановки, движения в определенную координату («Move to») и смещения на определенное расстояние («Shift on»).

На вкладке справа расположен диалог «Move settings» для демонстрации загрузки и сохранения настроек. По нажатию кнопки «GET» текущие настройки движения загружаются в числовые поля под этой кнопкой, а по нажатию кнопки «SET» числа из соответствующих этой кнопке полей загружаются в контроллер.

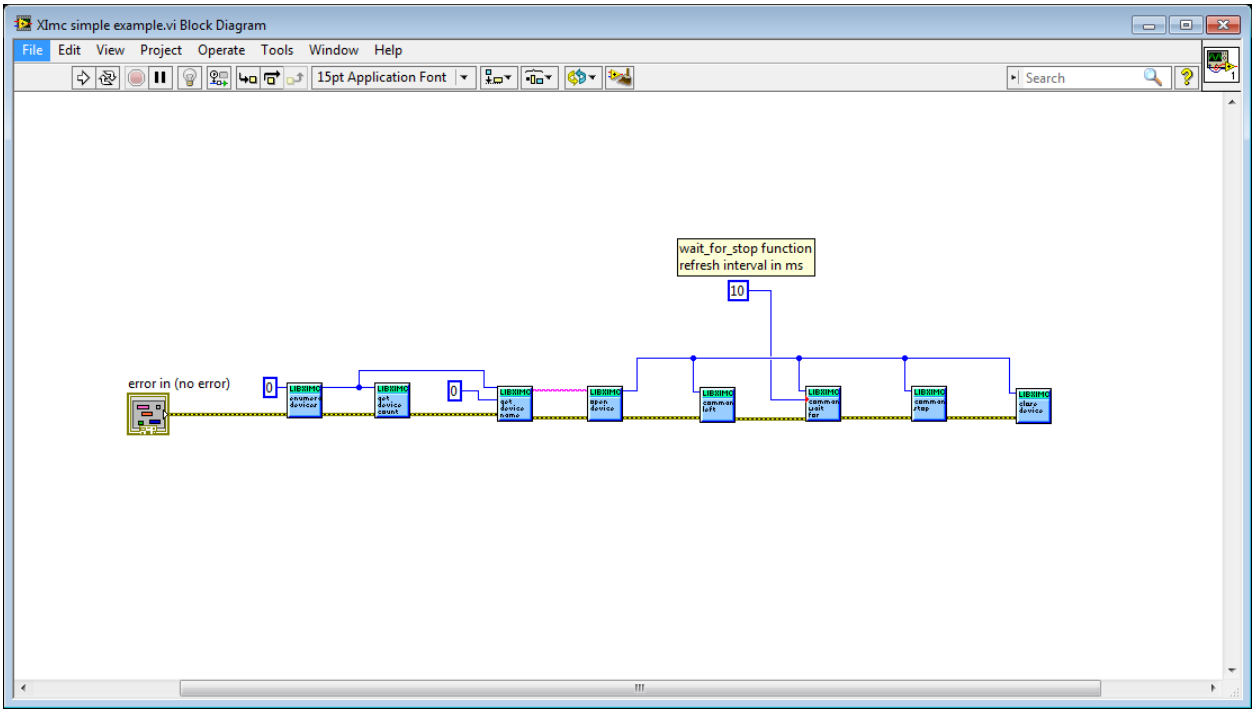


Исходный код примера можно посмотреть, войдя в режим редактирования. Программа выполняет бесконечный цикл опроса состояния контроллера и вывода на экран. Если нажата какая-либо кнопка в интерфейсе, то выполняется соответствующий этой кнопке функциональный блок.



Для взаимодействия с контроллерами LabVIEW использует библиотеку libximc. Интерфейс, доступный через LabVIEW, полностью аналогичен интерфейсу библиотеки libximc. Подробное описание функций, структур данных и флагов можно загрузить по этой ссылке. Каждой функции соответствует свой subVI модуль, имеющий входы и выходы соответствующие входным и выходным параметрам этой функции. Для того чтобы вызывать какую-либо функцию нужно сначала выполнить опрос устройств (enumerate_devices), после этого выбрать какое-либо устройство из доступных, открыть его с помощью open_device и передать полученный идентификатор и необходимые параметры желаемой функции. После использования необходимо закрыть устройство функцией close_device.

Рассмотрим создание простейшей программы на Labview для работы с libximc на примере «Ximc simple example.vi».



В начале программы вызывается функция enumerate_devices, которой передаются флаги открытия (подробнее см. *Руководство по программированию*). Результат выполнения функции enumerate_devices, являющийся закрытым указателем, передается функции get_device_name с параметром номера устройства, имя которого мы хотим получить (количество обнаруженных устройств можно получить функцией get_device_count из этого же указателя). Результат выполнения функции get_device_name передается функции open_device. Эта последовательность не является обязательной - при желании сформированную определенным образом строку с именем открываемого устройства можно передать функции open_device напрямую. Результатом выполнения функции open_device является хендл устройства или определенная в ximc.h константа device_undefined, возвращаемое при невозможности открыть устройство. Хендл устройства передается всем функциям чтения значений из контроллера, записи значений в контроллер и подачи команд контроллеру, вместе с другими параметрами, если таковые требуются в соответствии с прототипом функции. В примере «Ximc simple example.vi» вызываются команды command_left, command_wait_for_stop и command_stop. После окончания работы с контроллером устройство необходимо закрыть, передав его хендл функции close_device, а после окончания работы с результатом функции enumerate_devices необходимо освободить память закрытого указателя функцией free_enumerate_devices (опущено в примере для краткости).

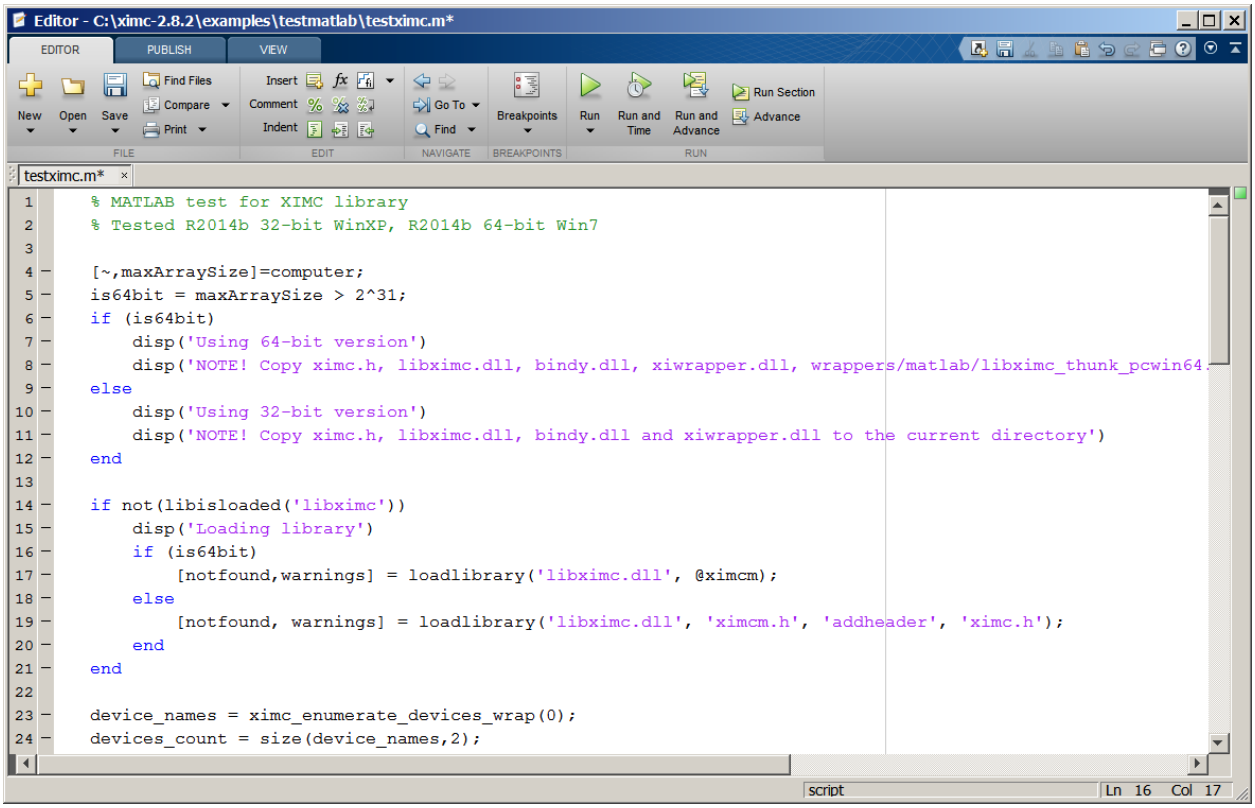
Примечание: Библиотека libximc открывает контроллеры в режиме эксклюзивного доступа. Любой контроллер, открытый с помощью libximc, необходимо закрыть, прежде чем он может быть использован другими процессами. Не останавливайте работу примера LabView или любой другой LabView программы использующей libximc с помощью кнопки «Abort execution» - это не дает возможности программе вызвать функцию close_device(), поэтому в этом случае все открытые в LabView контроллеры будут заблокированы до полного закрытия среды LabView.

Предупреждение: Поддержка контроллеров 8DCMC1 и 8SMC1 с 2016 года прекращена компанией STANDA. Примеры (в частности для LabVIEW) для этих контроллеров обновляться не будут!

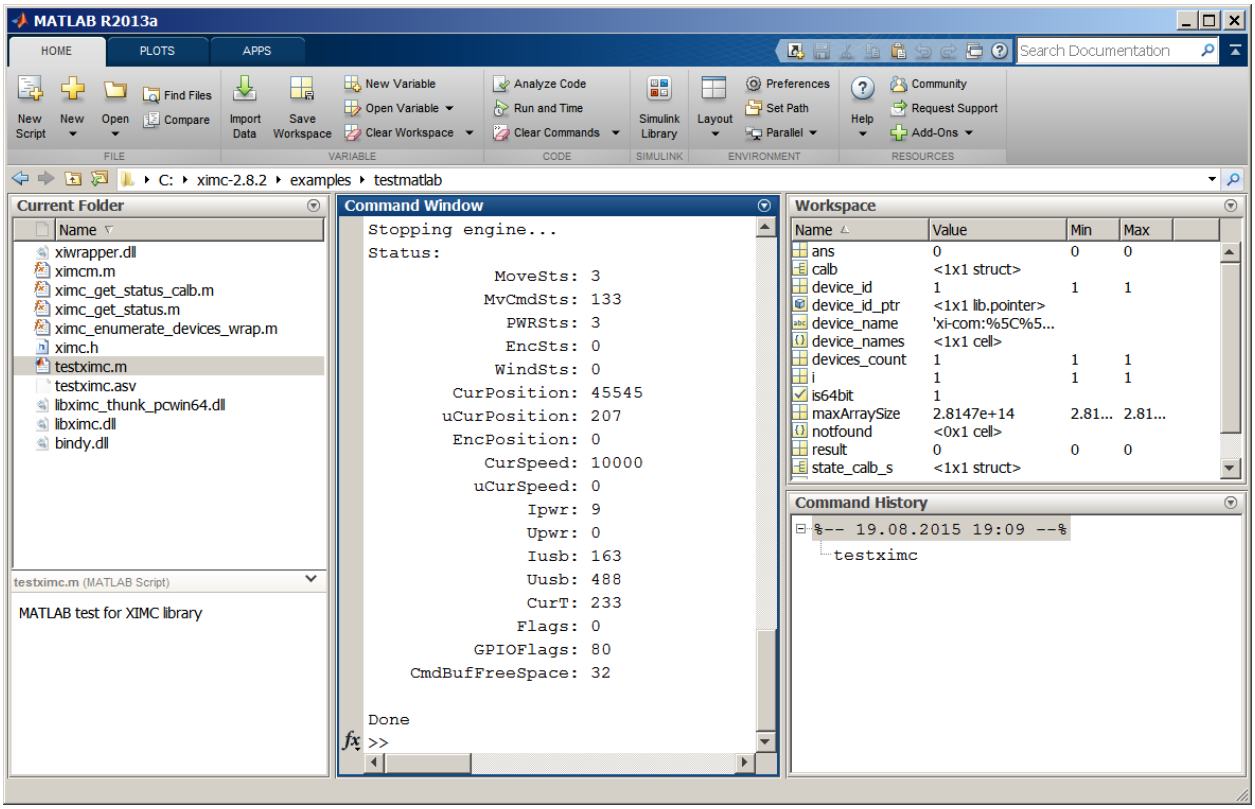
6.1.4 Работа с контроллером в среде Matlab

Библиотека libximc может использоваться для работы с контроллерами моторов в среде Matlab.

Примечание: Для работы с SDK требуется Microsoft Visual C++ Redistributable Package 2013 (поставляется с SDK, файлы vcredist_x86 или vcredist_x64).



Распакуйте архив комплекта разработчика библиотеки libximc, скопируйте **ximc.h**, **win64\libximc.dll**, **win64\bindy.dll**, **win64\xiwrapper.dll**, **win64\wrappers\matlab\libximc_thunk_pcwin64.dll** и **win64\wrappers\matlab\ximc.m** в директорию **examples\testmatlab** если вы используете 64-битную среду или **ximc.h**, **win32\libximc.dll**, **win32\bindy.dll**, **win32\xiwrapper.dll** и **win32\wrappers\matlab\ximcm.h** если вы используете 32-битную. Подключите контроллер моторов к компьютеру и запустите пример testximc.m.



В окне выполненных команд вы увидите результат опроса контроллеров.

В окне выполненных команд вы увидите результат опроса контроллеров. Вызов функций libximc из Matlab-программы производится следующим образом: определите путь к библиотекам **libximc.dll**, **bindy.dll**, **xiwrapper.dll** и заголовочному файлу **ximc.h**, а в случае 64-битной среды ещё к **libximc_thunk_pcwin64.dll** и **ximc.m**. Однократно используйте функцию Matlab loadlibrary чтобы загрузить библиотеку libximc и далее используйте функцию **calllib** чтобы вызвать необходимую функцию libximc. Список имен функций libximc, принимаемые и возвращаемые ими параметры вы можете найти в *Руководстве по программированию*.

6.1.5 Работа с контроллером в ScanImage

Библиотека libximc может использоваться для работы с контроллерами моторов в ScanImage.

ScanImage это открытое программное обеспечение для лазерной микроскопии, электрофизиологии, лазерной сканирующей фотостимуляции и других физиологических методов, ориентированных преимущественно на нейробиологию.

Контроллеры Standa прямо сейчас могут быть использованы для точного лазерного сканирования. Вы можете загрузить сопутствующее программное обеспечение со страницы Программное обеспечение и присоединиться к одной из 200 лабораторий, использующих ScanImage.

Примечание: Сопутствующее ПО было протестировано на Windows 10 64 бит, Matlab 2017, ScanImage 2018 и DAQmx 2017.ScanImage 2018 и DAQmx 2017.

6.1.6 Работа с контроллером в среде LabWindows

Скачайте пример программы для LabWindows со страницы Программное обеспечение.

Архив содержит два примера для LabWindows: - Пример **testcli** - представляет собой консольное приложение для среды LabWindows CVI использующее библиотеку libximc. - Пример **testgui** - представляет собой приложение с User Interface Editor для среды LabWindows CVI использующее библиотеку libximc

---

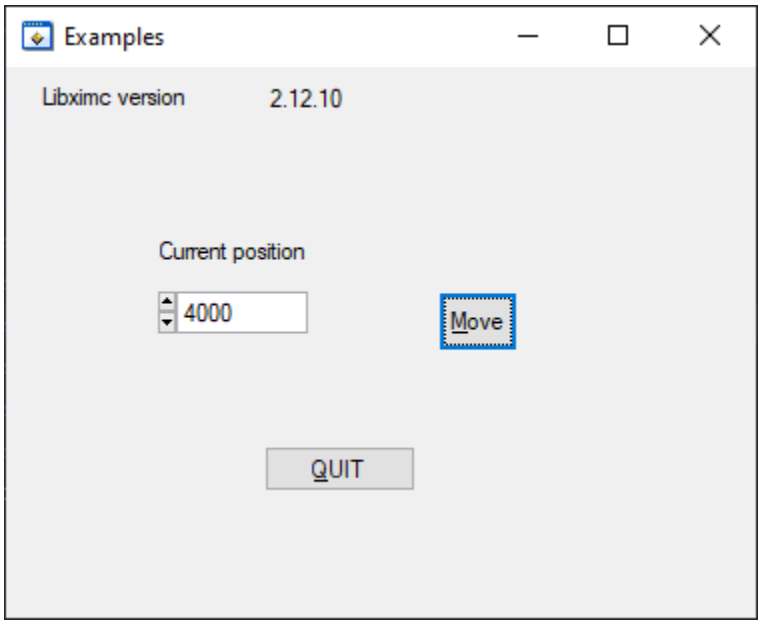
Важно: Работа примеров проверялась на версии LabWindows CVI 2017.

Откройте проект *..\examples\testlabwindows\testcli\testcli.prj* или *..\examples\testlabwindows\testgui\simple.prj* в LabWindows CVI. Чтобы скомпилировать пример, выполните команду *Build -> Rebuild*.

Скопируйте **..ximc\win64\libximc.dll**, **..ximc\win64\bindy.dll** и **..ximc\win64\xiwrapper.dll** в директорию, где лежит *.exe* файл.

Примечание: По умолчанию в примерах установлена 64-битная сборка, и к примеру подключена 64-битная версия библиотеки libximc. Если необходимо собрать 32-битную версию приложения выберите одну из 32-битных сборок в меню *Build -> Configuration*. Удалите 64-битную версию библиотеки libximc.lib из проекта и подключите 32-битную. Библиотека libximc с зависимостями находится в папках ximc/win32 и ximc/win64.

Программа **«simple.exe»**



После запуска откроется среда LabWindows. Вы увидите графический интерфейс. В верхней части выводится используемая версия libximc. В поле *«Current position»* отображается позиция в *шагах/с*. Изменив значение в поле и нажав кнопку *Move*, будет вызвана команда движения в позицию.

Примечание: Библиотека Libximc открывает контроллеры в режиме эксклюзивного доступа. Любой контроллер, открытый с помощью библиотеки libximc, должен быть закрыт, прежде чем он может быть использован другим процессом. Не останавливайте пример LabWindows или любую другую программу, использующую libximc по кнопке «Закрыть» - это не дает возможности программе вызвать функцию close_device(), поэтому в этом случае все открытые в Labwindows контроллеры будут заблокированы

до полного закрытия среды Labwindows.

Программа **«testcli-64.exe»**



Перед Вами консольная версия примера для LabWindows. Программа *«testcli»* сообщает версию используемой библиотеки, а также указывает, какой порт она удерживает. После этого программа *«testcli»* отправляет в контроллер команду command_movr. После успешного выполнения команды движения, вызывается команда close_device

Демонстрация работы примеров

https://youtu.be/iRab6qlWSo0

6.1.7 Краткое описание работы с поддерживаемыми языками программирования

- *Visual C++*
- *.NET (C# и Visual Basic)*
- *Delphi*
- *Matlab*
- *Java*
- *Python*
- *Python (Jupyter Notebook)*

Для приобретения первых навыков использования библиотеки создано простое тестовое приложение *testapp*. Языки, отличные от С-подобных, поддерживаются с помощью вызовов с преобразованием аргументов типа *stdcall*.

Простое тестовое приложение на языке C расположено в директории *«examples/test_C/testapp_C»*, проект на C# - в *«examples/test_CSharp»*, на VB.NET - в *«examples/test_VBNET»*, для Delphi 6 - в *«example/test_Delphi»*, для MATLAB - *«examples/test_MATLAB»*, для Java - *«examples/test_Java»*, для Python - *«examples/test_Python»*. Библиотеки, заголовочные файлы и другие необходимые файлы расположены в директориях *«win32»/»win64»*, **«MacOSX»** и подобных. В комплект разработчика также входят уже скомпилированные примеры: *testapp* и *testappeasy* в варианте **32** и **64 бита** под **Windows** и только **64 бита** под **OSX**, *test_CSharp*, *test_VBNET*, *test_Delphi* - **только 32 бита**, *test_Java* - кроссплатформенный, **test_MATLAB и test_Python не требуют компиляции**. Также руководство по программированию можно скачать по этой ссылке.

Примечание: Для работы с SDK требуется Microsoft Visual C++ Redistributable Package 2013 (поставляется с SDK, файлы vcredist_x86 или vcredist_x64)

6.1.7.1 Visual C++

Тестовое приложение может быть собрано с помощью *testapp.sln*. Для компиляции необходимо использовать также MS Visual C++, **mingw-library не поддерживается**. Убедитесь, что Microsoft Visual C++ Redistributable Package 2013 установлен.

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

6.1.7.2 .NET (C# и Visual Basic)

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

Тестовые приложения на языке C# для Visual Studio 2013 расположены в директориях *testcs* (*для C#*) и *testvbnet* (*для VB.NET*). Откройте проекты и соберите.

6.1.7.3 Delphi

Обертка для использования в Delphi *libximc.dll* предлагается как модуль *wrappers/pascal/ximc.pas*. Консольное тестовое приложение размещено в директории *«testdelphi»*. *Проверено с Delphi 6 на 32-битной системе*. Просто скомпилируйте, разместите DLL в директории с исполняемым модулем и запустите его.

6.1.7.4 Matlab

Тестовая программа на MATLAB *testximc.m* располагается в директории *examples/testmatlab*.

Перед запуском:

На OS X: скопируйте *ximc/macosx/libximc.framework*, *ximc/macosx/wrappers/ximcm.h*, *ximc/ximc.h* в директорию *examples/matlab*. Установите XCode, совместимый с Matlab.

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

Для проверки совместимых XCode и gcc проверьте документы или похожие.

На Windows: перед запуском ничего делать не нужно. Измените текущую директорию в MATLAB на *examples/matlab*. Затем запустите в MATLAB:

6.1.7.5 Java

Как запустить пример на Linux. Перейдите в *ximc-2.x.x/examples/testjava/compiled/* и выполните:

```
$ java -cp /usr/share/java/libjximc.jar:testjava.jar ru.ximc.TestJava
```

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

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

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

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

Затем пересоберите исходные тексты:

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

или для Windows или Mac:

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

Затем соберите jar:

```
$ jar cmf META-INF/MANIFEST.MF testjava.jar ru
```

6.1.7.6 Python

Измените текущую директорию на *examples/test_Python/xxxtest*.

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

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:`pwd`
```

На Windows и MacOS: перед запуском ничего делать не нужно. Запустите Python 2 или Python 3:

```
python xxxx.py
```

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

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

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

6.1.7.7 Python (Jupyter Notebook)

Мы добавили поддержку Jupyter Notebook, чтобы упростить программирование на Python. В приведенном ниже блокноте показано, как работать с контроллерами двигателей Standa 8SMC5 через привязку libximc python.

Если вы не знакомы с Jupyter Notebooks, обратитесь к официальной документации.

Открыть в Colab Открыть в HTML-формате

Примечание: По умолчанию этот блокнот использует виртуальный контроллер. Для демонстрации его работы реальный контроллер вам не потребуется, но если он у вас есть, вы также можете протестировать его с помощью этого блокнота.

Комплект разработчика можно скачать на странице программного обеспечения. Он включает в себя скомпилированную библиотеку libximc для систем Windows, Linux и Mac OS, руководство по программированию и примеры. Libximc - это кроссплатформенная библиотека, которая поддерживает языки C++, C#, Delphi, Visual Basic, Matlab, Java и Python. Примеры, включенные в пакет библиотеки, предназначены для быстрого ознакомления с программированием для контроллеров XIMC. Исходники Libximc также доступны для скачивания.

Примеры для работы с LabView предоставляются отдельно.

Внимание: Руководство по программированию включено в архив libximc 2.X.X, где 2.X.X - номер версии. Руководство находится в /ximc-2.X.X/ximc/doc-ru/libximc7-ru.pdf. Также руководство libximc можно скачать по этой ссылке. Руководство по программированию создано в системе Doxygen.

6.2 Описание протокола обмена

Описание протокола v20.8

- *Описание протокола*
- *Исполнение команд*
- *Обработка ошибок на стороне контроллера*
  - *Неверные команды или данные*
  - *Расчёт CRC*
  - *Сбои передачи*
  - *Восстановление синхронизации методом таймаута*
  - *Восстановление синхронизации методом очистительных нулей*

- *Обработка ошибок на стороне библиотеки*
  - *Возможные значения ответа библиотеки*
  - *Процедура синхронизации очистительными нулями*
- *Коды ошибок ответов контроллера*
  - *ERRC*
  - *ERRD*
  - *ERRV*
- *Все команды контроллера*
  - *Команда GACC*
  - *Команда GBRK*
  - *Команда GCAL*
  - *Команда GCTL*
  - *Команда GCTP*
  - *Команда GEAS*
  - *Команда GEDS*
  - *Команда GEIO*
  - *Команда GEMF*
  - *Команда GENG*
  - *Команда GENI*
  - *Команда GENS*
  - *Команда GENT*
  - *Команда GEST*
  - *Команда GFBS*
  - *Команда GGRI*
  - *Команда GGRS*
  - *Команда GHOM*
  - *Команда GHSI*
  - *Команда GHSS*
  - *Команда GJOY*
  - *Команда GMOV*
  - *Команда GMTI*
  - *Команда GMTS*
  - *Команда GNET*
  - *Команда GNME*
  - *Команда GNMF*

- *Команда GNVM*
- *Команда GPID*
- *Команда GPWD*
- *Команда GPWR*
- *Команда GSEC*
- *Команда GSNI*
- *Команда GSNO*
- *Команда GSTI*
- *Команда GSTS*
- *Команда GURT*
- *Команда SACC*
- *Команда SBRK*
- *Команда SCAL*
- *Команда SCTL*
- *Команда SCTP*
- *Команда SEAS*
- *Команда SEDS*
- *Команда SEIO*
- *Команда SEMF*
- *Команда SENG*
- *Команда SENI*
- *Команда SENS*
- *Команда SENT*
- *Команда SEST*
- *Команда SFBS*
- *Команда SGRI*
- *Команда SGRS*
- *Команда SHOM*
- *Команда SHSI*
- *Команда SHSS*
- *Команда SJOY*
- *Команда SMOV*
- *Команда SMTI*
- *Команда SMTS*
- *Команда SNET*

- *Команда SNME*
- *Команда SNMF*
- *Команда SNVM*
- *Команда SPID*
- *Команда SPWD*
- *Команда SPWR*
- *Команда SSEC*
- *Команда SSNI*
- *Команда SSNO*
- *Команда SSTI*
- *Команда SSTS*
- *Команда SURT*
- *Команда ASIA*
- *Команда CLFR*
- *Команда CONN*
- *Команда DBGR*
- *Команда DBGW*
- *Команда DISC*
- *Команда EERD*
- *Команда EESV*
- *Команда GBLV*
- *Команда GETC*
- *Команда GETI*
- *Команда GETM*
- *Команда GETS*
- *Команда GFWV*
- *Команда GOFW*
- *Команда GPOS*
- *Команда GSER*
- *Команда GUID*
- *Команда HASF*
- *Команда HOME*
- *Команда IRND*
- *Команда LEFT*
- *Команда LOFT*

– *Команда MOVE*
– *Команда MOVR*
– *Команда PWOF*
– *Команда RDAN*
– *Команда READ*
– *Команда RERS*
– *Команда REST*
– *Команда RIGT*
– *Команда SARS*
– *Команда SAVE*
– *Команда SPOS*
– *Команда SSER*
– *Команда SSTP*
– *Команда STMS*
– *Команда STOP*
– *Команда UPDF*
– *Команда WDAT*
– *Команда WKEY*
– *Команда ZERO*

6.2.1 Описание протокола

Управление контроллером с ПК происходит по интерфейсу последовательного порта (COM-порт). На стороне контроллера жёстко установлены следующие параметры COM-порта:

- Скорость – 115200 бод
- Длина кадра – 8 бит
- Стоп-биты – 2 бита
- Чётность – нет
- Контроль потока – нет (Xon/Xoff, CTS/RTS не используются)
- Таймаут на получение, между байтами одного пакета – 400 мсек
- Порядок следования бит – LittleEndian
- Многобайтовые типы данных передаются младшим байтом вперёд

6.2.2 Исполнение команд

Базовый принцип протокола - «Запрос-Ответ», причём все обмены данными инициируются ПК, т.е. ПК посылает команды в контроллер, но не наоборот. Каждая команда подразумевает получение ответа от контроллера (кроме редких случаев специальных команд), т.е. нельзя послать несколько команд подряд, без ожидания ответа на них.

Все команды делятся на сервисные, штатные управляющие и штатные информационные. Команды выполняются сразу после их поступления в контроллер. Установленные командой SXXX параметры начинают влиять на текущее движение в течение 1 мс после установки. Обработка команды не влияет на своевременность выполнения контроллером действий связанных с оперативным управление и контролем двигателя (работа ШИМ, взаимодействие с энкодером и т.п.).

И контроллер и ПК обладают буфером обмена. Принятые команды и данные, в случае их наличия в команде, обрабатываются один раз. То есть, после обработки эти данные удаляются из буфера и обрабатываются уже новые пришедшие байты. Каждая команда состоит из четырёхбайтной строки, данных (если команда их предусматривает) и двухбайтного кода контроля CRC если команда содержит данные. Данные могут пересылаться как из компьютера, так и контроллером. Команда передаётся на обработку если она распознана и, в случае передачи данных, код CRC верный. После обработки пришедшей без ошибок команды контроллер посылает в компьютер четырехбайтную строку – наименование выполненной команды, затем данные, если формат команды это предусматривает, затем два байта CRC (если есть данные).

6.2.3 Обработка ошибок на стороне контроллера

6.2.3.1 Неверные команды или данные

Если пришедшая в контроллер команда не может быть интерпретирована, как определенная команда управления, то в компьютер посылается строка «errc», команда игнорируется, в данных текущего состояния контроллера выставляется бит «команда не распознана». Если неопознанная команда содержала данные, то возможно неверная интерпретация принятых данных как новых команд. Необходима синхронизация.

Если пришедшая в контроллер команда интерпретирована верно, команда предусматривала данные, они пришли, но два байта CRC не соответствует полученным с ней данным, то в данных текущего состояния контроллера устанавливается флаг ошибки CRC пришедших данных, в компьютер посылается строка «errd», текущая команда игнорируется. Синхронизация приёма/передачи с компьютером не нужна.

6.2.3.2 Расчёт CRC

CRC рассчитывается для передаваемых данных. Четыре байта команды в расчёте не участвуют. Алгоритм CRC на языке Си:

```
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;
}
```

Функция получает указатель на массив данных pbuf, длину данных в байтах n. Функция возвращает двубайтное слово - код CRC.

Пример расчёта CRC:

Код команды (CMD): «home» или 0x656D6F68

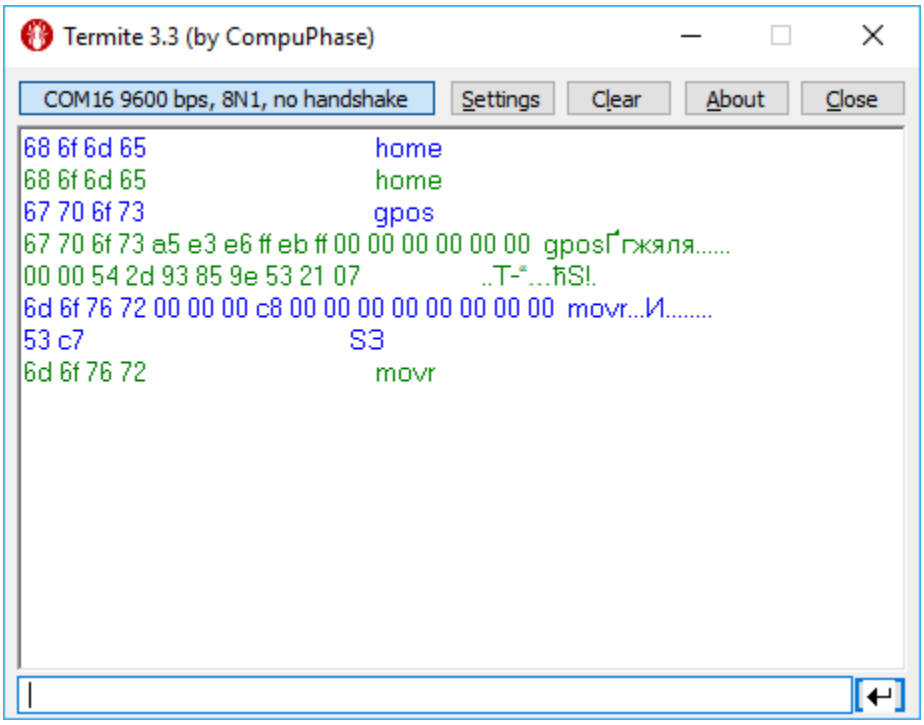
```
0x68 0x6F 0x6D 0x65
CMD
```

Код команды (CMD): «gpos» или 0x736F7067

```
0x67 0x70 0x6F 0x73
CMD
```

Код команды (CMD): «movr» или 0x72766F6D

```
0x6D 0x6F 0x76 0x72  0x00 0x00 0x00 0xC8  0x00 0x00  0x00 0x00 0x00 0x00 0x00 0x00  0x53 0xc7
CMD                  DeltaPosition        uDPos      Reserved                       CRC
```



6.2.3.3 Сбои передачи

Наиболее вероятны следующие сбои в канале связи: исчезновение байта при приёме или передаче контроллером, возникновение лишнего байта при приёме или передаче контроллером и изменение принятого или посланного байта.Сбои происходят при нестандартных условиях и обычно не наблюдаются вообще.

Регулярные сбои возможны при некачественном, сломанном кабеле USB или соединительном кабеле между платами. Протокол не разрабатывался для штатного применения в условиях сильно нестабильной связи. В частности в таких условиях редко возможно выполнение не той команды, что была послана.

Исчезновение байта на стороне контроллера

Байт, ожидаемый, но не полученный контроллером, приводит к таймауту компьютера. Посылка команды считается компьютером неуспешной. На этот момент синхронизация передачи данных будет

нарушена, но восстановится по таймауту (если таймаут контроллера меньше таймаута компьютера с учётом времени пересылки).

Исчезновение байта на стороне компьютера

Байт, не полученный компьютером, приводит к таймауту компьютера. Синхронизация не нарушена.

Возникновение байта на стороне контроллера

Лишний байт, возникший при приёме контроллером, приводит к получению компьютером одного или нескольких «errc» либо «errd» (очень редко сочетания «errc» и «errd»). Посылка команды считается неуспешной. В приёмном буфере компьютера может появиться несколько «errc» или «errd» ответов контроллера. На этот момент синхронизация нарушена.

Возникновение байта на стороне компьютера

Байт, возникший при приёме компьютером, приводит к неверно принятой команде или неверному коду CRC. Кроме того, в приёмном буфере останется лишний байт. На этот момент синхронизация нарушена.

Изменение байта на стороне контроллера

Байт, изменившийся при приёме контроллером, приводит к получению компьютером одного или нескольких «errc» либо «errd» (очень редко сочетания «errc» и «errd»). Посылка команды считается неуспешной. В приёмном буфере компьютера может появиться несколько «errc» либо «errd» ответов контроллера. Обычно синхронизация не нарушается, но редко она может быть нарушена.

Изменение байта на стороне компьютера

Байт, изменившийся при приёме компьютером, приводит к неверно принятой команде или неверному коду CRC. На этот момент синхронизация не нарушена.

6.2.3.4 Восстановление синхронизации методом таймаута

Если при получении пакета, время между получением одного или нескольких байт выходит за рамки таймаута, то полученные данные игнорируются, входной буфер очищается. Время таймаута контроллера должно быть меньше таймаута компьютера с учетом погрешности на время пересылки.

6.2.3.5 Восстановление синхронизации методом очистительных нулей

Ни одна команда не начинается нулём („\0“). Поэтому возможен такой метод синхронизации: контроллер на каждый полученный первый байт команды равный нулю отвечает нулём, а компьютер игнорирует первые байт ответа если он равен нулю и переходит к рассмотрению следующего. Тогда в случая когда синхронизация нарушена на стороне компьютера или контроллера, но еще не прошло время таймаута контроллера, возможен следующий алгоритм:

Если компьютером в ответ на переданную команду с данными или без, получен от контроллера ответ не на ту команду, «errc» либо «errd» , то с компьютера в контроллер средствами библиотеки посылается от 4 до 250 нулей (ограничение в 250 байт связано с длиной приёмного буфера и протоколом передачи данных по I2C, а передача менее 4 нулей часто не приведёт к восстановлению синхронизации). При этом происходит постоянное считывание приходящих байт от контроллера до появления первого нуля. После этого и считывание и посылка прекращаются.

Принятый нуль обычно не является частью предыдущей передачи, так как в моменты ошибок контроллер получает ответы «errc» / «errd». В редких случаях (особое изменение байта на стороне контроллера) возможна синхронизация с некоторой попытки. Таким образом, приход первого нуля обычно означает, что приёмный буфер контроллера чист и уже не заполнится, пока не придёт первая значимая команда. Сразу после прихода первого нуля от контроллера компьютер готов передавать следующую команду. Остальные нули, находящиеся в пересылке, будут проигнорированы, так как придут до ответа контроллера.

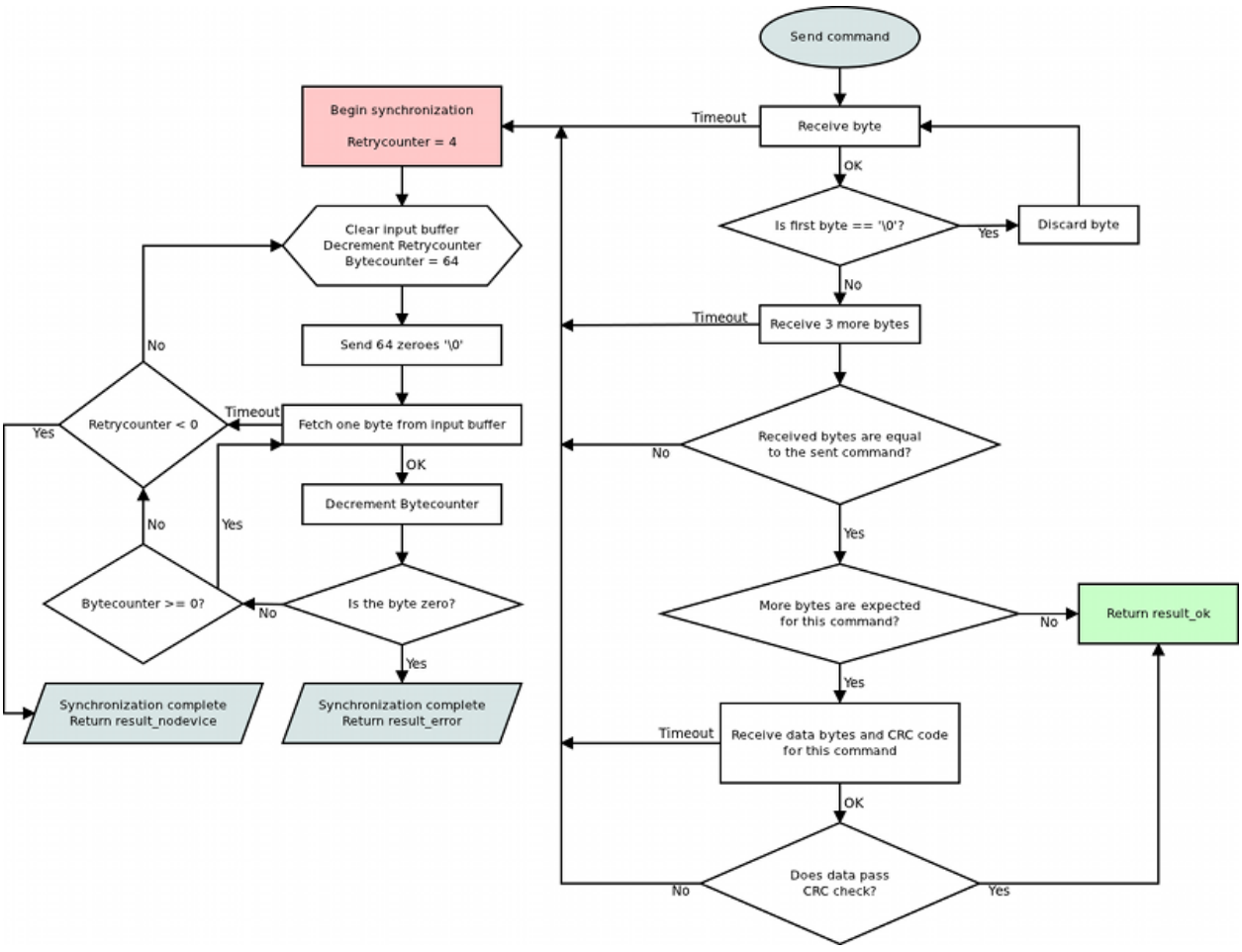
Синхронизация завершена.

6.2.4 Обработка ошибок на стороне библиотеки

Практически каждая функция библиотеки возвращает статус выполнения типа *result_t*.

После посылки запроса контроллеру библиотека проверяет первые приходящие байты пока не встретит первое ненулевое значение. Все нулевые байты игнорируются. Остальные приходящие байты считаются значимыми. Библиотека ожидает первые 4 байта ответа. Далее она сравнивает их с кодом запроса и, при необходимости, ожидает остальные байты пакета данных. Если полученные 4 байта не соответствуют запросу, то запускается процедура синхронизации очистительными нулями, команда выполнена неуспешно. Если полученные первые 4 байта совпадают с кодом запроса и в ответе есть еще данные, то после их получения проверяется CRC код. Если код неверный, то запускается синхронизация очистительными нулями, выполнение команды считается неуспешным.

Если ошибок не обнаружено, то команда считается выполненной успешно и возвращается *result_ok*.



6.2.4.1 Возможные значения ответа библиотеки

- *result_ok*. Ошибок нет.
- *result_error*. Общая ошибка. Может быть связана с аппаратными проблемами, отсутствием данных в буфере порта, превышением таймаутов. Также может означать сбой синхронизации, который был устранён. Такой сбой мог быть вызван помехами на линии связи с контроллером. Еще одной причиной может быть несоответствие протоколов в прошивке и в контроллере.

- *result_nodevice*. Невозможность открытия устройства, потеря связи с ним в процессе передачи данных, неудачная синхронизация. Требуется повторное открытие устройства или вмешательство пользователя.

Если функция возвращает ошибку, любые переданные в неё структуры для записи считаются неопределёнными. Возврат кода ошибки может сопровождаться записью подробного сообщения в системный лог на unix или в stderr на windows.

6.2.4.2 Процедура синхронизации очистительными нулями

Восстановление синхронизации осуществляется посылкой нулевых байтов и считывания принимаемых байт до появления первого нулевого значения („\0“). Опционально можно в конце синхронизации очистить буфер порта. Посылается изначально 64 нулевых байта. Если от контроллера не пришло ни одного нулевого байта за время таймаута, то 64 байта посылаются еще 3 раза. После 4 посылки и неполучения нулевого байта устройство считается потерянным и библиотека должна вернуть код ошибки *result_nodevice*. В случае удачной синхронизации возвращаемый код ошибки *result_error*.

6.2.5 Коды ошибок ответов контроллера

6.2.5.1 ERRC

Ответ: (4 байт)

Код: «errc» или 0x63727265

uint32_t	errc	Команда недоступна

Описание

Ответ на команду, в случае если команда неизвестна, либо не может быть выполнена и/или обработана в данный момент (в данном состоянии). Устанавливает соответствующий бит в поле «flags» структуры состояния.

6.2.5.2 ERRD

Ответ: (4 байт)

Код: «errd» или 0x64727265

uint32_t	errd	Неверные данные

Описание

Ответ на команду «errd», устанавливается в том случае, если вычислиные контроллером данные CRC не совпадают с полученым полем CRC. В этом случае устанавливается бит соответствия в поле «flags» структуры состояния.

6.2.5.3 ERRV

Ответ: (4 байт)

Код: «errv» или 0x76727265

uint32_t	errv	Неверное значение

Описание

Ответ на команду, в случае если команда корректна, контрольная сумма правильная, но передаваемые значения (хотя бы одно из них) выходят за допустимый диапазон и не могут быть приняты. При этом неверное значение заменяется одним из верных методами округления, ограничения или сбрасывания в некое стандартное состояние. Устанавливает соответствующий бит в поле «flags» структуры состояния.

6.2.6 Все команды контроллера

6.2.6.1 Команда GACC

Код команды (CMD): «gacc» или 0x63636167.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (114 байт)

uint32_t	CMD	Команда
int8_t	MagneticBrakeInfo	Производитель и номер магнитного тормоза, Максимальная длина строки: 24 символов.
float	MBRatedVoltage	Номинальное напряжение для управления магнитным тормозом (В). Тип данных: float.
float	MBRatedCurrent	Номинальный ток для управления магнитным тормозом (А). Тип данных: float.
float	MBTorque	Удерживающий момент (мН м). Тип данных: float.
uint32_t	MBSettings	Флаги настроек магнитного тормоза. Это битовая маска для побитовых операций.
	0x1 - MB_AVAILABLE	Если флаг установлен, то магнитный тормоз доступен
	0x2 - MB_POWERED_HOLD	Если флаг установлен, то магнитный тормоз находится в режиме удержания (активен) при подаче питания
int8_t	TemperatureSensorInfo	Производитель и номер температурного датчика, Максимальная длина строки: 24 символов.
float	TSMin	Минимальная измеряемая температура (град Цельсия). Тип данных: float.
float	TSMax	Максимальная измеряемая температура (град Цельсия) Тип данных: float.
float	TSGrad	Температурный градиент (В/град Цельсия). Тип данных: float.

Continued on next page

Таблица 6.7 – continued from previous page

uint32_t	TSSettings	Флаги настроек температурного датчика. Это битовая маска для побитовых операций.
	0x7 - TS_TYPE_BITS	Биты, отвечающие за тип температурного датчика
	0x0 - TS_TYPE_UNKNOWN	Неизвестный сенсор
	0x1 - TS_TYPE_THERMOCOUPLE	Термопара
	0x2 - TS_TYPE_SEMICONDUCTOR	Полупроводниковый температурный датчик
	0x8 - TS_AVAILABLE	Если флаг установлен, то датчик температуры доступен
uint32_t	LimitSwitchesSettings	Флаги настроек концевых выключателей. Это битовая маска для побитовых операций.
	0x1 - LS_ON_SW1_AVAILABLE	Если флаг установлен, то концевой переключатель, подключенный к ножке SW1, доступен
	0x2 - LS_ON_SW2_AVAILABLE	Если флаг установлен, то концевой переключатель, подключенный к ножке SW2, доступен
	0x4 - LS_SW1_ACTIVE_LOW	Если флаг установлен, то концевой переключатель, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте
	0x8 - LS_SW2_ACTIVE_LOW	Если флаг установлен, то концевой переключатель, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте
	0x10 - LS_SHORTED	Если флаг установлен, то концевые переключатели замкнуты
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение информации о дополнительных аксессуарах из EEPROM.

6.2.6.2 Команда GBRK

Код команды (CMD): «gbrk» или 0x6B726267.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (25 байт)

uint32_t	CMD	Команда
uint16_t	t1	Время в мс между включением питания мотора и отключением тормоза.

Continued on next page

Таблица 6.9 – continued from previous page

uint16_t	t2	Время в мс между отключением тормоза и готовностью к движению. Все команды движения начинают выполняться только по истечении этого времени.
uint16_t	t3	Время в мс между остановкой мотора и включением тормоза.
uint16_t	t4	Время в мс между включением тормоза и отключением питания мотора.
uint8_t	BrakeFlags	Флаги. Это битовая маска для побитовых операций.
	0x1 - BRAKE_ENABLED	Управление тормозом включено, если флаг установлен.
	0x2 - BRAKE_ENG_PWROFF	Тормоз отключает питание шагового мотора, если флаг установлен.
uint8_t	Reserved [10]	Зарезервировано (10 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек управления тормозом.

6.2.6.3 Команда GCAL

Код команды (CMD): «gcal» или 0x6C616367.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (118 байт)

uint32_t	CMD	Команда
float	CSS1_A	Коэффициент масштабирования для аналоговых измерений тока в обмотке A.
float	CSS1_B	Коэффициент сдвига для аналоговых измерений тока в обмотке A.
float	CSS2_A	Коэффициент масштабирования для аналоговых измерений тока в обмотке B.
float	CSS2_B	Коэффициент сдвига для аналоговых измерений тока в обмотке B.
float	FullCurrent_A	Коэффициент масштабирования для аналоговых измерений полного тока.
float	FullCurrent_B	Коэффициент сдвига для аналоговых измерений полного тока.
uint8_t	Reserved [88]	Зарезервировано (88 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда чтения калибровочных коэффициентов. Команда только для производителя. Эта функция заполняет структуру калибровочных коэффициентов. Эти коэффициенты используются для

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

6.2.6.4 Команда GCTL

Код команды (CMD): «gctl» или 0x6C746367.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (93 байт)

uint32_t	CMD	Команда
uint32_t	MaxSpeed	Массив скоростей (в полных шагах), использующийся при управлении джойстиком или кнопками влево/вправо. Диапазон: 0..100000.
uint8_t	uMaxSpeed	Массив скоростей (в микрошагах), использующийся при управлении джойстиком или кнопками влево/вправо. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint16_t	Timeout	Timeout[i] - время в мс, по истечении которого устанавливается скорость max_speed[i+1] (используется только при управлении кнопками).
uint16_t	MaxClickTime	Максимальное время клика (в мс). До истечения этого времени первая скорость не включается.
uint16_t	Flags	Флаги. Это битовая маска для побитовых операций.
	0x3 - CONTROL_MODE_BITS	Биты управления мотором с помощью джойстика или кнопок влево/вправо.
	0x0 - CONTROL_MODE_OFF	Управление отключено.
	0x1 - CONTROL_MODE_JOY	Управление с помощью джойстика.
	0x2 - CONTROL_MODE_LR	Управление с помощью кнопок влево/вправо.
	0x4 - CONTROL_BTN_LEFT_PUSHED_OPEN	Нажатая левая кнопка соответствует открытому контакту, если этот флаг установлен.
	0x8 - CONTROL_BTN_RIGHT_PUSHED_OPEN	Нажатая правая кнопка соответствует открытому контакту, если этот флаг установлен.

Continued on next page

Таблица 6.13 – continued from previous page

int32_t	DeltaPosition	Смещение (дельта) позиции (в полных шагах)
int16_t	uDeltaPosition	Дробная часть смещения в микрошагах. Используется только с шаговым двигателем. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint8_t	Reserved [9]	Зарезервировано (9 байт)
uint16_t	CRC	Контрольная сумма

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

6.2.6.5 Команда GCTP

Код команды (CMD): «gctp» или 0x70746367.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (18 байт)

uint32_t	CMD	Команда
uint8_t	CTPMinError	Минимальное отличие шагов ШД от положения энкодера, устанавливающее флаг STATE_RT_ERROR. Измеряется в шагах ШД.
uint8_t	CTPFlags	Флаги. Это битовая маска для побитовых операций.
	0x1 - CTP_ENABLED	Контроль позиции включен, если флаг установлен.
	0x2 - CTP_BASE	Управление положением основано на датчике вращения, если установлен этот флаг; в противном случае - на энкодере.
	0x4 - CTP_ALARM_ON_ERROR	Войти в состояние ALARM при расхождении позиции, если флаг установлен.

Continued on next page

Таблица 6.15 – continued from previous page

	0x8 - REV_SENS_INV	Сенсор считается активным, когда на нём 0, инвертирование делает активным уровень 1. То есть если не инвертировать, то действует обычная логика - 0 это срабатывание/активация/активное состояние.
	0x10 - CTP_ERROR_CORRECTION	Корректировать ошибки, возникающие при проскальзывании, если флаг установлен. Работает только с энкодером. Несовместимо с флагом CTP_ALARM_ON_ERROR.
uint8_t	Reserved [10]	Зарезервировано (10 байт)
uint16_t	CRC	Контрольная сумма

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

6.2.6.6 Команда GEAS

Код команды (CMD): «geas» или 0x73616567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (54 байт)

uint32_t	CMD	Команда
uint16_t	stepcloseloop_Kw	Коэффициент смешения реальной и заданной скорости, диапазон [0, 100], значение по умолчанию 50.
uint16_t	stepcloseloop_Kp_low	Обратная связь по позиции в зоне малых скоростей, диапазон [0, 65535], значение по умолчанию 1000.
uint16_t	stepcloseloop_Kp_high	Обратная связь по позиции в зоне больших скоростей, диапазон [0, 65535], значение по умолчанию 33.
uint8_t	Reserved [42]	Зарезервировано (42 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение расширенных настроек.

6.2.6.7 Команда GEDS

Код команды (CMD): «geds» или 0x73646567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (26 байт)

uint32_t	CMD	Команда
uint8_t	BorderFlags	Флаги, определяющие тип границ и поведение мотора при их достижении. Это битовая маска для побитовых операций.
	0x1 - BORDER_IS_ENCODER	Если флаг установлен, границы определяются предустановленными точками на шкале позиции. Если флаг сброшен, границы определяются концевыми выключателями.
	0x2 - BORDER_STOP_LEFT	Если флаг установлен, мотор останавливается при достижении левой границы.
	0x4 - BORDER_STOP_RIGHT	Если флаг установлен, мотор останавливается при достижении правой границы.
	0x8 - BORDERS_SWAP_MISSET_DETECTION	Если флаг установлен, мотор останавливается при достижении обоих границ. Нужен для предотвращения поломки двигателя при неправильных настройках концевых выключателей
uint8_t	EnderFlags	Флаги, определяющие настройки концевых выключателей. Это битовая маска для побитовых операций.
	0x1 - ENDER_SWAP	Если флаг установлен, первый концевой выключатель находится справа; иначе - слева.
	0x2 - ENDER_SW1_ACTIVE_LOW	1 - Концевой переключатель, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте.
	0x4 - ENDER_SW2_ACTIVE_LOW	1 - Концевой переключатель, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте.
int32_t	LeftBorder	Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.

Continued on next page

Таблица 6.19 – continued from previous page

int16_t	uLeftBorder	Позиция левой границы в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
int32_t	RightBorder	Позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER.
int16_t	uRightBorder	Позиция правой границы в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек границ и концевых выключателей.

6.2.6.8 Команда GEIO

Код команды (CMD): «geio» или 0x6F696567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (18 байт)

uint32_t	CMD	Команда
uint8_t	EXTIOSetupFlags	Флаги настройки работы внешнего ввода-вывода. Это битовая маска для побитовых операций.
	0x1 - EXTIO_SETUP_OUTPUT	Если флаг установлен, то ножка в состоянии вывода, иначе - ввода.
	0x2 - EXTIO_SETUP_INVERT	Если флаг установлен, то нули считаются активным состоянием выхода, а спадающие фронты как момент подачи входного сигнала.
uint8_t	EXTIOModeFlags	Флаги настройки режимов внешнего ввода-вывода. Это битовая маска для побитовых операций.
	0xf - EXTIO_SETUP_MODE_IN_BITS	Биты, отвечающие за поведение при переходе сигнала в активное состояние.
	0x0 - EXTIO_SETUP_MODE_IN_NOP	Ничего не делать.

Continued on next page

Таблица 6.21 – continued from previous page

	0x1 - EXTIO_SETUP_MODE_IN_STOP	По переднему фронту входного сигнала делается остановка двигателя (эквивалент команды STOP).
	0x2 - EXTIO_SETUP_MODE_IN_PWOF	Выполняет команду PWOF, обесточивая обмотки двигателя.
	0x3 - EXTIO_SETUP_MODE_IN_MOVR	Выполняется команда MOVR с последними настройками.
	0x4 - EXTIO_SETUP_MODE_IN_HOME	Выполняется команда HOME.
	0x5 - EXTIO_SETUP_MODE_IN_ALARM	Войти в состояние ALARM при переходе сигнала в активное состояние.
	0xf0 - EXTIO_SETUP_MODE_OUT_BITS	Биты выбора поведения на выходе.
	0x0 - EXTIO_SETUP_MODE_OUT_OFF	Ножка всегда в неактивном состоянии.
	0x10 - EXTIO_SETUP_MODE_OUT_ON	Ножка всегда в активном состоянии.
	0x20 - EXTIO_SETUP_MODE_OUT_MOVING	Ножка находится в активном состоянии при движении.
	0x30 - EXTIO_SETUP_MODE_OUT_ALARM	Ножка находится в активном состоянии при нахождении в состоянии ALARM.
	0x40 - EXTIO_SETUP_MODE_OUT_MOTOR_ON	Ножка находится в активном состоянии при подаче питания на обмотки.
uint8_t	Reserved [10]	Зарезервировано (10 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда чтения параметров настройки режимов внешнего ввода/вывода.

6.2.6.9 Команда GEMF

Код команды (CMD): «gemf» или 0x666D6567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (48 байт)

uint32_t	CMD	Команда
float	L	Индуктивность обмоток двигателя.
float	R	Сопротивление обмоток двигателя.
float	Km	Электромеханический коэффициент двигателя.
uint8_t	BackEMFFlags	Флаги автонастроек шагового двигателя. Это битовая маска для побитовых операций.
	0x1 - BACK_EMF_INDUCTANCE_AUTO	Флаг автоопределения индуктивности обмоток двигателя.
	0x2 - BACK_EMF_RESISTANCE_AUTO	Флаг автоопределения сопротивления обмоток двигателя.

Continued on next page

Таблица 6.23 – continued from previous page

	0x4 - BACK_EMF_KM_AUTO	Флаг автоопределения электромеханического коэффициента двигателя.
uint8_t	Reserved [29]	Зарезервировано (29 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение электромеханических настроек шагового двигателя. Настройки различны для разных двигателей.

6.2.6.10 Команда GENG

Код команды (CMD): «geng» или 0x676E6567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (34 байт)

uint32_t	CMD	Команда
uint16_t	NomVoltage	Номинальное напряжение мотора в десятках мВ. Контроллер будет сохранять напряжение на моторе не выше номинального, если установлен флаг ENGINE_LIMIT_VOLT (используется только с DC двигателем).
uint16_t	NomCurrent	Номинальный ток через мотор (в мА). Ток стабилизируется для шаговых и может быть ограничен для DC(если установлен флаг ENGINE_LIMIT_CURR). Диапазон: 15..8000
uint32_t	NomSpeed	Номинальная (максимальная) скорость (в целых шагах/с или rpm для DC и шагового двигателя в режиме ведущего энкодера). Контроллер будет сохранять скорость мотора не выше номинальной, если установлен флаг ENGINE_LIMIT_RPM. Диапазон: 1..100000.
uint8_t	uNomSpeed	Микрошаговая часть номинальной скорости мотора (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

Continued on next page

Таблица 6.25 – continued from previous page

uint16_t	EngineFlags	Флаги, управляющие работой мотора. Это битовая маска для побитовых операций.
	0x1 - ENGINE_REVERSE	Флаг реверса. Связывает направление вращения мотора с направлением счета текущей позиции. При сброшенном флаге (по умолчанию) прикладываемое к мотору положительное напряжение увеличивает счетчик позиции. И наоборот, при установленном флаге счетчик позиции увеличивается, когда к мотору приложено отрицательное напряжение. Измените состояние флага, если положительное вращение мотора уменьшает счетчик позиции.
	0x2 - ENGINE_CURRENT_AS_RMS	Флаг интерпретации значения тока. Если флаг снят, то задаваемое значение тока интерпретируется как максимальная амплитуда тока. Если флаг установлен, то задаваемое значение тока интерпретируется как среднеквадратичное значение тока (для шагового) или как значение тока, посчитанное из максимального тепловыделения (BLDC).
	0x4 - ENGINE_MAX_SPEED	Флаг максимальной скорости. Если флаг установлен, движение происходит на максимальной скорости.
	0x8 - ENGINE_ANTIPLAY	Компенсация люфта. Если флаг установлен, позиционер будет подходить к заданной точке всегда с одной стороны. Например, при подходе слева никаких дополнительных действий не совершается, а при подходе справа позиционер проходит целевую позицию на заданное расстояния и возвращается к ней опять же справа.
	0x10 - ENGINE_ACCEL_ON	Ускорение. Если флаг установлен, движение происходит с ускорением.
	0x20 - ENGINE_LIMIT_VOLT	Номинальное напряжение мотора. Если флаг установлен, напряжение на моторе ограничивается заданным номинальным значением(используется только с DC двигателем).

Continued on next page

Таблица 6.25 – continued from previous page

	0x40 - ENGINE_LIMIT_CURR	Номинальный ток мотора. Если флаг установлен, ток через мотор ограничивается заданным номинальным значением(используется только с DC двигателем).
	0x80 - ENGINE_LIMIT_RPM	Номинальная частота вращения мотора. Если флаг установлен, частота вращения ограничивается заданным номинальным значением.
int16_t	Antiplay	Количество шагов двигателя или импульсов энкодера, на которое позиционер будет отъезжать от заданной позиции для подхода к ней с одной и той же стороны. Используется, если установлен флаг ENGINE_ANTIPLAY.
uint8_t	MicrostepMode	Настройки микрошагового режима(используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Это битовая маска для побитовых операций.
	0x1 - MICROSTEP_MODE_FULL	Полношаговый режим.
	0x2 - MICROSTEP_MODE_FRAC_2	Деление шага 1/2.
	0x3 - MICROSTEP_MODE_FRAC_4	Деление шага 1/4.
	0x4 - MICROSTEP_MODE_FRAC_8	Деление шага 1/8.
	0x5 - MICROSTEP_MODE_FRAC_16	Деление шага 1/16.
	0x6 - MICROSTEP_MODE_FRAC_32	Деление шага 1/32.
	0x7 - MICROSTEP_MODE_FRAC_64	Деление шага 1/64.
	0x8 - MICROSTEP_MODE_FRAC_128	Деление шага 1/128.
	0x9 - MICROSTEP_MODE_FRAC_256	Деление шага 1/256.
uint16_t	StepsPerRev	Количество полных шагов на оборот(используется только с шаговым двигателем). Диапазон: 1..65535.
uint8_t	Reserved [12]	Зарезервировано (12 байт)
uint16_t	CRC	Контрольная сумма

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

6.2.6.11 Команда GENI

Код команды (CMD): «geni» или 0x696E6567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (70 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель. Максимальная длина строки: 16 символов.
int8_t	PartNumber	Серия и номер модели. Максимальная длина строки: 24 символа.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение информации об энкодере из EEPROM.

6.2.6.12 Команда GENS

Код команды (CMD): «gens» или 0x736E6567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (54 байт)

uint32_t	CMD	Команда
float	MaxOperatingFrequency	Максимальная частота (кГц). Тип данных: float.
float	SupplyVoltageMin	Минимальное напряжение питания (В). Тип данных: float.
float	SupplyVoltageMax	Максимальное напряжение питания (В). Тип данных: float.
float	MaxCurrentConsumption	Максимальное потребление тока (мА). Тип данных: float.
uint32_t	PPR	Количество отсчётов на оборот
uint32_t	EncoderSettings	Флаги настроек энкодера. Это битовая маска для побитовых операций.
	0x1 - ENCSET_DIFFERENTIAL_OUTPUT	Если флаг установлен, то энкодер имеет дифференциальный выход, иначе - несимметричный выход
	0x4 - ENCSET_PUSHPULL_OUTPUT	Если флаг установлен, то энкодер имеет двухтактный выход, иначе - выход с открытым коллектором
	0x10 - ENCSET_INDEXCHANNEL_PRESENT	Если флаг установлен, то энкодер имеет дополнительный индексный канал, иначе - он отсутствует
	0x40 - ENCSET_REVOLUTIONSENSOR_PRESENT	Если флаг установлен, то энкодер имеет датчик оборотов, иначе - он отсутствует

Continued on next page

Таблица 6.29 – continued from previous page

	0x100 - ENCSET_REVOLUTIONSENSOR_ACTIVE_HIGH	Если флаг установлен, то активное состояние датчика оборотов соответствует логической 1, иначе - логическому 0
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек энкодера из EEPROM.

6.2.6.13 Команда GENT

Код команды (CMD): «gent» или 0x746E6567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (14 байт)

uint32_t	CMD	Команда
uint8_t	EngineType	Тип мотора. Это битовая маска для побитовых операций.
	0x0 - ENGINE_TYPE_NONE	Это значение не нужно использовать.
	0x1 - ENGINE_TYPE_DC	Мотор постоянного тока.
	0x2 - ENGINE_TYPE_2DC	Два мотора постоянного тока, что приводит к эмуляции двух контроллеров.
	0x3 - ENGINE_TYPE_STEP	Шаговый мотор.
	0x4 - ENGINE_TYPE_TEST	Продолжительность включения фиксирована. Используется только производителем.
	0x5 - ENGINE_TYPE_BRUSHLESS	Бесщеточный мотор.
uint8_t	DriverType	Тип силового драйвера. Это битовая маска для побитовых операций.
	0x1 - DRIVER_TYPE_DISCRETE_FET	Силовой драйвер на дискретных мосфет-ключах. Используется по умолчанию.
	0x2 - DRIVER_TYPE_INTEGRATE	Силовой драйвер с использованием ключей, интегрированных в микросхему.
	0x3 - DRIVER_TYPE_EXTERNAL	Внешний силовой драйвер.
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

Описание: Возвращает информацию о типе мотора и силового драйвера.

6.2.6.14 Команда GEST

Код команды (CMD): «gest» или 0x74736567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (46 байт)

uint32_t	CMD	Команда
uint16_t	Param1	
uint8_t	Reserved [38]	Зарезервировано (38 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение расширенных настроек. В настоящее время не используется.

6.2.6.15 Команда GFBS

Код команды (CMD): «gfbs» или 0x73626667.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (18 байт)

uint32_t	CMD	Команда
uint16_t	IPS	Количество отсчётов энкодера на оборот вала. Диапазон: 1..65535. Поле устарело, рекомендуется записывать 0 в IPS и использовать расширенное поле CountsPerTurn. Может потребоваться обновление микропрограммы контроллера до последней версии.
uint8_t	FeedbackType	Тип обратной связи. Это битовая маска для побитовых операций.
	0x1 - FEEDBACK_ENCODER	Обратная связь с помощью энкодера.
	0x4 - FEEDBACK_EMF	Обратная связь по ЭДС.
	0x5 - FEEDBACK_NONE	Обратная связь отсутствует.
	0x6 - FEEDBACK_ENCODER_MEDIATED	Обратная связь по энкодеру, опосредованному относительно двигателя механической передачей (например, винтовой передачей).
uint8_t	FeedbackFlags	Флаги. Это битовая маска для побитовых операций.
	0x1 - FEEDBACK_ENC_REVERSE	Обратный счет у энкодера.
	0xc0 - FEEDBACK_ENC_TYPE_BITS	Биты, отвечающие за тип энкодера.
	0x0 - FEEDBACK_ENC_TYPE_AUTO	Определяет тип энкодера автоматически.
	0x40 - FEEDBACK_ENC_TYPE_SINGLE_ENDED	Недифференциальный энкодер.
	0x80 - FEEDBACK_ENC_TYPE_DIFFERENTIAL	Дифференциальный энкодер.

Continued on next page

Таблица 6.35 – continued from previous page

uint32_t	CountsPerTurn	Количество отсчётов энкодера на оборот вала. Диапазон: 1..4294967295. Для использования поля CountsPerTurn нужно записать 0 в поле IPS, иначе будет использоваться значение из поля IPS.
uint8_t	Reserved [4]	Зарезервировано (4 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек обратной связи.

6.2.6.16 Команда GGRI

Код команды (CMD): «ggri» или 0x69726767.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (70 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель. Максимальная длина строки: 16 символов.
int8_t	PartNumber	Серия и номер модели. Максимальная длина строки: 24 символа.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение информации о редукторе из EEPROM.

6.2.6.17 Команда GGRS

Код команды (CMD): «ggrs» или 0x73726767.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (58 байт)

uint32_t	CMD	Команда
float	ReductionIn	Входной коэффициент редуктора. (Выход = (ReductionOut/ReductionIn) вход) Тип данных: float.
float	ReductionOut	Выходной коэффициент редуктора. (Выход = (ReductionOut/ReductionIn) вход) Тип данных: float.

Continued on next page

Таблица 6.39 – continued from previous page

float	RatedInputTorque	Максимальный крутящий момент (Н м). Тип данных: float.
float	RatedInputSpeed	Максимальная скорость на входном валу редуктора (об/мин). Тип данных: float.
float	MaxOutputBacklash	Выходной люфт редуктора (градус). Тип данных: float.
float	InputInertia	Эквивалентная входная инерция редуктора(г см2). Тип данных: float.
float	Efficiency	КПД редуктора (%). Тип данных: float.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек редуктора из EEPROM.

6.2.6.18 Команда GHOM

Код команды (CMD): «ghom» или 0x6D6F6867.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (33 байт)

uint32_t	CMD	Команда
uint32_t	FastHome	Скорость первого движения (в полных шагах). Диапазон: 0..100000.
uint8_t	uFastHome	Дробная часть скорости первого движения в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint32_t	SlowHome	Скорость второго движения (в полных шагах). Диапазон: 0..100000.
uint8_t	uSlowHome	Дробная часть скорости второго движения в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
int32_t	HomeDelta	Расстояние отхода от точки останова (в полных шагах).

Continued on next page

Таблица 6.41 – continued from previous page

int16_t	uHomeDelta	Дробная часть расстояния отхода от точки останова в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint16_t	HomeFlags	Набор флагов, определяющие такие параметры, как направление и условия останова. Это битовая маска для побитовых операций.
	0x1 - HOME_DIR_FIRST	Определяет направление первоначального движения мотора после поступления команды HOME. Если флаг установлен - вправо; иначе - влево.
	0x2 - HOME_DIR_SECOND	Определяет направление второго движения мотора. Если флаг установлен - вправо; иначе - влево.
	0x4 - HOME_MV_SEC_EN	Если флаг установлен, реализуется второй этап доводки в домашнюю позицию; иначе - этап пропускается.
	0x8 - HOME_HALF_MV	Если флаг установлен, в начале второго движения первые пол оборота сигналы завершения движения игнорируются.
	0x30 - HOME_STOP_FIRST_BITS	Биты, отвечающие за выбор сигнала завершения первого движения.
	0x10 - HOME_STOP_FIRST_REV	Первое движение завершается по сигналу с Revolution sensor.
	0x20 - HOME_STOP_FIRST_SYN	Первое движение завершается по сигналу со входа синхронизации.
	0x30 - HOME_STOP_FIRST_LIM	Первое движение завершается по сигналу с концевого переключателя.
	0xc0 - HOME_STOP_SECOND_BITS	Биты, отвечающие за выбор сигнала завершения второго движения.
	0x40 - HOME_STOP_SECOND_REV	Второе движение завершается по сигналу с Revolution sensor.
	0x80 - HOME_STOP_SECOND_SYN	Второе движение завершается по сигналу со входа синхронизации.
	0xc0 - HOME_STOP_SECOND_LIM	Второе движение завершается по сигналу с концевого переключателя.
	0x100 - HOME_USE_FAST	Если флаг установлен, используется быстрый поиск домашней позиции; иначе - традиционный.
uint8_t	Reserved [9]	Зарезервировано (9 байт)

Continued on next page

Таблица 6.41 – continued from previous page

uint16_t	CRC	Контрольная сумма

Описание: Команда чтения настроек для подхода в home position. Эта функция заполняет структуру настроек, использующихся для калибровки позиции, в память контроллера.

6.2.6.19 Команда GHSI

Код команды (CMD): «ghsi» или 0x69736867.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (70 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель. Максимальная длина строки: 16 символов.
int8_t	PartNumber	Серия и номер модели. Максимальная длина строки: 24 символа.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение информации о датчиках Холла из EEPROM.

6.2.6.20 Команда GHSS

Код команды (CMD): «ghss» или 0x73736867.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (50 байт)

uint32_t	CMD	Команда
float	MaxOperatingFrequency	Максимальная частота (кГц). Тип данных: float.
float	SupplyVoltageMin	Минимальное напряжение питания (В). Тип данных: float.
float	SupplyVoltageMax	Максимальное напряжение питания (В). Тип данных: float.
float	MaxCurrentConsumption	Максимальное потребление тока (мА). Тип данных: float.
uint32_t	PPR	Количество отсчётов на оборот
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек датчиков Холла из EEPROM.

6.2.6.21 Команда GJOY

Код команды (CMD): «gjoy» или 0x796F6A67.

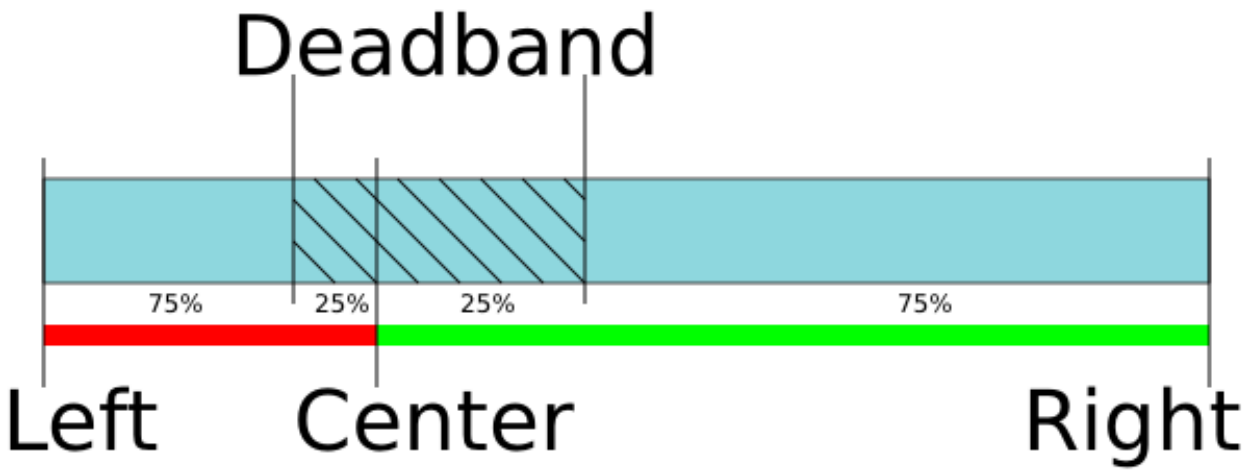
Запрос: (4 байт)

uint32_t	CMD	Команда

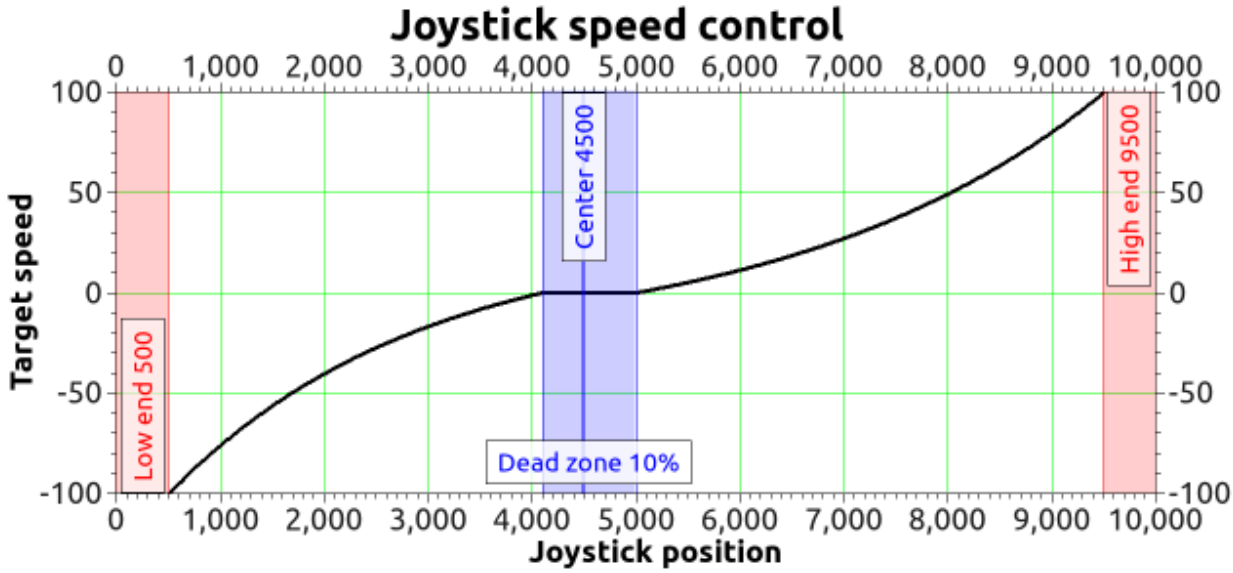
Ответ: (22 байт)

uint32_t	CMD	Команда
uint16_t	JoyLowEnd	Значение в шагах джойстика, соответствующее нижней границе диапазона отклонения устройства. Должно лежать в пределах. Диапазон: 0..10000.
uint16_t	JoyCenter	Значение в шагах джойстика, соответствующее неотклонённому устройству. Должно лежать в пределах. Диапазон: 0..10000.
uint16_t	JoyHighEnd	Значение в шагах джойстика, соответствующее верхней границе диапазона отклонения устройства. Должно лежать в пределах. Диапазон: 0..10000.
uint8_t	ExpFactor	Фактор экспоненциальной нелинейности отклика джойстика.
uint8_t	DeadZone	Отклонение от среднего положения, которое не вызывает начала движения (в десятых долях процента). Максимальное мёртвое отклонение +-25.5%, что составляет половину рабочего диапазона джойстика.
uint8_t	JoyFlags	Флаги управления джойстиком. Это битовая маска для побитовых операций.
	0x1 - JOY_REVERSE	Реверс воздействия джойстика. Отклонение джойстика к большим значениям приводит к отрицательной скорости и наоборот.
uint8_t	Reserved [7]	Зарезервировано (7 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек джойстика. При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость, а 100% отклонения соответствует MaxSpeed [i] (см. команду SCTL), где i=0, если предыдущим использованием этого режима не было выбрано другое i. Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Диапазоны DeadZone показаны на следующем рисунке.



Зависимость между отклонением и скоростью экспоненциальная, что позволяет без переключения режимов скорости сочетать высокую подвижность и точность. На графике ниже показан пример экспоненциальной зависимости скорости и работы мертвой зоны:



Параметр нелинейнойсти можно менять. Нулевой параметр нелинейности соответствует линейной зависимости.

6.2.6.22 Команда GMOV

Код команды (CMD): «gmov» или 0x766F6D67.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (30 байт)

uint32_t	CMD	Команда
uint32_t	Speed	Заданная скорость (для ШД: шагов/с, для DC: rpm). Диапазон: 0..100000.
uint8_t	uSpeed	Заданная скорость в единицах деления микрошага в секунду. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым мотором.
uint16_t	Accel	Ускорение, заданное в шагах в секунду^2 (ШД) или в оборотах в минуту за секунду (DC). Диапазон: 1..65535.
uint16_t	Decel	Торможение, заданное в шагах в секунду^2 (ШД) или в оборотах в минуту за секунду (DC). Диапазон: 1..65535.
uint32_t	AntiplaySpeed	Скорость в режиме антилюфта, заданная в целых шагах/с (ШД) или в оборотах/с(DC). Диапазон: 0..100000.
uint8_t	uAntiplaySpeed	Скорость в режиме антилюфта, выраженная в микрошагах в секунду. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым мотором.
uint8_t	MoveFlags	Флаги, управляющие настройкой движения. Это битовая маска для побитовых операций.
	0x1 - RPM_DIV_1000	Флаг указывает на то что рабочая скорость указанная в команде задана в милли rpm. Применим только для режима обратной связи ENCODER и только для BLDC моторов.
uint8_t	Reserved [9]	Зарезервировано (9 байт)
uint16_t	CRC	Контрольная сумма

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

6.2.6.23 Команда GMTI

Код команды (CMD): «gmti» или 0x69746D67.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (70 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель. Максимальная длина строки: 16 символов.
int8_t	PartNumber	Серия и номер модели. Максимальная длина строки: 24 символа.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение информации о двигателе из EEPROM.

6.2.6.24 Команда GMTS

Код команды (CMD): «gmts» или 0x73746D67.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (112 байт)

uint32_t	CMD	Команда
uint8_t	MotorType	Тип двигателя. Это битовая маска для побитовых операций.
	0x0 - MOTOR_TYPE_UNKNOWN	Неизвестный двигатель
	0x1 - MOTOR_TYPE_STEP	Шаговый двигатель
	0x2 - MOTOR_TYPE_DC	DC двигатель
	0x3 - MOTOR_TYPE_BLDC	BLDC двигатель
uint8_t	ReservedField	Зарезервировано
uint16_t	Poles	Кол-во пар полюсов у DC или BLDC двигателя или кол-во шагов на оборот для шагового двигателя.
uint16_t	Phases	Кол-во фаз у BLDC двигателя.
float	NominalVoltage	Номинальное напряжение на обмотке (В). Тип данных: float
float	NominalCurrent	Максимальный постоянный ток в обмотке для DC и BLDC двигателей, номинальный ток в обмотке для шаговых двигателей (А). Тип данных: float.
float	NominalSpeed	Не используется. Номинальная скорость (об/мин). Применяется для DC и BLDC двигателей. Тип данных: float.

Continued on next page

Таблица 6.53 – continued from previous page

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

Continued on next page

Таблица 6.53 – continued from previous page

float	MaxCurrentTime	Безопасная длительность максимального тока в обмотке (мс). Тип данных: float.
float	NoLoadCurrent	Ток потребления в холостом режиме (А). Применяется для DC и BLDC двигателей. Тип данных: float.
float	NoLoadSpeed	Скорость в холостом режиме (об/мин). Применяется для DC и BLDC двигателей. Тип данных: float.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек двигателя из EEPROM.

6.2.6.25 Команда GNET

Код команды (CMD): «gnet» или 0x74656E67.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (38 байт)

uint32_t	CMD	Команда
uint8_t	DHCPEnabled	Определяет способ получения IP-адреса каналов. Может принимать значения: 0 — статически, 1 — через DHCP.
uint8_t	IPv4Address	IP-адрес устройства в формате x.x.x.x.
uint8_t	SubnetMask	Маска подсети в формате x.x.x.x.
uint8_t	DefaultGateway	Шлюз сети по умолчанию в формате x.x.x.x.
uint8_t	Reserved [19]	Зарезервировано (19 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда чтения сетевых настроек. Только для производителя. Эта функция возвращает текущие сетевые настройки.

6.2.6.26 Команда GNME

Код команды (CMD): «gnme» или 0x656D6E67.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (30 байт)

uint32_t	CMD	Команда
int8_t	PositionerName	Пользовательское имя подвижки. Может быть установлено пользователем для его удобства. Максимальная длина строки: 16 символов.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение пользовательского имени подвижки из EEPROM.

6.2.6.27 Команда GNMF

Код команды (CMD): «gnmf» или 0x666D6E67.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (30 байт)

uint32_t	CMD	Команда
int8_t	ControllerName	Пользовательское имя контроллера. Может быть установлено пользователем для его удобства. Максимальная длина строки: 16 символов.
uint8_t	CtrlFlags	Настройки контроллера. Это битовая маска для побитовых операций.
	0x1 - EEPROM_PRECEDENCE	Если флаг установлен, то настройки в EEPROM подвижки имеют приоритет над текущими настройками и заменяют их при обнаружении EEPROM.
uint8_t	Reserved [7]	Зарезервировано (7 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение пользовательского имени контроллера и настроек из FRAM.

6.2.6.28 Команда GNVM

Код команды (CMD): «gnvm» или 0x6D766E67.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (36 байт)

uint32_t	CMD	Команда

Continued on next page

Таблица 6.61 – continued from previous page

uint32_t	UserData	Пользовательские данные. Могут быть установлены пользователем для его удобства. Каждый элемент массива хранит только 32 бита пользовательских данных. Это важно на системах где тип int содержит больше чем 4 байта. Например это все системы amd64.
uint8_t	Reserved [2]	Зарезервировано (2 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение пользовательских данных из FRAM.

6.2.6.29 Команда GPID

Код команды (CMD): «gpid» или 0x64697067.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (48 байт)

uint32_t	CMD	Команда
uint16_t	KpU	Пропорциональный коэффициент ПИД контура по напряжению
uint16_t	KiU	Интегральный коэффициент ПИД контура по напряжению
uint16_t	KdU	Дифференциальный коэффициент ПИД контура по напряжению
float	Kpf	Пропорциональный коэффициент ПИД контура по позиции для BLDC
float	Kif	Интегральный коэффициент ПИД контура по позиции для BLDC
float	Kdf	Дифференциальный коэффициент ПИД контура по позиции для BLDC
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение ПИД коэффициентов. Эти коэффициенты определяют поведение позиционера. Коэффициенты различны для разных позиционеров.

6.2.6.30 Команда GPWD

Код команды (CMD): «gpwd» или 0x64777067.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (36 байт)

uint32_t	CMD	Команда
int8_t	UserPassword	Строчка-пароль для доступа к веб-странице, который пользователь может поменять с помощью USB команды или на веб-странице.
uint8_t	Reserved [10]	Зарезервировано (10 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда чтения пароля к веб-странице. Только для производителя. Эта функция пользователяет прочитать пользовательский пароль к веб-странице из памяти контроллера.

6.2.6.31 Команда GPWR

Код команды (CMD): «gpwr» или 0x72777067.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (20 байт)

uint32_t	CMD	Команда
uint8_t	HoldCurrent	Ток мотора в режиме удержания, в процентах от номинального. Диапазон: 0..100.
uint16_t	CurrReductDelay	Время в мс от перехода в состояние STOP до уменьшения тока.
uint16_t	PowerOffDelay	Время в с от перехода в состояние STOP до отключения питания мотора.
uint16_t	CurrentSetTime	Время в мс, требуемое для набора номинального тока от 0% до 100%.
uint8_t	PowerFlags	Флаги параметров управления питанием. Это битовая маска для побитовых операций.
	0x1 - POWER_REDUCT_ENABLED	Если флаг установлен, уменьшить ток по прошествии CurrReductDelay. Иначе - не уменьшать.
	0x2 - POWER_OFF_ENABLED	Если флаг установлен, снять напряжение с обмоток по прошествии PowerOffDelay. Иначе - не снимать.

Continued on next page

Таблица 6.67 – continued from previous page

	0x4 - POWER_SMOOTH_CURRENT	Если установлен, то запитывание обмоток, снятие питания или снижение/повышение тока происходят плавно со скоростью CurrentSetTime, а только потом выполняется та задача, которая вызвала это плавное изменение.
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда чтения параметров питания мотора. Используется только с шаговым двигателем. Используется только с шаговым двигателем.

6.2.6.32 Команда GSEC

Код команды (CMD): «gsec» или 0x63657367.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (28 байт)

uint32_t	CMD	Команда
uint16_t	LowUpwrOff	Нижний порог напряжения на силовой части для выключения, десятки мВ.
uint16_t	CriticalIpwr	Максимальный ток силовой части, вызывающий состояние ALARM, в мА.
uint16_t	CriticalUpwr	Максимальное напряжение на силовой части, вызывающее состояние ALARM, десятки мВ.
uint16_t	CriticalT	Максимальная температура контроллера, вызывающая состояние ALARM, в десятых долях градуса Цельсия.
uint16_t	CriticalIusb	Максимальный ток USB, вызывающий состояние ALARM, в мА.
uint16_t	CriticalUusb	Максимальное напряжение на USB, вызывающее состояние ALARM, десятки мВ.
uint16_t	MinimumUusb	Минимальное напряжение на USB, вызывающее состояние ALARM, десятки мВ.
uint8_t	Flags	Флаги критических параметров. Это битовая маска для побитовых операций.

Continued on next page

Таблица 6.69 – continued from previous page

	0x1 - ALARM_ON_DRIVER_OVERHEATING	Если флаг установлен, то войти в состояние Alarm при получении сигнала подступающего перегрева с драйвера. Иначе - игнорировать подступающий перегрев с драйвера.
	0x2 - LOW_UPWR_PROTECTION	Если установлен, то выключать силовую часть при напряжении меньшем LowUpwrOff.
	0x4 - H_BRIDGE_ALERT	Если установлен, то выключать силовую часть при сигнале неполадки в одном из транзисторных мостов.
	0x8 - ALARM_ON_BORDERS_SWAP_MISSET	Если флаг установлен, то войти в состояние Alarm при получении сигнала с противоположного концевого выключателя
	0x10 - ALARM_FLAGS_STICKING	Если флаг установлен, то только по команде STOP возможен сброс всех флагов ALARM
	0x20 - USB_BREAK_RECONNECT	Если флаг установлен, то будет включен блок перезагрузки USB при поломке связи
	0x40 - ALARM_WINDING_MISMATCH	Если флаг установлен, то войти в состояние Alarm при получении сигнала рассогласования обмоток
	0x80 - ALARM_ENGINE_RESPONSE	Если флаг установлен, то войти в состояние Alarm при получении сигнала ошибки реакции двигателя на управляющее воздействие
uint8_t	Reserved [7]	Зарезервировано (7 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда записи установок защит.

6.2.6.33 Команда GSNI

Код команды (CMD): «gsni» или 0x696E7367.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (28 байт)

uint32_t	CMD	Команда
uint8_t	SyncInFlags	Флаги синхронизации входа. Это битовая маска для побитовых операций.
	0x1 - SYNCIN_ENABLED	Включение необходимости импульса синхронизации для начала движения.

Continued on next page

Таблица 6.71 – continued from previous page

	0x2 - SYNCIN_INVERT	Если установлен - срабатывает по переходу из 1 в 0. Иначе - из 0 в 1.
	0x4 - SYNCIN_GOTOPOSITION	Если флаг установлен, то двигатель смещается к позиции, установленной в Position и uPosition, иначе двигатель смещается на Position и uPosition
uint16_t	ClutterTime	Минимальная длительность входного импульса синхронизации для защиты от дребезга (мкс).
int32_t	Position	Желаемая позиция или смещение (в полных шагах)
int16_t	uPosition	Дробная часть позиции или смещения в микрошагах. Используется только с шаговым двигателем. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint32_t	Speed	Заданная скорость (для ШД: шагов/с, для DC: rpm). Диапазон: 0..100000.
uint8_t	uSpeed	Заданная скорость в микрошагах в секунду. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым мотором.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек для входного импульса синхронизации. Эта функция считывает структуру с настройками синхронизации, определяющими поведение входа синхронизации, в память контроллера.

6.2.6.34 Команда GSNO

Код команды (CMD): «gsno» или 0x6F6E7367.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (16 байт)

uint32_t	CMD	Команда

Continued on next page

Таблица 6.73 – continued from previous page

uint8_t	SyncOutFlags	Флаги синхронизации выхода. Это битовая маска для побитовых операций.
	0x1 - SYNCOUT_ENABLED	Синхронизация выхода работает согласно настройкам, если флаг установлен. В ином случае значение выхода фиксировано и подчиняется SYNCOUT_STATE.
	0x2 - SYNCOUT_STATE	Когда значение выхода управляется напрямую (см. флаг SYNCOUT_ENABLED), значение на выходе соответствует значению этого флага.
	0x4 - SYNCOUT_INVERT	Нулевой логический уровень является активным, если флаг установлен, а единичный - если флаг сброшен.
	0x8 - SYNCOUT_IN_STEPS	Если флаг установлен использовать шаги/импульсы энкодера для выходных импульсов синхронизации вместо миллисекунд.
	0x10 - SYNCOUT_ONSTART	Генерация синхронизирующего импульса при начале движения.
	0x20 - SYNCOUT_ONSTOP	Генерация синхронизирующего импульса при остановке.
	0x40 - SYNCOUT_ONPERIOD	Выдает импульс синхронизации после прохождения SyncOutPeriod отсчётов.
uint16_t	SyncOutPulseSteps	Определяет длительность выходных импульсов в шагах/импульсах энкодера, когда установлен флаг SYNCOUT_IN_STEPS, или в микросекундах если флаг сброшен.
uint16_t	SyncOutPeriod	Период генерации импульсов (в шагах/отсчетах энкодера), используется при установленном флаге SYNCOUT_ONPERIOD.
uint32_t	Accuracy	Это окрестность вокруг целевой координаты, попадание в которую считается попаданием в целевую позицию и генерируется импульс по остановке.
uint8_t	uAccuracy	Это окрестность вокруг целевой координаты в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек для выходного импульса синхронизации. Эта функция считывает структуру с настройками синхронизации, определяющими поведение выхода синхронизации, в память контроллера.

6.2.6.35 Команда GSTI

Код команды (CMD): «gsti» или 0x69747367.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (70 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель. Максимальная длина строки: 16 символов.
int8_t	PartNumber	Серия и номер модели. Максимальная длина строки: 24 символа.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение информации о позиционере из EEPROM. Не поддерживается.

6.2.6.36 Команда GSTS

Код команды (CMD): «gsts» или 0x73747367.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (70 байт)

uint32_t	CMD	Команда
float	LeadScrewPitch	Шаг ходового винта в мм. Тип данных: float.
int8_t	Units	Единицы измерения расстояния, используемые в полях MaxSpeed и TravelRange (шаги, градусы, мм, . . .), Максимальная длина строки: 8 символов.
float	MaxSpeed	Максимальная скорость (Units/c). Тип данных: float.
float	TravelRange	Диапазон перемещения (Units). Тип данных: float.
float	SupplyVoltageMin	Минимальное напряжение питания (В). Тип данных: float.
float	SupplyVoltageMax	Максимальное напряжение питания (В). Тип данных: float.
float	MaxCurrentConsumption	Максимальный ток потребления (А). Тип данных: float.

Continued on next page

Таблица 6.77 – continued from previous page

float	HorizontalLoadCapacity	Горизонтальная грузоподъемность (кг). Тип данных: float.
float	VerticalLoadCapacity	Вертикальная грузоподъемность (кг). Тип данных: float.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение настроек позиционера из EEPROM.

6.2.6.37 Команда GURT

Код команды (CMD): «gurt» или 0x74727567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (16 байт)

uint32_t	CMD	Команда
uint32_t	Speed	Скорость UART (в бодах)
uint16_t	UARTSetupFlags	Флаги настройки UART. Это битовая маска для побитовых операций.
	0x3 - UART_PARITY_BITS	Биты, отвечающие за выбор четности.
	0x0 - UART_PARITY_BIT_EVEN	Бит 1, если четный
	0x1 - UART_PARITY_BIT_ODD	Бит 1, если нечетный
	0x2 - UART_PARITY_BIT_SPACE	Бит четности всегда 0
	0x3 - UART_PARITY_BIT_MARK	Бит четности всегда 1
	0x4 - UART_PARITY_BIT_USE	Бит чётности не используется, если „0“; бит четности используется, если „1“
	0x8 - UART_STOP_BIT	Если установлен, один стоповый бит; иначе - 2 стоповых бита
uint8_t	Reserved [4]	Зарезервировано (4 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда чтения настроек UART. Эта функция заполняет структуру настроек UART.

6.2.6.38 Команда SACC

Код команды (CMD): «sacc» или 0x63636173.

Запрос: (114 байт)

uint32_t	CMD	Команда
int8_t	MagneticBrakeInfo	Производитель и номер магнитного тормоза, Максимальная длина строки: 24 символов.

Continued on next page

Таблица 6.80 – continued from previous page

float	MBRatedVoltage	Номинальное напряжение для управления магнитным тормозом (В). Тип данных: float.
float	MBRatedCurrent	Номинальный ток для управления магнитным тормозом (А). Тип данных: float.
float	MBTorque	Удерживающий момент (мН м). Тип данных: float.
uint32_t	MBSettings	Флаги настроек магнитного тормоза. Это битовая маска для побитовых операций.
	0x1 - MB_AVAILABLE	Если флаг установлен, то магнитный тормоз доступен
	0x2 - MB_POWERED_HOLD	Если флаг установлен, то магнитный тормоз находится в режиме удержания (активен) при подаче питания
int8_t	TemperatureSensorInfo	Производитель и номер температурного датчика, Максимальная длина строки: 24 символов.
float	TSMin	Минимальная измеряемая температура (град Цельсия). Тип данных: float.
float	TSMax	Максимальная измеряемая температура (град Цельсия) Тип данных: float.
float	TSGrad	Температурный градиент (В/град Цельсия). Тип данных: float.
uint32_t	TSSettings	Флаги настроек температурного датчика. Это битовая маска для побитовых операций.
	0x7 - TS_TYPE_BITS	Биты, отвечающие за тип температурного датчика
	0x0 - TS_TYPE_UNKNOWN	Неизвестный сенсор
	0x1 - TS_TYPE_THERMOCOUPLE	Термопара
	0x2 - TS_TYPE_SEMICONDUCTOR	Полупроводниковый температурный датчик
	0x8 - TS_AVAILABLE	Если флаг установлен, то датчик температуры доступен
uint32_t	LimitSwitchesSettings	Флаги настроек концевых выключателей. Это битовая маска для побитовых операций.
	0x1 - LS_ON_SW1_AVAILABLE	Если флаг установлен, то концевой переключатель, подключенный к ножке SW1, доступен
	0x2 - LS_ON_SW2_AVAILABLE	Если флаг установлен, то концевой переключатель, подключенный к ножке SW2, доступен

Continued on next page

Таблица 6.80 – continued from previous page

	0x4 - LS_SW1_ACTIVE_LOW	Если флаг установлен, то концевой переключатель, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте
	0x8 - LS_SW2_ACTIVE_LOW	Если флаг установлен, то концевой переключатель, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте
	0x10 - LS_SHORTED	Если флаг установлен, то концевые переключатели замкнуты
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись информации о дополнительных аксессуарах в EEPROM. Функция должна использоваться только производителем.

6.2.6.39 Команда SBRK

Код команды (CMD): «sbrk» или 0x6B726273.

Запрос: (25 байт)

uint32_t	CMD	Команда
uint16_t	t1	Время в мс между включением питания мотора и отключением тормоза.
uint16_t	t2	Время в мс между отключением тормоза и готовностью к движению. Все команды движения начинают выполняться только по истечении этого времени.
uint16_t	t3	Время в мс между остановкой мотора и включением тормоза.
uint16_t	t4	Время в мс между включением тормоза и отключением питания мотора.
uint8_t	BrakeFlags	Флаги. Это битовая маска для побитовых операций.
	0x1 - BRAKE_ENABLED	Управление тормозом включено, если флаг установлен.
	0x2 - BRAKE_ENG_PWROFF	Тормоз отключает питание шагового мотора, если флаг установлен.
uint8_t	Reserved [10]	Зарезервировано (10 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек управления тормозом.

6.2.6.40 Команда SCAL

Код команды (CMD): «scal» или 0x6C616373.

Запрос: (118 байт)

uint32_t	CMD	Команда
float	CSS1_A	Коэффициент масштабирования для аналоговых измерений тока в обмотке A.
float	CSS1_B	Коэффициент сдвига для аналоговых измерений тока в обмотке A.
float	CSS2_A	Коэффициент масштабирования для аналоговых измерений тока в обмотке B.
float	CSS2_B	Коэффициент сдвига для аналоговых измерений тока в обмотке B.
float	FullCurrent_A	Коэффициент масштабирования для аналоговых измерений полного тока.
float	FullCurrent_B	Коэффициент сдвига для аналоговых измерений полного тока.
uint8_t	Reserved [88]	Зарезервировано (88 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда записи калибровочных коэффициентов. Команда только для производителя. Эта функция записывает структуру калибровочных коэффициентов в память контроллера. Эти коэффициенты используются для пересчёта кодов АЦП в токи обмоток и полный ток потребления. Коэффициенты сгруппированы в пары, XXX_A и XXX_B; пары представляют собой коэффициенты линейного уравнения. Первый коэффициент - тангенс угла наклона, второй - постоянное смещение. Таким образом, XXX_Current[mA] = XXX_A[mA/ADC]XXX_ADC_CODE[ADC] + XXX_B[mA].

6.2.6.41 Команда SCTL

Код команды (CMD): «sctl» или 0x6C746373.

Запрос: (93 байт)

uint32_t	CMD	Команда
uint32_t	MaxSpeed	Массив скоростей (в полных шагах), использующийся при управлении джойстиком или кнопками влево/вправо. Диапазон: 0..100000.

Continued on next page

Таблица 6.86 – continued from previous page

uint8_t	uMaxSpeed	Массив скоростей (в микрошагах), использующийся при управлении джойстиком или кнопками влево/вправо. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint16_t	Timeout	Timeout[i] - время в мс, по истечении которого устанавливается скорость max_speed[i+1] (используется только при управлении кнопками).
uint16_t	MaxClickTime	Максимальное время клика (в мс). До истечения этого времени первая скорость не включается.
uint16_t	Flags	Флаги. Это битовая маска для побитовых операций.
	0x3 - CONTROL_MODE_BITS	Биты управления мотором с помощью джойстика или кнопок влево/вправо.
	0x0 - CONTROL_MODE_OFF	Управление отключено.
	0x1 - CONTROL_MODE_JOY	Управление с помощью джойстика.
	0x2 - CONTROL_MODE_LR	Управление с помощью кнопок влево/вправо.
	0x4 - CONTROL_BTN_LEFT_PUSHED_OPEN	Нажатая левая кнопка соответствует открытому контакту, если этот флаг установлен.
	0x8 - CONTROL_BTN_RIGHT_PUSHED_OPEN	Нажатая правая кнопка соответствует открытому контакту, если этот флаг установлен.
int32_t	DeltaPosition	Смещение (дельта) позиции (в полных шагах)
int16_t	uDeltaPosition	Дробная часть смещения в микрошагах. Используется только с шаговым двигателем. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint8_t	Reserved [9]	Зарезервировано (9 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек управления мотором. При выборе CTL_MODE=1 включается управление мотором с помощью джойстика. В этом режиме при отклонении джойстика на максимум двигатель стремится двигаться со скоростью MaxSpeed [i], где i=0, если предыдущим использованием этого режима не было выбрано другое i. Кнопки переключают номер скорости i. При выборе CTL_MODE=2

включается управление мотором с помощью кнопок left/right. При нажатии на кнопки двигатель начинает двигаться в соответствующую сторону со скоростью MaxSpeed[0], по истечении времени Timeout[i] мотор двигается со скоростью MaxSpeed [i+1]. При переходе от MaxSpeed [i] на MaxSpeed [i+1] действует ускорение, как обычно.

6.2.6.42 Команда SCTP

Код команды (CMD): «sctp» или 0x70746373.

Запрос: (18 байт)

uint32_t	CMD	Команда
uint8_t	CTPMinError	Минимальное отличие шагов ШД от положения энкодера, устанавливающее флаг STATE_RT_ERROR. Измеряется в шагах ШД.
uint8_t	CTPFlags	Флаги. Это битовая маска для побитовых операций.
	0x1 - CTP_ENABLED	Контроль позиции включен, если флаг установлен.
	0x2 - CTP_BASE	Управление положением основано на датчике вращения, если установлен этот флаг; в противном случае - на энкодере.
	0x4 - CTP_ALARM_ON_ERROR	Войти в состояние ALARM при расхождении позиции, если флаг установлен.
	0x8 - REV_SENS_INV	Сенсор считается активным, когда на нём 0, инвертирование делает активным уровень 1. То есть если не инвертировать, то действует обычная логика - 0 это срабатывание/активация/активное состояние.
	0x10 - CTP_ERROR_CORRECTION	Корректировать ошибки, возникающие при проскальзывании, если флаг установлен. Работает только с энкодером. Несовместимо с флагом CTP_ALARM_ON_ERROR.
uint8_t	Reserved [10]	Зарезервировано (10 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек контроля позиции(для шагового двигателя). При управлении ШД с энкодером (CTP_BASE 0) появляется возможность обнаруживать потерю шагов. Контроллер знает кол-во шагов на оборот (GENG::StepsPerRev) и разрешение энкодера (GFBS::IPT). При включении контроля (флаг CTP_ENABLED), контроллер запоминает текущую позицию в шагах ШД и текущую позицию энкодера. Далее, на каждом шаге позиция энкодера преобразовывается в шаги и если разница оказывается больше CTPMinError, устанавливается флаг STATE_CTP_ERROR. При управлении ШД с датчиком оборотов (CTP_BASE 1), позиция контролируется по нему. По активному фронту на входе

синхронизации контроллер запоминает текущее значение шагов. Далее, при каждом обороте проверяет, на сколько шагов сместились. При рассогласовании более CTPMinError устанавливается флаг STATE_CTP_ERROR.

6.2.6.43 Команда SEAS

Код команды (CMD): «seas» или 0x73616573.

Запрос: (54 байт)

uint32_t	CMD	Команда
uint16_t	stepcloseloop_Kw	Коэффициент смешения реальной и заданной скорости, диапазон [0, 100], значение по умолчанию 50.
uint16_t	stepcloseloop_Kp_low	Обратная связь по позиции в зоне малых скоростей, диапазон [0, 65535], значение по умолчанию 1000.
uint16_t	stepcloseloop_Kp_high	Обратная связь по позиции в зоне больших скоростей, диапазон [0, 65535], значение по умолчанию 33.
uint8_t	Reserved [42]	Зарезервировано (42 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись расширенных настроек.

6.2.6.44 Команда SEDS

Код команды (CMD): «seds» или 0x73646573.

Запрос: (26 байт)

uint32_t	CMD	Команда
uint8_t	BorderFlags	Флаги, определяющие тип границ и поведение мотора при их достижении. Это битовая маска для побитовых операций.
	0x1 - BORDER_IS_ENCODER	Если флаг установлен, границы определяются предустановленными точками на шкале позиции. Если флаг сброшен, границы определяются концевыми выключателями.
	0x2 - BORDER_STOP_LEFT	Если флаг установлен, мотор останавливается при достижении левой границы.
	0x4 - BORDER_STOP_RIGHT	Если флаг установлен, мотор останавливается при достижении правой границы.

Continued on next page

Таблица 6.92 – continued from previous page

	0x8 - BORDERS_SWAP_MISSET_DETECTION	Если флаг установлен, мотор останавливается при достижении обоих границ. Нужен для предотвращения поломки двигателя при неправильных настройках концевых выключателей
uint8_t	EnderFlags	Флаги, определяющие настройки концевых выключателей. Это битовая маска для побитовых операций.
	0x1 - ENDER_SWAP	Если флаг установлен, первый концевой выключатель находится справа; иначе - слева.
	0x2 - ENDER_SW1_ACTIVE_LOW	1 - Концевой переключатель, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте.
	0x4 - ENDER_SW2_ACTIVE_LOW	1 - Концевой переключатель, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте.
int32_t	LeftBorder	Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.
int16_t	uLeftBorder	Позиция левой границы в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
int32_t	RightBorder	Позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER.
int16_t	uRightBorder	Позиция правой границы в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек границ и концевых выключателей.

6.2.6.45 Команда SEIO

Код команды (CMD): «seio» или 0x6F696573.

Запрос: (18 байт)

uint32_t	CMD	Команда
uint8_t	EXTIOSetupFlags	Флаги настройки работы внешнего ввода-вывода. Это битовая маска для побитовых операций.
	0x1 - EXTIO_SETUP_OUTPUT	Если флаг установлен, то ножка в состоянии вывода, иначе - ввода.
	0x2 - EXTIO_SETUP_INVERT	Если флаг установлен, то нули считаются активным состоянием выхода, а спадающие фронты как момент подачи входного сигнала.
uint8_t	EXTIOModeFlags	Флаги настройки режимов внешнего ввода-вывода. Это битовая маска для побитовых операций.
	0xf - EXTIO_SETUP_MODE_IN_BITS	Биты, отвечающие за поведение при переходе сигнала в активное состояние.
	0x0 - EXTIO_SETUP_MODE_IN_NOP	Ничего не делать.
	0x1 - EXTIO_SETUP_MODE_IN_STOP	По переднему фронту входного сигнала делается остановка двигателя (эквивалент команды STOP).
	0x2 - EXTIO_SETUP_MODE_IN_PWOF	Выполняет команду PWOF, обесточивая обмотки двигателя.
	0x3 - EXTIO_SETUP_MODE_IN_MOVR	Выполняется команда MOVR с последними настройками.
	0x4 - EXTIO_SETUP_MODE_IN_HOME	Выполняется команда HOME.
	0x5 - EXTIO_SETUP_MODE_IN_ALARM	Войти в состояние ALARM при переходе сигнала в активное состояние.
	0xf0 - EXTIO_SETUP_MODE_OUT_BITS	Биты выбора поведения на выходе.
	0x0 - EXTIO_SETUP_MODE_OUT_OFF	Ножка всегда в неактивном состоянии.
	0x10 - EXTIO_SETUP_MODE_OUT_ON	Ножка всегда в активном состоянии.
	0x20 - EXTIO_SETUP_MODE_OUT_MOVING	Ножка находится в активном состоянии при движении.
	0x30 - EXTIO_SETUP_MODE_OUT_ALARM	Ножка находится в активном состоянии при нахождении в состоянии ALARM.
	0x40 - EXTIO_SETUP_MODE_OUT_MOTOR_ON	Ножка находится в активном состоянии при подаче питания на обмотки.
uint8_t	Reserved [10]	Зарезервировано (10 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда записи параметров настройки режимов внешнего ввода/вывода. Входные события обрабатываются по фронту. Выходные состояния сигнализируются логическим состоянием. По умолчанию нарастающий фронт считается моментом подачи входного сигнала, а единичное состояние считается активным выходом.

6.2.6.46 Команда SEMF

Код команды (CMD): «semf» или 0x666D6573.

Запрос: (48 байт)

uint32_t	CMD	Команда
float	L	Индуктивность обмоток двигателя.
float	R	Сопротивление обмоток двигателя.
float	Km	Электромеханический коэффициент двигателя.
uint8_t	BackEMFFlags	Флаги автонастроек шагового двигателя. Это битовая маска для побитовых операций.
	0x1 - BACK_EMF_INDUCTANCE_AUTO	Флаг автоопределения индуктивности обмоток двигателя.
	0x2 - BACK_EMF_RESISTANCE_AUTO	Флаг автоопределения сопротивления обмоток двигателя.
	0x4 - BACK_EMF_KM_AUTO	Флаг автоопределения электромеханического коэффициента двигателя.
uint8_t	Reserved [29]	Зарезервировано (29 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись электромеханических настроек шагового двигателя. Настройки различны для разных двигателей. Пожалуйста, загружайте новые настройки, когда вы меняете мотор.

6.2.6.47 Команда SENG

Код команды (CMD): «seng» или 0x676E6573.

Запрос: (34 байт)

uint32_t	CMD	Команда
uint16_t	NomVoltage	Номинальное напряжение мотора в десятках мВ. Контроллер будет сохранять напряжение на моторе не выше номинального, если установлен флаг ENGINE_LIMIT_VOLT (используется только с DC двигателем).

Continued on next page

Таблица 6.98 – continued from previous page

uint16_t	NomCurrent	Номинальный ток через мотор (в мА). Ток стабилизируется для шаговых и может быть ограничен для DC(если установлен флаг ENGINE_LIMIT_CURR). Диапазон: 15..8000
uint32_t	NomSpeed	Номинальная (максимальная) скорость (в целых шагах/с или rpm для DC и шагового двигателя в режиме ведущего энкодера). Контроллер будет сохранять скорость мотора не выше номинальной, если установлен флаг ENGINE_LIMIT_RPM. Диапазон: 1..100000.
uint8_t	uNomSpeed	Микрошаговая часть номинальной скорости мотора (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint16_t	EngineFlags	Флаги, управляющие работой мотора. Это битовая маска для побитовых операций.
	0x1 - ENGINE_REVERSE	Флаг реверса. Связывает направление вращения мотора с направлением счета текущей позиции. При сброшенном флаге (по умолчанию) прикладываемое к мотору положительное напряжение увеличивает счетчик позиции. И наоборот, при установленном флаге счетчик позиции увеличивается, когда к мотору приложено отрицательное напряжение. Измените состояние флага, если положительное вращение мотора уменьшает счетчик позиции.
	0x2 - ENGINE_CURRENT_AS_RMS	Флаг интерпретации значения тока. Если флаг снят, то задаваемое значение тока интерпретируется как максимальная амплитуда тока. Если флаг установлен, то задаваемое значение тока интерпретируется как среднеквадратичное значение тока (для шагового) или как значение тока, посчитанное из максимального тепловыделения (BLDC).
	0x4 - ENGINE_MAX_SPEED	Флаг максимальной скорости. Если флаг установлен, движение происходит на максимальной скорости.

Continued on next page

Таблица 6.98 – continued from previous page

	0x8 - ENGINE_ANTIPLAY	Компенсация люфта. Если флаг установлен, позиционер будет подходить к заданной точке всегда с одной стороны. Например, при подходе слева никаких дополнительных действий не совершается, а при подходе справа позиционер проходит целевую позицию на заданное расстояния и возвращается к ней опять же справа.
	0x10 - ENGINE_ACCEL_ON	Ускорение. Если флаг установлен, движение происходит с ускорением.
	0x20 - ENGINE_LIMIT_VOLT	Номинальное напряжение мотора. Если флаг установлен, напряжение на моторе ограничивается заданным номинальным значением(используется только с DC двигателем).
	0x40 - ENGINE_LIMIT_CURR	Номинальный ток мотора. Если флаг установлен, ток через мотор ограничивается заданным номинальным значением(используется только с DC двигателем).
	0x80 - ENGINE_LIMIT_RPM	Номинальная частота вращения мотора. Если флаг установлен, частота вращения ограничивается заданным номинальным значением.
int16_t	Antiplay	Количество шагов двигателя или импульсов энкодера, на которое позиционер будет отъезжать от заданной позиции для подхода к ней с одной и той же стороны. Используется, если установлен флаг ENGINE_ANTIPLAY.
uint8_t	MicrostepMode	Настройки микрошагового режима(используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Это битовая маска для побитовых операций.
	0x1 - MICROSTEP_MODE_FULL	Полношаговый режим.
	0x2 - MICROSTEP_MODE_FRAC_2	Деление шага 1/2.
	0x3 - MICROSTEP_MODE_FRAC_4	Деление шага 1/4.
	0x4 - MICROSTEP_MODE_FRAC_8	Деление шага 1/8.
	0x5 - MICROSTEP_MODE_FRAC_16	Деление шага 1/16.
	0x6 - MICROSTEP_MODE_FRAC_32	Деление шага 1/32.
	0x7 - MICROSTEP_MODE_FRAC_64	Деление шага 1/64.
	0x8 - MICROSTEP_MODE_FRAC_128	Деление шага 1/128.

Continued on next page

Таблица 6.98 – continued from previous page

	0x9 - MICROSTEP_MODE_FRAC_256	Деление шага 1/256.
uint16_t	StepsPerRev	Количество полных шагов на оборот(используется только с шаговым двигателем). Диапазон: 1..65535.
uint8_t	Reserved [12]	Зарезервировано (12 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

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

6.2.6.48 Команда SENI

Код команды (CMD): «seni» или 0x696E6573.

Запрос: (70 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель. Максимальная длина строки: 16 символов.
int8_t	PartNumber	Серия и номер модели. Максимальная длина строки: 24 символа.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись информации об энкодере в EEPROM. Функция должна использоваться только производителем.

6.2.6.49 Команда SENS

Код команды (CMD): «sens» или 0x736E6573.

Запрос: (54 байт)

uint32_t	CMD	Команда
float	MaxOperatingFrequency	Максимальная частота (кГц). Тип данных: float.
float	SupplyVoltageMin	Минимальное напряжение питания (В). Тип данных: float.
float	SupplyVoltageMax	Максимальное напряжение питания (В). Тип данных: float.

Continued on next page

Таблица 6.102 – continued from previous page

float	MaxCurrentConsumption	Максимальное потребление тока (мА). Тип данных: float.
uint32_t	PPR	Количество отсчётов на оборот
uint32_t	EncoderSettings	Флаги настроек энкодера. Это битовая маска для побитовых операций.
	0x1 - ENCSET_DIFFERENTIAL_OUTPUT	Если флаг установлен, то энкодер имеет дифференциальный выход, иначе - несимметричный выход
	0x4 - ENCSET_PUSHPULL_OUTPUT	Если флаг установлен, то энкодер имеет двухтактный выход, иначе - выход с открытым коллектором
	0x10 - ENCSET_INDEXCHANNEL_PRESENT	Если флаг установлен, то энкодер имеет дополнительный индексный канал, иначе - он отсутствует
	0x40 - ENCSET_REVOLUTIONSENSOR_PRESENT	Если флаг установлен, то энкодер имеет датчик оборотов, иначе - он отсутствует
	0x100 - ENCSET_REVOLUTIONSENSOR_ACTIVE_HIGH	Если флаг установлен, то активное состояние датчика оборотов соответствует логической 1, иначе - логическому 0
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек энкодера в EEPROM. Функция должна использоваться только производителем.

6.2.6.50 Команда SENT

Код команды (CMD): «sent» или 0x746E6573.

Запрос: (14 байт)

uint32_t	CMD	Команда
uint8_t	EngineType	Тип мотора. Это битовая маска для побитовых операций.
	0x0 - ENGINE_TYPE_NONE	Это значение не нужно использовать.
	0x1 - ENGINE_TYPE_DC	Мотор постоянного тока.
	0x2 - ENGINE_TYPE_2DC	Два мотора постоянного тока, что приводит к эмуляции двух контроллеров.
	0x3 - ENGINE_TYPE_STEP	Шаговый мотор.
	0x4 - ENGINE_TYPE_TEST	Продолжительность включения фиксирована. Используется только производителем.
	0x5 - ENGINE_TYPE_BRUSHLESS	Бесщеточный мотор.

Continued on next page

Таблица 6.104 – continued from previous page

uint8_t	DriverType	Тип силового драйвера. Это битовая маска для побитовых операций.
	0x1 - DRIVER_TYPE_DISCRETE_FET	Силовой драйвер на дискретных мосфет-ключах. Используется по умолчанию.
	0x2 - DRIVER_TYPE_INTEGRATE	Силовой драйвер с использованием ключей, интегрированных в микросхему.
	0x3 - DRIVER_TYPE_EXTERNAL	Внешний силовой драйвер.
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись информации о типе мотора и типе силового драйвера.

6.2.6.51 Команда SEST

Код команды (CMD): «sest» или 0x74736573.

Запрос: (46 байт)

uint32_t	CMD	Команда
uint16_t	Param1	
uint8_t	Reserved [38]	Зарезервировано (38 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись расширенных настроек. В настоящее время не используется.

6.2.6.52 Команда SFBS

Код команды (CMD): «sfbs» или 0x73626673.

Запрос: (18 байт)

uint32_t	CMD	Команда
uint16_t	IPS	Количество отсчётов энкодера на оборот вала. Диапазон: 1..65535. Поле устарело, рекомендуется записывать 0 в IPS и использовать расширенное поле CountsPerTurn. Может потребоваться обновление микропрограммы контроллера до последней версии.

Continued on next page

Таблица 6.108 – continued from previous page

uint8_t	FeedbackType	Тип обратной связи. Это битовая маска для побитовых операций.
	0x1 - FEEDBACK_ENCODER	Обратная связь с помощью энкодера.
	0x4 - FEEDBACK_EMF	Обратная связь по ЭДС.
	0x5 - FEEDBACK_NONE	Обратная связь отсутствует.
	0x6 - FEEDBACK_ENCODER_MEDIATED	Обратная связь по энкодеру, опосредованному относительно двигателя механической передачей (например, винтовой передачей).
uint8_t	FeedbackFlags	Флаги. Это битовая маска для побитовых операций.
	0x1 - FEEDBACK_ENC_REVERSE	Обратный счет у энкодера.
	0xc0 - FEEDBACK_ENC_TYPE_BITS	Биты, отвечающие за тип энкодера.
	0x0 - FEEDBACK_ENC_TYPE_AUTO	Определяет тип энкодера автоматически.
	0x40 - FEEDBACK_ENC_TYPE_SINGLE_ENDED	Недифференциальный энкодер.
	0x80 - FEEDBACK_ENC_TYPE_DIFFERENTIAL	Дифференциальный энкодер.
uint32_t	CountsPerTurn	Количество отсчётов энкодера на оборот вала. Диапазон: 1..4294967295. Для использования поля CountsPerTurn нужно записать 0 в поле IPS, иначе будет использоваться значение из поля IPS.
uint8_t	Reserved [4]	Зарезервировано (4 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Чтение настроек обратной связи.

6.2.6.53 Команда SGRI

Код команды (CMD): «sgri» или 0x69726773.

Запрос: (70 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель. Максимальная длина строки: 16 символов.
int8_t	PartNumber	Серия и номер модели. Максимальная длина строки: 24 символа.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись информации о редукторе в EEPROM. Функция должна использоваться только производителем.

6.2.6.54 Команда SGRS

Код команды (CMD): «sgrs» или 0x73726773.

Запрос: (58 байт)

uint32_t	CMD	Команда
float	ReductionIn	Входной коэффициент редуктора. (Выход = (ReductionOut/ReductionIn) вход) Тип данных: float.
float	ReductionOut	Выходной коэффициент редуктора. (Выход = (ReductionOut/ReductionIn) вход) Тип данных: float.
float	RatedInputTorque	Максимальный крутящий момент (Н м). Тип данных: float.
float	RatedInputSpeed	Максимальная скорость на входном валу редуктора (об/мин). Тип данных: float.
float	MaxOutputBacklash	Выходной люфт редуктора (градус). Тип данных: float.
float	InputInertia	Эквивалентная входная инерция редуктора(г см2). Тип данных: float.
float	Efficiency	КПД редуктора (%). Тип данных: float.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек редуктора в EEPROM. Функция должна использоваться только производителем.

6.2.6.55 Команда SHOM

Код команды (CMD): «shom» или 0x6D6F6873.

Запрос: (33 байт)

uint32_t	CMD	Команда
uint32_t	FastHome	Скорость первого движения (в полных шагах). Диапазон: 0..100000.

Continued on next page

Таблица 6.114 – continued from previous page

uint8_t	uFastHome	Дробная часть скорости первого движения в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint32_t	SlowHome	Скорость второго движения (в полных шагах). Диапазон: 0..100000.
uint8_t	uSlowHome	Дробная часть скорости второго движения в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
int32_t	HomeDelta	Расстояние отхода от точки останова (в полных шагах).
int16_t	uHomeDelta	Дробная часть расстояния отхода от точки останова в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint16_t	HomeFlags	Набор флагов, определяющие такие параметры, как направление и условия останова. Это битовая маска для побитовых операций.
	0x1 - HOME_DIR_FIRST	Определяет направление первоначального движения мотора после поступления команды HOME. Если флаг установлен - вправо; иначе - влево.
	0x2 - HOME_DIR_SECOND	Определяет направление второго движения мотора. Если флаг установлен - вправо; иначе - влево.
	0x4 - HOME_MV_SEC_EN	Если флаг установлен, реализуется второй этап доводки в домашнюю позицию; иначе - этап пропускается.
	0x8 - HOME_HALF_MV	Если флаг установлен, в начале второго движения первые пол оборота сигналы завершения движения игнорируются.
	0x30 - HOME_STOP_FIRST_BITS	Биты, отвечающие за выбор сигнала завершения первого движения.

Continued on next page

Таблица 6.114 – continued from previous page

	0x10 - HOME_STOP_FIRST_REV	Первое движение завершается по сигналу с Revolution sensor.
	0x20 - HOME_STOP_FIRST_SYN	Первое движение завершается по сигналу со входа синхронизации.
	0x30 - HOME_STOP_FIRST_LIM	Первое движение завершается по сигналу с концевого переключателя.
	0xc0 - HOME_STOP_SECOND_BITS	Биты, отвечающие за выбор сигнала завершения второго движения.
	0x40 - HOME_STOP_SECOND_REV	Второе движение завершается по сигналу с Revolution sensor.
	0x80 - HOME_STOP_SECOND_SYN	Второе движение завершается по сигналу со входа синхронизации.
	0xc0 - HOME_STOP_SECOND_LIM	Второе движение завершается по сигналу с концевого переключателя.
	0x100 - HOME_USE_FAST	Если флаг установлен, используется быстрый поиск домашней позиции; иначе - традиционный.
uint8_t	Reserved [9]	Зарезервировано (9 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда записи настроек для подхода в home position. Эта функция записывает структуру настроек, использующихся для калибровки позиции, в память контроллера.

6.2.6.56 Команда SHSI

Код команды (CMD): «shsi» или 0x69736873.

Запрос: (70 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель. Максимальная длина строки: 16 символов.
int8_t	PartNumber	Серия и номер модели. Максимальная длина строки: 24 символа.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись информации о датчиках Холла в EEPROM. Функция должна использоваться только производителем.

6.2.6.57 Команда SHSS

Код команды (CMD): «shss» или 0x73736873.

Запрос: (50 байт)

uint32_t	CMD	Команда
float	MaxOperatingFrequency	Максимальная частота (кГц). Тип данных: float.
float	SupplyVoltageMin	Минимальное напряжение питания (В). Тип данных: float.
float	SupplyVoltageMax	Максимальное напряжение питания (В). Тип данных: float.
float	MaxCurrentConsumption	Максимальное потребление тока (мА). Тип данных: float.
uint32_t	PPR	Количество отсчётов на оборот
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек датчиков Холла в EEPROM. Функция должна использоваться только производителем.

6.2.6.58 Команда SJOY

Код команды (CMD): «sjoy» или 0x796F6A73.

Запрос: (22 байт)

uint32_t	CMD	Команда
uint16_t	JoyLowEnd	Значение в шагах джойстика, соответствующее нижней границе диапазона отклонения устройства. Должно лежать в пределах. Диапазон: 0..10000.
uint16_t	JoyCenter	Значение в шагах джойстика, соответствующее неотклонённому устройству. Должно лежать в пределах. Диапазон: 0..10000.
uint16_t	JoyHighEnd	Значение в шагах джойстика, соответствующее верхней границе диапазона отклонения устройства. Должно лежать в пределах. Диапазон: 0..10000.
uint8_t	ExpFactor	Фактор экспоненциальной нелинейности отклика джойстика.

Continued on next page

Таблица 6.120 – continued from previous page

uint8_t	DeadZone	Отклонение от среднего положения, которое не вызывает начала движения (в десятых долях процента). Максимальное мёртвое отклонение +-25.5%, что составляет половину рабочего диапазона джойстика.
uint8_t	JoyFlags	Флаги управления джойстиком. Это битовая маска для побитовых операций.
	0x1 - JOY_REVERSE	Реверс воздействия джойстика. Отклонение джойстика к большим значениям приводит к отрицательной скорости и наоборот.
uint8_t	Reserved [7]	Зарезервировано (7 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек джойстика. При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость, а 100% отклонения соответствует MaxSpeed [i] (см. команду SCTL), где i=0, если предыдущим использованием этого режима не было выбрано другое i. Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Подробнее см. раздел „Управление с помощью джойстика“ на сайте https://doc.xisupport.com.

6.2.6.59 Команда SMOV

Код команды (CMD): «smov» или 0x766F6D73.

Запрос: (30 байт)

uint32_t	CMD	Команда
uint32_t	Speed	Заданная скорость (для ШД: шагов/с, для DC: rpm). Диапазон: 0..100000.
uint8_t	uSpeed	Заданная скорость в единицах деления микрошага в секунду. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым мотором.

Continued on next page

Таблица 6.122 – continued from previous page

uint16_t	Accel	Ускорение, заданное в шагах в секунду^2 (ШД) или в оборотах в минуту за секунду (DC). Диапазон: 1..65535.
uint16_t	Decel	Торможение, заданное в шагах в секунду^2 (ШД) или в оборотах в минуту за секунду (DC). Диапазон: 1..65535.
uint32_t	AntiplaySpeed	Скорость в режиме антилюфта, заданная в целых шагах/с (ШД) или в оборотах/с(DC). Диапазон: 0..100000.
uint8_t	uAntiplaySpeed	Скорость в режиме антилюфта, выраженная в микрошагах в секунду. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым мотором.
uint8_t	MoveFlags	Флаги, управляющие настройкой движения. Это битовая маска для побитовых операций.
	0x1 - RPM_DIV_1000	Флаг указывает на то что рабочая скорость указанная в команде задана в милли rpm. Применим только для режима обратной связи ENCODER и только для BLDC моторов.
uint8_t	Reserved [9]	Зарезервировано (9 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

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

6.2.6.60 Команда SMTI

Код команды (CMD): «smti» или 0x69746D73.

Запрос: (70 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель. Максимальная длина строки: 16 символов.
int8_t	PartNumber	Серия и номер модели. Максимальная длина строки: 24 символа.

Continued on next page

Таблица 6.124 – continued from previous page

uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись информации о двигателе в EEPROM. Функция должна использоваться только производителем.

6.2.6.61 Команда SMTS

Код команды (CMD): «smts» или 0x73746D73.

Запрос: (112 байт)

uint32_t	CMD	Команда
uint8_t	MotorType	Тип двигателя. Это битовая маска для побитовых операций.
	0x0 - MOTOR_TYPE_UNKNOWN	Неизвестный двигатель
	0x1 - MOTOR_TYPE_STEP	Шаговый двигатель
	0x2 - MOTOR_TYPE_DC	DC двигатель
	0x3 - MOTOR_TYPE_BLDC	BLDC двигатель
uint8_t	ReservedField	Зарезервировано
uint16_t	Poles	Кол-во пар полюсов у DC или BLDC двигателя или кол-во шагов на оборот для шагового двигателя.
uint16_t	Phases	Кол-во фаз у BLDC двигателя.
float	NominalVoltage	Номинальное напряжение на обмотке (В). Тип данных: float
float	NominalCurrent	Максимальный постоянный ток в обмотке для DC и BLDC двигателей, номинальный ток в обмотке для шаговых двигателей (А). Тип данных: float.
float	NominalSpeed	Не используется. Номинальная скорость (об/мин). Применяется для DC и BLDC двигателей. Тип данных: float.
float	NominalTorque	Номинальный крутящий момент (мН м). Применяется для DC и BLDC двигателей. Тип данных: float.
float	NominalPower	Номинальная мощность(Вт). Применяется для DC и BLDC двигателей. Тип данных: float.
float	WindingResistance	Сопротивление обмотки DC двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC двигателя (Ом). Тип данных: float.

Continued on next page

Таблица 6.126 – continued from previous page

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

Continued on next page

Таблица 6.126 – continued from previous page

uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек двигателя в EEPROM. Функция должна использоваться только производителем.

6.2.6.62 Команда SNET

Код команды (CMD): «snet» или 0x74656E73.

Запрос: (38 байт)

uint32_t	CMD	Команда
uint8_t	DHCPEnabled	Определяет способ получения IP-адреса каналов. Может принимать значения: 0 — статически, 1 — через DHCP.
uint8_t	IPv4Address	IP-адрес устройства в формате x.x.x.x.
uint8_t	SubnetMask	Маска подсети в формате x.x.x.x.
uint8_t	DefaultGateway	Шлюз сети по умолчанию в формате x.x.x.x.
uint8_t	Reserved [19]	Зарезервировано (19 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда записи сетевых настроек. Только для производителя. Эта функция меняет сетевые настройки на заданные.

6.2.6.63 Команда SNME

Код команды (CMD): «snme» или 0x656D6E73.

Запрос: (30 байт)

uint32_t	CMD	Команда
int8_t	PositionerName	Пользовательское имя подвижки. Может быть установлено пользователем для его удобства. Максимальная длина строки: 16 символов.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись пользовательского имени подвижки в EEPROM.

6.2.6.64 Команда SNMF

Код команды (CMD): «snmf» или 0x666D6E73.

Запрос: (30 байт)

uint32_t	CMD	Команда
int8_t	ControllerName	Пользовательское имя контроллера. Может быть установлено пользователем для его удобства. Максимальная длина строки: 16 символов.
uint8_t	CtrlFlags	Настройки контроллера. Это битовая маска для побитовых операций.
	0x1 - EEPROM_PRECEDENCE	Если флаг установлен, то настройки в EEPROM подвижки имеют приоритет над текущими настройками и заменяют их при обнаружении EEPROM.
uint8_t	Reserved [7]	Зарезервировано (7 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись пользовательского имени контроллера и настроек в FRAM.

6.2.6.65 Команда SNVM

Код команды (CMD): «snvm» или 0x6D766E73.

Запрос: (36 байт)

uint32_t	CMD	Команда
uint32_t	UserData	Пользовательские данные. Могут быть установлены пользователем для его удобства. Каждый элемент массива хранит только 32 бита пользовательских данных. Это важно на системах где тип int содержит больше чем 4 байта. Например это все системы amd64.
uint8_t	Reserved [2]	Зарезервировано (2 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись пользовательских данных во FRAM.

6.2.6.66 Команда SPID

Код команды (CMD): «spid» или 0x64697073.

Запрос: (48 байт)

uint32_t	CMD	Команда
uint16_t	KpU	Пропорциональный коэффициент ПИД контура по напряжению
uint16_t	KiU	Интегральный коэффициент ПИД контура по напряжению
uint16_t	KdU	Дифференциальный коэффициент ПИД контура по напряжению
float	Kpf	Пропорциональный коэффициент ПИД контура по позиции для BLDC
float	Kif	Интегральный коэффициент ПИД контура по позиции для BLDC
float	Kdf	Дифференциальный коэффициент ПИД контура по позиции для BLDC
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись ПИД коэффициентов. Эти коэффициенты определяют поведение позиционера. Коэффициенты различны для разных позиционеров. Пожалуйста, загружайте новые настройки, когда вы меняете мотор или позиционер.

6.2.6.67 Команда SPWD

Код команды (CMD): «spwd» или 0x64777073.

Запрос: (36 байт)

uint32_t	CMD	Команда
int8_t	UserPassword	Строчка-пароль для доступа к веб-странице, который пользователь может поменять с помощью USB команды или на веб-странице.
uint8_t	Reserved [10]	Зарезервировано (10 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда записи пароля к веб-странице. Только для производителя. Эта функция меняет пользовательский пароль к веб-странице.

6.2.6.68 Команда SPWR

Код команды (CMD): «spwr» или 0x72777073.

Запрос: (20 байт)

uint32_t	CMD	Команда
uint8_t	HoldCurrent	Ток мотора в режиме удержания, в процентах от номинального. Диапазон: 0..100.
uint16_t	CurrReductDelay	Время в мс от перехода в состояние STOP до уменьшения тока.
uint16_t	PowerOffDelay	Время в с от перехода в состояние STOP до отключения питания мотора.
uint16_t	CurrentSetTime	Время в мс, требуемое для набора номинального тока от 0% до 100%.
uint8_t	PowerFlags	Флаги параметров управления питанием. Это битовая маска для побитовых операций.
	0x1 - POWER_REDUCT_ENABLED	Если флаг установлен, уменьшить ток по прошествии CurrReductDelay. Иначе - не уменьшать.
	0x2 - POWER_OFF_ENABLED	Если флаг установлен, снять напряжение с обмоток по прошествии PowerOffDelay. Иначе - не снимать.
	0x4 - POWER_SMOOTH_CURRENT	Если установлен, то запитывание обмоток, снятие питания или снижение/повышение тока происходят плавно со скоростью CurrentSetTime, а только потом выполняется та задача, которая вызвала это плавное изменение.
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда записи параметров питания мотора. Используется только с шаговым двигателем.

6.2.6.69 Команда SSEC

Код команды (CMD): «ssec» или 0x63657373.

Запрос: (28 байт)

uint32_t	CMD	Команда
uint16_t	LowUpwrOff	Нижний порог напряжения на силовой части для выключения, десятки мВ.
uint16_t	CriticalIpwr	Максимальный ток силовой части, вызывающий состояние ALARM, в мА.
uint16_t	CriticalUpwr	Максимальное напряжение на силовой части, вызывающее состояние ALARM, десятки мВ.
uint16_t	CriticalT	Максимальная температура контроллера, вызывающая состояние ALARM, в десятых долях градуса Цельсия.
uint16_t	CriticalIusb	Максимальный ток USB, вызывающий состояние ALARM, в мА.
uint16_t	CriticalUusb	Максимальное напряжение на USB, вызывающее состояние ALARM, десятки мВ.
uint16_t	MinimumUusb	Минимальное напряжение на USB, вызывающее состояние ALARM, десятки мВ.
uint8_t	Flags	Флаги критических параметров. Это битовая маска для побитовых операций.
	0x1 - ALARM_ON_DRIVER_OVERHEATING	Если флаг установлен, то войти в состояние Alarm при получении сигнала подступающего перегрева с драйвера. Иначе - игнорировать подступающий перегрев с драйвера.
	0x2 - LOW_UPWR_PROTECTION	Если установлен, то выключать силовую часть при напряжении меньшем LowUpwrOff.
	0x4 - H_BRIDGE_ALERT	Если установлен, то выключать силовую часть при сигнале неполадки в одном из транзисторных мостов.
	0x8 - ALARM_ON_BORDERS_SWAP_MISSET	Если флаг установлен, то войти в состояние Alarm при получении сигнала с противоположного концевого выключателя
	0x10 - ALARM_FLAGS_STICKING	Если флаг установлен, то только по команде STOP возможен сброс всех флагов ALARM
	0x20 - USB_BREAK_RECONNECT	Если флаг установлен, то будет включен блок перезагрузки USB при поломке связи

Continued on next page

Таблица 6.142 – continued from previous page

	0x40 - ALARM_WINDING_MISMATCH	Если флаг установлен, то войти в состояние Alarm при получении сигнала рассогласования обмоток
	0x80 - ALARM_ENGINE_RESPONSE	Если флаг установлен, то войти в состояние Alarm при получении сигнала ошибки реакции двигателя на управляющее воздействие
uint8_t	Reserved [7]	Зарезервировано (7 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда записи установок защит.

6.2.6.70 Команда SSNI

Код команды (CMD): «ssni» или 0x696E7373.

Запрос: (28 байт)

uint32_t	CMD	Команда
uint8_t	SyncInFlags	Флаги синхронизации входа. Это битовая маска для побитовых операций.
	0x1 - SYNCIN_ENABLED	Включение необходимости импульса синхронизации для начала движения.
	0x2 - SYNCIN_INVERT	Если установлен - срабатывает по переходу из 1 в 0. Иначе - из 0 в 1.
	0x4 - SYNCIN_GOTOPOSITION	Если флаг установлен, то двигатель смещается к позиции, установленной в Position и uPosition, иначе двигатель смещается на Position и uPosition
uint16_t	ClutterTime	Минимальная длительность входного импульса синхронизации для защиты от дребезга (мкс).
int32_t	Position	Желаемая позиция или смещение (в полных шагах)
int16_t	uPosition	Дробная часть позиции или смещения в микрошагах. Используется только с шаговым двигателем. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

Continued on next page

Таблица 6.144 – continued from previous page

uint32_t	Speed	Заданная скорость (для ШД: шагов/с, для DC: rpm). Диапазон: 0..100000.
uint8_t	uSpeed	Заданная скорость в микрошагах в секунду. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым мотором.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек для входного импульса синхронизации. Эта функция записывает структуру с настройками входного импульса синхронизации, определяющими поведение входа синхронизации, в память контроллера.

6.2.6.71 Команда SSNO

Код команды (CMD): «ssno» или 0x6F6E7373.

Запрос: (16 байт)

uint32_t	CMD	Команда
uint8_t	SyncOutFlags	Флаги синхронизации выхода. Это битовая маска для побитовых операций.
	0x1 - SYNCOUT_ENABLED	Синхронизация выхода работает согласно настройкам, если флаг установлен. В ином случае значение выхода фиксировано и подчиняется SYNCOUT_STATE.
	0x2 - SYNCOUT_STATE	Когда значение выхода управляется напрямую (см. флаг SYNCOUT_ENABLED), значение на выходе соответствует значению этого флага.
	0x4 - SYNCOUT_INVERT	Нулевой логический уровень является активным, если флаг установлен, а единичный - если флаг сброшен.
	0x8 - SYNCOUT_IN_STEPS	Если флаг установлен использовать шаги/импульсы энкодера для выходных импульсов синхронизации вместо миллисекунд.

Continued on next page

Таблица 6.146 – continued from previous page

	0x10 - SYNCOUT_ONSTART	Генерация синхронизирующего импульса при начале движения.
	0x20 - SYNCOUT_ONSTOP	Генерация синхронизирующего импульса при остановке.
	0x40 - SYNCOUT_ONPERIOD	Выдает импульс синхронизации после прохождения SyncOutPeriod отсчётов.
uint16_t	SyncOutPulseSteps	Определяет длительность выходных импульсов в шагах/импульсах энкодера, когда установлен флаг SYNCOUT_IN_STEPS, или в микросекундах если флаг сброшен.
uint16_t	SyncOutPeriod	Период генерации импульсов (в шагах/отсчетах энкодера), используется при установленном флаге SYNCOUT_ONPERIOD.
uint32_t	Accuracy	Это окрестность вокруг целевой координаты, попадание в которую считается попаданием в целевую позицию и генерируется импульс по остановке.
uint8_t	uAccuracy	Это окрестность вокруг целевой координаты в микрошагах (используется только с шаговым двигателем). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек для выходного импульса синхронизации. Эта функция записывает структуру с настройками выходного импульса синхронизации, определяющими поведение вывода синхронизации, в память контроллера.

6.2.6.72 Команда SSTI

Код команды (CMD): «ssti» или 0x69747373.

Запрос: (70 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель. Максимальная длина строки: 16 символов.
int8_t	PartNumber	Серия и номер модели. Максимальная длина строки: 24 символа.
uint8_t	Reserved [24]	Зарезервировано (24 байт)

Continued on next page

Таблица 6.148 – continued from previous page

uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись информации о позиционере в EEPROM. Не поддерживается. Функция должна использоваться только производителем.

6.2.6.73 Команда SSTS

Код команды (CMD): «ssts» или 0x73747373.

Запрос: (70 байт)

uint32_t	CMD	Команда
float	LeadScrewPitch	Шаг ходового винта в мм. Тип данных: float.
int8_t	Units	Единицы измерения расстояния, используемые в полях MaxSpeed и TravelRange (шаги, градусы, мм, ...), Максимальная длина строки: 8 символов.
float	MaxSpeed	Максимальная скорость (Units/c). Тип данных: float.
float	TravelRange	Диапазон перемещения (Units). Тип данных: float.
float	SupplyVoltageMin	Минимальное напряжение питания (В). Тип данных: float.
float	SupplyVoltageMax	Максимальное напряжение питания (В). Тип данных: float.
float	MaxCurrentConsumption	Максимальный ток потребления (А). Тип данных: float.
float	HorizontalLoadCapacity	Горизонтальная грузоподъемность (кг). Тип данных: float.
float	VerticalLoadCapacity	Вертикальная грузоподъемность (кг). Тип данных: float.
uint8_t	Reserved [24]	Зарезервировано (24 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись настроек позиционера в EEPROM. Функция должна использоваться только производителем

6.2.6.74 Команда SURT

Код команды (CMD): «surt» или 0x74727573.

Запрос: (16 байт)

uint32_t	CMD	Команда
uint32_t	Speed	Скорость UART (в бодах)
uint16_t	UARTSetupFlags	Флаги настройки UART. Это битовая маска для побитовых операций.
	0x3 - UART_PARITY_BITS	Биты, отвечающие за выбор четности.
	0x0 - UART_PARITY_BIT_EVEN	Бит 1, если четный
	0x1 - UART_PARITY_BIT_ODD	Бит 1, если нечетный
	0x2 - UART_PARITY_BIT_SPACE	Бит четности всегда 0
	0x3 - UART_PARITY_BIT_MARK	Бит четности всегда 1
	0x4 - UART_PARITY_BIT_USE	Бит чётности не используется, если „0“; бит четности используется, если „1“
	0x8 - UART_STOP_BIT	Если установлен, один стоповый бит; иначе - 2 стоповых бита
uint8_t	Reserved [4]	Зарезервировано (4 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда записи настроек UART. Эта функция записывает структуру настроек UART в память контроллера.

6.2.6.75 Команда ASIA

Код команды (CMD): «asia» или 0x61697361.

Запрос: (22 байт)

uint32_t	CMD	Команда
int32_t	Position	Желаемая позиция или смещение (в полных шагах)
int16_t	uPosition	Дробная часть позиции или смещения в микрошагах. Используется только с шаговым двигателем. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint32_t	Time	Время, за которое требуется достичь требуемой позиции, в микросекундах.
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Это команда добавляет один элемент в буфер FIFO команд, выполняемых при получении входного импульса синхронизации. Каждый импульс синхронизации либо выполнится то действие, которое описано в SSNI, если буфер пуст, либо самое старое из загруженных в буфер действий временно подменяет скорость и координату в SSNI. В последнем случае это действие стирается из буфера. Количество оставшихся пустыми элементов буфера можно узнать в структуре GETS.

6.2.6.76 Команда CLFR

Код команды (CMD): «clfr» или 0x72666C63.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда очистки FRAM контроллера. Память очищается путем заполнения всего объема памяти байтами 0x00. После очистки контроллер перезагружается. Ответа на эту команду нет.

6.2.6.77 Команда CONN

Код команды (CMD): «conn» или 0x6E6E6F63.

Запрос: (14 байт)

uint32_t	CMD	Команда
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (15 байт)

uint32_t	CMD	Команда
uint8_t	sresult	Результат выполнения команды.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда служит для открытия сеанса ISP (in-system programming) при загрузке прошивки. Result = RESULT_OK, если команда выполнена загрузчиком. Result = RESULT_SOFT_ERROR, если во время выполнения команды произошла ошибка. Result не доступен через функцию библиотеки command_update_firmware, значение поля обрабатывается внутри функции.

6.2.6.78 Команда DBGR

Код команды (CMD): «dbgr» или 0x72676264.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (142 байт)

uint32_t	CMD	Команда
uint8_t	DebugData	Отладочные данные.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение данных из прошивки для отладки и поиска неисправностей. Команда только для производителя. Получаемые данные зависят от версии прошивки, истории и контекста использования.

6.2.6.79 Команда DBGW

Код команды (CMD): «dbgw» или 0x77676264.

Запрос: (142 байт)

uint32_t	CMD	Команда
uint8_t	DebugData	Отладочные данные.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись данных в прошивку для отладки и поиска неисправностей. Команда только для производителя.

6.2.6.80 Команда DISC

Код команды (CMD): «disc» или 0x63736964.

Запрос: (14 байт)

uint32_t	CMD	Команда
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (15 байт)

uint32_t	CMD	Команда
uint8_t	sresult	Результат выполнения команды.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда служит для закрытия сеанса ISP (in-system programming) при загрузке прошивки. Result = RESULT_OK, если команда выполнена загрузчиком. Result = RESULT_HARD_ERROR, если во время выполнения команды произошла аппаратная ошибка. Result = RESULT_SOFT_ERROR, если во время выполнения команды произошла программная ошибка. Result не доступен через функцию библиотеки command_update_firmware, значение поля обрабатывается внутри функции.

6.2.6.81 Команда EERD

Код команды (CMD): «eerd» или 0x64726565.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Чтение настроек контроллера из EEPROM памяти позиционера. Эта операция также автоматически выполняется при подключении позиционера с EEPROM памятью. Функция должна использоваться только производителем.

6.2.6.82 Команда EESV

Код команды (CMD): «eesv» или 0x76736565.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

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

6.2.6.83 Команда GBLV

Код команды (CMD): «gblv» или 0x766C6267.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (10 байт)

uint32_t	CMD	Команда
uint8_t	Major	Мажорный номер версии загрузчика
uint8_t	Minor	Минорный номер версии загрузчика
uint16_t	Release	Номер релиза версии загрузчика
uint16_t	CRC	Контрольная сумма

Описание: Чтение номера версии загрузчика контроллера.

6.2.6.84 Команда GETC

Код команды (CMD): «getc» или 0x63746567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (38 байт)

uint32_t	CMD	Команда
int16_t	WindingVoltageA	В случае ШД, напряжение на обмотке A (в десятках мВ); в случае бесщеточного, напряжение на первой обмотке; в случае DC - на единственной.
int16_t	WindingVoltageB	В случае ШД, напряжение на обмотке B (в десятках мВ); в случае бесщеточного, напряжение на второй обмотке; в случае DC не используется.
int16_t	WindingVoltageC	В случае бесщеточного, напряжение на третьей обмотке (в десятках мВ); в случае ШД и DC не используется.
int16_t	WindingCurrentA	В случае ШД, ток в обмотке A (в мА); в случае бесщеточного, ток в первой обмотке; в случае DC в единственной.
int16_t	WindingCurrentB	В случае ШД, ток в обмотке B (в мА); в случае бесщеточного, ток в второй обмотке; в случае DC не используется.
int16_t	WindingCurrentC	В случае бесщеточного, ток в третьей обмотке (в мА); в случае ШД и DC не используется.
uint16_t	Pot	Значение на аналоговом входе. Диапазон: 0..10000
uint16_t	Joy	Положение джойстика в десятитысячных долях. Диапазон: 0..10000
int16_t	DutyCycle	Коэффициент заполнения ШИМ.
uint8_t	Reserved [14]	Зарезервировано (14 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда чтения состояния обмоток и других не часто используемых данных. Предназначена в первую очередь для получения данных для построения графиков в паре с командой GETS.

6.2.6.85 Команда GETI

Код команды (CMD): «geti» или 0x69746567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (36 байт)

uint32_t	CMD	Команда
int8_t	Производитель	Производитель
int8_t	ManufacturerId	Идентификатор производителя
int8_t	ProductDescription	Описание продукта
uint8_t	Major	Основной номер версии железа.
uint8_t	Minor	Второстепенный номер версии железа.
uint16_t	Release	Номер правок этой версии железа.
uint8_t	Reserved [12]	Зарезервировано (12 байт)
uint16_t	CRC	Контрольная сумма

Описание: Возвращает информацию об устройстве. Доступна как из прошивки, так и из бутлоадера.

6.2.6.86 Команда GETM

Код команды (CMD): «getm» или 0x6D746567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (216 байт)

uint32_t	CMD	Команда
int32_t	Speed	Текущая скорость в микрошагах в секунду (целые шаги пересчитываются с учетом текущего режима деления шага) или отсчетах энкодера в секунду.
int32_t	Error	Текущая скорость в микрошагах в секунду (целые шаги пересчитываются с учетом текущего режима деления шага) или отсчетах энкодера в секунду.
uint32_t	Length	Длина фактических данных в буфере.
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда чтения буфера данных для построения графиков скорости и ошибки следования. Заполнение буфера начинается по команде „start_measurements“. Буфер вмещает 25 точек, точки снимаются с периодом 1 мс. Для создания устойчивой системы следует считывать данные каждые 20 мс, если буфер полностью заполнен, то рекомендуется повторять считывания каждые 5 мс до момента пока буфер вновь не станет заполнен 20-ю точками.

6.2.6.87 Команда GETS

Код команды (CMD): «gets» или 0x73746567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (54 байт)

uint32_t	CMD	Команда
uint8_t	MoveSts	Состояние движения. Это битовая маска для побитовых операций.
	0x1 - MOVE_STATE_MOVING	Если флаг установлен, то контроллер пытается вращать двигателем. Не используйте этот флаг для ожидания завершения команды движения. Вместо него используйте MVCMD_RUNNING из поля MvCmdSts.
	0x2 - MOVE_STATE_TARGET_SPEED	Флаг устанавливается при достижении заданной скорости.
	0x4 - MOVE_STATE_ANTIPLAY	Выполняется компенсация люфта, если флаг установлен.
uint8_t	MvCmdSts	Состояние команды движения (касается command_move, command_movr, command_left, command_right, command_stop, command_home, command_loft). Это битовая маска для побитовых операций.
	0x3f - MVCMD_NAME_BITS	Битовая маска активной команды.
	0x0 - MVCMD_UKNWN	Неизвестная команда.
	0x1 - MVCMD_MOVE	Команда move.
	0x2 - MVCMD_MOVR	Команда movr.
	0x3 - MVCMD_LEFT	Команда left.
	0x4 - MVCMD_RIGHT	Команда rigt.
	0x5 - MVCMD_STOP	Команда stop.
	0x6 - MVCMD_HOME	Команда home.
	0x7 - MVCMD_LOFT	Команда loft.
	0x8 - MVCMD_SSTP	Команда плавной остановки(SSTP).
	0x40 - MVCMD_ERROR	Состояние завершения движения (1 - команда движения выполнена с ошибкой, 0 - команда движения выполнена корректно). Имеет смысл если MVCMD_RUNNING указывает на завершение движения.
	0x80 - MVCMD_RUNNING	Состояние команды движения (0 - команда движения выполнена, 1 - команда движения сейчас выполняется).

Continued on next page

Таблица 6.179 – continued from previous page

uint8_t	PWRSts	Состояние питания шагового двигателя (используется только с шаговым двигателем). Это битовая маска для побитовых операций.
	0x0 - PWR_STATE_UNKNOWN	Неизвестное состояние, которое не должно никогда реализовываться.
	0x1 - PWR_STATE_OFF	Обмотки мотора разомкнуты и не управляются драйвером.
	0x3 - PWR_STATE_NORM	Обмотки запитаны номинальным током.
	0x4 - PWR_STATE_REDUCT	Обмотки намеренно запитаны уменьшенным током от рабочего для снижения потребляемой мощности.
	0x5 - PWR_STATE_MAX	Обмотки двигателя питаются от максимального тока, который драйвер может обеспечить при этом напряжении.
uint8_t	EncSts	Состояние энкодера. Это битовая маска для побитовых операций.
	0x0 - ENC_STATE_ABSENT	Энкодер не подключен.
	0x1 - ENC_STATE_UNKNOWN	Состояние энкодера неизвестно.
	0x2 - ENC_STATE_MALFUNC	Энкодер подключен и неисправен.
	0x3 - ENC_STATE_REVERS	Энкодер подключен и исправен, но считает в другую сторону.
	0x4 - ENC_STATE_OK	Энкодер подключен и работает должным образом.
uint8_t	WindSts	Состояние обмоток. Это битовая маска для побитовых операций.
	0x0 - WIND_A_STATE_ABSENT	Обмотка A не подключена.
	0x1 - WIND_A_STATE_UNKNOWN	Состояние обмотки A неизвестно.
	0x2 - WIND_A_STATE_MALFUNC	Короткое замыкание на обмотке A.
	0x3 - WIND_A_STATE_OK	Обмотка A работает адекватно.
	0x0 - WIND_B_STATE_ABSENT	Обмотка B не подключена.
	0x10 - WIND_B_STATE_UNKNOWN	Состояние обмотки B неизвестно.
	0x20 - WIND_B_STATE_MALFUNC	Короткое замыкание на обмотке B.
	0x30 - WIND_B_STATE_OK	Обмотка B работает адекватно.
int32_t	CurPosition	Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь. В случае работы с DC-мотором в этом поле находится текущая позиция по данным с энкодера, в случае работы с ШД-мотором в режиме, когда первичными являются импульсы, подаваемые на мотор, в этом поле содержится целое значение шагов текущей позиции.

Continued on next page

Таблица 6.179 – continued from previous page

int16_t	uCurPosition	Дробная часть текущей позиции в микрошагах. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым двигателем.
int64_t	EncPosition	Текущая позиция по данным с энкодера в импульсах энкодера, используется только если энкодер установлен, активизирован и не является основным датчиком положения, например при использовании энкодера совместно с шаговым двигателем для контроля проскальзования.
int32_t	CurSpeed	Текущая скорость.
int16_t	uCurSpeed	Дробная часть текущей скорости в микрошагах. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым двигателем.
int16_t	Ipwr	Ток потребления силовой части, мА.
int16_t	Upwr	Напряжение на силовой части, десятки мВ.
int16_t	Iusb	Ток потребления по USB, мА.
int16_t	Uusb	Напряжение на USB, десятки мВ.
int16_t	CurT	Температура процессора в десятых долях градусов Цельсия.
uint32_t	Flags	Флаги состояний. Это битовая маска для побитовых операций.
	0x3f - STATE_CONTR	Флаги состояния контроллера.
	0x1 - STATE_ERRC	Недопустимая команда. Полученная команда отсутствует в списке известных команд контроллера. Наиболее вероятной причиной является устаревшая прошивка.
	0x2 - STATE_ERRD	Обнаружена ошибка целостности данных. Данные внутри команды и ее CRC-код не соответствуют, поэтому данные не могут считаться действительными. Эта ошибка может быть вызвана электромагнитными помехами в интерфейсе UART/RS232.

Continued on next page

Таблица 6.179 – continued from previous page

	0x4 - STATE_ERRV	Недопустимое значение данных. Обнаружена ошибка в значении. Значения в команде не могут быть применены без коррекции, поскольку они выходят за допустимый диапазон. Вместо исходных значений были использованы исправленные значения.
	0x10 - STATE_EEPROM_CONNECTED	Подключена память EEPROM с настройками. Встроенный профиль подвижки загружается из микросхемы памяти EEPROM, что позволяет подключать различные подвижки к контроллеру с автоматической настройкой.
	0x20 - STATE_IS_HOMED	Калибровка выполнена. Это означает, что шкала относительного положения откалибрована с помощью аппаратного датчика абсолютного положения, такого как концевой переключатель.
	0x1b3ffc0 - STATE_SECUR	Флаги опасности.
	0x40 - STATE_ALARM	Контроллер находится в состоянии ALARM, показывая, что случилась какая-то опасная ситуация. В состоянии ALARM все команды игнорируются пока не будет послана команда STOP и состояние ALARM деактивируется.
	0x80 - STATE_CTP_ERROR	Контроль позиции нарушен(используется только с шаговым двигателем). Флаг устанавливается, когда положение энкодера и положение шага слишком далеки друг от друга.
	0x100 - STATE_POWER_OVERHEAT	Перегрев силового драйвера. Управление двигателем отключено до восстановления рабочей температуры драйвера. Этого не должно происходить в коробочных версиях контроллера. Это может произойти в версии контроллера с «голой» платой и с пользовательским радиатором. Решение: используйте другой радиатор.
	0x200 - STATE_CONTROLLER_OVERHEAT	Перегрелась микросхема контроллера.
	0x400 - STATE_OVERLOAD_POWER_VOLTAGE	Превышено напряжение на силовой части.
	0x800 - STATE_OVERLOAD_POWER_CURRENT	Превышен максимальный ток потребления силовой части.
	0x1000 - STATE_OVERLOAD_USB_VOLTAGE	Превышено напряжение на USB.

Continued on next page

Таблица 6.179 – continued from previous page

	0x2000 - STATE_LOW_USB_VOLTAGE	Слишком низкое напряжение на USB.
	0x4000 - STATE_OVERLOAD_USB_CURRENT	Превышен максимальный ток потребления USB.
	0x8000 - STATE_BORDERS_SWAP_MISSET	Достижение неверной границы.
	0x10000 - STATE_LOW_POWER_VOLTAGE	Напряжение на силовой части ниже чем напряжение Low Voltage Protection
	0x20000 - STATE_H_BRIDGE_FAULT	Получен сигнал от драйвера о неисправности
	0x100000 - STATE_WINDING_RES_MISMATCH	Сопротивления обмоток слишком сильно отличаются друг от друга. Обычно это происходит с поврежденным шаговым двигателем у которого полностью или частично закорочены обмотки.
	0x200000 - STATE_ENCODER_FAULT	Получен сигнал от энкодера о неисправности
	0x800000 - STATE_ENGINE_RESPONSE_ERROR	Ошибка реакции двигателя на управляющее воздействие. Отказ алгоритма управления двигателем означает, что он не может определять правильные решения с помощью полученных данных обратной связи. Единичный отказ может быть вызван механической проблемой. Повторяющийся сбой может быть вызван неправильной настройкой двигателя.
	0x1000000 - STATE_EXTIO_ALARM	Ошибка вызвана внешним входным сигналом EXTIO.
uint32_t	GPIOFlags	Флаги состояний GPIO входов. Это битовая маска для побитовых операций.
	0xffff - STATE_DIG_SIGNAL	Флаги цифровых сигналов.
	0x1 - STATE_RIGHT_EDGE	Достижение правой границы.
	0x2 - STATE_LEFT_EDGE	Достижение левой границы.
	0x4 - STATE_BUTTON_RIGHT	Состояние кнопки „вправо“ (1, если нажата).
	0x8 - STATE_BUTTON_LEFT	Состояние кнопки „влево“ (1, если нажата).
	0x10 - STATE_GPIO_PINOUT	Если флаг установлен, ввод/вывод общего назначения работает как выход; если флаг сброшен, ввод/вывод работает как вход.
	0x20 - STATE_GPIO_LEVEL	Состояние ввода/вывода общего назначения.
	0x200 - STATE_BRAKE	Состояние вывода управления тормозом. Флаг „1“ - если тормоз не запитан(зажат), „0“ - если на тормоз подаётся питание(разжат).

Continued on next page

Таблица 6.179 – continued from previous page

	0x400 - STATE_REV_SENSOR	Состояние вывода датчика оборотов(флаг „1“, если датчик активен).
	0x800 - STATE_SYNC_INPUT	Состояние входа синхронизации(1, если вход синхронизации активен).
	0x1000 - STATE_SYNC_OUTPUT	Состояние выхода синхронизации(1, если выход синхронизации активен).
	0x2000 - STATE_ENC_A	Состояние ножки A энкодера(флаг „1“, если энкодер активен).
	0x4000 - STATE_ENC_B	Состояние ножки B энкодера(флаг „1“, если энкодер активен).
uint8_t	CmdBufFreeSpace	Данное поле служебное. Оно показывает количество свободных ячеек буфера цепочки синхронизации.
uint8_t	Reserved [4]	Зарезервировано (4 байт)
uint16_t	CRC	Контрольная сумма

Описание: Возвращает информацию о текущем состоянии устройства.

6.2.6.88 Команда GFWV

Код команды (CMD): «gfwv» или 0x76776667.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (10 байт)

uint32_t	CMD	Команда
uint8_t	Major	Мажорный номер версии прошивки
uint8_t	Minor	Минорный номер версии прошивки
uint16_t	Release	Номер релиза версии прошивки
uint16_t	CRC	Контрольная сумма

Описание: Чтение номера версии прошивки контроллера.

6.2.6.89 Команда GOFW

Код команды (CMD): «gofw» или 0x77666F67.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (15 байт)

uint32_t	CMD	Команда
uint8_t	sresult	Результат выполнения команды.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда инициирует передачу управления прошивке. Эта команда так же доступна из прошивки, для совместимости. Только для производителя. Result = RESULT_OK, если переход из загрузчика в прошивку возможен. После ответа на эту команду выполняется переход. Result = RESULT_NO_FIRMWARE, если прошивка не найдена. Result = RESULT_ALREADY_IN_FIRMWARE, если эта команда была вызвана из прошивки.

6.2.6.90 Команда GPOS

Код команды (CMD): «gpos» или 0x736F7067.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (26 байт)

uint32_t	CMD	Команда
int32_t	Position	Позиция в основных шагах двигателя
int16_t	uPosition	Позиция в микрошагах (используется только с шаговыми двигателями). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
int64_t	EncPosition	Позиция энкодера.
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

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

6.2.6.91 Команда GSER

Код команды (CMD): «gser» или 0x72657367.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (10 байт)

uint32_t	CMD	Команда
uint32_t	SerialNumber	Серийный номер платы.
uint16_t	CRC	Контрольная сумма

Описание: Чтение серийного номера контроллера.

6.2.6.92 Команда GUID

Код команды (CMD): «guid» или 0x64697567.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (40 байт)

uint32_t	CMD	Команда
uint32_t	UniqueID0	Уникальный ID 0.
uint32_t	UniqueID1	Уникальный ID 1.
uint32_t	UniqueID2	Уникальный ID 2.
uint32_t	UniqueID3	Уникальный ID 3.
uint8_t	Reserved [18]	Зарезервировано (18 байт)
uint16_t	CRC	Контрольная сумма

Описание: Считывает уникальный идентификатор каждого чипа, это значение не является случайным. Только для производителя. Уникальный идентификатор может быть использован в качестве инициализационного вектора для операций шифрования бутлоадера или в качестве серийного номера для USB и других применений.

6.2.6.93 Команда HASF

Код команды (CMD): «hasf» или 0x66736168.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (15 байт)

uint32_t	CMD	Команда
uint8_t	sresult	Результат выполнения команды.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда определяет наличие в контроллере ПО. Только для производителя. Данная команда доступна так же из прошивки. Result = RESULT_NO_FIRMWARE, если прошивка не найдена. Result = RESULT_HAS_FIRMWARE, если прошивка найдена.

6.2.6.94 Команда HOME

Код команды (CMD): «home» или 0x656D6F68.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Движение в домашнюю позицию. Алгоритм движения: 1) Двигает мотор согласно скоростям FastHome, uFastHome и флагу HOME_DIR_FAST до достижения концевого выключателя, если флаг HOME_STOP_ENDS установлен. Или двигает до достижения сигнала с входа синхронизации, если установлен флаг HOME_STOP_SYNC. Или до поступления сигнала с датчика оборотов, если установлен флаг HOME_STOP_REV_SN 2) далее двигает согласно скоростям SlowHome, uSlowHome и флагу HOME_DIR_SLOW до достижения сигнала с входа синхронизации, если установлен флаг HOME_MV_SEC. Если флаг HOME_MV_SEC сброшен, пропускаем этот пункт. 3) далее двигает мотор согласно скоростям FastHome, uFastHome и флагу HOME_DIR_SLOW на расстояние HomeDelta, uHomeDelta. Описание флагов и переменных см. описание команд GHOM/SHOM.

6.2.6.95 Команда IRND

Код команды (CMD): «irnd» или 0x646E7269.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (24 байт)

uint32_t	CMD	Команда
uint8_t	key	Случайный ключ.
uint8_t	Reserved [2]	Зарезервировано (2 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение случайного числа из контроллера. Только для производителя.

6.2.6.96 Команда LEFT

Код команды (CMD): «left» или 0x7466656C.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: При получении команды „left“ двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), влево.

6.2.6.97 Команда LOFT

Код команды (CMD): «loft» или 0x74666F6C.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: При получении команды „loft“ двигатель смещается из текущей точки на расстояние Antiplay, заданное в настройках мотора (engine_settings), затем двигается в ту же точку.

6.2.6.98 Команда MOVE

Код команды (CMD): «move» или 0x65766F6D.

Запрос: (18 байт)

uint32_t	CMD	Команда
int32_t	Position	Желаемая позиция (в целых шагах или отсчетах энкодера).
int16_t	uPosition	Дробная часть позиции в микрошагах. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым двигателем.
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: При получении команды „move“ двигатель начинает перемещаться (если не используется режим „ТТЛСинхроВхода“), с заранее установленными параметрами (скорость, ускорение, удержание), к точке указанной в полях Position, uPosition. Для шагового мотора uPosition задает значение микрошага, для DC мотора это поле не используется.

6.2.6.99 Команда MOVR

Код команды (CMD): «movr» или 0x72766F6D.

Запрос: (18 байт)

uint32_t	CMD	Команда
int32_t	DeltaPosition	Смещение (дельта) позиции (в целых шагах или отсчетах энкодера)

Continued on next page

Таблица 6.202 – continued from previous page

int16_t	uDeltaPosition	Дробная часть смещения в микрошагах, используется только с шаговым двигателем. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
uint8_t	Reserved [6]	Зарезервировано (6 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Перемещение на заданное смещение. При получении команды „movr“ двигатель начинает смещаться (если не используется режим „ТТЛСинхроВхода“), с заранее установленными параметрами (скорость, ускорение, удержание), влево или вправо (зависит от знака DeltaPosition) на количество импульсов указанное в полях DeltaPosition, uDeltaPosition. Для шагового мотора uDeltaPosition задает значение микрошага, для DC мотора это поле не используется.

6.2.6.100 Команда PWOF

Код команды (CMD): «pwof» или 0x666F7770.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Немедленное отключение питания двигателя вне зависимости от его состояния. Команда предназначена для ручного управления питанием двигателя. Не следует использовать эту команду для отключения двигателя во время движения, так как питание может снова включиться для завершения движения. Для автоматического управления питанием двигателя и его отключения после остановки следует использовать систему управления электропитанием.

6.2.6.101 Команда RDAN

Код команды (CMD): «rdan» или 0x6E616472.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (76 байт)

uint32_t	CMD	Команда

Continued on next page

Таблица 6.207 – continued from previous page

uint16_t	A1Voltage_ADC	„Выходное напряжение на 1 выводе обмотки A“ необработанные данные с АЦП.
uint16_t	A2Voltage_ADC	„Выходное напряжение на 2 выводе обмотки A“ необработанные данные с АЦП.
uint16_t	B1Voltage_ADC	„Выходное напряжение на 1 выводе обмотки B“ необработанные данные с АЦП.
uint16_t	B2Voltage_ADC	„Выходное напряжение на 2 выводе обмотки B“ необработанные данные с АЦП.
uint16_t	SupVoltage_ADC	„Напряжение питания ключей H-моста“ необработанные данные с АЦП.
uint16_t	ACurrent_ADC	„Ток через обмотку A“ необработанные данные с АЦП.
uint16_t	BCurrent_ADC	„Ток через обмотку B“ необработанные данные с АЦП.
uint16_t	FullCurrent_ADC	„Полный ток“ необработанные данные с АЦП.
uint16_t	Temp_ADC	Напряжение с датчика температуры, необработанные данные с АЦП.
uint16_t	Joy_ADC	Джойстик, необработанные данные с АЦП.
uint16_t	Pot_ADC	Напряжение на аналоговом входе, необработанные данные с АЦП
uint16_t	L5_ADC	Напряжение питания USB после current sense резистора, необработанные данные с АЦП.
uint16_t	H5_ADC	Напряжение питания USB, необработанные данные с АЦП
int16_t	A1Voltage	„Выходное напряжение на 1 выводе обмотки A“ откалиброванные данные (в десятках мВ).
int16_t	A2Voltage	„Выходное напряжение на 2 выводе обмотки A“ откалиброванные данные (в десятках мВ).
int16_t	B1Voltage	„Выходное напряжение на 1 выводе обмотки B“ откалиброванные данные (в десятках мВ).
int16_t	B2Voltage	„Выходное напряжение на 2 выводе обмотки B“ откалиброванные данные (в десятках мВ).
int16_t	SupVoltage	„Напряжение питания ключей H-моста“ откалиброванные данные (в десятках мВ).
int16_t	ACurrent	„Ток через обмотку A“ откалиброванные данные (в мА).
int16_t	BCurrent	„Ток через обмотку B“ откалиброванные данные (в мА).

Continued on next page

Таблица 6.207 – continued from previous page

int16_t	FullCurrent	„Полный ток“ откалиброванные данные (в мА).
int16_t	Temp	Температура, откалиброванные данные (в десятых долях градуса Цельсия).
int16_t	Joy	Джойстик во внутренних единицах. Диапазон: 0..10000
int16_t	Pot	Аналоговый вход во внутренних единицах. Диапазон: 0..10000
int16_t	L5	Напряжение питания USB после current sense резистора (в десятках мВ).
int16_t	H5	Напряжение питания USB (в десятках мВ).
uint16_t	deprecated	
int32_t	R	Сопротивление обмоток двигателя(для шагового двигателя), в мОм.
int32_t	L	Псевдоиндуктивность обмоток двигателя(для шагового двигателя), в мкГн.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Описание: Чтение аналоговых данных, содержащих данные с АЦП и нормированные значения величин. Эта функция используется для тестирования и калибровки устройства.

6.2.6.102 Команда READ

Код команды (CMD): «read» или 0x64616572.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

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

6.2.6.103 Команда RERS

Код команды (CMD): «rers» или 0x73726572.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

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

6.2.6.104 Команда REST

Код команды (CMD): «rest» или 0x74736572.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда сброса контроллера и перехода в режим загрузчика, добавлена для совместимости с Протоколом Обмена Загрузчика. Ответа на эту команду нет.

6.2.6.105 Команда RIGT

Код команды (CMD): «rigt» или 0x74676972.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: При получении команды „rigt“ двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), вправо.

6.2.6.106 Команда SARS

Код команды (CMD): «sars» или 0x73726173.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: При получении команды контроллер выполняет операцию сохранения важных настроек (калибровочные коэффициенты и т.п.) во встроенную энергонезависимую память контроллера. Только для производителя.

6.2.6.107 Команда SAVE

Код команды (CMD): «save» или 0x65766173.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: При получении команды контроллер выполняет операцию сохранения текущих настроек во встроенную энергонезависимую память контроллера.

6.2.6.108 Команда SPOS

Код команды (CMD): «spos» или 0x736F7073.

Запрос: (26 байт)

uint32_t	CMD	Команда
int32_t	Position	Позиция в основных шагах двигателя
int16_t	uPosition	Позиция в микрошагах (используется только с шаговыми двигателями). Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
int64_t	EncPosition	Позиция энкодера.
uint8_t	PosFlags	Флаги. Это битовая маска для побитовых операций.
	0x1 - SETPOS_IGNORE_POSITION	Если установлен, то позиция в шагах и микрошагах не обновляется.
	0x2 - SETPOS_IGNORE_ENCODER	Если установлен, то счётчик энкодера не обновляется.
uint8_t	Reserved [5]	Зарезервировано (5 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

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

6.2.6.109 Команда SSER

Код команды (CMD): «sser» или 0x72657373.

Запрос: (50 байт)

uint32_t	CMD	Команда
uint32_t	SN	Новый серийный номер платы.
uint8_t	Key	Ключ защиты для установки серийного номера (256 бит).
uint8_t	Major	Основной номер версии железа.
uint8_t	Minor	Второстепенный номер версии железа.
uint16_t	Release	Номер правок этой версии железа.
uint8_t	Reserved [4]	Зарезервировано (4 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Запись серийного номера и версии железа во flash память контроллера. Вместе с новым серийным номером и версией железа передаётся „Ключ“, только при совпадении которого происходит изменение и сохранение. Функция используется только производителем.

6.2.6.110 Команда SSTP

Код команды (CMD): «sstp» или 0x70747373.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Плавная остановка. Двигатель останавливается с ускорением замедления.

6.2.6.111 Команда STMS

Код команды (CMD): «stms» или 0x736D7473.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Начать измерения и буферизацию скорости, ошибки следования.

6.2.6.112 Команда STOP

Код команды (CMD): «stop» или 0x706F7473.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Немедленная остановка двигателя, переход в состояние STOP, ключи в режиме BREAK (обмотки накоротко замкнуты), режим „удержания“ дезактивируется для DC двигателей, удержание тока в обмотках для шаговых двигателей (с учётом Power management настроек). При вызове этой команды сбрасывается флаг ALARM.

6.2.6.113 Команда UPDF

Код команды (CMD): «updf» или 0x66647075.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Команда переводит контроллер в режим обновления прошивки. Только для производителя. Получив такую команду, прошивка платы устанавливает флаг (для загрузчика), отправляет эхо-ответ и перезагружает контроллер.

6.2.6.114 Команда WDAT

Код команды (CMD): «wdat» или 0x74616477.

Запрос: (142 байт)

uint32_t	CMD	Команда
uint8_t	Data	Закодированная прошивка.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (4 байт)

uint32_t	CMD	Команда

Описание: Записывает данные (прошивку) во Flash память контроллера. Не возвращает результат выполнения, хотя может завершаться ошибкой. Ошибочность заливки и тип ошибки можно узнать при завершении заливки.

6.2.6.115 Команда WKEY

Код команды (CMD): «wkey» или 0x79656B77.

Запрос: (46 байт)

uint32_t	CMD	Команда
uint8_t	Key	Ключ защиты для установки серийного номера (256 бит).
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Ответ: (15 байт)

uint32_t	CMD	Команда
uint8_t	sresult	Результат выполнения команды.
uint8_t	Reserved [8]	Зарезервировано (8 байт)
uint16_t	CRC	Контрольная сумма

Описание: Команда записи ключа для расшифровки прошивки. Result = RESULT_OK, если команда выполнена загрузчиком. Result = RESULT_HARD_ERROR, если во время выполнения команды произошла ошибка. Result не доступен через функцию библиотеки write_key, значение поля обрабатывается внутри функции. Функция используется только производителем.

6.2.6.116 Команда ZERO

Код команды (CMD): «zero» или 0x6F72657A.

Запрос: (4 байт)

uint32_t	CMD	Команда

Ответ: (4 байт)

uint32_t	CMD	Команда

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

Об этом документе

6.3 Совместимость с ПО для 8SMC1-USBhF

Важно: Протоколы связи для контроллеров 8SMC1 и 8SMC4/5 несовместимы. Протокол связи для 8SMC4/5 описан в *Описание протокола обмена*. Из-за различий в протоколах определенное количество

функций как 8SMC1, так и 8SMC4/5 не может быть сопоставлено друг с другом. Рекомендуемый способ работы с 8SMC4/5 - адаптировать ваше программное обеспечение для использования библиотеки libximc.

Программное обеспечение XiLab предназначено для контроллеров 8SMC4 и 8SMC5, **оно не будет работать с 8SMC1**. Для 8SMC1 есть два пакета программ: MicroSMC и SMCVieW.

- SMCView - это графический интерфейс на основе Labview для контроллеров 8SMC1. Для работы с 8SMC1 используется драйвер одного типа.
- MicroSMC - это программный драйвер для контроллеров 8SMC1. Он использует драйвер другого типа для работы с 8SMC1 (не тот, который использует SMCView).

Новые контроллеры можно использовать с программами, написанными для контроллеров 8SMC1-USBhF. Для этого есть две возможности:

1. Рекомендуемая. Использование версии ПО MicroSMC, которая поддерживает как 8SMC1-USBhF так и новые контроллеры. В этом случае ПО MicroSMC открывает все найденные новые контроллеры в режиме эксклюзивного доступа для обеспечения разделения доступа и взаимодействие с ними возможно только с помощью функций API контроллеров 8SMC1-USBhF через библиотеку USMCDLL, использующую MicroSMC. Скачать версию MicroSMC с поддержкой новых контроллеров можно на странице *Программное обеспечение*.
2. Управление только новыми контроллерами без ПО MicroSMC с помощью переходной библиотеки USMCDLL/libximc. Используйте эту возможность, если вам требуется одновременная работа ПО MicroSMC (для управления контроллерами 8SMC1-USBhF) и работа с контроллерами новыми с помощью библиотеки libximc. Скачать переходную библиотеку USMCDLL/libximc можно по этой ссылке.

При переходе с контроллера 8SMC1 на 8SMC4/5 мы советуем заменить функции вашей программы на используемые функции библиотеки libximc. Ниже в таблице описано соответствие функций API контроллеров 8SMC1-USBhF и новых контроллеров: в первом столбце указана функция библиотеки USMCDLL, во втором - соответствующий параграф в описании контроллера 8SMC1-USBhF, в третьем - особенности этой функции в переходной библиотеке при использовании с новыми контроллерами.

Функция	8SMC1-USBhF	Новый контроллер
USMC_Init	7.5.3	Использует функцию libximc *enumerate_devices*, которая опрашивает все COM-порты в системе, а также функции *get_enumerate_device_information*,* get_enumerate_device_serial, get_device_count*, *get_device_name*, *open_device*.
USMC_GetState	7.5.4	Использует функции libximc *get_status* и *get_engine_settings*. Флаги AReset, EMReset, RotTrErr возвращаемой структуры USMC_State всегда принимают значение false.
USMC_SaveParametersToFlash	7.5.5	Использует функцию libximc *command_save_settings*.

Continued on next page

Таблица 6.238 – continued from previous page

Функция	8SMC1-USBhF	Новый контроллер
USMC_GetMode	7.5.6	Использует функции libximc *get_edges_settings*, *get_power_settings*, *get_control_settings*, *get_ctp_settings*, *get_sync_out_settings*. В возвращаемой структуре USMC_Mode флаги EMReset, ResetRT, SyncOUTR, EncoderEn, EncoderInv, ResBEnc, ResEnc всегда принимают значение false, флаг SyncINOp всегда равен true.
USMC_SetMode	7.5.7	Использует те же функции, что и USMC_GetMode, а также их «set_» аналоги, игнорирует флаги RotTeEn, RotTrOp, ResetRT, SyncOUTR, SyncINOp, EncoderEn.
USMC_GetParameters	7.5.8	Использует функции 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*. Возвращает нулевые значения параметров BTimeoutR, BTimeoutD, MinP, MaxLoft, StartPos.
USMC_SetParameters	7.5.9	Использует те же функции, что и USMC_GetParameters, а также их «set_» аналоги, игнорирует параметры BTimeoutR, BTimeoutD, MinP, MaxLoft, StartPos.
USMC_GetStartParameters	7.5.10	Использует функции libximc *get_move_settings*, *get_engine_settings*. Флаги WSyncIN, SyncOUTR, ForceLoft возвращаемой структуры USMC_StartParameters всегда принимают значение false.
USMC_Start	7.5.11	Использует функции libximc *get_move_settings*, *get_engine_settings*, а также их «set_» аналоги и функцию *command_move*.
USMC_Stop	7.5.12	Использует функцию libximc *command_stop*.
USMC_SetCurrentPosition	7.5.13	Использует функцию libximc *set_position*.
USMC_GetEncoderState	7.5.14	Использует функцию libximc *get_status*.
USMC_GetLastErr	7.5.15	Не модифицирует передаваемый ей параметр. Ошибку выполнения функций USMC_ можно отследить по ненулевому коду возврата.
USMC_Close	7.5.16	Использует функцию libximc *close_device*.

Переходная библиотека USMCDLL не требует для своей работы запущенного в фоне приложения MicroSMC.exe и использует библиотеку libximc.dll для взаимодействия с новыми контроллерами.

Все функции USMC_, включающие в себя несколько вызовов функций libximc, прерывают своё выполнение, если одна из составляющих их функций libximc возвращает ошибку. В этом случае возможна неполная запись настроек в контроллер. Ненулевой код возврата функции USMC_ равен коду возврата выполненной с ошибкой функции libximc.

Тестовая программа: usmcdll_libximc_test.zip

Чтобы контроллер 8SMC1-USBhF работал под управлением Windows 10, необходимо обновить микропрограмму контроллера до последней версии (25.05). Все соответствующие программы можно найти на этой странице. Полный пакет программного обеспечения и поддержки для Windows 8. Последняя прошивка v.2505 - обновлена для windows 8 и 10.

Важно: Для обновления прошивки вам потребуется или Windows 7 или Windows 8.1. Скорее всего, вы не сможете обновить прошивку под управлением Windows 10

Если вы хотите отремонтировать контроллеры 8SMC1-USBhF, обратитесь к менеджеру, который продал оборудование. Если контакты менеджера были утеряны, вы можете написать на sales@standa.lt. **Важно отметить**, что стоимость ремонта может быть сопоставима со стоимостью нового контроллера.

6.4 Таймауты libximc

При работе с программой *XiLab* или написании собственных приложений с использованием *libximc* действуют таймауты для детектирования ошибок или более стабильной работы контроллера. Ниже приведён список таймаутов, их длительность и условия применения. Таймауты оптимизированы для работы через соединение USB на современном компьютере. При создании собственной цепи передачи управляющего сигнала необходимо учитывать задержки линии связи, чтобы таймауты не срабатывали.

Когда происходит	Название	Время в миллисекундах
Таймаут при перечислении устройств. Если не удаётся определить тип устройства.	ENUMERATE_TIMEOUT_TIME	100
Попытка открыть порт.	DEFAULT_TIMEOUT_TIME	5000
Ожидание данных от устройства.	DEFAULT_TIMEOUT_TIME	5000
От открытия устройства до начала работы с ним.	RESET_TIME/2	50
Ожидание появления устройства при запуске процедуры перезаливки и его перезагрузке.	RESET_TIME * 1.2 + DEFAULT_TIMEOUT_TIME	5120
Ожидание после записи сектора флэш памяти при перезаливке.	FLASH_SECTIONWRITE_TIME	100
Таймаут попыток установить связь с контроллером после его перезагрузки для перезаливки.	XISM_PORT_DETECT_TIME	60000

6.5 Скрипты XILab

- *Краткое описание языка*
  - *Типы данных*
  - *Инструкции*
  - *Объявление переменных*
  - *Ключевые и зарезервированные слова*
  - *Функции*
- *Подсветка синтаксиса*
- *Дополнительные функции, предоставляемые XILab*

  - *Запись в лог XILab*
  - *Задержка выполнения скрипта*
  - *Создание объекта типа «ось»*
  - *Создание объекта типа «файл»*
  - *Создание структуры калибровки*
  - *Получение следующего серийного номера*
  - *Ожидание остановки движения*
  - *функции библиотеки libximc*
- *Примеры*
  - *Скрипт-пример работы с битовыми масками*
  - *Скрипт сканирования и записи в файл*
  - *Многоосный скрипт циклического движения*
  - *Одноосный скрипт циклического движения*
  - *Скрипт проверки калибровки домашней позиции*
  - *Скрипт для поиска серийных номеров контроллеров*
  - *Скрипт перемещения и ожидания*
  - *Скрипт случайного сдвига*
  - *Скрипт установки нулевой позиции*
  - *Скрипт для автотестирования*
  - *Тест на пересечение границ*
  - *Тест настройки с замкнутым контуром*
  - *Скрипт дискретного движения*
  - *Экспоненциальное изменение позиции использующие user units*
  - *Шаговый скрипт использующий user units*
  - *Шаговый скрипт*
  - *Тест калибровки домашней позиции сигналу со входа EXTIO*
  - *Скрипт движения по sin*
  - *Скрипт перемещения по сигналу со входа EXTIO. Движение осуществляется в user units*
  - *Вероятные тесты*
  - *Скрипт выполняющий ряд смещений с калибровкой*
  - *Тест на пропуск шагов*
  - *Скрипт тестирования синхронизации*
  - *Скрипт тестирования ошибок синхронизации*

Скриптовый язык Xilab реализован с помощью QtScript, он в свою очередь основан на ECMAScript. ECMAScript — это встраиваемый расширяемый не имеющий средств ввода/вывода язык программиро-

вания, используемый в качестве основы для построения других скриптовых языков. Стандартизирован международной организацией ECMA в спецификации ECMA-262.

QtScript (и, соответственно, XILab) использует третью редакцию стандарта ECMAScript.

6.5.1 Краткое описание языка

6.5.1.1 Типы данных

В ECMAScript поддерживаются девять типов данных. Три из них (Reference, List, и Completion) используются только как промежуточные результаты расчета значений выражений. Оставшиеся шесть типов это:

- Неопределённый,
- Нулевой,
- Логический,
- Строковый,
- Числовой,
- Объектный.

6.5.1.2 Инструкции

Наиболее распространенные конструкции языка ECMAScript представлены в таблице ниже:

Название	Применение	Краткие сведения
Блок	{[<список инструкций>]}	Несколько инструкций можно объединить в один блок фигурными скобками.
Объявление переменной	var <список объявления переменных>	Переменная объявляется с помощью ключевого слова «var».
Пустая инструкция	;	Точка с запятой является пустой инструкцией. Заканчивать строки точкой с запятой не обязательно.
Условие	if (<условие>) <инструкция> [else <инструкция>]	Условное выполнение можно производить с помощью ключевых слов «if ... else». Если условие верно, то выполняется инструкция блока if, в противном случае выполняется инструкция блока else, если он присутствует.
Цикл	do <тело цикла> while (<условие>) while (<условие>) <тело цикла> for ([<выражение 1>]; [<условие>]; [<выражение 2>]) <тело цикла>	Цикл можно реализовать несколькими различными способами. Форма «do ... while ...» выполняет тело цикла как минимум один раз и пока условие верно. Форма «while ... do ...» выполняет тело цикла пока условие верно. Форма «for ...» выполняет выражение до начала (выражение 1), а затем выполняет тело цикла каждый раз после выполнения итеративного выражения (выражение 2) и проверки условия на истинность.

Continued on next page

Таблица 6.240 – continued from previous page

Название	Применение	Краткие сведения
Возврат	return [<выражение>]	Прекращает выполнение функции и возвращает выражение как результат.
Генерация исключения	throw <выражение>	Генерирует исключение, которое может быть обработано конструкцией try (см.ниже).
Блок try	try <блок> catch (<идентификатор>) <блок> try <блок> finally <блок> try <блок> catch (<идентификатор>) <блок> finally <блок>	Используется совместно с исключениями. Конструкция «try ... catch ... finally» пытается выполнить блок try. Если в этом блоке происходит исключение идентификатор, то выполняется содержимое блока catch. После всего безусловно выполняется блок finally. Один из блоков «catch» и «finally» может отсутствовать.

6.5.1.3 Объявление переменных

Переменные определяются с помощью ключевого слова var. При объявлении переменная помещается в область видимости, соответствующую функции, в которой она объявляется. Если переменная объявляется вне функций, она помещается в глобальную область видимости. Создание переменной происходит при получении управления функцией с её объявлением. Или программой, если переменная глобальна. При создании переменной в ECMAScript она приобретает значение undefined. Если переменная объявлена с инициализацией, инициализация происходит не в момент создания переменной, а при выполнении строки с инструкцией var.

6.5.1.4 Ключевые и зарезервированные слова

Следующие слова являются ключевыми в языке и не могут быть использованы как идентификаторы:

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

Следующие слова используются как ключевые в предлагаемых расширениях и зарезервированы:

```
abstract  enum        int         short
boolean   export      interface   static
byte      extends     long        super
char      final       native      synchronized
class     float       package     throws
const     goto        private     transient
debugger  implements  protected   volatile
double    import      public
```

6.5.1.5 Функции

Функции в ECMAScript являются объектами. Функции, как и любые другие объекты, могут храниться в переменных, объектах и массивах, могут передаваться как аргументы в другие функции и могут возвращаться функциями. Функции, как и любые другие объекты, могут иметь свойства. Существенной специфической чертой функций является то, что они могут быть вызваны.

В тексте программы именованную функцию в ECMAScript обычно определяют следующим способом:

```
function sum(arg1, arg2) { // a function which takes two parameters
    return arg1 + arg2;    // and returns their sum
}
```

6.5.2 Подсветка синтаксиса

Шрифт текста в окне скрипта имеет подсветку синтаксиса. Цвета:

Тип выражения	*цвет*	*пример отображения*
Произвольные функции	фиолетовый	my_function();
Функции XILab	синий	get_status();
Положительные числа	зеленый	a = 100;
Отрицательные числа	красный	b = -200;
Комментарии	серый	// a comment
Все остальное	черный	var s = "a string";

Во время выполнения скрипта фон строки с последней выполненной командой меняется на тёмно-серый с частотой обновления 1 раз в 20 мс.

6.5.3 Дополнительные функции, предоставляемые XILab

На данной картинке изображены функции, которые XILab предоставляет для использования в скриптах в дополнение к стандартным функциям языка.

```
var s = "a string";
```

- log(string text [, int loglevel]) - запись в лог XILab
- msleep(int ms) - задержка выполнения скрипта
- new_axis(int serial_number) - создание объекта типа «ось»
- new_file(string filename) - создание объекта типа «файл»
- new_calibration(int A, int Microstep) - создание структуры калибровки для передачи калиброванным функциям
- get_next_serial(int serial) - получение следующего серийного номера
- command_wait_for_stop(int refresh_period) - ожидание остановки движения
- а также все функции библиотеки libximc (см. *Руководство по программированию*)

Кроме этого, в скриптах определены и доступны для использования все константы протокола обмена. *Пример использования*.

6.5.3.1 Запись в лог XILab

Производится вызовом функции log(string text [, int loglevel]). Дописывает в лог XiLab строку *text*. Если передаётся второй параметр *loglevel*, то сообщение получает соответствующий уровень логгирования и отображается соответствующим цветом.

Loglevel	Тип
1	Error
2	Warning
3	Info

Пример:

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

Пример использования.

Замечание: не рекомендуется вызывать функции интерфейса Xilab (запись в лог) чаще одного раза в 20 мс.

6.5.3.2 Задержка выполнения скрипта

Производится вызовом функции msleep(int ms). Скрипт делает паузу в данном месте выполнения длиной ms миллисекунд.

Пример:

```
msleep(200);
```

Пример использования.

6.5.3.3 Создание объекта типа «ось»

Многоосевой интерфейс XILab также предоставляет возможность управлять контроллерами посредством скриптов. Отличия состоят в том, что необходимо явно указывать, какому контроллеру посылается команда. Для этого вводится новый тип объекта «ось», который имеет методы, совпадающие по именам с функциями *библиотеки* . Идентификация происходит по серийному номеру. контроллера.

Пример:

```
var x = new_axis(123);
x.command_move(50);
```

В этом примере в первой строке скрипта происходит создание оси с именем переменной x, которая соответствует контроллеру с серийным номером «123». Если такой контроллер не подключен к компьютеру, то скрипт выдаст ошибку выполнения и завершится. Во второй строке оси x подается команда переместиться в координату 50 [шагов].

Пример использования.

6.5.3.4 Создание объекта типа «файл»

Скрипты Xilab имеют возможность чтения из файла и записи в файл. Для этого необходимо создать объект типа файл и работать с ним. Имя файла указывается при создании в конструкторе. Объект имеет следующие функции:

Тип_возврата Имя_функции	Краткие сведения
bool open()	Открывает файл. Файл открывается на чтение-запись, если это возможно; если нет, то только на чтение.
void close()	Закрывает файл.
Number size()	Возвращает размер файла в байтах.
bool seek(Number pos)	Устанавливает текущую позицию в файле в *pos* байт[1].
bool resize(Number size)	Изменяет размер файла до *size* байт. Если *size* меньше текущего размера, то файл обрезается, если больше, то дополняется нулями.
bool remove()	Удаляет файл.
String read(Number maxsize)	Читает строку из файла, но не более *maxsize* байт. Данные читаются в кодировке utf-8.
Number write(String s, Number maxsize)	Записывает строку в файл, но не более *maxsize* байт. Данные записываются в кодировке utf-8, символ конца строки пользователь должен записать самостоятельно. Возвращает количество записанных байт, либо -1, если произошла ошибка.

Все функции работы с файлами, возвращающие значение типа *bool*, возвращают «true» в случае успеха, в противном случае «false».

Используйте символ «/» как разделитель путей для работы скриптов на всех платформах (Windows/Linux/Mac).

Пример:

```
var winf = new_file("C:/file.txt"); // An example of file name and path on Windows
var linf = new_file("/home/user/Desktop/file.txt"); // An example of file name and path on Linux
var macf = new_file("/Users/macuser/file.txt"); // An example of file name and path on Mac

var f = winf; // Pick a file name
if (f.open()) { // Try to open the file
  f.write( "some text" ); // If successful, then write desired data to the file
  f.close(); // Close the file
} else { // If file open failed for some reason
  log( "Failed opening file" ); // Log an error
}
```

Пример использования.

6.5.3.5 Создание структуры калибровки

функция new_calibration(double A, int Microstep) принимает в качестве параметров коэффициент A пересчета из шагов в пользовательские единицы и деление микрошага Microstep (либо полученное ранее вызовом функции get_engine_settings в поле *MicrostepMode*, либо задаваемое одной из констант MICROSTEP_MODE_) и возвращает структуру типа calibration_t, которую необходимо передать в калиброванные get_/set_* функции для получения/задания величин в пользовательских единицах. Следующие две записи функционально эквивалентны:

[1] Выйти из файла: если позиция находится за пределами файла, то seek () не должен немедленно расширять файл. Если запись выполняется в этой позиции, файл должен быть расширен. Содержимое файла между предыдущим концом файла и новыми записанными данными НЕ УКАЗАНО и варьируется между платформами и файловыми системами.

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

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

Пример использования.

6.5.3.6 Получение следующего серийного номера

функция get_next_serial(int serial) принимает в качестве параметра число и возвращает наименьший серийный номер из списка серийных номеров открытых устройств, который больше переданного ей параметра. Если такого серийного номера нет, то возвращается 0. Эта функция удобна для автоматического создания объекта типа «ось» без задания фиксированного серийного номера.

Пример:

```
var first_serial = get_next_serial(0);
var x = new_axis(first_serial);
var y = new_axis(get_next_serial(first_serial));
```

В этом примере в первой строке происходит получение первого серийного номера, во второй - создание объекта оси по этому серийному номеру, в третьей - получение следующего серийного номера и создание оси для него.

Пример использования.

6.5.3.7 Ожидание остановки движения

Функция command_wait_for_stop(int refresh_period) останавливает выполнение скрипта до тех пор, пока контроллер не прекратит движение, то есть, пока флаг MVCMD_RUNNING в члене MvCmdSts структуры, возвращаемой функцией get_status(), не будет снят. Функция скриптов command_wait_for_stop напрямую использует функцию command_wait_for_stop библиотеки libximc и принимает в качестве параметра число, означающее периодичность (в миллисекундах) считывания состояния контроллера.

Эта функция также присутствует как метод объекта типа «ось».

Пример использования.

6.5.3.8 функции библиотеки libximc

Функции библиотеки libximc начинающиеся на «get_» считывают из контроллера настройки и возвращают соотвествующую команде структуру данных. Функции библиотеки libximc начинающиеся на «set_» принимают как параметр структуру данных и записывают эти настройки в контроллер. Заполнить структуру данных для set-функций можно двумя способами:

- вызвать соответствующую get-функцию и модифицировать необходимые поля

```
// set settings: type 1
var m = get_move_settings();
m.Speed = 100;
set_move_settings(m);
```

- создать объект (*Object*) и заполнить все его свойства (*properties*) с именами членов структуры с учетом регистра.

```
// set settings: type 2
var m = new Object;
m.Speed = 100;
m.uSpeed = 0;
m.Accel = 300;
m.Decel = 500;
m.AntiplaySpeed = 10;
m.uAntiplaySpeed = 0;
set_move_settings(m);
```

Необходимо помнить, что при использовании первого способа в контроллер посылается дополнительная команда (вызывается не только set-функция, но и get- перед этим), а при использовании второго способа необходимо инициализировать все свойства объекта, соответствующие именам членов структуры. Если в объекте какое-то свойство будет пропущено, то оно будет считаться равным нулю. Если в объекте будет объявлено какое-либо дополнительное свойство, не входящее в структуру данных, то оно будет проигнорировано. Если тип данных какого-либо поля не будет соответствовать типу данных структуры, то будет произведено приведение типов по правилам EcmaScript. Структуры данных для всех команд описаны в главе *Описание протокола*.

Пример использования.

6.5.4 Примеры

В этом разделе приведены примеры типичных действий, которые можно выполнять с помощью скриптов XILab.

6.5.4.1 Скрипт-пример работы с битовыми масками

```
/*
* Bit mask example script
*
* Description of the script:
*  This script clearly shows how to work with bit masks.
*  This script or part of it may be needed when working with any of our other commands that use␣
↪bit masks. For example, the  «set_home_settings» command
*
* To run the script, upload it to the XILab software
*/

var a = new_axis(get_next_serial(0)); // take first found axis
var gets = a.get_status(); // read status once and reuse it

var gpio = gets.GPIOFlags;
var left = STATE_LEFT_EDGE;
var right = STATE_RIGHT_EDGE;
var mask = left | right;
var result = gpio & mask;
log( to_binary(left) + " = left limit switch flag" );
log( to_binary(right) + " = right limit switch flag" );
log( to_binary(mask) + " = OR operation on flags gives the mask" );
log( to_binary(gpio) + " = gpio state" );
log( to_binary(result) + " = AND operation on state and mask gives result" );
if ( result ) {
  log("At least one limit switch is on");
} else {
  log("Both limit switches are off");
}
```

```
// Binary representation function
function to_binary(i)
{
  bits = 32;
  x = i >>> 0; // coerce to unsigned in case we need to print negative ints
  str = x.toString(2); // the binary representation string
  return (repeat("0", bits) + str).slice (-bits); // pad with zeroes and return
}

// String repeat function
function repeat(str, times)
{
  var result="";
  var pattern=str;
  while (times > 0) {
    if (times&1) {
      result+=pattern;
    }
    times>>=1;
    pattern+=pattern;
  }
  return result;
}
```

6.5.4.2 Скрипт сканирования и записи в файл

```
/*
* A script which scans and writes data to the file
*
* Description of the script:
*  This script scans and writes the data to a .csv file.
*  The script can be useful if you are using the system to scan an area and/or capture frames
*
* To run the script, upload it to the XILab software
*/

var start = 0; // Starting coordinate in steps
var step = 10; // Shift amount in steps
var end = 100; // Ending coordinate in steps

var speed = 300; // maximum movement speed in steps / second
var accel = 100; // acceleration value in steps / second^2
var decel = 100; // deceleration value in steps / second^2
var delay = 100;

var m = get_move_settings(); // read movement settings from the controller
m.Speed = speed; // set movement speed
m.Accel = accel; // set acceleration
m.Decel = decel; // set deceleration
set_move_settings(m); // write movement settings into the controller

var f = new_file("C:/a.csv"); // Choose a file name and path
f.open(); // Open a file
f.seek( 0 ); // Seek to the beginning of the file

command_move(start); // Move to the starting position
```

```
command_wait_for_stop(delay); // Wait until controller stops moving

while (get_status().CurPosition < end) {
  f.write( get_status().CurPosition + "," + get_chart_data().Pot + "," + Date.now() + "\n" ); //␣
↪Get current position, potentiometer value and date and write them to file
  command_movr(step); // Move to the next position
  command_wait_for_stop(delay); // Wait until controller stops moving
}
f.close(); // Close the file
```

move_and_sleep.csv

- пример файла для использования с приведенным выше скриптом

6.5.4.3 Многоосный скрипт циклического движения

```
/*
* Multi axis cyclic movement script
*
* Description of the script:
*  Does cyclic movement between two border points with set values of acceleration,
*  deceleration and top speed, for all axes found. The script is similar to the "Cyclic"
*  button in XILab
*
* To run the script, upload it to the XILab software
*/

var axes = [];
var number_of_axes = 0;
var last_serial = 0;

while (serial = get_next_serial(last_serial))   // Get next serial number and repeat for each axes
{
  axes[number_of_axes] = new_axis(serial);
  log("Found axis " + number_of_axes + " with serial number " + serial);
  number_of_axes++;
  last_serial = serial;
}

for (var i = 0; i < number_of_axes; i++)
{
  axis_configure(axes[i]);
}

while (1)
{
  for (var i = 0; i < number_of_axes; i++)
  {
    go_first_border(axes[i]);
    go_second_border(axes[i]);
  }

  msleep(100);
}

function axis_configure(axis)
{
  var speed = 1000;   // Maximum movement speed in steps / second
```

```
  var accel = 2000;   // Acceleration value in steps / second^2
  var decel = 5000;   // Deceleration value in steps / second^2

  axis.command_stop(); // send STOP command (does immediate stop)
  axis.command_zero(); // send ZERO command (sets current position and encoder value to zero)
  var m = axis.get_move_settings(); // read movement settings from the controller
  m.Speed = speed; // set movement speed
  m.Accel = accel; // set acceleration
  m.Decel = decel; // set deceleration
  axis.set_move_settings(m); // write movement settings into the controller
}

function go_first_border(axis)
{
  var first_border = 0; // first border coordinate in steps
  var GETS = axis.get_status();

  if (!(GETS.MvCmdSts & MVCMD_RUNNING) && (GETS.CurPosition != first_border))
  {
    axis.command_move(first_border);  // move towards one border
  }
}

function go_second_border(axis)
{
  var second_border = 25000; // second border coordinate in steps
  var GETS = axis.get_status();

  if (!(GETS.MvCmdSts & MVCMD_RUNNING) && (GETS.CurPosition != second_border))
  {
    axis.command_move(second_border);   // move towards another border
  }
}
```

6.5.4.4 Одноосный скрипт циклического движения

```
/*
* Single axis cyclic movement script
*
* Description of the script:
*  Does cyclic movement between two border points with set values of acceleration,
*  deceleration and top speed. The script is similar to the "Cyclic" button in XILab
*
* To run the script, upload it to the XILab software
*/

var first_border = -10; // first border coordinate in mm
var second_border = 10; // second border coordinate in mm
var mm_per_step = 0.005; // steps to distance translation coefficient
var delay = 100; // delay in milliseconds
var calb = new_calibration(mm_per_step, get_engine_settings().MicrostepMode); // create␣
↪calibration structure
command_stop(); // send STOP command (does immediate stop)
command_zero(); // send ZERO command (sets current position and encoder value to zero)
while (1) { // infinite loop
  command_move_calb(first_border, calb); // move towards one border
  command_wait_for_stop(delay); // wait until controller stops moving
```

```
  command_move_calb(second_border, calb); // move towards another border
  command_wait_for_stop(delay); // wait until controller stops moving
}
```

6.5.4.5 Скрипт проверки калибровки домашней позиции

```
/*
* Homing test script
*
* Description of the script:
*  This script tests homing function by repeatedly moving to a random position,
*  doing quick stop and then homing, for all axes found. The script is similar to the GO
*  Home button in XILab
*
* To run the script, upload it to the XILab software
*/

var axes = [];
var number_of_axes = 0;
var last_serial = 0;
while (serial = get_next_serial(last_serial)) // get next serial number and repeat for each axes.
{
    axes[number_of_axes] = new_axis(serial);
   log("Found axis " + number_of_axes + " with serial number " + serial);
   number_of_axes++;
   last_serial = serial;
}

while (1) { // infinite loop
    for (var i = 0; i < number_of_axes; i++)
   {
        homing_test(axes[i]);
    }
}

function homing_test(axis)
{
    var shift_low = 0; // minimum shift distance in steps
   var shift_high = 10000; // maximum shift distance in steps
   var speed_low = 100; // minimum movement speed in steps / second
   var speed_high = 5000; // maximum movement speed in steps / second
   var time_low = 1; // minimum wait time in seconds
   var time_high = 10; // maximum wait time in seconds

    axis.command_home(); // send HOME command (find home position)
    axis.command_wait_for_stop(100); // wait until controller stops moving
    var m = axis.get_move_settings(); // read movement settings from the controller
    m.Speed = rnd(speed_low, speed_high); // set random speed from a range of speeds between
↪"speed_low" and "speed_high"
    axis.set_move_settings(m); // write movement settings into the controller
    var shift = rnd(shift_low, shift_high); // pick random shift value from a range of distances␣
↪between "shift_low" and "shift_high"
    if (Math.random() < 0.5) { // pick random direction
       shift = -shift;
    }
    axis.command_movr(shift); // send MOVR command (does a relative shift)
    msleep( rnd(time_low*1000, time_high*1000) ); // pause for a random time from a range between
↪"time_low" and "time_high"
```

```
    axis.command_stop(); // send STOP command (does immediate stop)
}

function rnd(min,max) { // "rnd" is a helper function which uses Math.random() and returns a␣
↪uniformly distributed integer random value between "min" and "max"
 var r = Math.random()*(max-min)+min;
 return Math.round(r);
}
```

6.5.4.6 Скрипт для поиска серийных номеров контроллеров

```
/*
* List axis serials script
*
* Description of the script:
*  An example of a script that searches for all the serial numbers of controllers
*  and outputs them to the log.
*
* To run the script, upload it to the XILab software
*/

var i = 0; // Declare loop iteration variable
var serial = 0; // Declare serial number variable
var axes = Array(); // Declare axes array
while (true) { // The loop
 serial = get_next_serial(serial); // Get next serial
 if (serial == 0) // If there are no more controllers then...
     break; // ...break out of the loop
 var a = new Object(); // Create an object
 a.serial = serial; // Assign serial number to its "serial" property
 a.handle = new_axis(serial); // Assign new axis object to its "handle" property
 axes[i] = a; // Add it to the array
 i++; // Increment counter
}
for (var k=0; k < axes.length; k++) { // Iterate through array elements
  log ( "Axis with S/N " + axes[k].serial + " is in position " + axes[k].handle.get_status().
↪CurPosition ); // For each element print saved axis serial and call a get_status() function
}
```

6.5.4.7 Скрипт перемещения и ожидания

```
/*
* Move and wait script
*
* Description of the script:
*  The script reads the next coordinate and the delay time from the csv file,
*  after which it moves to the specified coordinate with a subsequent delay,
*  and so on until the end of reading the entire file.
*     The script can be useful if you are using the system to scan an area and/or capture frames
* To run the script, upload it to the XILab software
*/

var axis = new_axis(get_next_serial(0)); // Use first available controller
var x; // A helper variable, represents coordinate
var ms; // A helper variable, represents wait time in milliseconds
var f = new_file("./move_and_sleep.csv"); // Choose a file name and path; this script uses a file␣
↪from examples in the installation directory
```

```
f.open(); // Open a file
while ( str = f.read(4096) ) { // Read file contents string by string, assuming each string is␣
↪less than 4 KiB long
  var ar = str.split(","); // Split the string into substrings with comma as a separator; the␣
↪result is an array of strings
  x = ar[0]; // Variable assignment
  ms = ar[1]; // Variable assignment
  log( "Moving to coordinate " + x ); // Log the event
  axis.command_move(x); // Move to the position
  axis.command_wait_for_stop(100); // Wait until the movement is complete
  log( "Waiting for " + ms + " ms" ); // Log the event
  msleep(ms); // Wait for the specified amount of time
}
log ( "The end." );
f.close(); // Close the file
```

6.5.4.8 Скрипт случайного сдвига

```
/*
* Random shift script
*
* Description of the script:
*  This script does shifts on random offset from a specified range of distances
*  with a random speed from a chosen range of speeds.

* To run the script, upload it to the XILab software
*/

var axes = [];
var number_of_axes = 0;
var last_serial = 0;
while (serial = get_next_serial(last_serial)) // get next serial number and repeat for each axes.
{
   axes[number_of_axes] = new_axis(serial);
   log("Found axis " + number_of_axes + " with serial number " + serial);
   number_of_axes++;
    last_serial = serial;
}

while (1) { // infinite loop
   for (var i = 0; i < number_of_axes; i++)
   {
       go_to_random_shift(axes[i]);
    }
}

function go_to_random_shift(axis)
{
   var shift_low = 0; // minimum shift distance in steps
   var shift_high = 10000; // maximum shift distance in steps
   var speed_low = 100; // minimum movement speed in steps / second
   var speed_high = 5000; // maximum movement speed in steps / second

   var m = axis.get_move_settings(); // read movement settings from the controller
   m.Speed = rnd(speed_low, speed_high); // set random speed from a range of speeds between "speed_
↪low" and "speed_high"
   axis.set_move_settings(m); // write movement settings into the controller
```

```
   var shift = rnd(shift_low, shift_high); // pick random shift value from a range of distances␣
↪between "shift_low" and "shift_high"
   if (Math.random() < 0.5) { // pick random direction
       shift = -shift;
   }
   axis.command_movr(shift); // send MOVR command (does a relative shift)
    axis.command_wait_for_stop(100); // wait until controller stops moving
}

function rnd(min,max) { // "rnd" is a helper function which uses Math.random() and returns a␣
↪uniformly distributed integer random value between "min" and "max"
  var r = Math.random()*(max-min)+min;
  return Math.round(r);
}
```

6.5.4.9 Скрипт установки нулевой позиции

```
/*
* Set zero script
*
* Description of the script:
*  This script changes "standoff" setting (found on "Home position" page in the XILab "Settings")␣
↪so that the current position becomes the home position.
*  The script is very convenient for calibrating and configuring new stages, as well as for␣
↪changing the zero position
*
* How to use:
*  - manually move your positioner to a desired position
*  - launch this script and wait for completion
*  As a result your positioner will return to the starting position and all subsequent calls to
↪"homing" function will bring it there.
*
* Note: homing settings are saved into RAM and will be lost when controller is powered down. If␣
↪you wish to save these settings to non-volatile memory you should either pick "Save settings to␣
↪flash" on the XILab main Settings page or call "command_save_settings()" at the end of the␣
↪script.
*
* To run the script, upload it to the XILab software
*/

var axes = [];
var number_of_axes = 0;
var last_serial = 0;
while (serial = get_next_serial(last_serial)) // get next serial number and repeat for each axes.
{
    axes[number_of_axes] = new_axis(serial);
   log("Found axis " + number_of_axes + " with serial number " + serial);
   number_of_axes++;
   last_serial = serial;
}

for (var i = 0; i < number_of_axes; i++)
{
   set_zero(axes[i]);
}

function set_zero(axis)
```

```
{
    axis.command_stop(); // send STOP command (does immediate stop)

   var h = axis.get_home_settings(); // read homing settings from the controller
   h.HomeDelta = 0; // set "HomeDelta" parameter in "home_position" structure to 0
   h.uHomeDelta = 0; // set "uHomeDelta" parameter in "home_position" structure to 0
   var saved_fast = h.FastHome; // save "FastHome" parameter from "home_position" structure to a␣
↪variable
   var saved_ufast = h.uFastHome; // save "uFastHome" parameter from "home_position" structure to␣
↪a variable
   if (h.HomeFlags & HOME_MV_SEC_EN != 0) // if homing settings have two homing phases turned on
   {
        h.FastHome = 100; // set "FastHome" parameter in "home_position" structure (first movement␣
↪speed) to 100 steps/s: this is required to avoid slip at the end of the first phase
       h.uFastHome = 0; // set "uFastHome" parameter in "home_position" structure to 0
    }
    axis.set_home_settings(h); // write homing settings into the controller

    var old_pos = axis.get_status().CurPosition; // save whole step part of the initial position␣
↪into a variable
    var old_upos = axis.get_status().uCurPosition; // save microstep part of the initial position␣
↪into a variable
    axis.command_home(); // send HOME command (find home position)
    do { msleep(100); } while (axis.get_status().MvCmdSts == (MVCMD_HOME | MVCMD_RUNNING)); //␣
↪query controller state every 100 ms while movement state is "homing command is being executed"
    if (axis.get_status().MvCmdSts != MVCMD_HOME)  // if current state is not "homing completed␣
↪successfully" (MVCMD_RUNNING unset, MVCMD_ERROR unset, last command MVCMD_HOME) then homing was␣
↪interrupted
    {
        h.FastHome = saved_fast; // set "FastHome" parameter in "home_position" structure to a␣
↪saved value
       h.uFastHome = saved_ufast; // set "uFastHome" parameter in "home_position" structure to a␣
↪saved value
       axis.set_home_settings(h); // write movement settings into the controller (this restores␣
↪the initial settings)
       throw "Script aborted: homing failed."; // throw an exception and terminate
    }

    var new_pos = axis.get_status().CurPosition; // read whole part of new position into "new_pos"␣
↪variable
    var new_upos = axis.get_status().uCurPosition; // read microstep part of new position into
↪"new_upos" variable
    h.HomeDelta = old_pos-new_pos; // set "HomeDelta" parameter in "home_position" structure to␣
↪this value
    h.uHomeDelta = old_upos-new_upos; // set "uHomeDelta" parameter in "home_position" structure␣
↪to this value
    h.FastHome = saved_fast; // set "FastHome" parameter in "home_position" structure to a saved␣
↪value
    h.uFastHome = saved_ufast; // set "uFastHome" parameter in "home_position" structure to a␣
↪saved value
    axis.set_home_settings(h); // write movement settings into the controller
    axis.command_move(old_pos, old_upos); // move to initial position
    do { msleep(100); } while (axis.get_status().MvCmdSts == (MVCMD_MOVE | MVCMD_RUNNING)); //␣
↪query controller state every 100 ms while movement state is "movement command is being executed"
    if (axis.get_status().MvCmdSts != MVCMD_MOVE)  // if current state is not "movements completed␣
↪successfully" (MVCMD_RUNNING unset, MVCMD_ERROR unset, last command MVCMD_MOVE) then movement␣
↪was interrupted
    {
```

```
        throw "Script aborted: return to position failed."; // throw an exception and terminate
    }
    axis.command_zero(); // send ZERO command (sets current position and encoder value to zero)
}

log("Done."); // log success
```

6.5.4.10 Скрипт для автотестирования

```
/*
* Autotester script
*
* Description of the script:
*  The script tests the controller with a stage using a set of tests.
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/
```

Посмотреть полный код

6.5.4.11 Тест на пересечение границ

```
/*
* Border crossing test
*
* Description of the script:
*  The script checks the correct operation of the connected external limit switch
*
* How to connect wires?
*  You must connect the limit switch to the DSub-15 connector (pins 8 and 9 on the controller␣
↪connector).
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

const MVCMD_ERROR = 0x40;
const MVCMD_RUNNING = 0x80;

var axis = new_axis(get_next_serial(0));
var m = axis.get_extio_settings();
var s = axis.get_status();

var count_error = 0;
var count_good = 0;

function BorderOff()
{
  m.EXTIOSetupFlags = 0x01;
  m.EXTIOModeFlags = 0x10;

  axis.set_extio_settings(m);
```

```
}

function BorderOn()
{
  m.EXTIOSetupFlags = 0x01;
  m.EXTIOModeFlags = 0x00;

  axis.set_extio_settings(m);
}

function BorderCycle()
{
  BorderOn();
  msleep(50);

  BorderOff();
  msleep(100);
}

log("You have to connect the common",2);
log("input/output pin on the backplane connector to",2);
log("the 2nd limit switch pin on the stage connector", 2);
log("Also its recommended to load the profile for your stage", 2);

log(">>> Start testing", 3);

while (1)
{
  axis.command_left();
  msleep(200);
  BorderOn();
  msleep(200);
  s = axis.get_status();

  if (s.MvCmdSts & MVCMD_RUNNING)
  {
    count_error++;

    log(">>> ALARM ! Crossing through the limit switch !" ,1);
  }
  else
  {
    count_good++;

    if (!(count_good % 50))
      log(">>> " + count_good + " cycles were done correct, " + count_error + " cycles were done␣
↪incorrect", 3);
  }

  BorderOff();
}
```

6.5.4.12 Тест настройки с замкнутым контуром

```
/*
* Closed loop tuning test
*
```

```
* Description of the script:
*  The script checks the parameters of a closed loop
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

var global_axis;

global_axis = new_axis(get_next_serial(0));
global_axis.command_stop();

const DEBUG = 1;
const MICROSTEPS = 256;
const SKIP_ENCODER_COUNT = 7;
const STORE_COUNT_MICROSTEPS = 19;
const TRUST_INDEX = 15;     // Always TRUST_INDEX < STORE_COUNT_MICROSTEPS

/*
* Save and overwrite settings
*/
var SFBS = global_axis.get_feedback_settings(); // Save information about encoder (IPS)
SFBS.FeedbackType = FEEDBACK_NONE;              // Overwrite feedback type, because in profile␣
↪feedback is encoder

var SENG = global_axis.get_engine_settings();   // Save information about engine (nominal current,␣
↪steps per revolution)

var SENT = global_axis.get_entype_settings();   // Save information about engine type

var SEDS = global_axis.get_edges_settings();    // Save information about edges

/*
* Clear FRAM for synchronization
* real full step with full step of firmware
*/
log("Clearing FRAM", 1);
global_axis.command_clear_fram();
msleep(4000);

/*
* Restore controller settings
*/
global_axis.set_feedback_settings(SFBS);
msleep(100);

global_axis.set_engine_settings(SENG);
msleep(100);

global_axis.set_entype_settings(SENT);
msleep(100);

global_axis.set_edges_settings(SEDS);
msleep(100);

/*
```

```
* Prepare and apply move settings
*/
var SMOV = global_axis.get_move_settings();
SMOV.Speed = 0;
SMOV.uSpeed = 16;
global_axis.set_move_settings(SMOV);
msleep(100);

/*
* Going to the full step and waiting 3 seconds for equilibration
*/
global_axis.command_move(0, 0);
msleep(3000);

/*
* Arrays for measurements and structure for GPOS
*/
var MicroStepsToRight = [];
var MicroStepsToLeft = [];
var EncToRight = [];
var EncToLeft = [];
var GPOS;

/*
* Start moving
*/
global_axis.command_right();

/*
* Skipping a some first counts for the stable experiment
*/
for (var i = 0; i < SKIP_ENCODER_COUNT; i++)
{
  GPOS = global_axis.get_position();
  var EncPos = GPOS.EncPosition;

  while (EncPos == GPOS.EncPosition)
  {
    msleep(40);
    GPOS = global_axis.get_position();
  }

  EncPos = GPOS.EncPosition;
}

/*
* Start measurements
*/
for (var i = 0; i < STORE_COUNT_MICROSTEPS; i++)
{
  GPOS = global_axis.get_position();
  var EncPos = GPOS.EncPosition;

  while (EncPos == GPOS.EncPosition)
  {
    msleep(40);
    GPOS = global_axis.get_position();
  }
```

```
  EncPos = GPOS.EncPosition;

  MicroStepsToRight[i] = GPOS.Position * MICROSTEPS + GPOS.uPosition;
  EncToRight[i] = GPOS.EncPosition;
}

/*
* Stop moving
*/
global_axis.command_stop();

/*
*  Start moving and measurements
*/
global_axis.command_left();

for (var i = 0; i < STORE_COUNT_MICROSTEPS; i++)
{
  GPOS = global_axis.get_position();
  var EncPos = GPOS.EncPosition;

  while (EncPos == GPOS.EncPosition)
  {
    msleep(40);
    GPOS = global_axis.get_position();
  }

  EncPos = GPOS.EncPosition;

  MicroStepsToLeft[STORE_COUNT_MICROSTEPS - 1 - i] = GPOS.Position * MICROSTEPS + GPOS.uPosition;
  EncToLeft[STORE_COUNT_MICROSTEPS - 1 - i] = GPOS.EncPosition;
}

/*
* Stop moving
*/
global_axis.command_stop();

/*
* Check all values
*/
for (var i = 0; i < STORE_COUNT_MICROSTEPS; i++)
{
  var diffMicrosteps = MicroStepsToRight[i] - MicroStepsToLeft[i];
  var diffConts = EncToRight[i] - EncToLeft[i];

  if (DEBUG)
  {
    log(MicroStepsToRight[i] + " - " + MicroStepsToLeft[i] + " = " + diffMicrosteps + "␣
↪microstep(s)\t" +
      EncToRight[i] + " - " + EncToLeft[i] + " = " + diffConts + " count(s)", 3);
  }

  if (diffConts != 1)
  {
    log("Script error! Try again", 1);
  }
```

```
}

var RightB = (MicroStepsToRight[TRUST_INDEX] * EncToRight[0] - MicroStepsToRight[0] *␣
↪EncToRight[TRUST_INDEX]) / (EncToRight[0] - EncToRight[TRUST_INDEX]);
var LeftB = (MicroStepsToLeft[TRUST_INDEX] * EncToLeft[0] - MicroStepsToLeft[0] * EncToLeft[TRUST_
↪INDEX]) / (EncToLeft[0] - EncToLeft[TRUST_INDEX]);

log("Right counts show B = " + RightB, 2);
log("Left counts show B = " + LeftB, 2);
log("Aver B = " + ((RightB + LeftB) / 2), 3);
```

6.5.4.13 Скрипт дискретного движения

```
/*
* Discrete motion script
*
* Description of the script:
*  The script opens two axes by serial numbers. Moves along the X axis to a certain coordinate.␣
↪Then it begins to shift discretely along the Y axis, with a programmable pause after each offset.
*  The script can be useful if you are using the system to scan an area and/or capture frames
*
* Warning: enter the serial numbers of your axes!
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

// Enter the serial numbers of your axes.
serial_number_x = 14889;
serial_number_y = 14888;

var x = new_axis(serial_number_x);
var y = new_axis(serial_number_y);

// Installing the source data
var x_target_coordinate = 5000; // first border coordinate
var delay = 100; // delay in milliseconds

var y_first_border = 0; // first border coordinate
var y_second_border = 5000; // second border coordinate
var y_step = 100
var y_direct = 1

// Calibration and positioning of the axes to their original positions.
x.command_stop(); // send STOP command (does immediate stop)
x.command_zero(); // send ZERO command (sets current position and encoder value to zero)
y.command_stop(); // send STOP command (does immediate stop)
y.command_zero(); // send ZERO command (sets current position and encoder value to zero)

x.command_move(x_target_coordinate); // move to wards one border
x.command_wait_for_stop(10); // wait until controller stops moving
y.command_move(y_first_border); // move towards one border
y.command_wait_for_stop(10); // wait until controller stops moving

// Movement in discrete samples along one axis from the end to the end
```

```
// with a delay after each movement.
while (1) { // infinite loop

// Choosing the direction of travel
if (y.get_position() >= y_second_border)
{
  y_direct = -1
}
if (y.get_status().CurPosition <= y_first_border)
{
  y_direct = 1
}

// Movement in a given direction
y.command_movr(y_step*y_direct); // move towards another border
y.command_wait_for_stop(10); // wait until controller stops moving
msleep(delay);
}
```

6.5.4.14 Экспоненциальное изменение позиции использующие user units

```
/*
* Exponential position change in user units script
*
* Description of the script:
*  The script performs discrete control of movement according to a certain law of motion. The␣
↪amplitude and the law of displacement are given. A correction speed is used to maintain the␣
↪positioning accuracy.
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

// Main characteristics
var time_discre = 10; // Discreteness of movement control(ms)
var end_err = 0.01; // The accuracy of reaching the final coordinate of the movement.

var full_move = 6; // Total movement in mm

// If you change the equation of motion, you will need to change the equation for the correction␣
↪velocity, since this change is not linear.
var K = 0.4;
var Glob_err = 0;

// Advanced setting.
var mm_per_step = 0.00125; // Distance in gr for 1 completed step.
var calb = new_calibration(mm_per_step, get_engine_settings().MicrostepMode); // create␣
↪calibration structure

// Setting the starting position.
command_stop(); // send STOP command (does immediate stop)
command_wait_for_stop(10); // wait until controller stops moving
command_zero(); // send ZERO command (sets current position and encoder value to zero)
```

```
log("Start:");
// Setting 0 speeds and accelerations.
zero_movesettings();

//
go_position(full_move, time_discre)


// Function for calculating the set speed and acceleration
function set_movesettings(time, corr_speed)
{
  // Speed, Accel, Decel setting.
  var m = get_move_settings_calb(calb); // read movement settings from the controller

  // The equation of speed is equal to the derivative of the equation of motion.
  m.Speed =  K*Math.exp(K*time/1000 )+ corr_speed;

  set_move_settings_calb(m, calb); // write movement settings into the controller

log("Speed = " +m.Speed);
log("corr_speed = " +corr_speed);
}

// Set the initial parameters of motion
function zero_movesettings()
{
  var m = get_move_settings_calb(calb); // read movement settings from the controller
  m.Speed = 0.1; // set movement speed

  set_move_settings_calb(m, calb); // write movement settings into the controller
}

// A function of calculating target coordinates from time
function current_target_coordinate(time)
{
  // The equation of motion. The time is set in milliseconds. The position at the initial time is␣
↪0.
  return (Math.exp(K*time/1000) - 1);
}

// The calculation of the correction speed
function speed_corr(err_pos)
{
  Glob_err = Glob_err + err_pos*0.35;
  return  Glob_err;
}

// The main moving
function go_position(full_time, time_discre)
{
  Glob_err = 0;
  var end_position = full_move;

  // Setting the movement to the desired coordinate.
  command_move_calb(end_position, calb);

  // Pause before starting to move, to turn on the power button.
  msleep(300);
```

```
  var mas = 1;
  var basetime = new Date();
  var curr_time = new Date();
  var err_pos = 0;
  var pos = 0;
  var pos1 = 0;
  var i = time_discre;
  do {
    // If you do not need position feedback, you can instead speed_corr(err_pos) write 0
    set_movesettings(i+1, speed_corr(err_pos));

    // Waiting for the end of a discrete time interval
    do {
      curr_time = new Date - basetime;
      msleep(1);
    }
    while ((curr_time) < i); //

    // Reading the actual and calculating the planned coordinate if used.
    pos =  get_position_calb(calb).Position;
    pos1 =  current_target_coordinate(i);

    // Calculation of the position error.
    err_pos =  (pos1 -  pos);

    log("time = " + i);
    log("err_pos = " + err_pos);

    i = i + time_discre
  }
  while (Math.abs(pos - end_position) > end_err); // Completion when the end of the movement is␣
↪reached with the specified accuracy.
}
```

6.5.4.15 Шаговый скрипт использующий user units

```
/*
* For calb step script
*
* Description of the script:
*  In the script, the user units are configured, after which there is a departure to the leftmost␣
↪position and then a certain number of shifts occur until the right position is reached. Each␣
↪position is recorded in a .csv file.
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

command_home(); // send HOME command (find home position)
command_wait_for_stop(100);
command_zero();
command_wait_for_stop(100);

// Setting the value to convert to user unit
var mm_per_step = 0.00125; // steps to distance translation coefficient
```

```
var calb = new_calibration(mm_per_step, get_engine_settings().MicrostepMode); // create␣
↪calibration structure

// Setting boundaries and movement step
// Boundaries can be set manually or taken from limit constraints
var edge = get_edges_settings_calb(calb);
var first_border = edge.LeftBorder; //0;
var second_board = edge.RightBorder;//23;
var shift = 5;//step move;
var delay = 2000;// The delay of movement

var f = new_file("file.csv"); // Choose a file name and path
f.open(); // Open a file
f.resize(0);
f.seek( 0 ); // Seek to the beginning of the file

command_move_calb(first_border, calb); // Move to the starting position
command_wait_for_stop(delay); // Wait until controller stops moving
var i = 1;
var time = 0;
var step = (second_board - first_border)/shift;
f.write(0+ ","  +  get_status().CurPosition + "," +  get_status().uCurPosition+ "," + "\n" ); //␣
↪Get current position, potentiometer value and date and write them to file
do {
  time = i*delay;
  command_movr_calb(shift, calb); // Move to the next position
  command_wait_for_stop(delay); // Wait until controller stops moving
  f.write(time + ","  +  get_position_calb(calb).Position + "," +  get_position_calb(calb).
↪EncPosition+ "," + "\n" ); // Get current position, potentiometer value and date and write them␣
↪to file
  i = i+1;
} while (  get_position_calb(calb).Position+shift < second_board )
f.close(); // Close the file
```

6.5.4.16 Шаговый скрипт

```
/*
* Step script
*
* Description of the script:
*  In the script, there is a departure to the leftmost position and then a certain number of␣
↪shifts occur until the right position is reached. Each position is recorded in a .csv file.
*
* Note: The script is similar to the "for_calb_step" script, only in this script the offset occurs␣
↪at the step mode
*
* To run the script, upload it to the XILab software
*/

command_home(); // send HOME command (find home position)
command_wait_for_stop(100);
command_zero();
command_wait_for_stop(100);
var edge = get_edges_settings();
var first_border = edge.LeftBorder;
var second_board = edge.RightBorder;
var count = 10;
```

```
var f = new_file("E:/a.csv"); // Choose a file name and path
f.open(); // Open a file
f.seek( 0 ); // Seek to the beginning of the file

var delay = 2000;
command_move(first_border); // Move to the starting position
command_wait_for_stop(delay); // Wait until controller stops moving
var i = 1;
var time = 0;
var shift = 6;
var step = (second_board - first_border)/shift;
f.write(0+ "," + get_status().CurPosition + "," + get_status().uCurPosition+ "," + "\n" ); //␣
↪Get current position, potentiometer value and date and write them to file
do {
  time = i*delay;
  command_movr(step, 0); // Move to the next position
  command_wait_for_stop(delay); // Wait until controller stops moving
  f.write(time + "," + get_status().CurPosition + "," + get_status().uCurPosition+ "," + "\n" );
↪ // Get current position, potentiometer value and date and write them to file
  i = i+1;
} while (i < shift )
f.close(); // Close the file
```

6.5.4.17 Тест калибровки домашней позиции сигналу со входа EXTIO

```
/*
* Homing test with extio
*
* Description of the script:
*  The script starts calibration when a signal is received from a general purpose digital input/
↪output (extio)
*
* How to connect wires?
*  You must connect to the HDB-26 connector (pin 25 on the controller).
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

const MVCMD_ERROR = 0x40;
const MVCMD_RUNNING = 0x80;

var axis = new_axis(get_next_serial(0));
var m = axis.get_extio_settings();
var s = axis.get_status();

var count_error = 0;
var count_good = 0;

function BorderOff()
{
  m.EXTIOSetupFlags = 0x01;
  m.EXTIOModeFlags = 0x10;

  axis.set_extio_settings(m);
```

```
}

function BorderOn()
{
  m.EXTIOSetupFlags = 0x01;
  m.EXTIOModeFlags = 0x00;

  axis.set_extio_settings(m);
}

function BorderCycle()
{
  BorderOn();
  msleep(50);

  BorderOff();
  msleep(2500);
}

log(">>> Start testing", 3);

while (1)
{
  axis.command_home();
  msleep(1500);

  BorderCycle();
  BorderCycle();

  command_wait_for_stop(100);

  s = axis.get_status();

  if (s.MvCmdSts & MVCMD_ERROR)
  {
    count_error++;

    log(">>> Alarm! Homing broken", 1);
  }
  else
  {
    count_good++;

    if (!(count_good % 50))
      log(">>> " + count_good + " cycles were done correct, " + count_error + " cycles were done␣
↪incorrect", 3);
  }
}
```

6.5.4.18 Скрипт движения по sin

```
/*
*  Motion by sin function
*
* Description of the script:
*  A script for moving with a change in speed according to the trigonometric law.
*  The script can be useful for precise positioning of a laser or motorized mirror
```

```
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

var delay = 100;

/* Definition of delta and initial phase*/
var df = 0.05;
var f = 0;

/* Definition of package */
var GETS = new Object();

// Initial installations
var move_set = get_move_settings();
var speed = 3000;
var amplitude = 1000;
var nomber = 100;
var time = 1000;
var pos = 0;
var Pi = 3.1415;
df = Pi/nomber;

command_zero();

var pos_read = new Object();
while (1)
{
    pos_read = get_position();

    // Movement to a point with an increase in the velocity amplitude
    for (i = 1; i <= nomber-1; i++)
    {
      f = df*i;

      move_set.Speed = speed * Math.sin(f);
      pos =  amplitude*i /nomber;

      set_move_settings(move_set);
      command_move(pos);
      while (Math.abs(pos_read.Position - pos)>move_set.Speed/10)
      {
        pos_read = get_position();
      }

    }

    // Movement to a point with a decrease in the velocity amplitude
    for (i = nomber-1; i >= 1; i--)
    {
      f = df*i;

      move_set.Speed = speed * Math.sin(f);
      pos =  amplitude*i /nomber;
```

```
      set_move_settings(move_set);
      command_move(pos);
      while (Math.abs(pos_read.Position - pos)>move_set.Speed/10)
      {
        pos_read = get_position();
      }

    }

  msleep(1000);

}
```

6.5.4.19 Скрипт перемещения по сигналу со входа EXTIO. Движение осуществляется в user units

```
/*
* Move EXTIO calb script
*
* Description of the script:
*  The script moves to one of the 2 specified points, depending on the state of the EXTIO input.
↪The movement is carried out in user units.
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and
↪structures.
*
* To run the script, upload it to the XILab software
*/

// Main characteristics
var time_discre = 10; // Discreteness of movement control(ms)
var nomspeed = 5;
var end_err = 0.015;

var low_position = 0; // Move to position for low EXTIO
var high_position = 180; // Move to position for high EXTIO

// Advanced setting.
var gr_per_step = 0.015; // Distance in gr for 1 completed step.
var calb = new_calibration(gr_per_step, get_engine_settings().MicrostepMode); // create
↪calibration structure

// Setting the starting position.
command_stop(); // send STOP command (does immediate stop)
command_wait_for_stop(10); // wait until controller stops moving
command_home();
command_zero(); // send ZERO command (sets current position and encoder value to zero)


log("Start:");
// Setting 0 speeds and accelerations.
movesettings();

extiosettings();
//
go_position(time_discre)

function extiosettings()
```

```
{
  var extsettings = get_extio_settings();
  extsettings.EXTIOSetupFlags = 0x00;
  extsettings.EXTIOModeFlags = 0x00;
  set_extio_settings(extsettings);
}

// Set the initial parameters of motion
function movesettings()
{
  var m = get_move_settings_calb(calb); // read movement settings from the controller
  m.Speed = nomspeed; // set movement speed

  set_move_settings_calb(m, calb); // write movement settings into the controller
}


// The main moving
function go_position(time_discre)
{
  var oldstate = 0;
  var maskstate = 32;

  // Setting the movement to the desired coordinate.
  command_move_calb(low_position, calb);

  // Pause before starting to move, to turn on the power button.
  msleep(300);

  while(1) {

  //
  var status = get_status_calb(calb);
  if ((status.GPIOFlags & maskstate) != oldstate)
  {
    log(status.GPIOFlags);
    if (oldstate)
      command_move_calb(high_position, calb);
    else
      command_move_calb(low_position, calb);

    oldstate = status.GPIOFlags & maskstate;
  }
  // Waiting for the end of a discrete time interval
  msleep(time_discre);
  }
}
```

6.5.4.20 Вероятные тесты

```
/*
* Probabilistic tests
*
* Description of the script:
*  The script runs a set of repeatable tests a certain number of times and is expected to fail.
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
```

```
*
* To run the script, upload it to the XILab software
*/
```

Посмотреть полный код

6.5.4.21 Скрипт выполняющий ряд смещений с калибровкой

```
/*
* Several shifts with calibration script
*
* Description of the script:
*  This program makes shifts given number of times to the specified distance, and stands the␣
↪appointed time after every shift. First it goes left, then it returns back to the origin and␣
↪repeats all movements to right.
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

const LEFT = -1;
const RIGHT = 1;

var axes = [];
var number_of_axes = 0;
var last_serial = 0;
while (serial = get_next_serial(last_serial)) // get next serial number and repeat for each axes.
{
    axes[number_of_axes] = new_axis(serial);
    log("Found axis " + number_of_axes + " with serial number " + serial);
    number_of_axes++;
    last_serial = serial;
}

// Start the main function for all avaliable axes
for (var i = 0; i < number_of_axes; i++)
{
    main(axes[i]);
}


function main(axis)
{
    axis.command_move(0,0); // Go back to the origin.
    msleep(500);

/* Creating and filling the calibration structure for specifying distances in um */
    var calibration = new Object;
    calibration.A = 5; // 1 step correspond to 5 um for 8MT50-100BS1
    calibration.MicrostepMode = axis.get_engine_settings().MicrostepMode; // Get MicrostepMode␣
↪from controller settings
/*******************************************************/

/* Main cycle */

    var N = 10; // Number of shifts
```

```
    var stand_time  = 3000; // Stand time in ms
    var shift = 10; // Distance of shift in um

    MakeShifts(axis, LEFT,   shift, N, stand_time, calibration); // Make 10 shifts to the left
    MakeShifts(axis, RIGHT, shift, N, stand_time, calibration); // Make 10 shifts to the right
    MakeShifts(axis, RIGHT, shift, N, stand_time, calibration); // Make 10 shifts to the right␣
↪again
    MakeShifts(axis, LEFT,   shift, N, stand_time, calibration); // Make 10 shifts to the left
}

function MakeShifts(axis, Direction, ShiftDistance, ShiftsQuantity, StandTime, Calibration)
{
/**
    This function makes shifts which number is specified by ShiftQuantity and length is specified␣
↪by ShiftDistance. After every shift it stands StandTime miliseconds. Calibration parameter is a␣
↪structure for convertation between steps and micrometers.
*/
    for (var i = 0; i < ShiftsQuantity; i++)
    {
        axis.command_movr_calb(Direction*ShiftDistance, Calibration);
        axis.command_wait_for_stop(100);
        msleep(StandTime);
    }
}
```

6.5.4.22 Тест на пропуск шагов

```
/*
* Steps loss test
*
* Description of the script:
*  The script was written to check for skipping steps
*  It can be useful for diagnosing problematic stages
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

function abs(x)
{
    return ((x > 0) ? x : -x);
}

/*
* Set home settings
*/
var SHOM = get_home_settings()
SHOM.FastHome = 500;
SHOM.uFastHome = 0;
SHOM.SlowHome = 20;
SHOM.uSlowHome = 0;
SHOM.HomeDelta = 300;
SHOM.uHomeDelta = 0;
SHOM.HomeFlags = HOME_STOP_FIRST_LIM;
set_home_settings(SHOM);
```

```
/*
* Check for encoder
*/
var encoder = 1
command_zero()
var first = get_status().EncPosition
command_movr(100)
command_wait_for_stop(300)
var second = get_status().EncPosition
if(abs(second - first) < 2)
{
  encoder = 0
  log("It seems like here are no encoder", 2)
}

command_home();
command_wait_for_stop(300);
command_zero();
msleep(200);
// Store old move settings
var MOV = get_move_settings();

// Set fast speed and accel\decel
var MOV2 =get_move_settings();
MOV2.Speed = MOV.Speed * 2;
MOV2.Accel = MOV.Accel * 2;
MOV2.Decel = MOV.Decel * 2;

for(var i=0; i < 1; i++) // Set default settings first
{
  set_move_settings(MOV2);

  // Move long...
  command_move(20000);
  command_wait_for_stop(300);

  // Set prev settings back
  set_move_settings(MOV);

  // Move back
  command_move(0);
  command_wait_for_stop(300);
}

if (encoder > 0)
{
  var lost = get_status().EncPosition
  if (abs(lost) > 1)
  {
    log("Lost " + lost + " pulses", 1)
  }
  else
  {
    log("All is OK");
  }
}
else
```

```
{
  log("Do final homing...")
  command_home()
  command_wait_for_stop(300)
  var  lost = get_status().CurPosition
  if (abs(lost) > 2)
  {
    log("Lost " + lost + " steps", 1)
  }
  else
  {
    log("All is OK");
  }
}
```

6.5.4.23 Скрипт тестирования синхронизации

```
/*
* Sync test script
*
* Description of the script:
*  The script is written to demonstrate the work of synchronization, and also checks its␣
↪operability
*
* For test you must short special pins on the controller. Pin 5 (sync in) and pin 6 (sync out).␣
↪See https://doc.xisupport.com/en/8smc5-usb/8SMCn-USB/Technical_specification/Appearance_and_
↪connectors/One_axis_system.html
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

const dX = 4.5;
const dot_num = 5;
const micro2mili = 1000;

var ASIA = [];

for (var i = 0; i < dot_num; i++)
  ASIA[i] = new Object();

var calb = new_calibration(1, MICROSTEP_MODE_FRAC_256);

function abs(x)
{
  return (x > 0) ? x : -x;
}

function set_default()
{
  // SSNI settings
  var SSNI = get_sync_in_settings_calb(calb);
  SSNI.SyncInFlags = SYNCIN_ENABLED | SYNCIN_GOTOPOSITION;
  set_sync_in_settings_calb(SSNI, calb);
```

```
  // SSNO settings
  var SSNO = get_sync_out_settings(calb);
  SSNO.SyncOutFlags = SYNCOUT_ENABLED;
  SSNO.Accuracy = 0.5;
  set_sync_out_settings(SSNO, calb);

  // SMOV settings
  var SMOV = get_move_settings_calb(calb);
  SMOV.Speed = 1000;
  SMOV.Accel = 500;
  SMOV.Decel = 1000;
  SMOV.AntiplaySpeed = 50;
  set_move_settings_calb(SMOV, calb);

  // SENG settings
  var SENG = get_engine_settings();
  SENG.NomSpeed = 5000;
  SENG.EngineFlags = ENGINE_ACCEL_ON | ENGINE_LIMIT_VOLT | ENGINE_LIMIT_CURR;
  SENG.Antiplay = 50;
  SENG.MicrostepMode = MICROSTEP_MODE_FRAC_256;
  SENG.StepsPerRev = 200;
  set_engine_settings(SENG);
}

function send_all_asia()
{
  for (var i = 0; i < dot_num; i++)
    command_add_sync_in_action_calb(ASIA[i], calb);
}

function check_all_asia()
{
  var GETS;

  for (var i = 0; i < dot_num; i++)
  {
    log("> Checking movement ASIA[" + i + "]", 3);
    msleep(ASIA[i].Time / micro2mili);
    GETS = get_status_calb(calb);
    if (abs(GETS.CurPosition - ASIA[i].Position) < dX)
      log(">>> OK! GETS.CurSpeed = " + GETS.CurSpeed, 3);
    else
      log(">>> Error! GETS.CurPosition = " + GETS.CurPosition + ", ASIA[" + i + "].Position = " +␣
↪ASIA[i].Position, 1);
    }
}

function test1()
{
  ASIA[0].Position = 22.5;
  ASIA[0].Time = 300000;

  ASIA[1].Position = 45.0;
  ASIA[1].Time = 300000;

  ASIA[2].Position = 32.5;
  ASIA[2].Time =300000;
```

```
  ASIA[3].Position = 10;
  ASIA[3].Time = 300000;

  ASIA[4].Position = -11.5;
  ASIA[4].Time = 300000;
}

function test2()
{
  ASIA[0].Position = -22.5;
  ASIA[0].Time = 300000;

  ASIA[1].Position = -45.0;
  ASIA[1].Time = 300000;

  ASIA[2].Position = -32.5;
  ASIA[2].Time =300000;

  ASIA[3].Position = -10;
  ASIA[3].Time = 300000;

  ASIA[4].Position = 11.5;
  ASIA[4].Time = 300000;
}

function test3()
{
  ASIA[0].Position = -6;
  ASIA[0].Time = 300000;

  ASIA[1].Position = 6;
  ASIA[1].Time = 300000;

  ASIA[2].Position = -6;
  ASIA[2].Time =300000;

  ASIA[3].Position = 6;
  ASIA[3].Time = 300000;

  ASIA[4].Position = -6;
  ASIA[4].Time = 300000;
}

command_zero();
set_default();
log(">>> Function test1() started", 3)
test1();
send_all_asia();
check_all_asia();
msleep(500);

command_zero();
set_default();
log(">>> Function test2() started", 3)
test2();
send_all_asia();
check_all_asia();
msleep(500);
```

```
command_zero();
set_default();
log(">>> Function test3() started", 3)
test3();
send_all_asia();
check_all_asia();
msleep(500);
```

6.5.4.24 Скрипт тестирования ошибок синхронизации

```
/*
* Sync bug test script
*
* Description of the script:
*  This script does synchronization test to find sync bug in firmware 3.9.9, 3.8.7, 3.9.10.
*  This test only for hardware from ver. 2.2.0 to ver. 2.2.4.
*  To begin the test, the first what you need to do is connect synchronization pin 15 to pin 16 on␣
↪CN1 jack.
*  The scripts automatically set all necessary settings for test, and automatically beginning the␣
↪test.
*  As a result you will see a log message with current position of motor and text OK! (green text)␣
↪or ERROR! (red text).
*
* Note: This is a rather difficult script to learn, since it uses a large number of commands and␣
↪structures.
*
* To run the script, upload it to the XILab software
*/

var axes = [];
var number_of_axes = 0;
var last_serial = 0;

while (serial = get_next_serial(last_serial)) // get next serial number and repeat for each axes.
{
  axes[number_of_axes] = new_axis(serial);
  number_of_axes++;
  last_serial = serial;
  log("Found axis " + number_of_axes + " with serial number " + serial);
}

// SSNI settings
var SSNI = get_sync_in_settings;
SSNI.SyncInFlags = SYNCIN_ENABLED | SYNCIN_GOTOPOSITION; //set flags for synchronization in for␣
↪enabled and absolute position
SSNI.ClutterTime = 4;// set sync in Clutter Time
SSNI.Position = 0;// set sync in absolute position
SSNI.Speed = 500;// set sync in speed
set_sync_in_settings(SSNI);// set synchronization in settings

// GSNO settings
var GSNO = get_sync_out_settings;
GSNO.SyncOutFlags = SYNCOUT_ENABLED | SYNCOUT_ONPERIOD;  // set flags for synchronization out for␣
↪enabled out and period
GSNO.SyncOutPeriod = 200;// set sync out period
GSNO.SyncOutPulseSteps = 2000;// set sync out Pusle width
```

```
set_sync_out_settings(GSNO);  // set synchronization out settings

for (i = 0; i < 10; i++)  // testing cycle
{
  GETS = get_status();// get information about device
  command_movr(201, 0);  // shifting position for 201 steps
  command_wait_for_stop(100); // Wait until the movement is complete

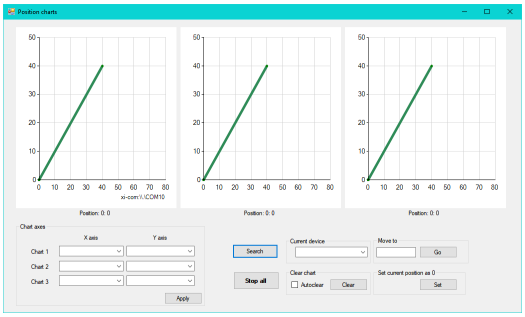
  if (GETS.CurPosition != 0)// get device current position and compare it with 0
  {
    log(">>> Error! GETS.CurPosition = " + GETS.CurPosition, 1);// log ERROR, bug is exist
  }
  else
  {
    log(">>> OK! GETS.CurPosition = " + GETS.CurPosition, 3); // log OK, bug does not exist
  }
}
```

6.6 Неподдерживаемые примеры

- *Пример шестиосного интерфейса XILab*
- *Примеры для работы с аттенюатором для Python и LabView*
- *Программа для отправки команд на контроллер XIMC с контроллера AVR*
- *Многоосевой интерфейс на Python*

Важно: Ниже приведены примеры, найденные в Интернете с открытым исходным кодом За работоспособность примеров отвечает их разработчики

6.6.1 Пример шестиосного интерфейса XILab



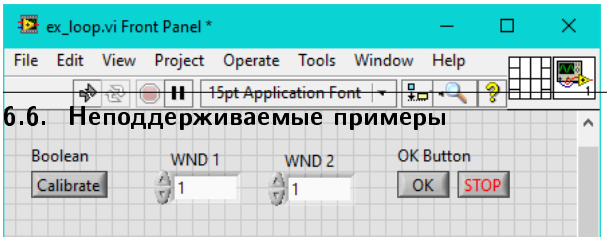
Программа позволяет работать с 6 устройствами в режиме реального времени на 3 графиках **Предварительно скомпилированные примеры были собраны с помощью Visual Studio 2013**.

Пример и руководство выложены на GitHub: https://github.com/sushchev/testwfa.

6.6.2 Примеры для работы с аттенюатором для Python и LabView

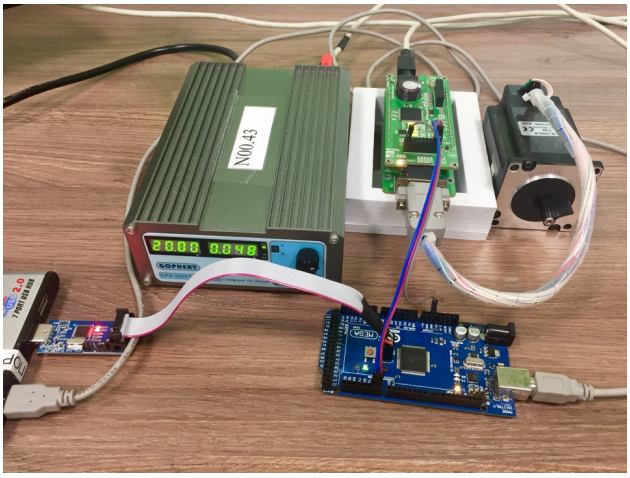
В этом проекте было реализовано несколько простых примеров использования протокола LibXIMC с контроллером 8SMC4/5-USB. В качестве управляемого устройства использовался вращающийся оптический аттенюатор. *Примеры кода были написаны на Python 3.4, а также в среде программирования LabView.*

Пример и руководство выложены на GitHub: https://github.com/Negrebetskiy/Attenuator.



6.6.3 Программа для отправки команд на контроллер XIMC с контроллера AVR

Важно: Написано в AtmelStudio7 для Atmel ATMEGA2560 16AU 1432 и используется с контроллером XIMC и шаговым двигателем ST5918L3008-B



Что делает эта программа:

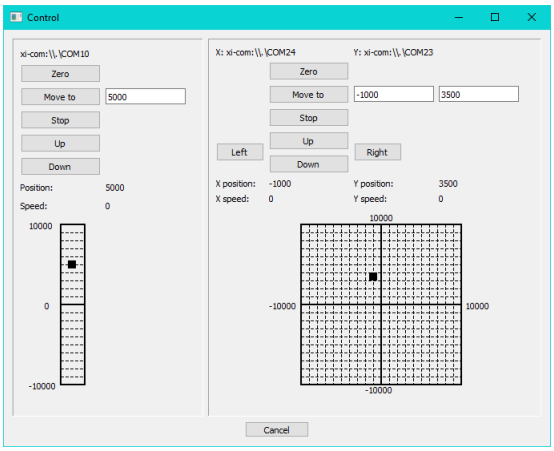
- Получает текущее положение шагового двигателя.
- Перемещает его влево на 3 секунды и останавливает шаговый двигатель.
- Перемещает его в исходное положение

Пример и руководство выложены на GitHub: https://github.com/ntarabrina/XIMC_command.

6.6.4 Многоосевой интерфейс на Python

Пример программы с графическим интерфейсом, предназначенной для работы с несколькими сервоконтроллерами (до шести включительно, при необходимости легко увеличить это число) с помощью протокола XIMC. GUI был создан через пакет PyQt5.

Пример и руководство выложены на GitHub: https://github.com/Negrebetskiy/MultiAxleGUI.



Глава 7

Управление контроллером по Ethernet

7.1 8Eth1 адаптер

7.1.1 Обзор Ethernet адаптера

Примечание: Адаптер **8SMC4-USB-Eth** был переименован в **8Eth1**[1]. Все имеющиеся инструкции в нашей документации актуальны и применимы к обоим устройствам.

7.1.1.1 Общая информация и внешний вид



Рис. 7.1: Внешний вид адаптера 8Eth1

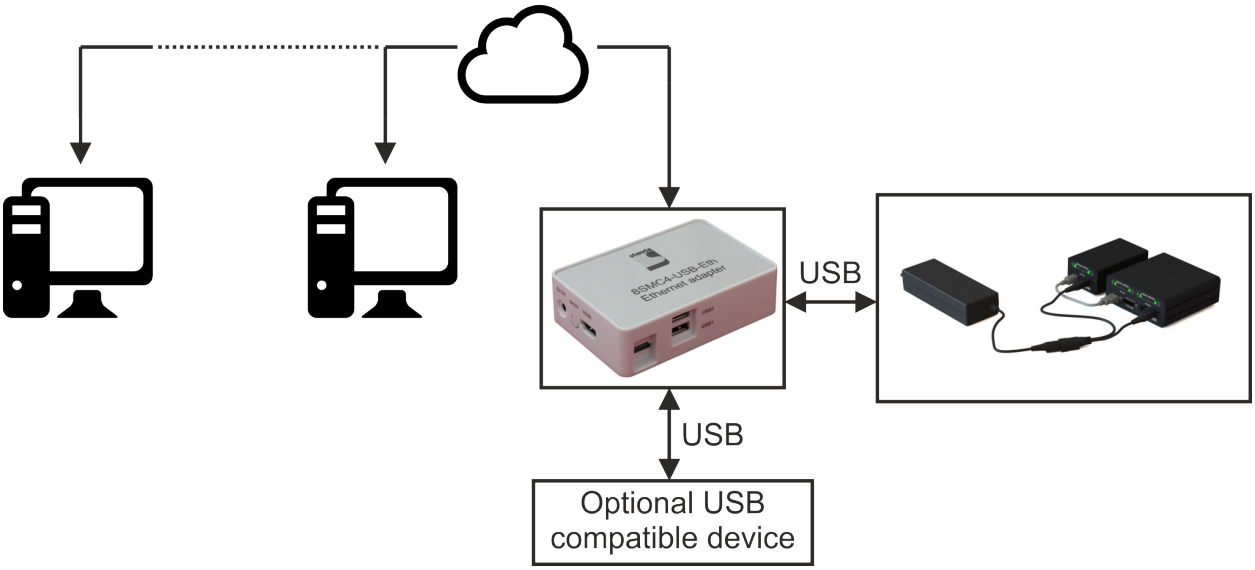
8Eth1[1] - это универсальное устройство, основанное на одноплатном компьютере Cubieboard2 под управлением ОС Linux, основной задачей которого является предоставление доступа к контроллерам моторов через различные Ethernet-ориентированные интерфейсы. Помимо этого, 8Eth1 представляет собой агрегатор сервисов, связанных с наиболее широко распространенными областями применения

[1] Артикул был изменен в 2020 г., ранее устройство имело артикул 8SMC4-USB-Eth.

контроллеров. Так, например, в комплектации по умолчанию устройство оснащено сервисом сетевой трансляции с подключенных web-камер, системой автоматического обнаружения в LAN, а также несколькими интерфейсами для удаленной работы с контроллерами:

1. Интерфейс системы управления *TANGO*.
2. Интерфейс *XIMC* (совместимый с XILab и libximc).

Помимо этого **8Eth1** также имеет встроенный *веб-интерфейс администрирования*, который позволяет с лёгкостью управлять устройством и следить за его состоянием.



Внешний вид системы с разных ракурсов (жирным отмечены разъёмы, используемые в текущей версии устройства):



Рис. 7.2: Вид адаптера спереди. Слева направо: **разъём питания**, **кнопка включения питания**, разъём HDMI



Рис. 7.3: Вид адаптера справа. Слева направо: разъём для микро-SD карты, **два разъёма USB type A**



Рис. 7.4: Вид адаптера сзади. Слева направо: **Ethernet разъём**, mini-USB типа B, кнопка перехода в режим FEL, аналоговые вход/выход для микрофона и наушников



Рис. 7.5: Вид адаптера слева. Отверстие с ИК приёмником

Используя USB хабы , вы можете подключить к этому Ethernet-адаптеру до 36 контроллеров.

Важно: Лучше использовать древовидную структуру, для подключения через USB. Чем меньше USB хабов вы используете, тем лучше.

Работа с многоосным контроллером 8SMC5-ETHERNET/RS232-B19 через Ethernet не отличается от работы с 8Eth1



Рис. 7.6: Внешний вид многоосного контроллера движения 8SMC5-ETHERNET/RS232-B19 (вид спереди)



Рис. 7.7: Внешний вид многоосного контроллера движения 8SMC5-ETHERNET/RS232-B19 (вид сзади)

Контроллер 8SMC4-ETHERNET/RS232-B19 разработан специально для управления до 12 мехатрон-

ными системами на основе стандартных технологий постоянного тока, сервопривода или шаговой технологии. Корпус контроллера может быть установлен в стандартные 19-ти дюймовые промышленные шкафы. Несколько корпусов могут быть подключены в одну сеть через порт Ethernet. система может быть автоматически установлена в одноранговым режиме без какой-либо специальной инфраструктуры или главных серверов для работы. Также каждая ось может управляться через отдельный порт RS232.

8SMC4-ETHERNET/RS232-B19 содержит все необходимые подсистемы, в том числе: управление, блоки питания и т. д., которые поддерживают одновременную работу всех осей. Контроллер может быть подключен к ряду рабочих станций на базе ПК через Ethernet. Все пользователи имеют доступ к настройке диапазона доступных осей и назначению прав доступа. С точки зрения программиста, работа со всеми осями системы очень похожа на работу с простой коробкой 8SMC4-USB. Количество осей, доступных пользователю и обрабатываемых системой, может достигать 50000. На передней панели расположены светодиодные индикаторы питания и состояния, кнопки ручного управления, кнопка блокировки системы и порт Ethernet. На задней панели расположены разъемы двигателя и отдельные порты RS232 для каждой оси, разъемы ввода-вывода синхронизации, разъем питания и кнопка включения/выключения. Система охлаждается через отверстия в решетке на верхней и нижней стенках коробки.

7.1.2 Администрирование

Примечание: Адаптер **8SMC4-USB-Eth** был переименован в **8Eth1**[1] . Все доступные инструкции в нашем руководстве актуальны и применимы к обоим устройствам.

7.1.2.1 Конфигурация сети

7.1.2.1.1 Подключение контроллера по сети, в которой нет DHCP-сервера

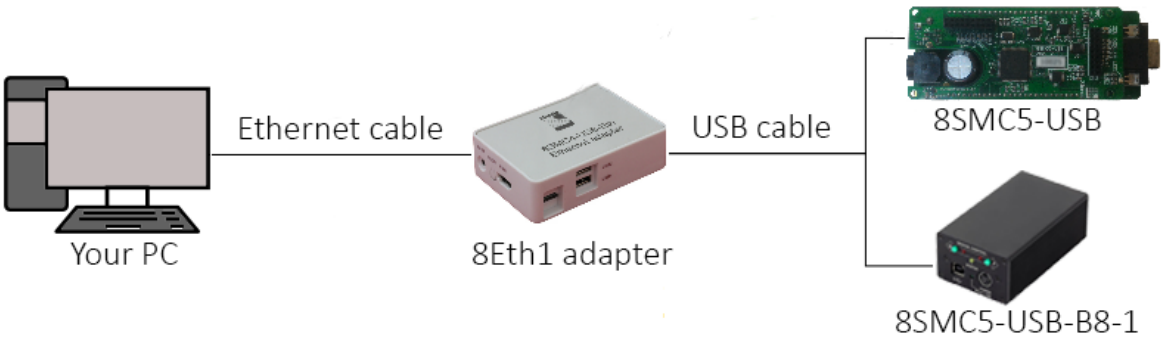


Рис. 7.8: Контроллер подключается к компьютеру напрямую. Компьютер не имеет выход в Интернет

- Подключите 8Eth1 адаптер к сетевому интерфейсу вашего компьютера
- Скачайте простой DHCP-сервер
- Настройте и запустите сервер в соответствии с инструкциями
- Используйте *revealer*, чтобы узнать выданный контроллеру IP-адрес, и открыть его в вашем веб-браузере.

[1] Вендор-код был изменен в 2020 году, ранее этот продукт имел вендор-код 8SMC4-USB-Eth.

7.1.2.1.2 Подключение контроллера по сети, в которой есть DHCP-сервер

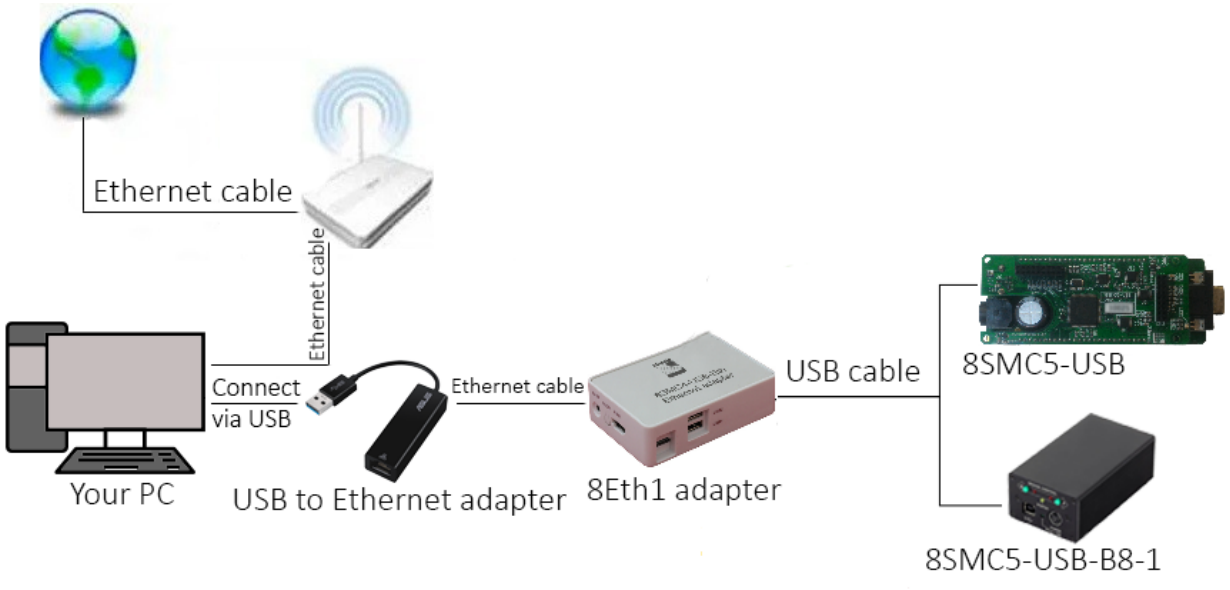


Рис. 7.9: Контроллер подключается к компьютеру через сетевую карту или через переходник. Компьютер имеет выход в Интернет

- Подключите 8Eth1 адаптер к сетевому интерфейсу вашего компьютера. Если у вас только один сетевой интерфейс, вы можете добавить дополнительную сетевую карту к вашему компьютеру или использовать USB-Ethernet адаптер

  > **Предупреждение:** Важно понимать, что добавление DHCP-сервера в локальную сеть, где DHCP уже доступен, может привести к сбою сети. Поэтому очень важно настроить/запустить новый DHCP-сервер на отдельном сетевом интерфейсе, который не подключен к локальной сети.

  Мы протестировали работу с такими адаптерами:

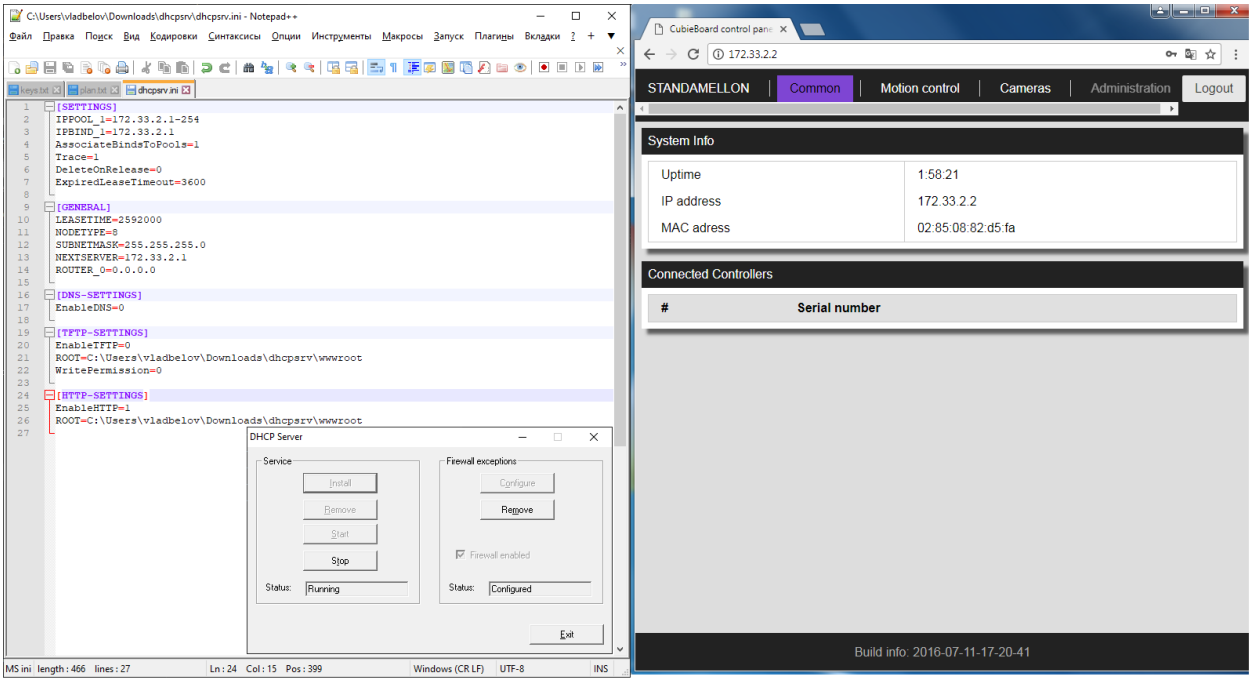


Рис. 7.10: USB-Ethernet адаптер ASUS - **работает**



Рис. 7.11: HUB/Адаптер/USB-C HUB 5 в 1 - **не работает**

- Подключите 8Eth1 адаптер к сетевому интерфейсу вашего компьютера
- Скачайте простой DHCP-сервер
- Настройте и запустите сервер в соответствии с инструкциями
- Используйте *revealer*, чтобы узнать выданный контроллеру IP-адрес, и открыть его в вашем веб-браузере.



Важно: Порт 49150 не должен быть заблокирован! Причиной блокирования зачастую может быть наличие антивирусов, программ, осуществляющих контроль и фильтрацию сетевых пакетов (фаерволы, брандмауэры).

Ethernet-переходник использует multicast и UDP-запросы, поэтому убедитесь, что такие запросы разрешены в вашей локальной сети. Наиболее распространенной причиной проблемы с обнаружением контроллеров в сети является блокировка таких запросов брандмауэром Windows или управляемым сетевым оборудованием.

Примечание: При необходимости вы можете использовать статический IP-адрес. Скачать инструкцию и образ со статическим IP-адресом можно здесь.

7.1.2.2 Автоматическое обнаружение устройств

Для того, чтобы облегчить задачу поиска подключенных к вашей локальной сети устройств **Standa 8Eth1**, мы предоставляем небольшую утилиту под названием «Revealer». Вы можете скачать её со страницы программного обеспечения.

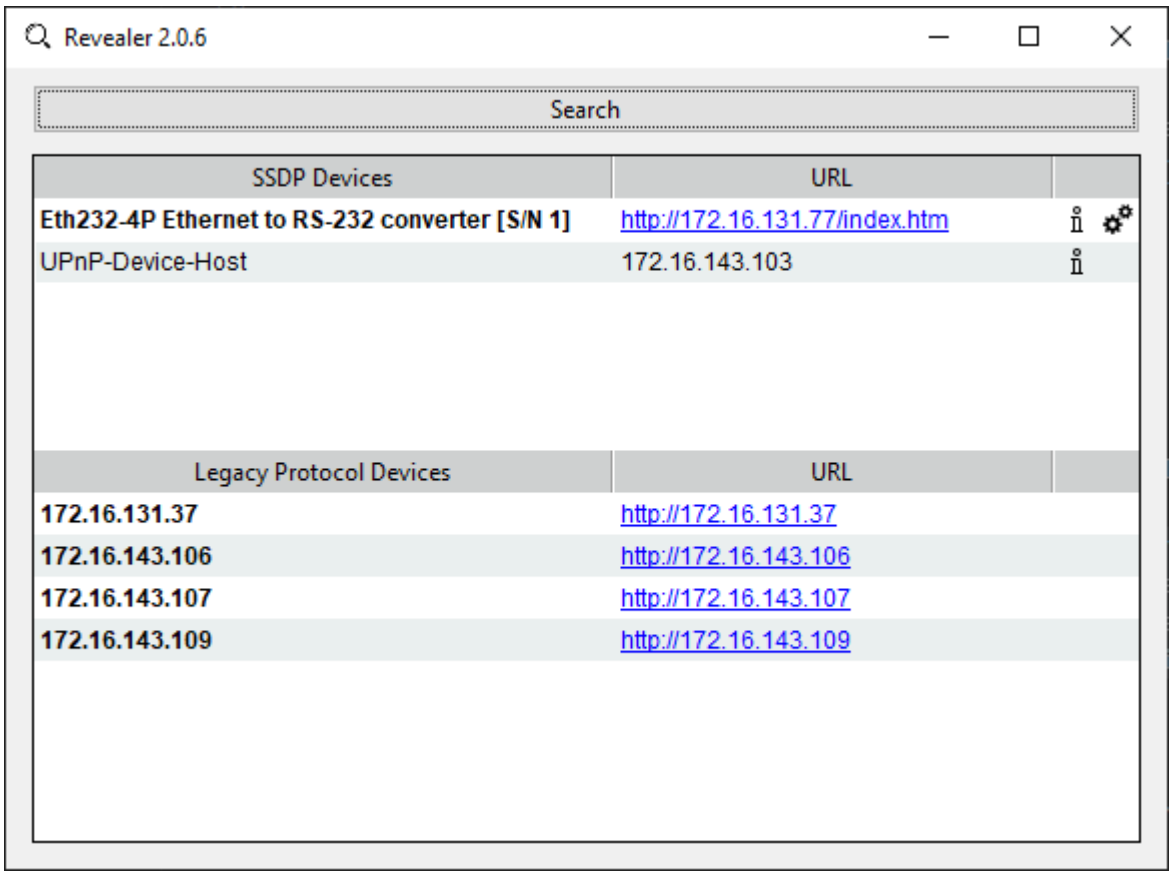


Рис. 7.12: Пользовательский интерфейс утилиты «Revealer»

Графический пользовательский интерфейс утилиты «Revealer» очень прост в использовании. Нажатие на кнопку *Search* запустит сканирование локальной сети, которое занимает приблизительно 3 секунды. В результате сканирования на панель *Legacy Protocol Devices* будет выведен список всех найденных устройств **Standa 8Eth1**. Нажатие на ссылку в этом списке приведёт к открытию нового окна (или вкладки) вашего стандартного системного браузера со страницей веб-интерфейса Администрирования соответствующего устройства.

Предупреждение: «Revealer» использует широковещательные UDP и SSDP запросы. При нажатии на кнопку *Search* используются оба протокола. Например: Для поиска **Standa 8Eth1** адаптера в вашей локальной сети используются широковещательные UDP запрос . Для поиска других устройств, например, принтеры, МФУ, сетевые диски, видеокамеры и т.п. используются SSDP запрос. Подробнее о SSDP можно прочитать здесь. **Важно отметить**, использование «Revealer» может быть нежелательным/невозможным в сетях, где в том или ином виде запрещены широковещательные UDP/SSDP запросы.

7.1.2.3 Обзор

Standa 8Eth1 и Multi-Axis Motion Controller 8SMC4-ETHERNET/RS232-B19 имеют предустановленный веб-интерфейс администрирования, который позволяет в интерактивном режиме управлять функционалом устройств, а также получать актуальную информацию об их состоянии.

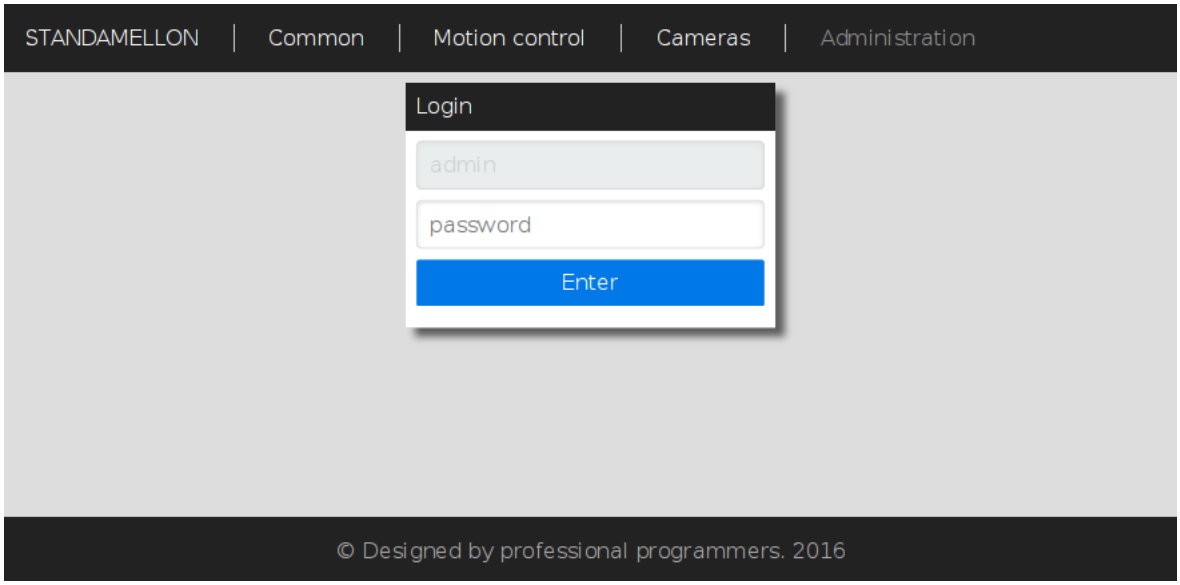


Рис. 7.13: Страница входа в интерфейс Администрирования

Для доступа к веб-интерфейсу устройства необходимо открыть URL *http://[address]*, где *[address]* является IP-адресом устройства в вашей локальной сети (узнать который можно в т.ч. используя программу «Revealer»). В случае если вы входите в веб-интерфейс первый раз (а также если вы не включали опцию запоминание паролей в браузере или выключили cookies), то вам будет необходимо пройти процедуру аутентификации.

Подсказка: Используйте «admin» в качестве логина и пароля

Интерфейс администрирования разделён на три функциональные секции.

7.1.2.3.1 Секция «Common»

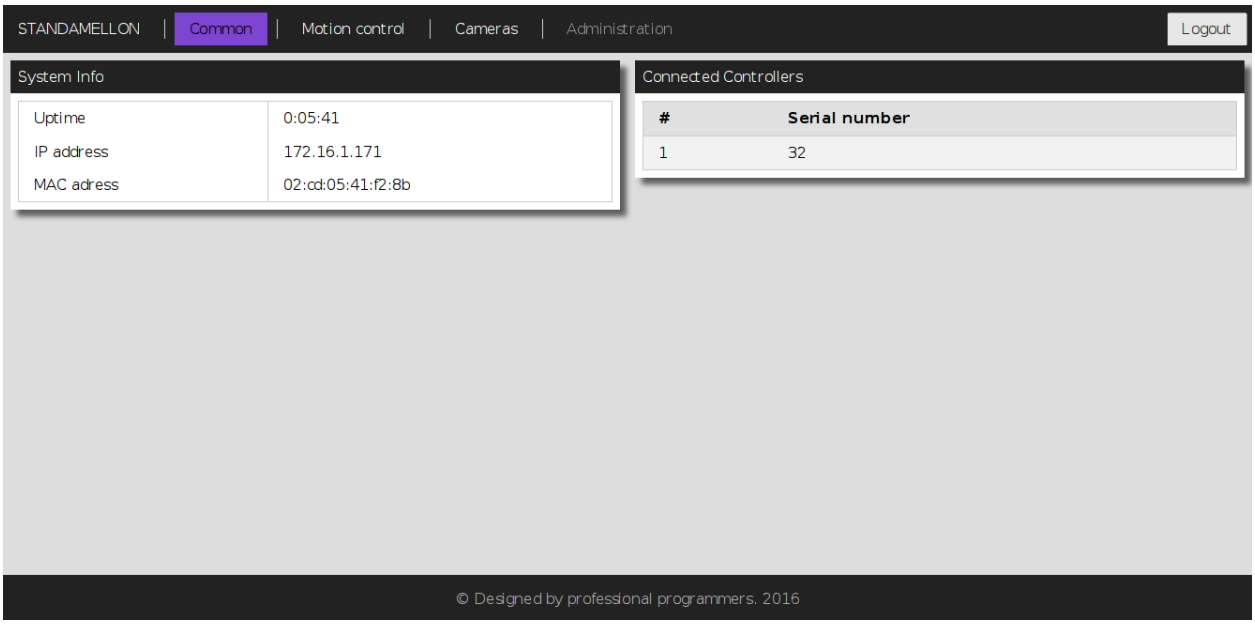


Рис. 7.14: Страница секции «Common»

Здесь содержится общая информация о системе и список серийных номеров подключенных к устройству контроллеров.

7.1.2.3.2 Секция «Motion Control»

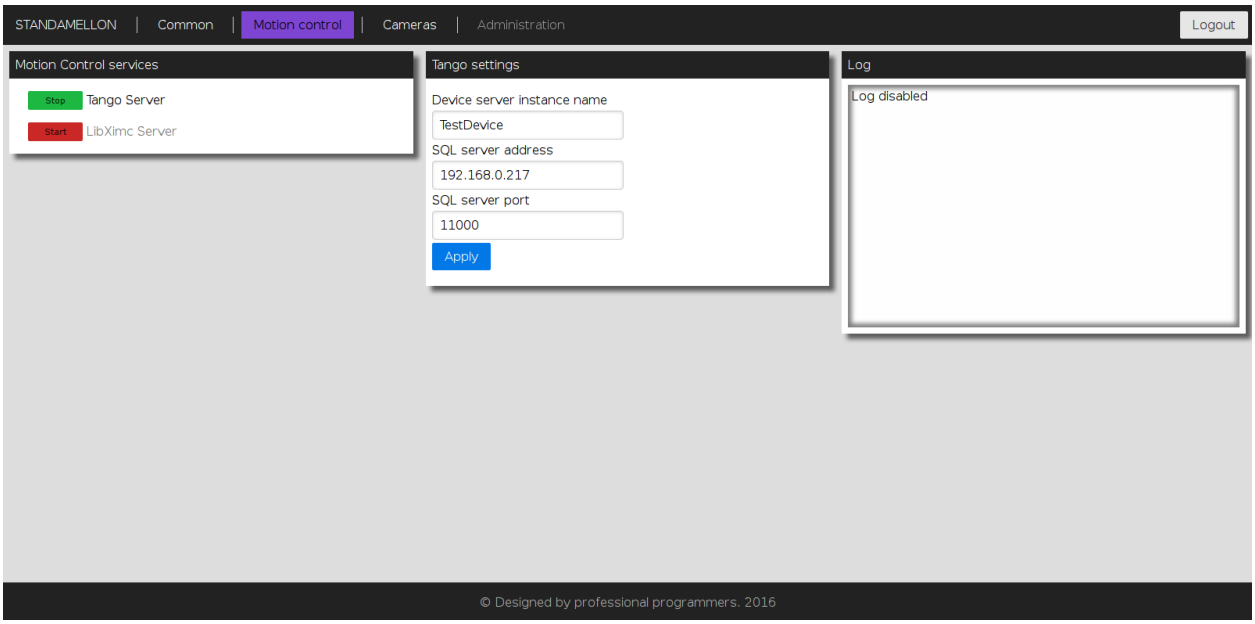


Рис. 7.15: Страница секции «Motion control»

Эта секция целиком посвящена поддерживаемым устройством сервисов управления движением. Панель «Motion Control services» содержит список всех доступных в данный момент сервисов управления движением. Нажатие на название какого-либо сервиса откроет соответствующую панель конфигурации (если сервис имеет какие-либо изменяемы настройки). Нажатие на кнопку «Apply» в панели настроек приведёт к сохранению настроек и полному перезапуску сервиса.

Примечание: В любой момент времени может быть активен только один из сервисов группы управления движением контроллеров. Запуск какого-либо сервиса приведёт к автоматической остановке предыдущего активного (при наличии такового).

7.1.2.3.3 Секция «Cameras»

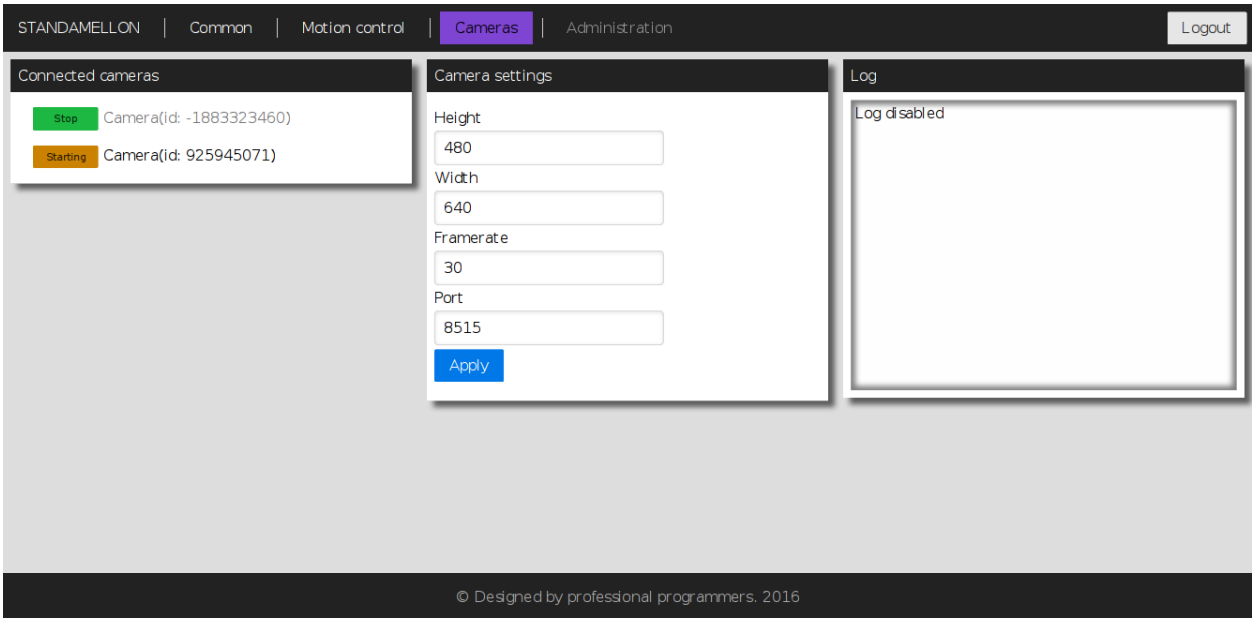


Рис. 7.16: Страница секции «Cameras»

Включает в себя список подключенных к устройству сетевых камер, настройки серверов трансляции, индикаторы и переключатели состояния для каждой из них.

7.1.2.4 Управление сервисами

Интерфейс Администрирования предназначен для того, чтобы конечный пользователь мог свободно конфигурировать и изменять состояние сервисов, доступных на устройстве. Для управления всеми сервисами - вне зависимости от их типа, функционала и предназначения - существует единый набор метафор. С каждым доступным для управления сервисом всегда ассоциирована соответствующая кнопка-индикатор, которая служит одновременно для отображения статуса сервиса и управления им:

- Красный цвет индикатора и надписать «Stopped» означают, что сервис остановлен; нажатие на кнопку начнёт запуск процесса.
- Оранжевый цвет индикатора и надписать «Starting» означают, что сервис в процессе запуска; нажатие на кнопку-индикатор остановит запуск процесса.
- Зелёный цвет кнопки-индикатора и надпись «Stop» означает, что сервис запущен; нажатие на кнопку начнёт остановку процесса.

- Оранжевый цвет кнопки-индикатора и надпись «Stopping» означает, что процесс останавливается; в этом состоянии кнопка недоступна для нажатия т.к. остановка процесса не может быть прервана.

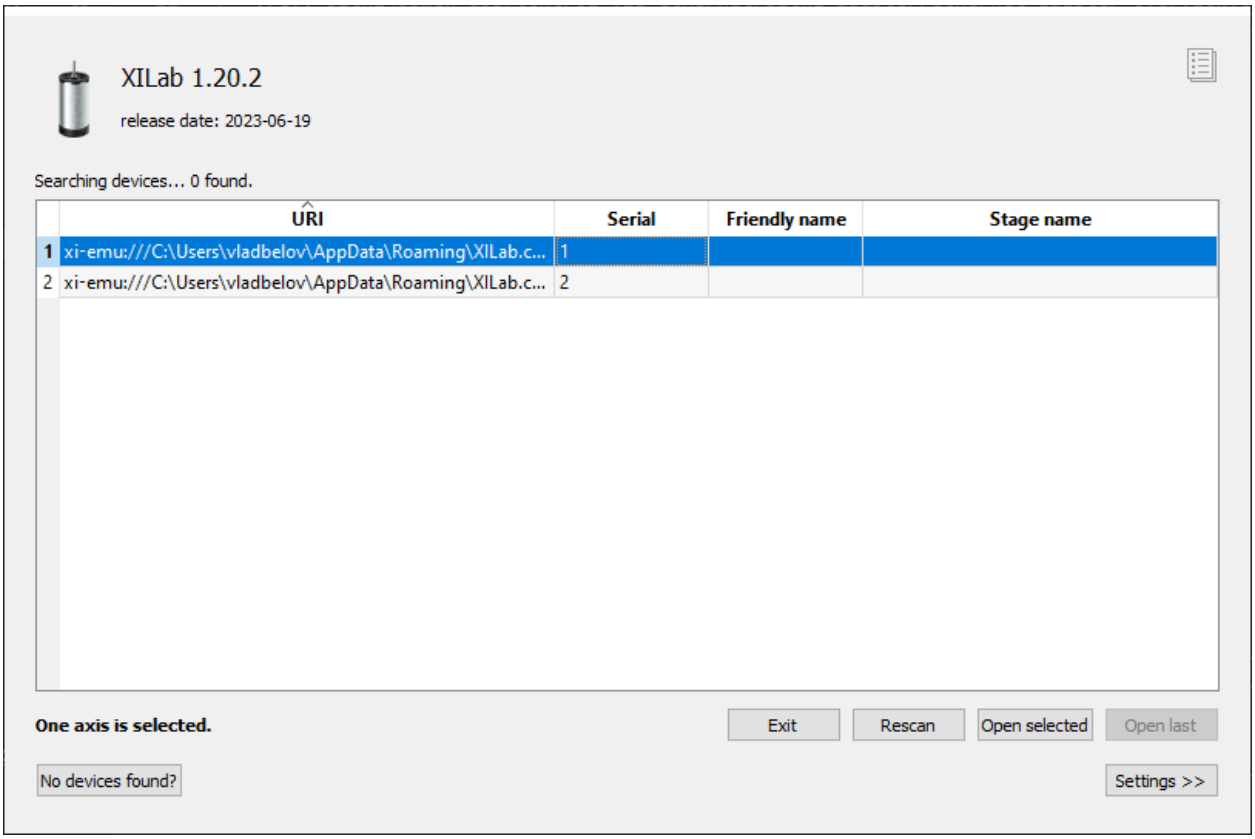
Примечание: Многие из сервисов имеют изменяемые настройки и в случае изменения какой-либо из них соответствующий процесс будет немедленно перезапущен (что, к примеру, может привести к потере связи с контроллером и потребовать переподключения в случае использования сервиса «LibXimc server»). Состояние сервиса можно отслеживать по цвету и надписи на кнопке-индикаторе.

7.1.2.5 Начало работы с помощью XILab

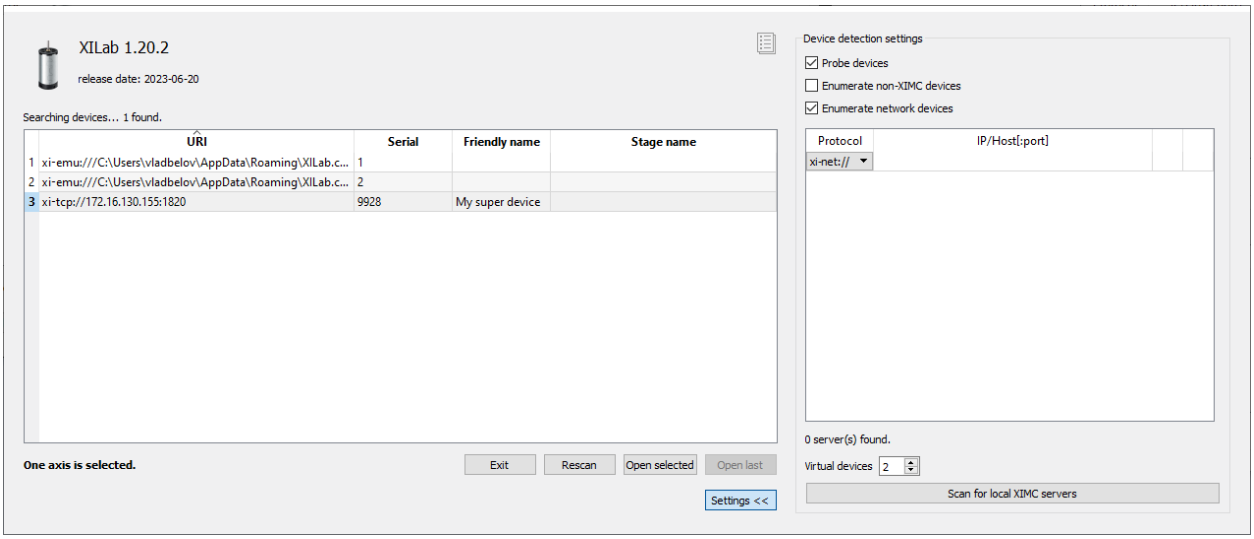
XILab способен работать с контроллерами через Ethernet. Для этого необходим либо адаптер *8Eth1*, либо исполняемый файл сервера libximc, который вы можете запустить на любом удобном для вас устройстве с подключенными к нему контроллерами.

- Если вы работаете с 8Eth1, подключите контроллеры к устройству с сервером XIMC. При этом предполагается, что двигатель может быть подключен к контроллерам и на него подано питание (подробнее см. *Краткое руководство и начало работы*).
- Подключите устройство с XIMC-сервером к той же подсети, что и управляющий компьютер, DHCP-сервер. В случае адаптера *8Eth1* подключите к нему адаптер питания (через разъем 5 В, 2 А) и подождите минуту, чтобы ОС Linux на одноплатном компьютере успела загрузиться.
- Включите XILab и проделайте следующую процедуру.

При первом запуске появится стартовое окно, в котором будут найдены два виртуальных устройства.



Нажмите *Settings*, установите флажок *Enumerate network devices* в правой вкладке и нажмите кнопку *Rescan* в левой вкладке, XILab найдет все оси, подключенные к системе. XILab будет использовать широковещательный запрос для поиска доступных серверов XIMC в вашей локальной сети. При желании вы также можете изменить список адресов вручную.



В окне обнаружения контроллера выберите нужную ось. Вы можете управлять им в *одноосном режиме* или в *многоосевом режиме*, если было выбрано более одной оси. Дополнительную информацию см. в *Руководстве по началу работы с программным обеспечением XILab* и *Руководство по программе XILab*.

Примечание: Когда IP-адрес устройства найден, следует понимать, что перемещение устройства в другое место может привести к изменению его IP-адреса.

Работа с несколькими адаптерами может привести к тому, что при широковещательном запросе первым будет отвечать один и тот же адаптер с подключёнными к нему контроллерами. Существует два способа решения этой проблемы:

- Отключить остальные 8Eth1 адаптеры, найти устройство в сети, подключить всё обратно.
- Многократно нажимать *Rescan* до тех пор, пока не будет найден нужный адаптер.

7.1.2.6 Обновление прошивки

Предупреждение: В ходе обновления прошивки содержимое вашей MicroSD карты и flash-памяти устройства 8Eth1 будет целиком стёрто. Убедитесь, что вы сделали резервные копии всех ваших данных.

1. Загрузите необходимую версию прошивки со страницы программного обеспечения.
2. Вам понадобится MicroSD на 4 Гбайт (или больше).
3. Загрузите и разархивируйте образ автоинсталлятора (в случае если вы используете Windows и у вас нет подходящего архиватора, вы можете воспользоваться 7-Zip (официальный сайт).
4. Запишите образ автоинсталлятора на MicroSD карту:

- Для **Unix-подобных ОС**:
  (a) Вы можете использовать стандартную системную утилиту dd (например:

```
dd if=autoinstall_image_2015-12-17.bin of=/dev/sdX bs=4M; sync
```

  но не забудьте изменить пути к файлам на ваши собственные).
- **ОС семейства Windows**:
  (a) Загрузите Win32 Disk Imager (официальный сайт) и запустите его.
  (b) Укажите путь к файлу автоинсталлятора в поле «Image File».
  (c) Выберите диск, который соответствует вашей MicroSD карте в поле «Device».
  (d) Нажмите кнопку «Write» и дождитесь окончания записи.

5. Вставьте MicroSD карту с записанным на неё автоинсталлятором в **8Eth1** и включите его.
6. Установка занимает приблизительно 10 минут. Вы можете узнать статус установки по LED-индикатору, который находится на плате. Его можно увидеть через отверстия для USB-разъемов в корпусе устройства:
   (a) После включения устройства LED горит или периодически загорается одинарными вспышками — автоинсталлятор запускается.
   (b) LED периодически загорается двойными зелёными вспышками — идёт установка.
   (c) LED горит зелёным — установка завершена.
7. Выключите устройство (вытащите кабель питания) и извлеките MicroSD карту.
8. Включите устройство и подождите 2-7 минут пока завершится инициализация после перепрошивки.

Предупреждение: Если после перепрошивки *revealer* не способен обнаружить устройство даже после указанного выше периода инициализации(и вы не можете подключиться к веб-интерфейсу администрирования и/или использовать другие сервисы), то попробуйте перезагрузить устройство. Если это не помогло, то повторите шаги 5-8.

После вышеописанных шагов ваш **8Eth1** перепрошит и может свободно использоваться в стандартном режиме.

7.2 Serial to Ethernet конвертер

7.2.1 Общая информация

Иногда возникает необходимость управления устройством через последовательный порт, а разместить компьютер рядом с устройством не всегда удобно, а иногда просто невозможно. В таких случаях на помощь приходят различные Serial to Ethernet конвертеры. Serial to Ethernet конвертеры устроены так, что устройства, имеющие последовательный тип подключения, при подключении к ним преобразуются в сетевые и открываются для доступа из любой точки сети. Кроме того, появляется функция многопользовательского доступа к одному устройству.

Вы можете использовать любой Serial to Ethernet конвертор, от самого дешевого до самого дорогого. Например, USR-TCP232-410s, USR-TCP232-302, ER108-R4U2, ER108-L4U2, NCOM-111-M, NCOM-211-M, NPort 5130, NPort 5110 и т.д. В этой главе в качестве Serial to Ethernet конвертора будет использоваться MOXA Nport 5110.



Рис. 7.17: Внешний вид MOXA конвертера

7.2.2 Подключение конвертера к контроллеру 8SMC5

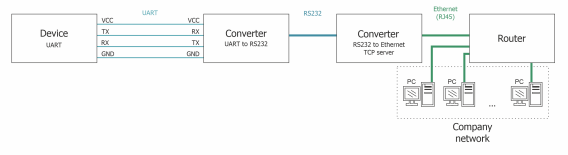


Рис. 7.18: Пример конфигурации

Чтобы создать схему, подобную приведенной выше, понадобится контроллер 8SMC5-USB (без корпуса), UART to RS232 конвертор (Waveshare RS232) и RS232 to Ethernet конвертор с TCP-сервером (MOXA Nport 5110). Полная реальная схема будет выглядеть так:

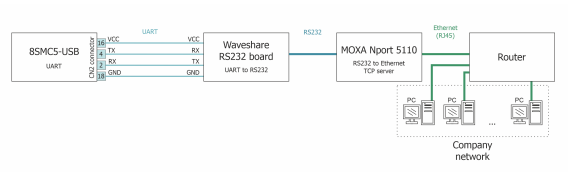


Рис. 7.19: Схема реальной конфигурации

- Подключите плату Waveshare RS232 к контроллеру 8SMC5-USB по следующей схеме:

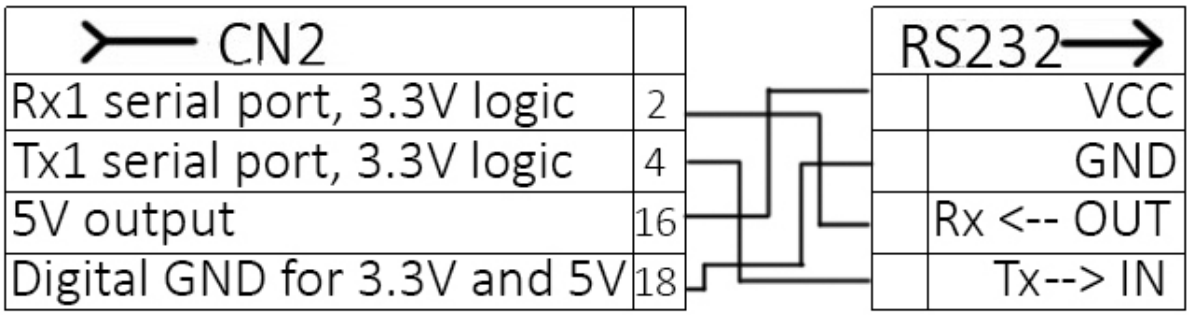


Рис. 7.20: Схема подключения Waveshare RS232 к контроллеру 8SMC5

Если вы используете контроллер *8SMC5-USB-B8-1* или *8SMC5-USB-B9-2*, вам все также понадобятся контакты RX, TX, 5V и GND, но брать их надо с разъема HDB-26.

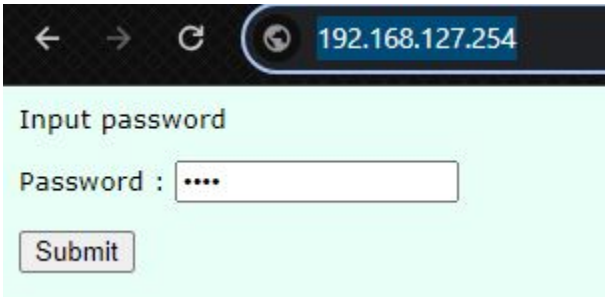
Важно: Для схемы из этой инструкции линии *Rx* и *Tx* должны быть подключены **как показано на схеме выше**. *Rx* -> *Rx*, *Tx* -> *Tx*. Если вы используете другой UART to RS232 конвертор, линии *Rx* и *Tx* могут быть подключены иначе, например *Rx* ->*Tx*, *Tx* -> *Rx*.

- Подключите плату Waveshare RS232 к преобразователю MOXA Nport 5110 с помощью разъема DB9
- Подключите Ethernet кабель и блок питания, который входит в комплект поставки MOXA Nport 5110
- Подключите блок питания и подвижку к контроллеру 8SMC5-USB
- Включите контроллер 8SMC5-USB и MOXA Nport 5110 конвертор

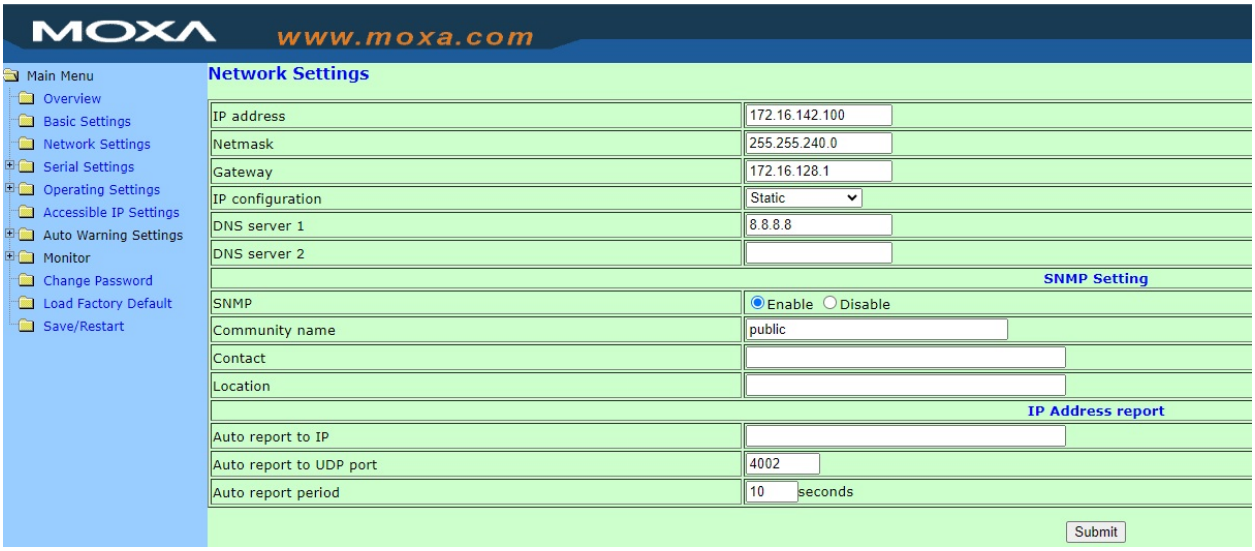
7.2.3 Настройка MOXA NPort 5110 конвертера через web интерфейс

- В адресной строке браузера введите *192.168.127.254* (IP-адрес устройства MOXA NPort 5110 по умолчанию). Введите пароль для доступа к устройству.

Подсказка: Пароль по умолчанию - «moxa»



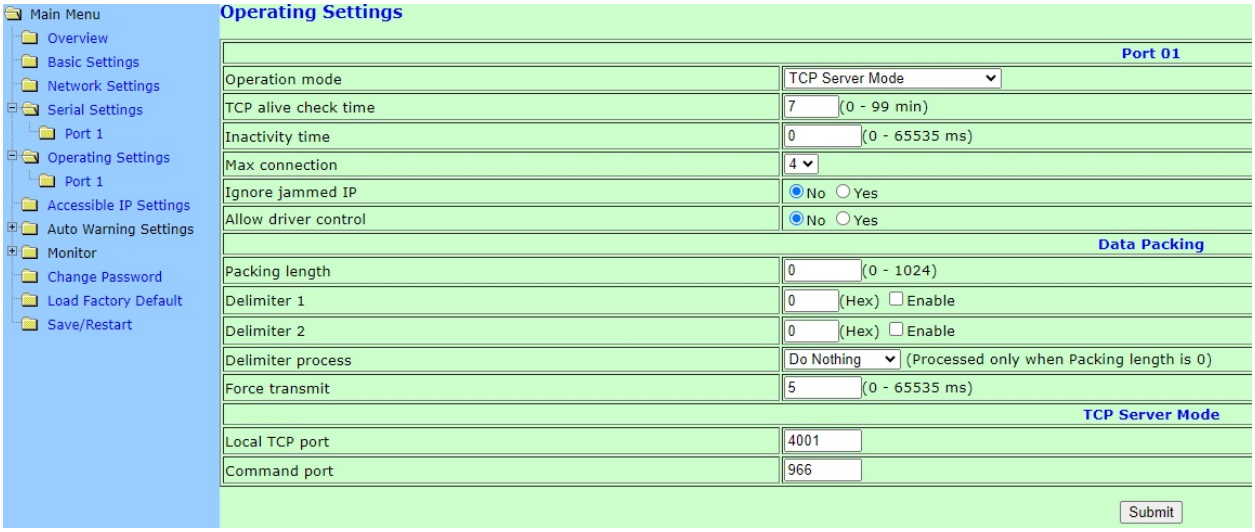
- В разделе *Network Settings* укажите IP-адрес, маску и шлюз в соответствии с параметрами вашей локальной сети.



После внесения любых изменений в настройки конвертера необходимо **сохранить и перезагрузить устройство**!

Важно: Пожалуйста, обратите внимание, что теперь вам нужно будет ввести новый IP-адрес устройства MOXA NPort 5110 в веб-браузере.

- В разделе *Operating Settings* выберите режим работы «TCP Server Mode».



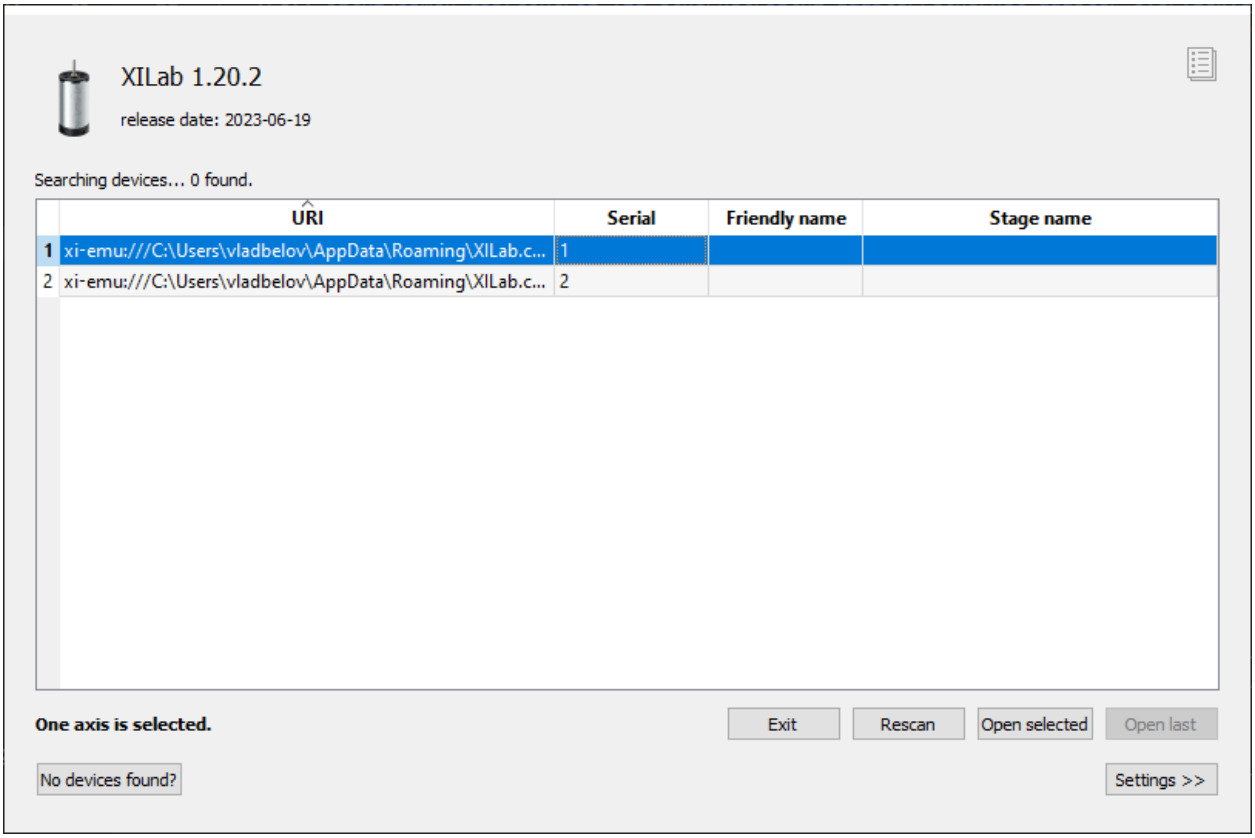
Не забудьте указать *Local TCP port*, через который будет устанавливаться соединение.



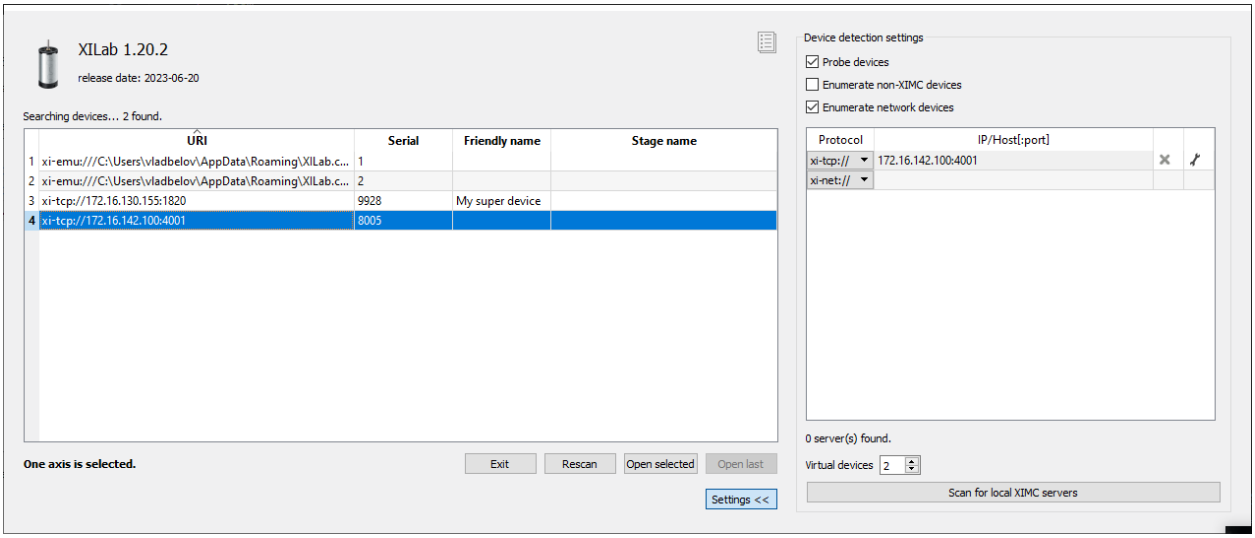
7.2.4 Начало работы с помощью XILab

Запустите XILab и сделайте следующее. *Предполагается, что вышеуказанные инструкции по подключению и настройке были соблюдены.*

При первом запуске XILab появится стартовое окно, в котором будут найдены два виртуальных устройства.



Нажмите *Settings* и установите флаг *Enumerate network devices* на правой вкладке. Выберете TCP протокол, введите IP адрес MOXA конвертера и **не забудьте указать порт**. Затем нажмите кнопку *Rescan* в левой части стартового окна, XILab найдет контроллер подключенный к конвертеру.



В окне обнаружения контроллера выберите нужную ось. Вы можете управлять им в *одноосном режиме* или в *многоосевом режиме*, если было выбрано более одной оси. Дополнительную информацию см. в *Руководстве по началу работы с программным обеспечением XILab* и *Руководство по программе XILab*.

Примечание: Когда IP-адрес устройства найден, следует понимать, что перемещение устройства в другое место может привести к изменению его IP-адреса.

7.3 8SMC5 Ethernet

7.3.1 Общая информация

Иногда возникает необходимость в удаленном управлении устройством, когда разместить компьютер рядом с устройством не всегда удобно или иногда просто невозможно. **Контроллер 8SMC5 с поддержкой Ethernet** - это устройство, которое позволяет конечному пользователю удаленно взаимодействовать с контроллером двигателя через сеть Ethernet. **Кроме того, появляется функция многопользовательского доступа к одному устройству.**

7.3.2 Примеры конфигурации для контроллера 8SMC5 с поддержкой Ethernet

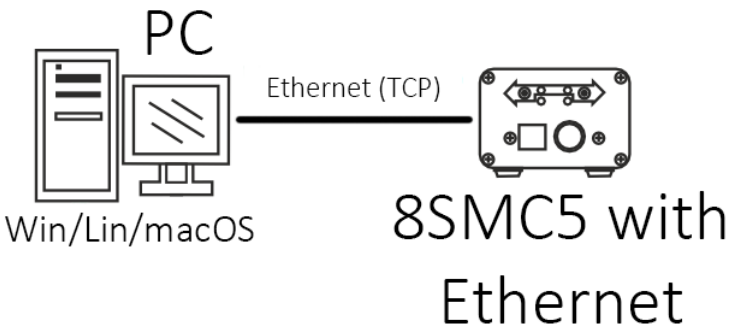


Рис. 7.21: На схеме показано, что, подключив контроллер к компьютеру (**который не имеет выхода в интернет**) с помощью кабеля Ethernet, вы можете найти его в сети и работать с ним

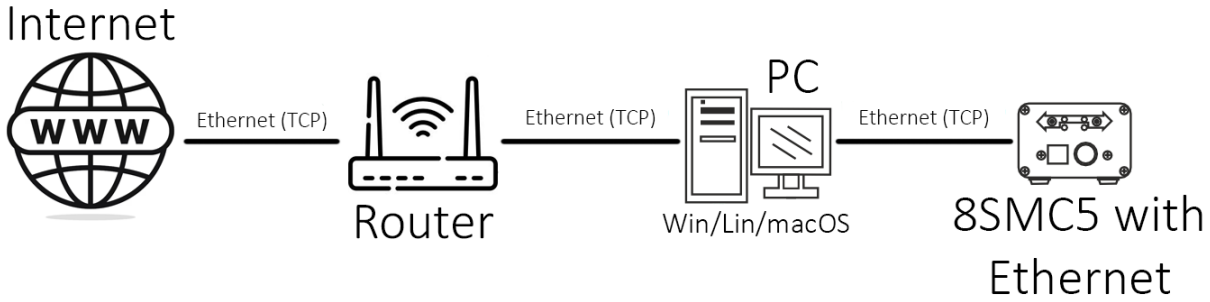


Рис. 7.22: На схеме показано, что, подключив контроллер к компьютеру с помощью кабеля Ethernet, вы можете найти его в сети и работать с ним

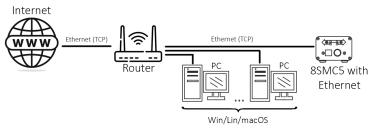


Рис. 7.23: На схеме показано, что, подключив контроллер к локальной сети с помощью кабеля Ethernet, вы можете найти его в сети и работать с ним. **Все компьютеры в сети могут иметь доступ к этому контроллеру**

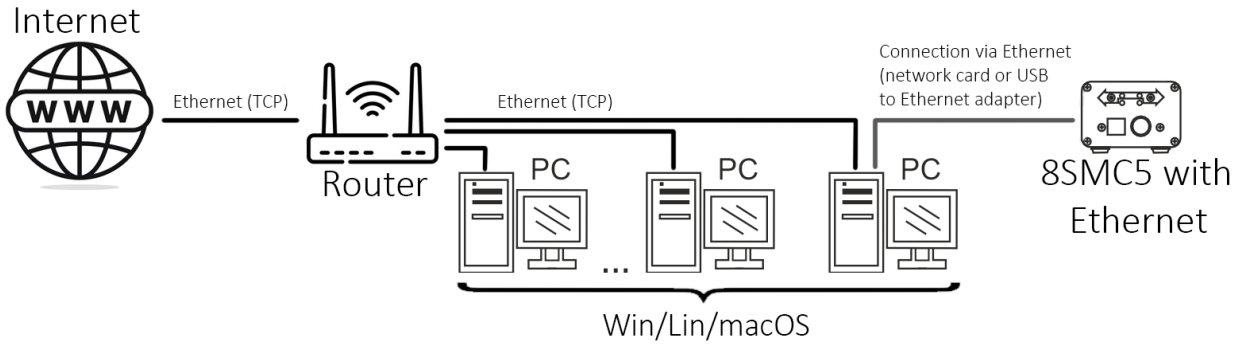


Рис. 7.24: На схеме показано, что, подключив контроллер к компьютеру с помощью кабеля Ethernet или USB - Ethernet адаптера, вы можете найти его в сети и работать с ним. **Все компьютеры в сети могут иметь доступ к этому контроллеру**

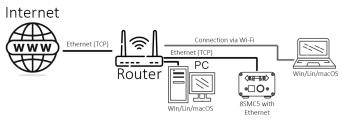


Рис. 7.25: На схеме показано, что, подключив контроллер к локальной сети с помощью кабеля Ethernet, вы можете найти его в сети и работать с ним. **Все компьютеры в сети могут иметь доступ к этому контроллеру**

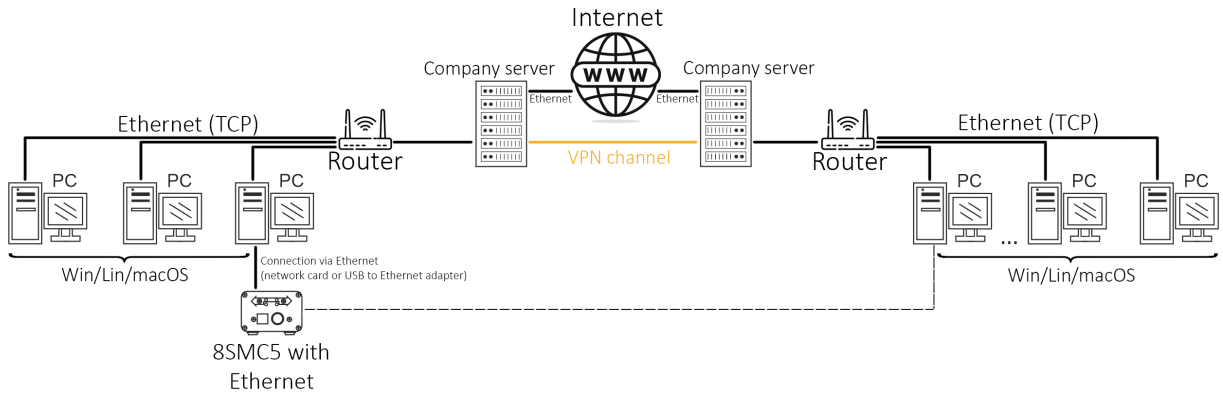


Рис. 7.26: На схеме показано, что, подключив контроллер к компьютеру с помощью кабеля Ethernet или с помощью USB - Ethernet адаптера, вы можете найти его в сети и работать с ним. При наличии VPN-канала, например, если вы находитесь в другом офисе, контроллер также будет вам доступен. **Все компьютеры в сети могут иметь доступ к этому контроллеру, включая ноутбуки, подключенные к этой сети через Wi-Fi**

7.3.3 Автоматическое обнаружение устройства

Для удобства мы предоставляем небольшую специальную утилиту под названием «Revealer», которая поможет вам мгновенно идентифицировать все контроллеры 8SMC5 с поддержкой Ethernet, подключенные к вашей локальной сети. Вы можете скачать его на нашей странице программного обеспечения.

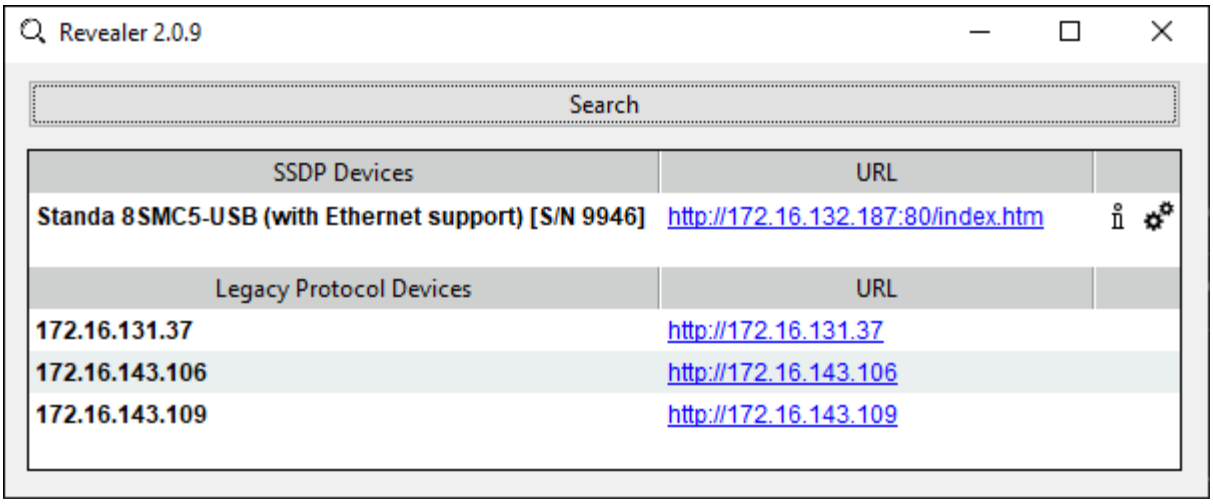


Рис. 7.27: Пользовательский интерфейс утилиты «Revealer»

Графический пользовательский интерфейс утилиты «Revealer» очень прост в использовании. Нажатие на кнопку *Search* запустит сканирование локальной сети. После этого все контроллеры 8SMC5 с поддержкой Ethernet, найденные в вашей локальной сети, будут перечислены на панели *SSDP Devices* в виде кликабельных ссылок. При нажатии на ссылку открывается ваш системный браузер по умолчанию и перенаправляется на веб-страницу интерфейса администрирования.

Предупреждение: «Revealer» использует широковещательные UDP и SSDP запросы. При нажатии на кнопку *Search* используются оба протокола. Например: Для поиска контроллеров 8SMC5 с поддержкой Ethernet, принтеров, МФУ, сетевых дисков, видеокамер и т.п. используются SSDP запросы. Подробнее о SSDP можно прочитать здесь. **Важно отметить**, использование «Revealer» может быть нежелательным/невозможным в сетях, где в том или ином виде запрещены широковещательные UDP/SSDP запросы.

7.3.4 Панель администрирования

Контроллер 8SMC5 с поддержкой Ethernet оснащен веб-интерфейсом администрирования, который позволяет конечному пользователю управлять службами устройства и отслеживать состояние системы.

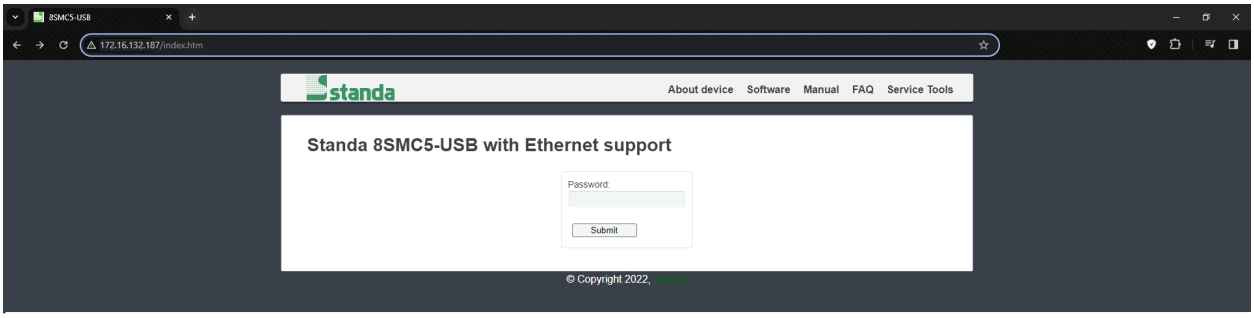


Рис. 7.28: Страница входа в интерфейс администрирования

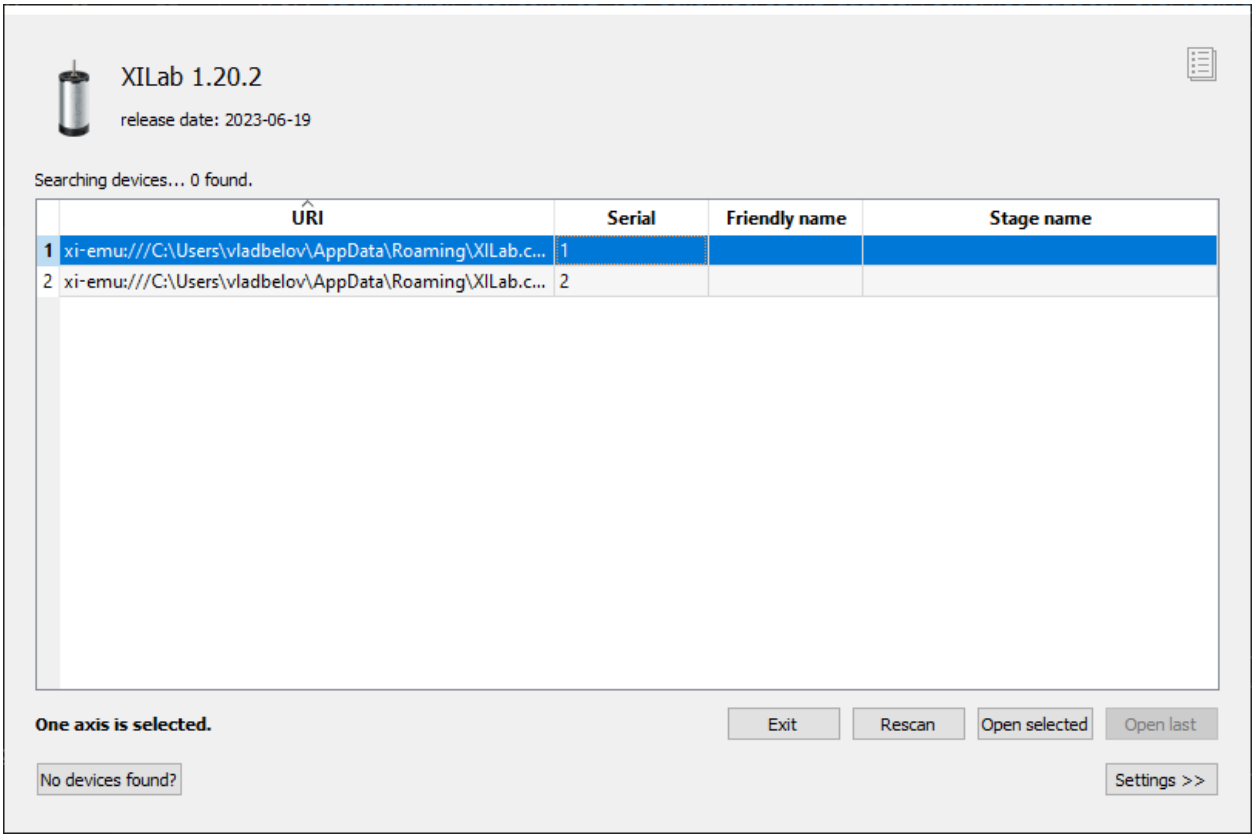
Для доступа к веб-интерфейсу устройства необходимо открыть URL *http://[address]*, где *[address]* является IP-адресом устройства в вашей локальной сети (узнать который можно в т.ч. используя программу

«Revealer»). Если вы делаете это впервые (или отключили сохранение файлов cookie/паролей в своем браузере), вам необходимо пройти аутентификацию.

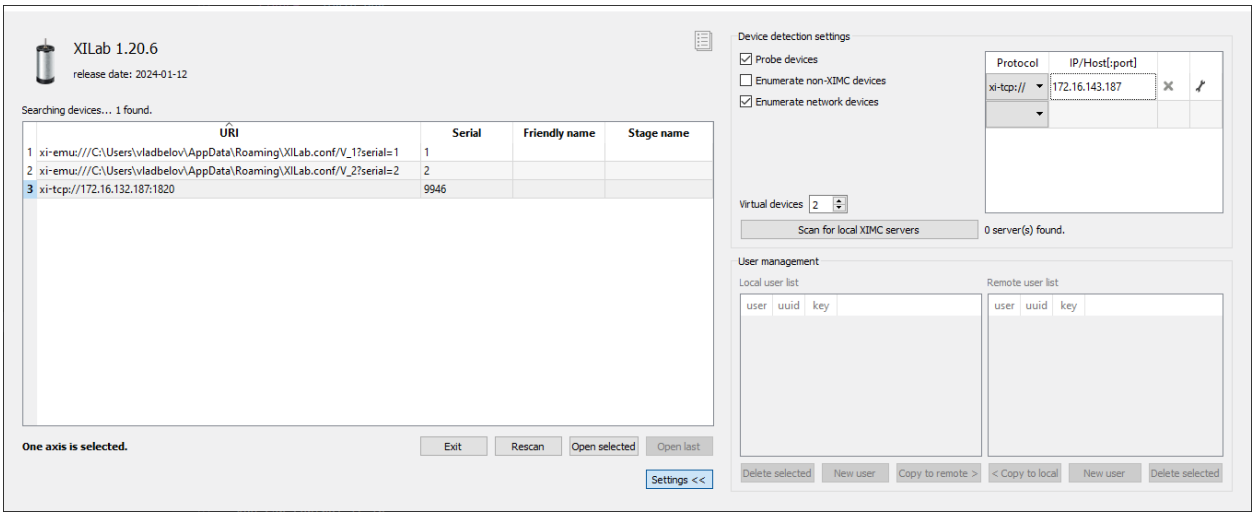
Подсказка: Используйте «0000» в качестве пароля по умолчанию

7.3.5 Начало работы с XILab

При первом запуске появится стартовое окно, в котором будут найдены два виртуальных устройства.



Нажмите *Settings*, установите флаг *Enumerate network devices* в правой вкладке. Выберите протокол TCP, введите IP-адрес контроллера 8SMC5 с поддержкой Ethernet. Затем нажмите кнопку *Rescan* в левой вкладке, XILab обнаружит контроллер, подключенный через Ethernet.



В окне обнаружения контроллера выберите нужную ось. Вы можете управлять им в *одноосном режиме* или в *многоосевом режиме*, если было выбрано более одной оси. Дополнительную информацию см. в *Руководстве по началу работы с программным обеспечением XILab* и *Руководство по программе XILab*.

Глава 8

FAQ

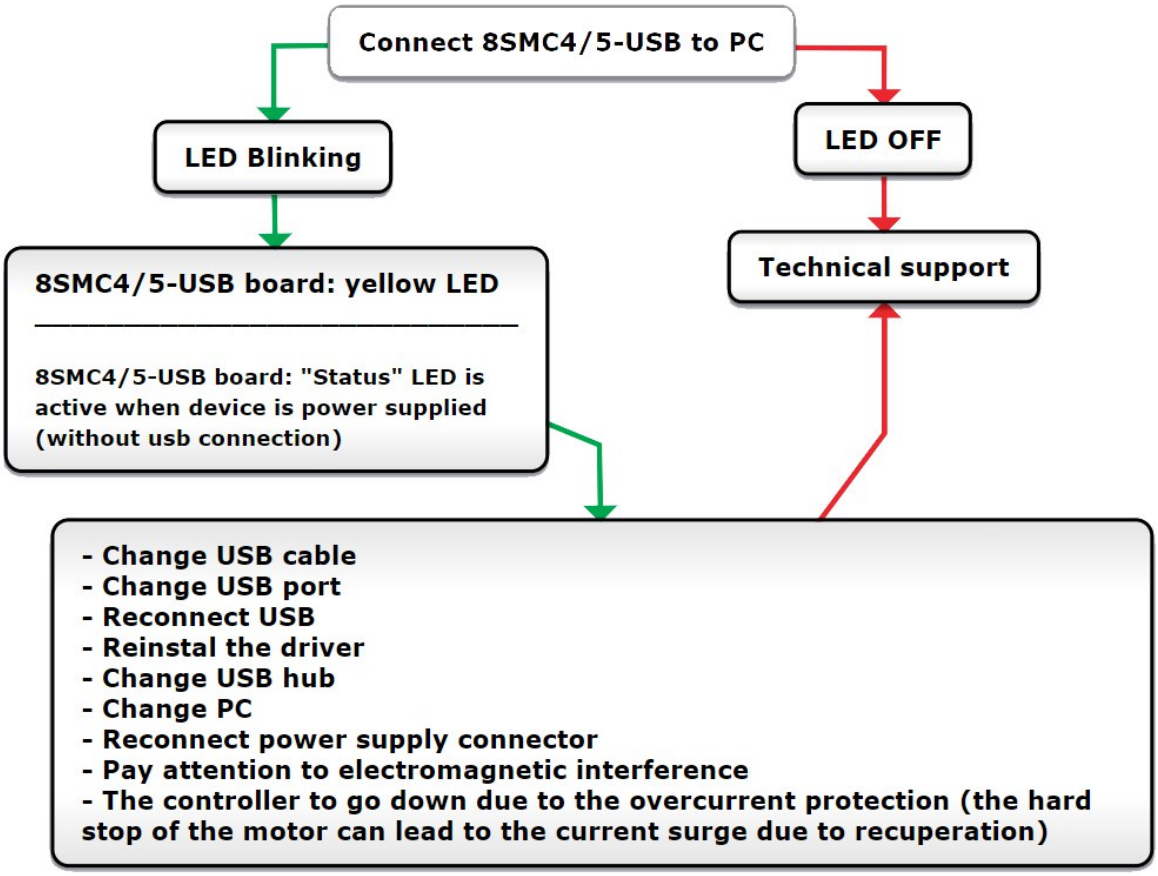
8.1 No device found / Can't open device

- *Подключение через USB*
- *Подключение через Ethernet*
- *Подключение через 8Eth1 адаптер*
- *Подключение через Serial to Ethernet конвертер*

8.1.1 Подключение через USB

XiLab или другое программное обеспечение не видит контроллер.

- Компьютер не обнаруживает контроллер по USB:



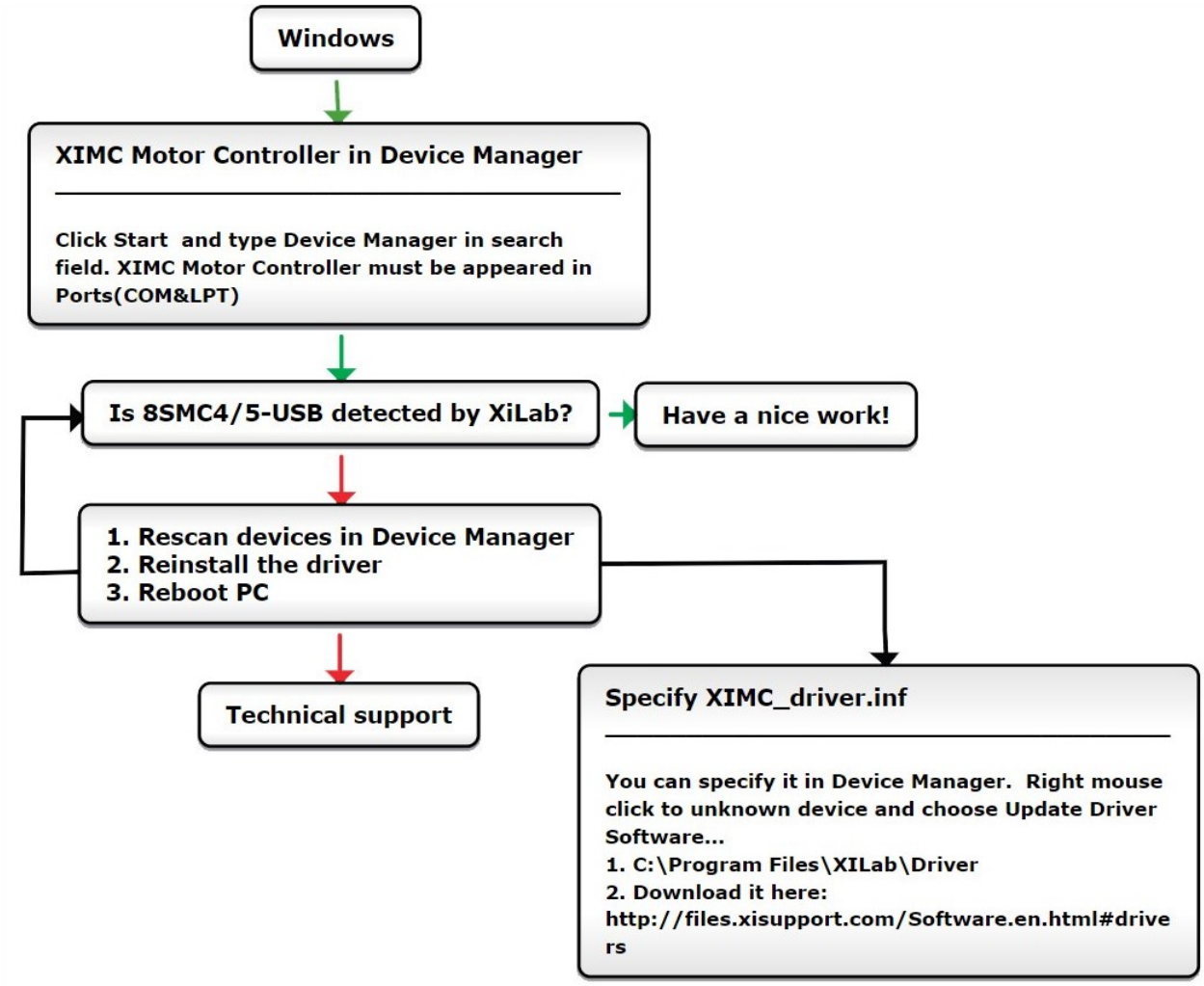
Комментарий к схеме:

Наиболее частая причина подобного рода ошибок - это проблемы в работе USB-хаба, кабеля или проблема определения виртуального COM-порта в операционной системе на используемом ПК. Попробуйте воспроизвести данную ошибку на другом компьютере или с другим USB-хабом, если он используется.

Предупреждение: Ошибка «Can't open device» или функция «open_device()» возвращает -1. Библиотека Libximc работает с контроллером в режиме эксклюзивного доступа. Каждый контроллер, открытый библиотекой libximc (XiLab тоже использует эту библиотеку) должен быть закрыт, прежде чем может быть использован другим процессом. Поэтому прежде чем попытаться открыть контроллер заново, проверьте, что XiLab или другое программное обеспечение, взаимодействующее с контроллером, закрыто.

Ниже приведены карты действий для не найденного контроллера.

Windows:



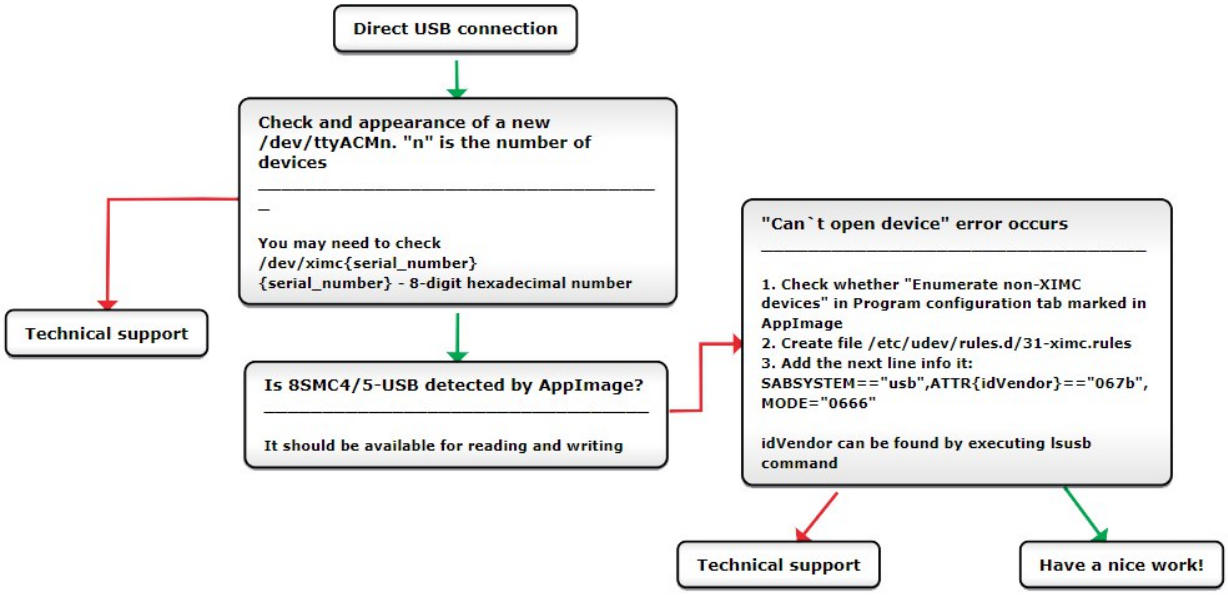
Комментарии к схеме:

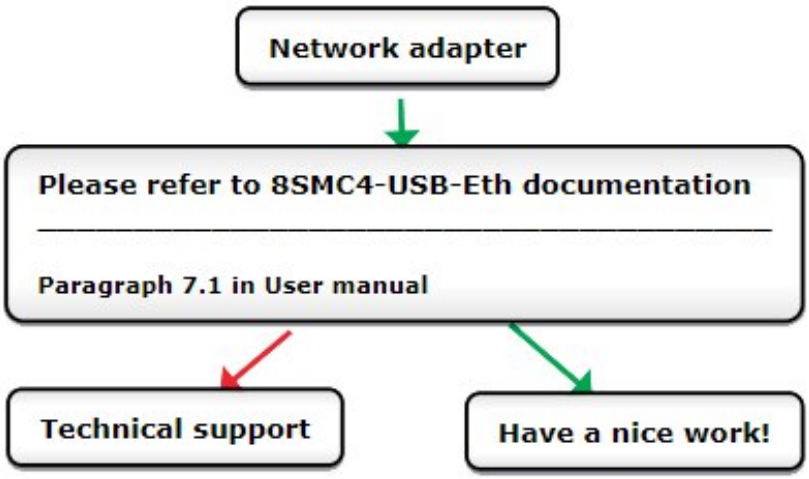
- Проверьте, что COM-порт, соответствующий вашему контроллеру, присутствует в Диспетчере устройств. Контроллер должен отображаться как «XIMC Motor Controller (COMn)». Если контроллер не распознан, попробуйте переустановить драйвер контроллера вручную.
- Попробуйте открыть COM-порт контроллера в любом простом последовательном эмуляторе (напрмер, Putty) и отправьте контроллеру одну из простых команд («stop», «sstp», «zero», *«GETS»*, *«GETI»*). Параметры подключения описаны *здесь*. Отсутствие ошибок означает, что контроллер работает правильно, и проблема вызвана используемым программным обеспечением.

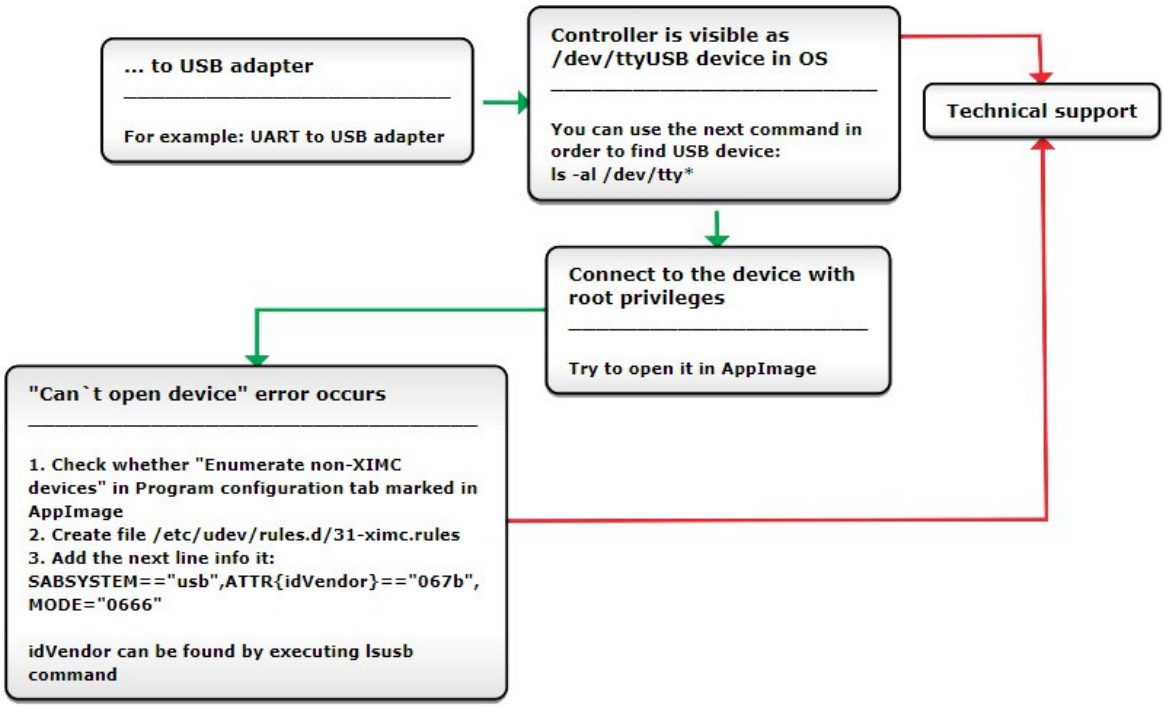
https://youtu.be/lzwDACaJHvQ

Видео-руководство по переустановке драйвера

Linux:


Direct USB connection
Check and appearance of a new /dev/ttyACMn. "n" is the number of devices
You may need to check /dev/ximc{serial_number} {serial_number} - 8-digit hexadecimal number
Technical support
Is 8SMC4/5-USB detected by AppImage?
It should be available for reading and writing
"Can`t open device" error occurs
1. Check whether "Enumerate non-XIMC devices" in Program configuration tab marked in AppImage
2. Create file /etc/udev/rules.d/31-ximc.rules
3. Add the next line info it: SABSYSTEM=="usb",ATTR{idVendor}=="067b", MODE="0666"
idVendor can be found by executing lsusb command
Technical support
Have a nice work!


Network adapter
Please refer to 8SMC4-USB-Eth documentation
Paragraph 7.1 in User manual
Technical support
Have a nice work!



Комментарий к решению проблемы «Can't open device» (2 ветка):

При работе с преобразователем USB-UART (а также USB-Ethernet, USB-Bluetooth и т. д.) в **Linux** он отображается как устройство **/dev/ttyUSB**. XiLab отображает его в списке, но при попытке открыть возникает ошибка «can't open device» из-за отсутствия соответствующих прав доступа к устройству.

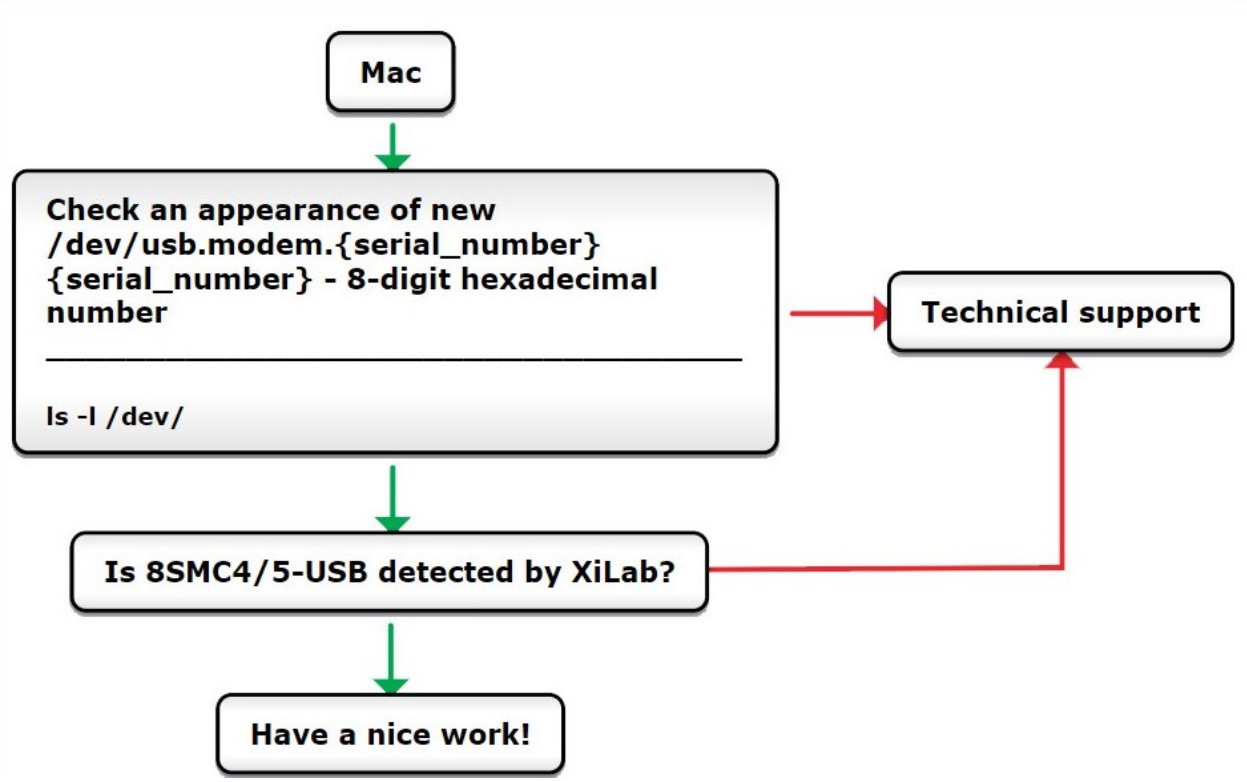
Для решения данной проблемы создайте файл: /etc/udev/rules.d/31-ximc.rules и добавьте в него следующую строку:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="067b", MODE="0666"
```

Идентификатор idVendor можно найти с помощью команды *lsusb*.

Примечание: Одним из возможных вариантов решения проблемы «**device not found**» является добавление пользователя в группу dialuot. **Важно**: после добавления в группу необходимо перезагрузить компьютер.

Mac OS:



8.1.2 Подключение через Ethernet

Примечание: При выполнении описанных ниже действий предполагается, что 8SMC5-ETH контроллер включен и работает

- **Виден ли ваш контроллер в программе «Revealer» ?**

  Если да:

  – Чтобы получить доступ к панели администрирования, нажмите на IP-адрес. Если панель управления открывается, значит ваш контроллер работает нормально!

  Подсказка: Используйте «0000» в качестве пароля по умолчанию

  Если нет:

  – **Подключите контроллер к компьютеру по USB:**

    * Используя Xilab, убедитесь, что установлена версия прошивки 7.0.11 или выше. Вы можете просмотреть версию прошивки во вкладке «About device» в настройках XILab. При необходимости обновите прошивку.
    * Убедитесь что подключенный по USB контроллер работает правильно. Для этого загрузите профиль и выполните любое движение. *Для проверки мы рекомендуем использовать XILab.*

  – Отключите «Брандмауэр Защитник Windows».

  – **Подключите контроллер к компьютеру с помощью Ethernet кабеля:**

    * Проверьте, виден ли контроллер в изолированной сети с помощью revealer. При необходимости используйте revealer для изменения IP-адреса вашего контроллера. Чтобы изменить настройки в окне revealer, нажмите на шестеренки. Например, вы можете установить статический IP-адрес для вашего контроллера.

- **В вашей сети установлен DHCP сервер?**

  Если да:

  – Убедитесь, что контроллеру был присвоен IP адрес.

  – Убедитесь, что контроллер находится в той же подсети, что и ваш компьютер.

  Если нет:

  – Вы можете установить *DHCP сервер*.

  – Используйте revealer, чтобы установить статический IP-адрес для вашего контроллера. Чтобы изменить настройки в окне revealer, нажмите на шестеренки.

8.1.3 Подключение через 8Eth1 адаптер

Примечание: При выполнении описанных ниже действий предполагается, что контроллер и адаптер 8Eth1 включены и работают. Контроллер, подключенный через 8Eth1 адаптер, можно открыть по протоколу xinet.

- **Виден ли ваш контроллер в программе «Revealer» ?**

  Если да:

  – Чтобы получить доступ к панели администрирования, нажмите на IP-адрес. Если панель управления открывается и на «Common panel» отобразится серийный номер вашего контроллера, это означает, что адаптер 8Eth1 работает нормально!

  Подсказка: Используйте «admin» в качестве логина и пароля по умолчанию

  Если нет:

  – Перезагрузите контроллер и 8Eth1[1] адаптер

  – В разделе «Motion control» запустите/перезапустите «libximc server»

  – Отключите «Брандмауэр Защитник Windows»

  – Порт 49150 не должен быть заблокирован. Причиной блокировки зачастую может быть наличие антивирусного ПО или программы, отслеживающей и фильтрующей сетевой трафик (брандмауэры)

- **В вашей сети установлен DHCP сервер?**

  Если да:

  – Убедитесь, что 8Eth1 адаптеру был присвоен IP-адрес

  – Убедитесь, что 8Eth1 адаптер находится в той же подсети, что и ваш компьютер

  Если нет:

  – Вы можете установить *DHCP сервер*

[1] Артикул был изменен в 2020 г., ранее устройство имело артикул 8SMC4-USB-Eth.

– Вы можете *назначить статический IP-адрес* вашему 8Eth1 адаптеру

8.1.4 Подключение через Serial to Ethernet конвертер

Примечание: При выполнении описанных ниже действий предполагается, что контроллер включен и работает. Контроллер, подключенный через Serial to Ethernet конвертер, как правило, можно открыть по протоколу tcp/udp.

- Как правило, преобразователи имеют светодиодную индикацию. Она работает? Если нет, то скорее всего ваш конвертер не работает по техническим причинам
- Конвертер передает данные? Для проверки требуется замкнуть контакты Rx и Tx между собой. Переданные данные должны быть возвращены обратно
- Правильно ли подключен конвертер? Как правило, используется соединение Rx -> Tx, Tx -> Rx, но некоторые производители требуют подключения Rx -> Rx, Tx -> Tx
- Установлен ли в вашей сети DHCP-сервер? Убедитесь, что конвертеру присвоен IP-адрес
- Конвертер подключен напрямую к компьютеру? Конвертер может находиться в другой подсети. Здесь помогут навыки системного администратора

Примечание: Отправляйте свои вопросы с подробным описанием проблемы в техподдержку:

- Задать вопрос
- Написать на почту: 8smc4@standa.lt
- Задайте вопрос в Telegram: @SMC5TechSupport

8.2 Не удаётся вращать мотором при помощи контроллера

- *Контроллер в состоянии Alarm*
- *Мотор вибрирует, вращения нет*
- *Механическое заклинивание*
- *Двигатель не реагирует на команды движения*

8.2.1 Контроллер в состоянии Alarm

Примечание: Попробуйте нажать *Stop* в главном окне XiLab. Это убирает состояние Alarm.

Если данный способ не помогает и Alarm появляется снова, то:

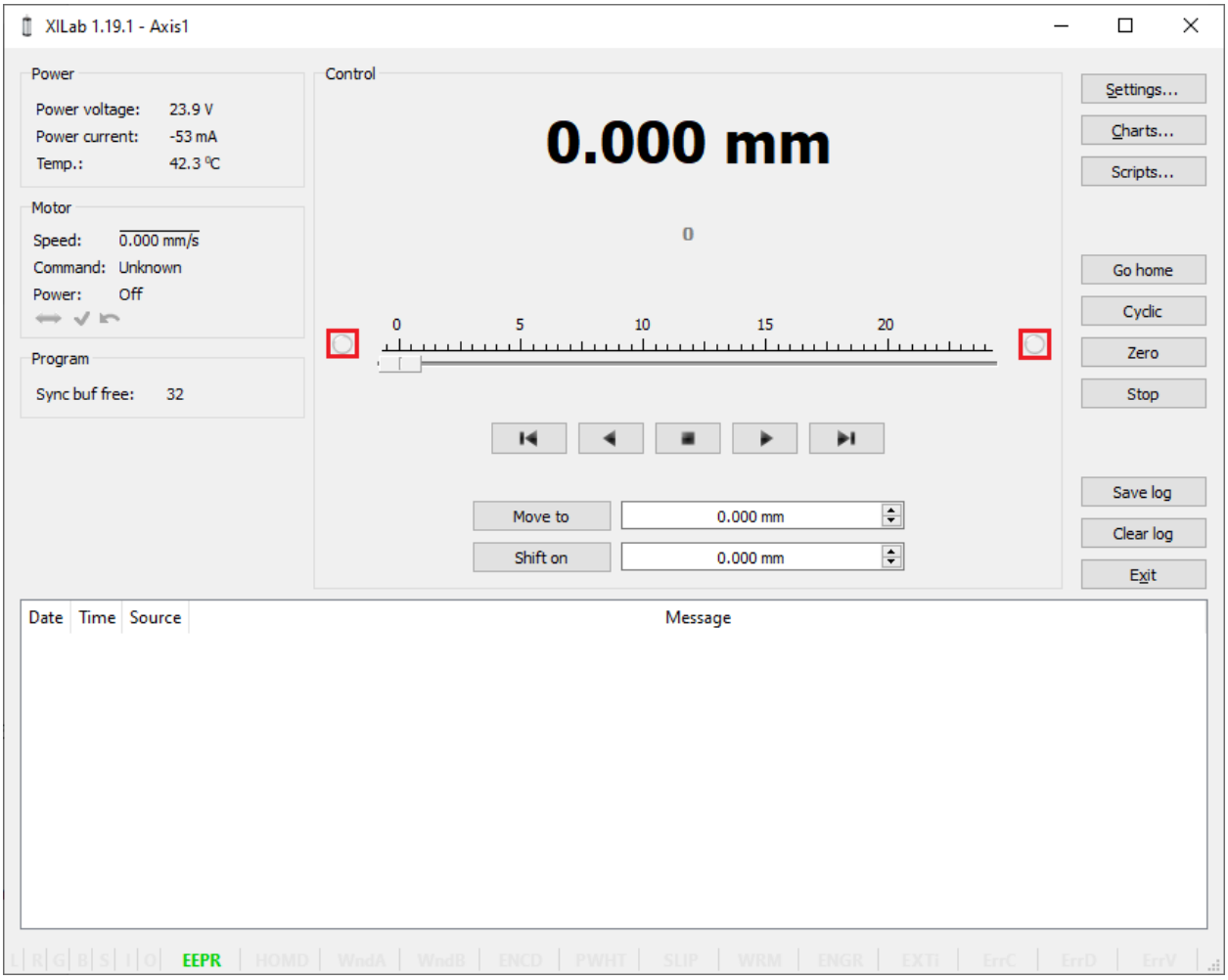
- Находясь в XiLab, перейдите во вкладку *Maximum ratings*.
- Поставте галочку напротив опции *Sticky Alarm flags*. Нажмите *Ок*.
- Нажмите *Stop* в главном окне XiLab, выйдя из состояния Alarm. Повторите последовательность действий, которая привела к Alarm.

- Сделайте скриншот главного окна XiLab и отправьте его в техподдержку с подробным описание проблемы.

8.2.2 Мотор вибрирует, вращения нет

У данной проблемы может быть несколько причин:

- Установлен **некорректный профиль** для вашего мотора/подвижки.
  - Поищите наиболее совпадающий по названию с используемой подвижкой профиль в папке с профилями в XiLab.
  - Рекомендуется сохранить текущую конфигурацию в файл. Для этого необходимо в окне *Settings* программы XiLab нажать *Save settings to file* (см. *настройки программы XiLab*), выбрать путь, куда сохранить настройки. Затем этот файл отправить в техподдержку с описанием проблемы.
- **Неправильно настроены концевые выключатели**, в результате чего подвижка уехала в концевик. Обычно это можно увидеть по загорающимся индикаторам в XiLab.



Основной причиной некорректной настройки концевиков является ошибочный профиль конфигурации для позиционера (см. предыдущий пункт). Информация по самостоятельной настройке находится в *разделе ручная настройка профиля*. При проблеме такого рода рекомендуется обратиться в техподдержку за дополнительной помощью.

  - Также одним из проявлений проблемы с концевиками может быть **механическое заклинивание** (см. следующий пункт).

- **Сгоревшая обмотка** мотора, проблемы с **контактом в разъёме** и т.п. Диагностировать проблемы такого рода можно самостоятельно. Для этого в XiLab есть возможность вывести графики напряжения и тока во время работы мотора. В исправном моторе ток в обмотках меняется по синусу или косинусу. В сломанном моторе будут заметны сильные отличия формы сигналов от гармонической.

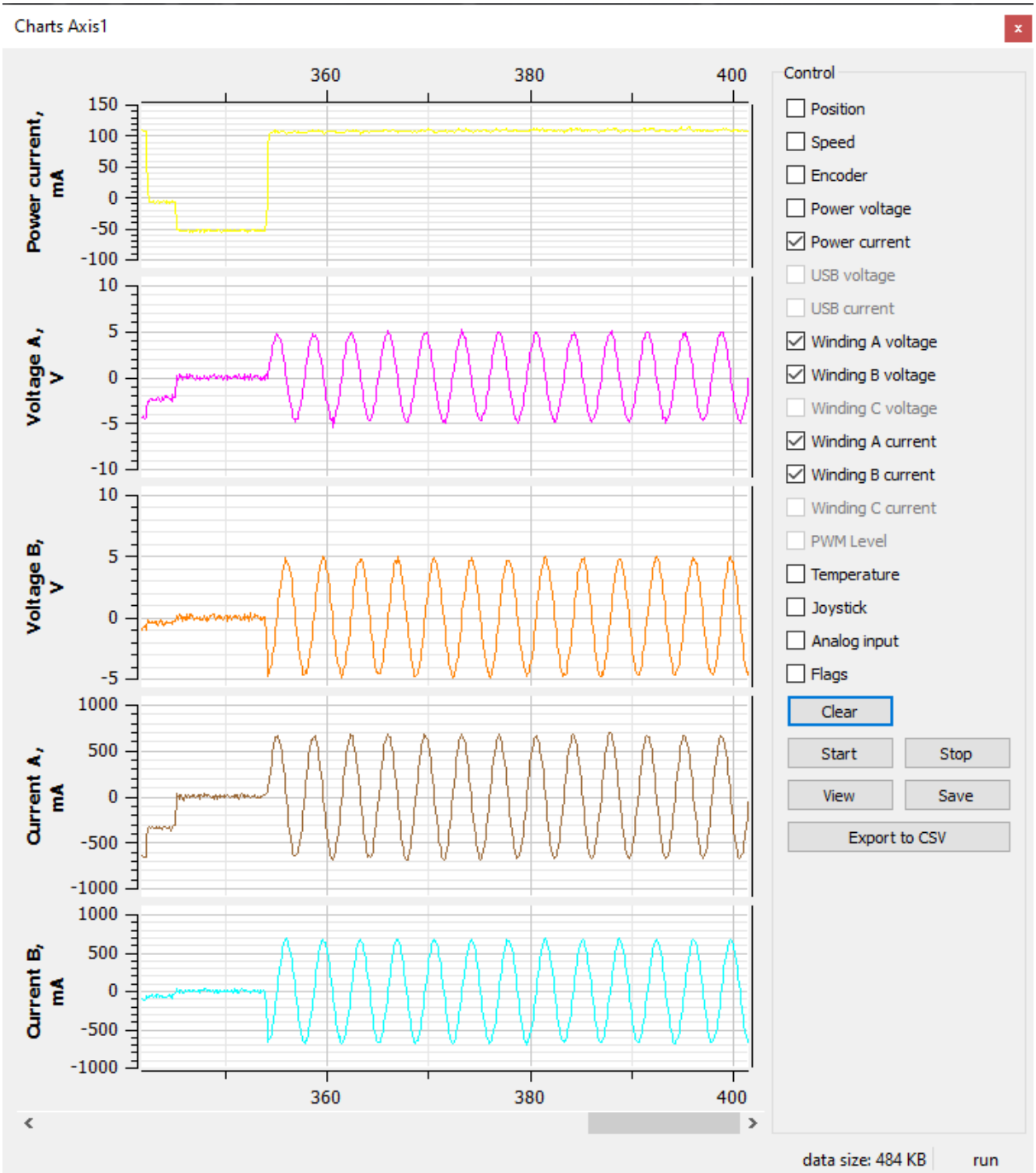


Рис. 8.1: Исправный случай

На графиках ниже видны проблемы. Например, отсутствует ток через обмотку B. Вероятно, в ней имеется разрыв. Также искажены остальные формы напряжений и токов.

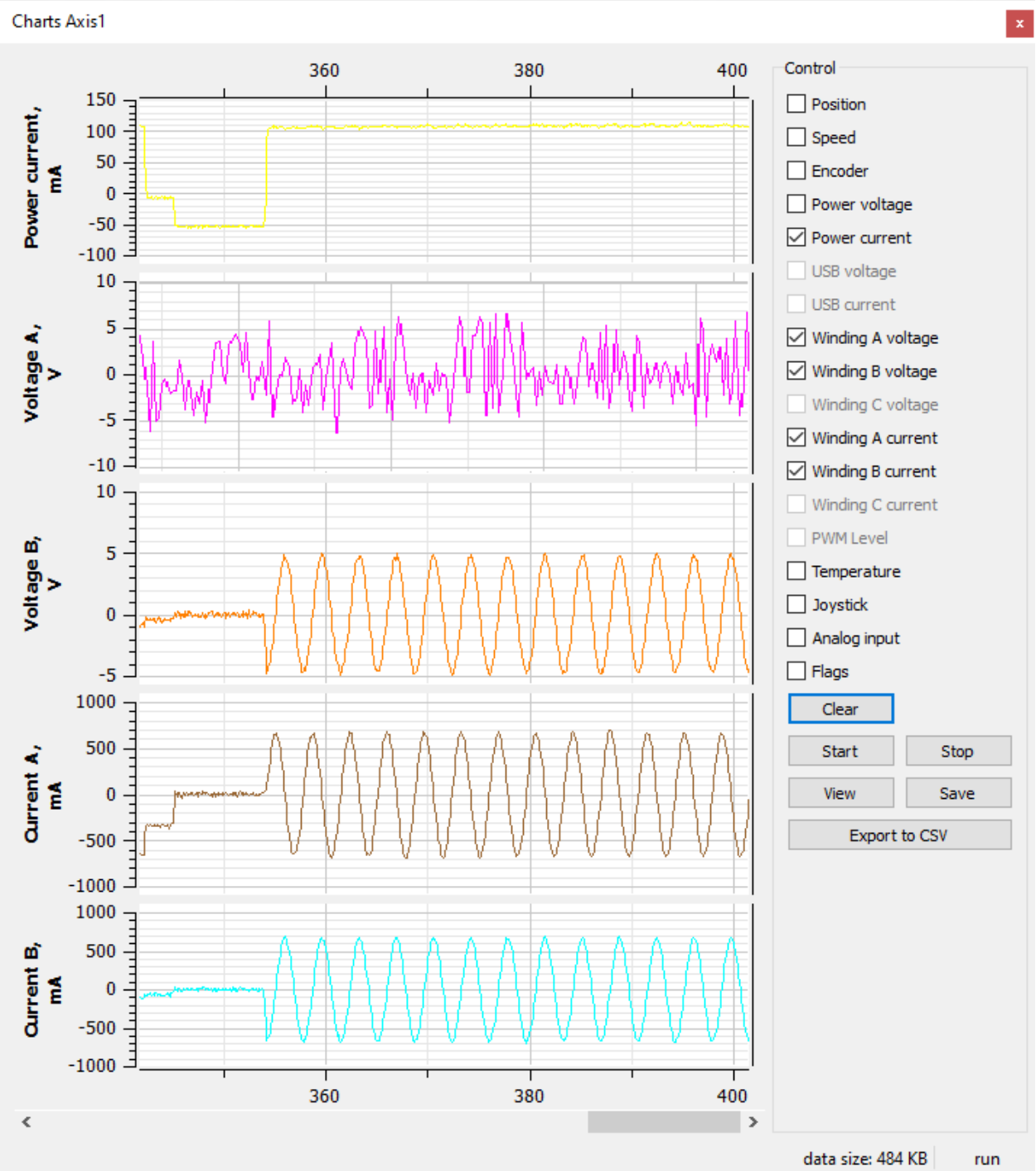


Рис. 8.2: Имеются проблемы с мотором

Для диагностики проблемы установите маленькую рабочую скорость (**1 ш/с** - оптимально) и подайте команду движения. Далее включите вывод графиков напряжений и токов в обмотках A и B и график полного тока в XiLab (кнопка *Charts*, далее отметьте галочками). Подождите некоторое время, пока графики построятся. После этого рекомендуется отправить их (кнопка *Save* для сохранения) в техподдержку с подробной формулировкой проблемы. Иногда при сгоревшей обмотке невозможно пользоваться XiLab из-за постоянной потери устройтва, и вывод графиков становится невозможен. В

этом случае также обратитесь в техподдержку с подробным описанием проблемы.

8.2.3 Механическое заклинивание

Существует два способа справиться с заклиниванием в позиционере:

- Провернуть подвижку руками, если это возможно.
- Увеличить ток в обмотке в 2-3 раза на короткое время (до 5-10 секунд) и подать команду движения в нужную сторону на невысокой скорости (**50-100 ш/с** - разумное значение). Через несколько секунд после вращения нажимать клавишу остановки (чёрный квадратик) в главом окне XiLab, до тех пор, пока не появится статус **power off**. Это позволит избежать перегрева мотора. **После выполнения данной операции не забудьте вернуть настройки обратно!**

8.2.4 Двигатель не реагирует на команды движения

Контроллер выглядит нормально, но двигатель не начинает движение по команде, оставляя сообщения об ошибках в журнале, контроллер перезагружается. Эта ошибка может возникнуть из-за существенно неправильных настроек калибровок контроллера. Это происходит, когда ожидаемые значения электрических параметров двигателя отличаются от реальных на несколько порядков. Установка неправильных калибровочных параметров может быть вызвана неравномерной механической нагрузкой на двигатель или различием механического трения по разным направлениям движения. В этом случае контроллер пытается сделать небольшое перемещение для калибровки двигателя (калибровка выполняется до включения питания двигателя) и выключается из-за срабатывания защиты по току.

Если вы столкнулись с этой проблемой, просто выполните следующие действия:

- Откройте XiLab, загрузите профиль для используемого двигателя.
- На вкладке *Stepper motor* в меню *Settings* выставьте номинальный ток **200 мА**, рабочую скорость **1 ш/с**, нажмите *Apply* и *Save to flash*.
- Попытайтесь начать движение, наблюдайте за текущими параметрами двигателя в окне *Charts*, как описано выше.
- Если диаграммы выглядят нормально, загрузите обычные настройки для используемого двигателя и работайте с ним, как обычно.

Примечание: Отправляйте свои вопросы с подробным описанием проблемы в техподдержку:

- Задать вопрос
- Написать на почту: 8smc4@standa.lt
- Задайте вопрос в Telegram: @SMC5TechSupport

8.3 Потеря USB-соединения

Наиболее распространенная причина такого рода проблем заключается в заземлении. Чтобы выяснить причину постоянной потери USB-соединения, следует:

- Если контроллер и/или подвижка крепятся к металлическому столу, временно подложите под них что-нибудь диэлектрическое или полностью перенесите их на диэлектрическую поверхность;
- Заземлите компьютер;
- Заземлите контроллер;
- Заземлите подвижку;

Примечание: Если описанные выше шаги устранили проблему потери USB-соединения, значит проблема заключалась в заземлении вашего металлического стола. На нем возникали колебания электрического потенциала.

- Замените USB-кабель. *Используйте только проверенные и заведомо работоспособные USB-кабели!* Неисправный или некачественный USB кабель может стать причиной неправильной работы контроллера, в том числе ошибок при вращении мотором или при распознавании устройства операционной системой. **Супер короткие кабели с толстыми проводами и экранированием** идеально подходят для надежного соединения;
- Используйте другой USB-порт;
- Используйте другой ПК.

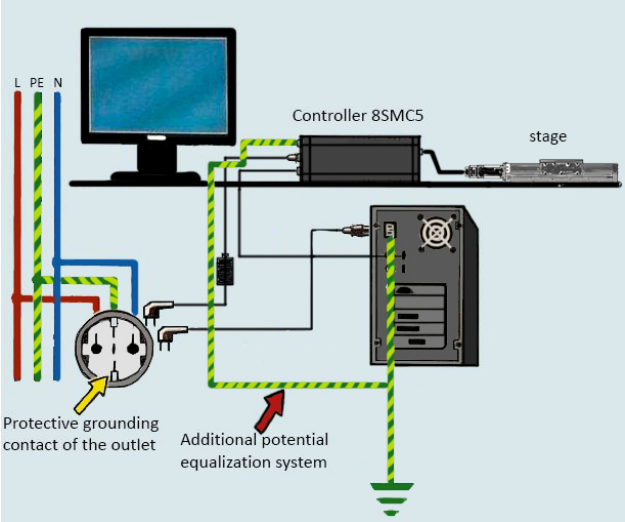


Рис. 8.3: Пример правильного заземления

Примечание: Отправляйте свои вопросы с подробным описанием проблемы в техподдержку:

- Задать вопрос
- Написать на почту: 8smc4@standa.lt
- Задайте вопрос в Telegram: @SMC5TechSupport

8.4 Самодиагностика адаптера 8Eth1

- *Инструкция по установке статического ip образа*
- *Самодиагностика статического ip-адреса на 8Eth1*

Примечание: Адаптер **8SMC4-USB-Eth** был переименован в **8Eth1**[1]. Все имеющиеся инструкции

[1] Артикул был изменен в 2020 г., ранее устройство имело артикул 8SMC4-USB-Eth.

в нашей документации актуальны и применимы к обоим устройствам.

Программное обеспечение со статическим ip-адресом для адаптера 8Eth1 можно скачать с нашего сайта

8.4.1 Инструкция по установке статического ip образа

Внимание: Каждый шаг инструкции является обязательным, и ни один шаг нельзя пропустить!

1. Сначала вы должны записать образ на microSD карту, которая находится в вашем Ethernet адаптере. Для этого извлеките microSD карту из 8Eth1 и с помощью кардридера подключите ее к компьютеру. С помощью программы *«win32diskimager»* (для Windows) или *«dd»* (для Linux) откройте образ в программе. Образ необходимо записать на microSD.
2. Затем нужно извлечь microSD карту из кардридера и вставить обратно в 8Eth1. При запуске Ethernet адаптер получит ip адрес через DHCP.
3. Далее нужно подключиться по SSH, порт 36000 (**не 22!**).

   Подсказка: логин - root, пароль - 12345678

4. Наконец, установка статического IP адреса. Введите в командной строке следующее:

```
mount -o remount,rw /
vi /etc/network/interfaces
```

5. Измените статический ip адрес на динамический, маску сети, шлюз и DNS-сервера. Ниже приведен пример. После всех вышеперечисленных действий вам необходимо перезагрузить адаптер.

```
auto eth0
iface eth0 inet static
address 172.16.1.101
netmask 255.255.255.0
gateway 172.16.1.1
dns-nameservers 172.16.1.1
:wq
sync
reboot
```

8.4.2 Самодиагностика статического ip-адреса на 8Eth1

После записи образа на SD-карту в конфигурационном файле по умолчанию устанавливается **динамический ip-адрес** . Этот адрес предоставляется вам автоматически после включения адаптера 8Eth1 (автоматически присваивается свободный ip-адрес, доступный в вашей сети). Если ip-адрес адаптера 8Eth1 отсутствует в Revealer после записи образа на SD-карту, необходимо выполнить самодиагностику:

Важно: Самодиагностику следует проводить в изолированной среде. Адаптер 8Eth1должен быть подключен к ПК. **Компьютер не должен быть подключен к интернету, и на нем должен быть запущен DHCP-сервер!**

1. Убедитесь, что образ правильно записан на SD-карту (в программе *win32diskimager* нажмите кнопку *«Verify Only»*);

2. Подключите монитор к порту HDMI, а клавиатуру - к порту USB. Убедитесь, что 8Eth1 адаптер загружается. Подождите, пока система загрузится, должно появиться поле для ввода логина;

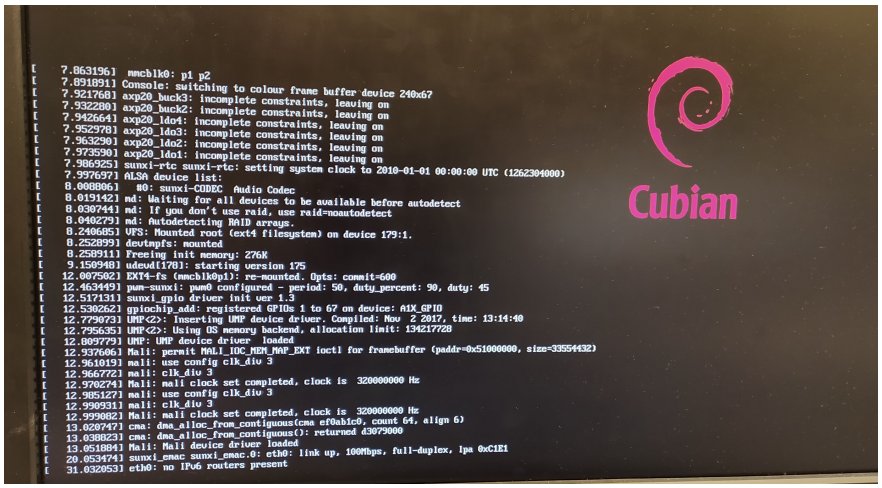


Рис. 8.4: Загрузка системы адаптера 8Eth1

3. Если системе не удается загрузиться, перезапустите адаптер 8Eth1. В какой-то момент вы увидите, что адаптер 8Eth1 запустился. Вы увидите поле для ввода логина;
4. Введите **логин - root**, **пароль - 12345678**;
5. Введите команду **ifconfig** и убедитесь, что указанному выше адаптеру был назначен динамический ip-адрес (вам нужно смотреть на «eth0»);

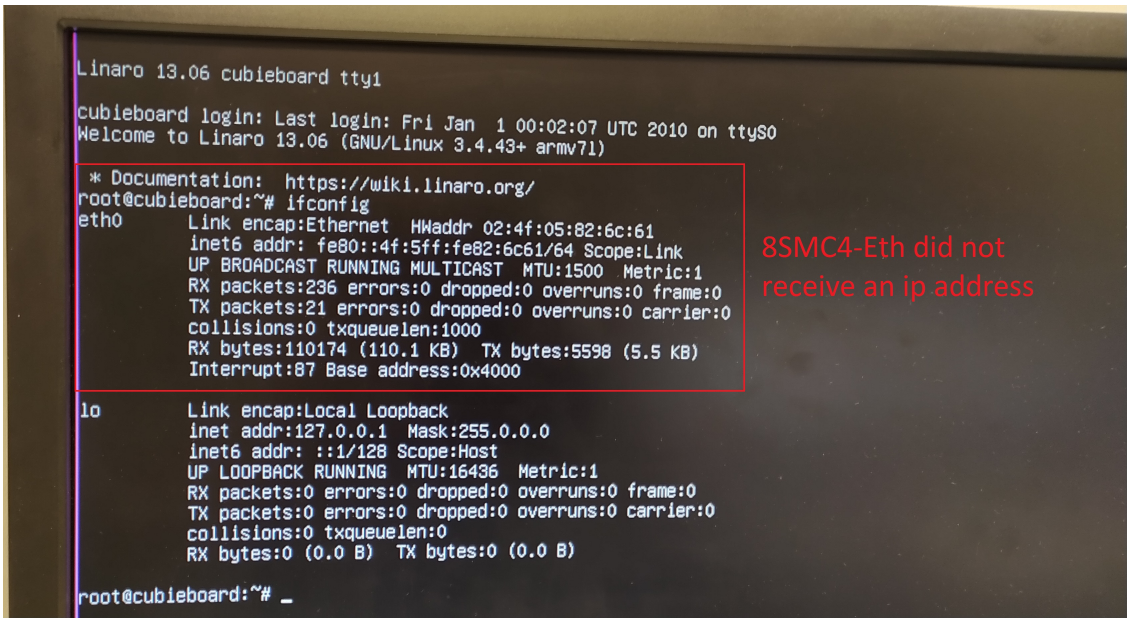


Рис. 8.5: Адаптер 8Eth1не получил ip-адрес

6. Присвойте адаптеру статический ip-адрес (инструкция по настройке статического ip-адреса написана выше);
7. Отсоедините монитор и клавиатуру от 8Eth1. Отключите 8Eth1 от компьютера и подключите его

к сети;

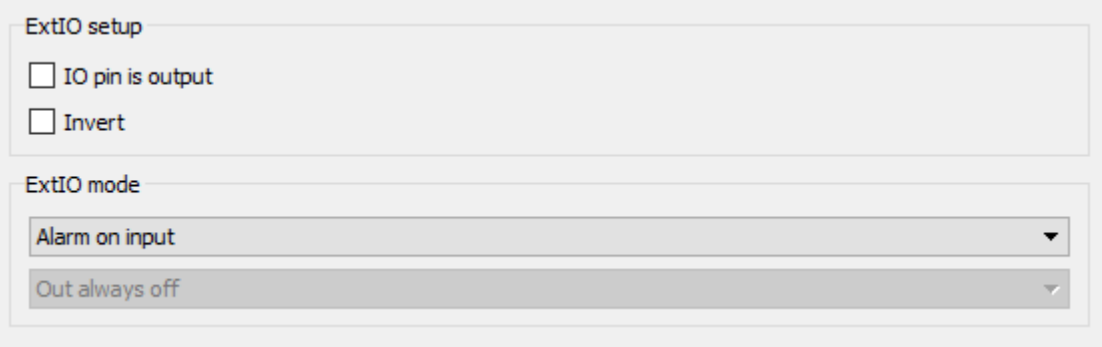
8. Не забудьте сохранить изменения и перезапустить адаптер;
9. Дождитесь перезагрузки и найдите статический ip-адрес вашего адаптера 8Eth1 в программе Revealer.

8.5 Как реализовать кнопку экстренной остановки?

Для реализации кнопки аварийной остановки вам потребуется использовать *цифровой вход/выход общего назначения*, (контакты **25. EXTGPIO_0** и **14. DGND, digital ground**), расположенный на *разъеме HDB-26*.

Используя библиотеку libximc, нужно будет установить флаг **0x5 - EXTIO_SETUP_MODE_IN_ALARM** (см. *Команду SEIO*).

Если вы используете XiLab, нужно снять флажок «IO pin is output» в *настройках EXTIO* , а затем из выпадающего списка выбрать «Alarm on input».



Важно: Для кнопки аварийной остановки рекомендуется использовать именно ALARM, так как ALARM не позволит выполнять какие-либо действия до тех пор, пока он не будет сброшен (сброс происходит с помощью кнопки stop или при вызове команды stop). Если вместо ALARM используется другая команда, например «stop» или «power off», то при вызове любой команды движения (MOVE/MOVR/LEFT/RIGT) движение продолжится, несмотря на то, что кнопка осталась нажатой.

В функционал аварийной кнопки не заложена плавная остановка!

8.6 Список паролей панели администратора 8SMC5 по умолчанию

Имя устройства	Пароль
Адаптер 8Eth1	admin
8SMC5 Ethernet	0000

8.7 Как вернуть окно xilab, которое скрылось за пределами экрана?

Все данные установленных программ хранятся в скрытой папке *AppData* (настройки, закладки, история, сохранения и тд.). Один из следующих шагов поможет вам восстановить потерянное окно xilab:

Настройки xilab по умолчанию. Перейдите в каталог *C:|Users|<your_user>|AppData|Roaming* -> найдите папку *xilab.conf* -> переименуйте или удалите ее. После этого настройки xilab будут восстановлены по умолчанию, и все потерянные окна должны вернуться.

Вы также можете вручную изменить размер окна. Для этого в папке *xilab.conf* найдите файл с серийным номером вашего контроллера, *например: SM123.cfg* , откройте его любым текстовым редактором и измените поля:

```
[settingsWindow_params]
position = @ Point (411 1057)
size = @ Size (722 286)
```

Расположите окна каскадом Щёлкните правой кнопкой мыши по панели задач. Выберите «Расположить окна каскадом». Все открытые программы появятся перед вами, и можно будет рассортировать их.

Включите обнаружение дисплеев Нажмите правой кнопкой мыши на рабочем столе и выберите «Параметры экрана». Затем щёлкните «Обнаружить». Windows вернёт пропавшие окна на экран. Помогает, если проблема возникла из-за того, что у вас несколько мониторов.

Измените разрешение экрана Щёлкните правой кнопкой мыши на рабочем столе и нажмите «Параметры экрана». В открывшемся окне измените разрешение на какое-нибудь другое, доступное вам. Windows переместит все вышедшие за пределы экрана окна обратно на дисплей. После этого можно вернуть то разрешение, что было у вас по умолчанию.

Используйте сочетание клавиш Первым делом, сделайте «сбежавшее» окно активным. То есть, выделите его мышкой на панели задач или через Alt+Tab переключитесь на него. Нажмите комбинацию Alt+пробел. Она открывает специальное системное меню активного окна. Далее нажимаем стрелку вниз на клавиатуре и выделяем второй пункт — **Переместить**. Нажимаем Enter. Теперь, после нажатия Enter, окно готово к перемещению. Нажмите клавишу влево или вправо на клавиатуре и начните перемещать окно. Вы увидите контур передвигаемой программы. Продолжайте удерживать клавишу со стрелкой до тех пор, пока весь контур не окажется на видимом рабочем столе. После этого нажмите Enter.

8.8 Где я могу найти руководство по программированию для контроллера 8SMC5?

Руководство по программированию включено в архив libximc 2.X.X, где 2.X.X - номер версии. Руководство находится в /ximc-2.X.X/ximc/doc-ru/libximc7-ru.pdf. Руководство по программированию можно найти на нашем втором сайте libximc.xisupport.com или Вы можете скачать его PDF версию. Руководство по программированию создано в системе Doxygen.

8.9 CRC алгоритм на Python

Ниже приведен пример используемого нами алгоритма **CRC-16/MODBUS**, который написан на Python.

```
def crc16(data: bytes):
 crc = 0xffff
 for cur_byte in data:
   crc = crc ^ cur_byte
   for _ in range(8):
      a = crc
      carry_flag = a & 0x0001
      crc = crc >> 1
      if carry_flag == 1:
         crc = crc ^ 0xa001
return bytes([crc % 256, crc >> 8 % 256])


data = b"\x00\x00\x00\xC8\x00\x00\x00\x00\x00\x00\x00\x00"
crc = crc16(data)

crc_str = " ".join("{:02x}".format(x) for x in crc)
print(crc_str)
```

Примечание: По этой ссылке Вы можете найти алгоритм CRC-16/MODBUS написанный на *C#*, *Java* и *PHP*, найденный в интернете с открытым исходным кодом.

За работоспособность кода отвечают разработчики.

8.10 Виртуальный контроллер, как в XILab

Вы можете использовать виртуальный контроллер в своих программах. Для этого используйте функцию:

```
device_t XIMC_API open_device (const char *uri)
```

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

Аргументы uri- уникальный идентификатор устройства. Uri устройства имеет вид:

```
"xi-com:port"               - # Serial port
"xi-net://host/serial"      - # XiNet connection
"xi-udp://<ip/host>:<port>" - # Raw UDP connection
"xi-tcp://<ip/host>:<port>" - # Raw TCP connection
"xi-emu:///file"            - # Virtual device
```

Пример:

```
"xi-com:\\.\COM3        # in Windows
"xi-com:/dev/tty.s123"` # in Linux/Mac
```

Для сетевого устройства «host» это IPv4 адрес или полностью определённое имя домена, «serial» это серийный номер устройства в шестнадцатеричной системе.

Пример:

```
"xi-net://192.168.0.1/00001234"
"xi-net://hostname.com/89ABCDEF"
```

Для работы по UDP/TCP протоколу используйте *«xi-udp://<ip/host>:<port>»* или *«xi-tcp://<ip/host>:<port>»*.

Пример:

```
"xi-udp://192.168.0.1:1818"
"xi-tcp://192.168.0.1:1820"
```

Для виртуального устройства *«file»* это путь к файлу с сохраненным состоянием устройства. Если файл не существует, он будет создан и инициализирован значениями по умолчанию.

Пример:

```
"xi-emu:///C:\Users\<user>\AppData\Roaming\XILab.conf/V_1"    # in Windows
"xi-emu:///home/<user>/.config/XILab.conf/V_1"               # in Linux/Mac
```

Вы также можете использовать виртуальный контроллер из программного обеспечения XILab с загруженным профилем. Для этого выберите и откройте виртуальный контроллер в XILab. После загрузите необходимый профиль (»*Settings*» -> «*Load setting from file...*») или просто установите необходимые параметры и нажмите «*Apply*». Затем файл будет сохранен в каталоге *C:UsersuserAppDataRoamingXILab.conf/V_x*, где x - номер виртуального контроллера.

В вашей программе вы можете открыть этот виртуальный контроллер, указав полный путь к файлу.
Например:

```
device_name = "xi-emu:///C:\Users\<user>\AppData\Roaming\XILab.conf/V_1"
device = open_device(device_name)
```



Рис. 8.6: Демонстрация работы виртуального контроллера

Внимание: Функция загрузки профиля реализована только в интерфейсе XILab. Если вам нужно изменить настройки этапа во время выполнения кода, вы можете использовать альтернативный вариант. Вы можете загрузить профиль на флэш-память контроллера (откройте XILab->Settings->Load setting from file... ->выберите свой профиль->OK->Apply->Save settings to flash). Затем вы можете изменить настройки в вашей программе. Как только вы захотите вернуть все настройки по умолчанию, выполните *Команду READ*

8.11 *probe_flag* - что это?

```
probe_flags = 1 + 4; % ENUMERATE_PROBE and ENUMERATE_NETWORK
```

«probe_flag» - это параметр, передаваемый в функцию библиотеки libximc *«enumerate_devices»*. Он контролирует, как *libximc.dll* выполняет поиск устройств.

```
define ENUMERATE PROBE 0x01
```

Проверяет, является ли устройство XIMC-совместимым. **Будьте осторожны с этим флагом, т.к он отправляет данные в устройство!**

```
define ENUMERATE NETWORK 0x04
```

Проверяет сетевые устройства.

Примечание: В зависимости от типа подключения контроллера вы можете убрать тот или иной флаг

8.12 Как проверить, установлено ли соединение с 8SMC5-USB и активно ли оно еще во время моего сеанса с помощью библиотеки libximc?

Чтобы постоянно проверять наличие соединения между контроллером и библиотекой libximc, регулярно отправляйте команду get_status в цикле.

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

Описанный выше метод может быть реализован в любой программе, использующую библиотеку libximc.

Примечание: Аналогичный метод проверки соединения реализован в XILab

---

8.13 Проблема компенсации люфта (пример из техподдержки)

Примечание: В этой главе мы наглядно покажем проблему люфта и ее решение. Описание и решение взяты из реального запроса пользователя в службу технической поддержки

Уважаемый инженер по обслуживанию! Мы используем одно из ваших зеркал с электроприводом (8MKVDOM-1) с контроллером 8SMC5-USB в очень чувствительном месте для юстировки нашего лазера. Чтобы настроить нашу систему, мы наблюдаем мощность на выходе нашей лазерной системы. Проблема, с которой мы сталкиваемся, заключается в том, что мы получаем разные результаты при сканировании в одном направлении и снова в обратном, то есть «оптимальное» положение изменяется. Мы наблюдаем это при использовании нашей самодельной программы Labview, а также программного обеспечения XILab. Использование команды «Go home» в программном обеспечении XILab не переводит двигатель в положение 0, а приводит к колебательному движению, останавливающемуся где-то около нуля. Это намеренно и связано ли это с нашей проблемой? Для остановки используется концевой

выключатель, мы не пытались использовать «датчик оборотов» или «импульс синхронизации». Это вариант? Также у вас есть предложения, как сделать мотор максимально точным?

С уважением, Марвин

https://youtu.be/pmSF7dO5rAc

Видео, демонстрирующее проблему с люфтом

Здравствуйте, дорогой Марвин! Прежде всего, создайте профиль для своей сцены. Откройте XILab -> Settings -> «Load setting from file...» -> откройте папку «Standa» -> выберите 8MKVDOM-1.cfg. Не забудьте нажать кнопки «Apply» и «Save settings to flash». После этого нажмите кнопку «Go home» в главном окне xilab и нажмите кнопку «Zero». Датчик оборотов или энкодер помогут вам лучше контролировать положение

В вашей механической системе перемещение в желаемое положение слева и справа не одно и то же, и есть люфт, но эту проблему можно устранить. Для этого установите флажок «Backlash compensation» на вкладке «Stepper motor» и укажите значение, превышающее величину люфта (в вашем случае это 8-10 шагов). Знак этой настройки определяет направление движения к позиции. Положительный знак означает движение слева, отрицательный - справа. В поле «Backlash compensation speed» установите скорость, с которой будет выполняться компенсационное движение. Это значение должно быть низким (достаточно 50 s/sec), чтобы избежать «дрейфа» во время компенсации люфта.

Важно : Режим компенсации люфта не предполагает коррекции положения оси, обеспечивая только выбор направления, из которого ступень должна приближаться к точке назначения, придерживаясь этого выбранного направления.

С уважением, Влад

Привет Влад, Круто! Мы немного поработали, и теперь это работает

Благодарю! Марвин

8.14 Управление Raspberry Pi

Важно: Поддерживаются почти все одноплатные компьютеры ARM (Raspberry Pi 1/2/3/4/..., NanoPi, Cubieboard и т.д.). Единственным ограничением является то, что ядро ARM должно быть версии 7 или выше

8.14.1 Работа с программным обеспечением XILab на процессоре ARM

Предупреждение: XILab не будет работать на процессоре ARM!

Если в вашем linux есть графическая оболочка, вы можете использовать *неподдерживаемые примеры*, среди которых есть примеры, напоминающие XILab.

8.14.2 Работа с библиотекой libximc на процессоре ARM

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

`dpkg -i filename.deb`, где «filename.deb» - это имя пакета (пакеты в Debian имеют расширение .deb). Запускать dpkg необходимо с правами суперпользователя (root).

В ОС на базе Linux контроллеры XIMC должны распознаваться как устройства **ttyACMn** и иметь символическую ссылку в **/dev/ximc/**

Контроллер может не видеться в системе из-за отсутствия прав доступа к устройству. Чтобы решить эту проблему, создайте файл: `/etc/udev/rules.d/31-ximc.rules` и добавьте в него следующую строку: `SUBSYSTEM=="usb ATTRS{idVendor}=="067b MODE="0666"`

Идентификатор *idVendor* можно узнать, выполнив команду `lsusb`. Также одним из возможных решений проблемы «no device found» является добавление пользователя в группу **dialout**. **Важно**: После добавления пользователя в группу необходимо перезагрузить компьютер.

Комплект разработчика можно скачать на странице Программное обеспечение. Он содержит скомпилированную библиотеку libximc для систем Windows, Linux и Mac OS, руководство по программированию и примеры. Libximc — это кроссплатформенная библиотека, поддерживающая языки C++, C#, Delphi, Visual Basic, Matlab, Java и Python. Примеры, включенные в пакет библиотеки, предназначены для быстрого ознакомления с программированием для контроллеров XIMC. Исходники Libximc также доступны для скачивания.

8.15 Зависание операционной системы при использовании библиотеки libximc и ядра Linux с версией менее 3.16

Комментарий: проблема является следствием ошибки в драйвере последовательного порта cdc-acm. Наблюдается при частом последовательном открытии и закрытии нескольких устройств. Зависание проявляется на Debian 7 (ядро 3.2), не проявляется на Debian 8 (ядро 3.16). Дополнительная информация о проблеме находится по следующей ссылке.

Решение: обновление текущей версии Linux.

Примечание: Отправляйте свои вопросы с подробным описанием проблемы в техподдержку:

- Задать вопрос
- Написать на почту: 8smc4@standa.lt
- Задайте вопрос в Telegram: @SMC5TechSupport
