Владимирский государственный университет

М. С. ДЕНИСОВ И. В. РУМЯНЦЕВ П. А. ЧЕБОТАРЕВ

КОМПЬЮТЕРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Лабораторный практикум

Владимир 2024

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

М. С. ДЕНИСОВ И. В. РУМЯНЦЕВ П. А. ЧЕБОТАРЕВ

КОМПЬЮТЕРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Лабораторный практикум

Электронное издание

Владимир 2024

ISBN 978-5-9984-1976-8

УДК 004.896
ББК 32.966

Авторы: М. С. Денисов (введение, лабораторные работы № 3 – 5, заключение, рекомендательный библиографический список), И. В. Румянцев (лабораторная работа № 2), П. А. Чеботарев (п. 1 – 3, лабораторная работа № 1)

Рецензенты:
Кандидат технических наук, доцент
доцент кафедры технологии машиностроения
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
А. В. Жданов

Кандидат технических наук, доцент
доцент кафедры автоматизированных технологических систем
Брянского государственного технического университета
В. А. Хандожко

Издается по решению редакционно-издательского совета ВлГУ

Денисов, М. С. Компьютерные системы управления [Электронный ресурс] : лаб. практикум / М. С. Денисов, И. В. Румянцев, П. А. Чеботарев ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2024. – 132 с. – ISBN 978-5-9984-1976-8. – Электрон. дан. (6,01 Мб). – 1 электрон. опт. диск (CD-ROM). – Систем. требования: Intel от 1,3 ГГц ; Windows XP/7/8/10 ; Adobe Reader ; дисковод CD-ROM. – Загл. с титул. экрана.

Посвящен изучению учебного стенда по автоматизации, собранного на основе современных высокопроизводительных комплектующих китайской фирмы Inovance.

Предназначен для студентов направления подготовки 15.03.04 «Автоматизация технологических процессов и производств» очной и заочной форм обучения.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС ВО.

Ил. 180. Табл. 6. Библиогр.: 8 назв.

ISBN 978-5-9984-1976-8

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В течение последних десяти лет наблюдается активный переход на компьютерные системы управления на предприятиях и производствах. Промышленные организации в условиях высокой конкурентной борьбы на рынке ищут более эффективные способы для быстрой адаптации к современным производственным тенденциям и пути снижения себестоимости производимой продукции. Использование компьютерных систем управления напрямую способствует достижению поставленных целей.

Компьютерные системы вырабатывают управляющие воздействия, которые поступают на исполнительные механизмы благодаря цифровым и аналоговым преобразователям. Компьютерные системы управления имеют ряд преимуществ относительно других систем управления. Во-первых, в них легко учитывается и компенсируется нелинейность характеристик датчиков, преобразователей, усилителей, а также исполнительных механизмов. Во-вторых, компьютерные системы управления адаптивны и оптимальны, поэтому могут быть реализованы только на основе электронно-вычислительной машины. В-третьих, электронно-вычислительная машина представляет собой универсальную систему, а ее переналадка на другую систему управления заключается только в изменении коэффициентов или загрузке новой программы. При использовании аналоговых систем может возникнуть необходимость в полном перепрограммировании, а следовательно, и в разработке новой конструкторской и технологической документации.

Если рассматривать операционные системы реального времени, то следует помнить о том, что они описываются специальным стандартом. Проблема управления процессом в режиме реального времени заключается в аппаратном аспекте. Для того чтобы создать многозадачную операционную систему реального времени, необходимо использовать электронно-вычислительные машины в масштабах цеха или предприятия.

Таким образом, применение компьютерных систем управления для автоматизации технологических процессов – один из самых удобных и выгодных вариантов.

Лабораторный практикум предназначен для изучения принципов компьютерных систем управления, а также обучения программированию алгоритмов технологических процессов и интерфейсов. В теоретической части на примере стенда Inovance описывается возможная аппаратная часть для компьютерных систем управления, а также среды для программирования контроллеров и панелей.

Представленные лабораторные работы нацелены на изучение способов программирования исполнительных устройств и контрольно-измерительных приборов, а также способов программирования интерфейсов и алгоритмов для компьютерных систем управления.

1. УЧЕБНЫЙ СТЕНД ПО АВТОМАТИЗАЦИИ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ INOVANCE

Компания Inovance (Китай) по объёму продаж занимает третье место после Siemens и ABB. Численность сотрудников, работающих в разных уголках мира, – около 18 тыс. человек. Компания производит преобразователи частоты различных серий, панели визуализации, контроллеры и модули ввода/вывода, ЧПУ системы.

На рис. 1.1 представлен стенд Inovance, который включает следующие компоненты:

AM600 – программируемый логический контроллер (ПЛК)

GR10-EC-6SW – разветвитель сети EtherCAT

GL10-PS2 – блок питания

GL10-1600END/GL10-0016ETP – модули дискретного ввода/вывода.

Разветвитель сети EtherNET

GL10-PS2 – блок питания

GL-10-RTU-ECTA – коплер EtherCAT

GL10-4AD/GL10-4DA – модули аналогового ввода/вывода

Преобразователи частоты SV660N

Сервомоторы серии MS1

Рис. 1.1. Стенд Inovance

Назначение стенда: знакомство с продукцией Inovance. Изучение программы для программирования контроллера и панели. Ознакомление с обменом данных: OPC UA, Modbus TCP между панелью и контроллером.

Программируемый логический контроллер

ПЛК – вычислительное устройство, предназначенное для автоматизации технологических процессов, осуществляющее функции автономного сбора, обработки, хранения информации, выработки команд управления (рис. 1.2).

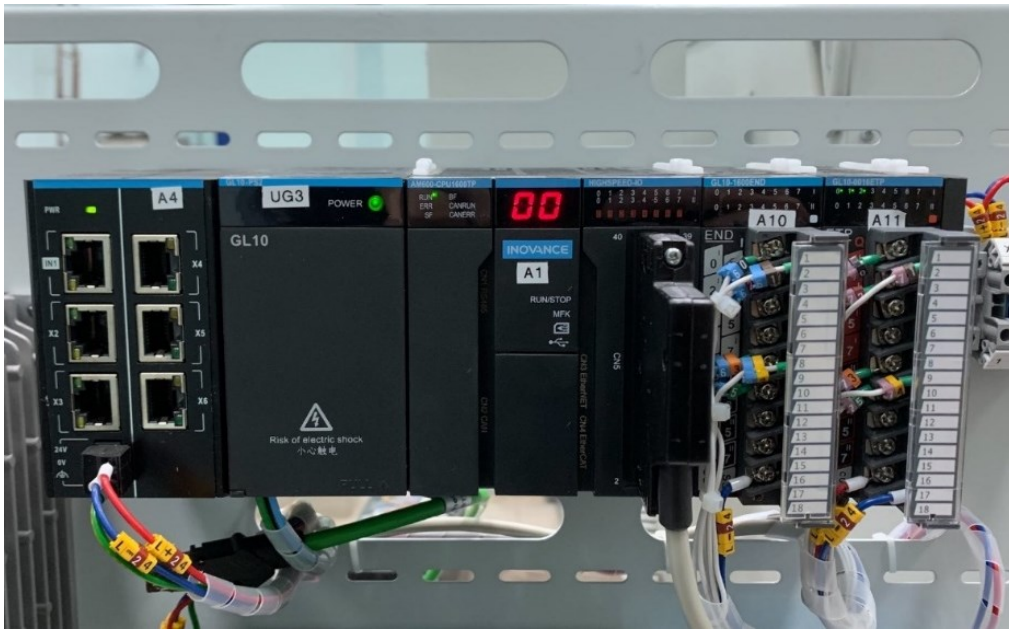


Рис. 1.2. Элементы автоматики Inovance

Стенд включает следующие средства автоматики (см. рис. 1.2):
(A4) GR10-EC-6SW – разветвитель сети EtherCAT
(UG3) GL10-PS2 – блок питания
AM600 – ПЛК
(A10-A11)GL10-1600END/GL10-0016ETP – модули дискретного ввода/вывода.

AM600. Контроллер средней производительности, предназначен для связи с объектами управления как через высокоскоростные входы/выходы, расположенные на борту контроллера, так и по цифровой сети EtherCAT. Питание контроллера осуществляется от специализированного блока питания.

Характеристики ПЛК:

– микропроцессор ARM-Cortex A8 1 ГГц;

– возможность управления до 32 осей – для двухточечного позиционирования (до 8 – для интерполированного движения);

– работа с математической точностью – IEEE 754 double;

– встроенный порт EtherCAT;

– возможность подключения до 16 дополнительных модулей ввода/вывода;

– языки программирования IEC 61131-3 (CODESYS);

– вход энкодера;

– имитация выходного сигнала энкодера;

– многозадачность.

Поддерживаемые интерфейсы и протоколы передачи данных:

– встроенный Ethernet;

– Modbus TCP/IP;

– OPC UA (сервер);

– интерфейс CANopen/Modbus RTU;

– CAM-функциональность.

(UG3) – блок питания GL10-PS2

Специальный модуль питания, имеет две клеммы. Одна из них является входным портом, другая – выходным.

Напряжение входного порта – 220 В.

Напряжение выходного порта – 1 – 24 В.

Напряжение выходного порта – 2 – 5 В.

Локальные модули, которые подключаются непосредственно на шину контроллера, также запитаны от блока питания. Пример подключения представлен на рис. 1.3.

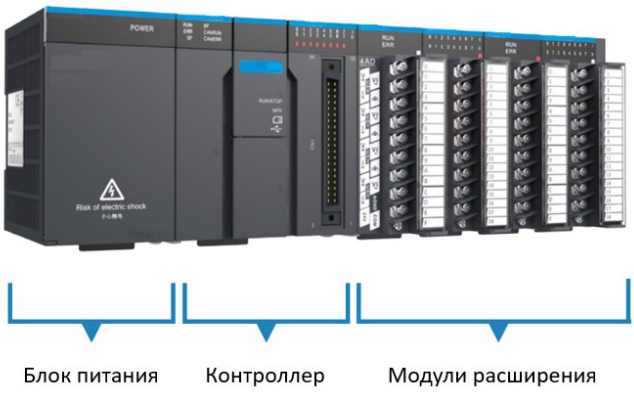


Рис. 1.3. Подключение блока питания

(A10-A11)GL10-1600END/GL10-0016ETP – модули дискретного ввода/вывода

Взаимодействие ПЛК с исполнительными механизмами, устройствами контроля осуществляется посредством модулей расширения и цифровых интерфейсов, позволяющих определить состояние объектов управления, сформировать внешние управляющие воздействия в соответствии с заложенной логикой.

GL10-1600END – модуль дискретного ввода

Характеристики:

– на борту модуля имеется индикация, соответствующая различным входным сигналам:

индикация ВКЛ: вход активен

индикация ВЫКЛ: вход неактивен;

– 16 входных дискретных каналов;

– режим управления как по плюсу, так и по минусу;

– класс входного напряжения – 24 В постоянного напряжения;

– внутренняя потребляемая мощность (типичная) – 5 В, 55 мА;

– входной ток (типичный) – 5,3 мА;

– время фильтрации порта и передачи данных в контроллер – 10 мс;

– входное сопротивление – 4,3 кОм.

GL10-0016ETP – модуль дискретного вывода

Характеристики:

– на борту модуля имеется индикация, соответствующая различным выходным сигналам:

индикация ВКЛ: вход активен

индикация ВЫКЛ: вход неактивен;

– 16 выходных дискретных каналов;

– режим управления по плюсу;

– источник выходного режима;

– напряжение питания – 24 В постоянного тока;

– внутреннее энергопотребление – 5 В, 65 мА;

– время отклика при включении модуля – менее 0,5 мс (для аппаратного обеспечения).

(A4) GR10-EC-6SW – разветвитель сети EtherCAT

Так как у контроллера имеется только один выход для управления устройствами по EtherCAT, то необходимо использовать разветвитель.

Модуль (рис. 1.4) позволяет принимать сигнал с одного устройства и переадресовывать его на один или несколько других портов.

Основные характеристики:

– номинальное рабочее напряжение – 24 В;

– внутренний потребляемый ток – 160 мА;

– протокол связи EtherCAT;

– канал EtherCAT с одним входом и пятью выходами;

– максимальная скорость связи – 100 Мбит/с;

– стандартный сетевой порт с сетевыми кабелями Cat 5e, длина кабеля – не более 100 м;

– рабочая температура – от –10 до +55 ℃;

– температура хранения – от –25 до +70 ℃;

– влажность – 10 – 95 %, без образования конденсата.

Устройства удаленного доступа подключены к контроллеру по протоколу EtherCAT.

Рис. 1.4. Удаленное подключение модулей ввода/вывода

На рис. 1.4 представлены следующие модули:

(A5) – разветвитель сети EtherNET

(UG4) GL10-PS2 – блок питания

(A20) GL-10-RTU-ECTA – коплер EtherCAT

(A21-A22) GL10-4AD/GL10-4DA – модули аналогового ввода/вывода

(A20) GL-10-RTU-ECTA – коплер EtherCAT

Модуль предназначен для удаленного подключения с помощью протокола EtherCAT к контроллеру модулей ввода/вывода.

Основные характеристики:

– источник питания – 24 В;

– протокол связи EtherCAT;

– максимальная скорость связи Ethernet – 100 Мбит/с;

– сетевой интерфейс: стандартный интерфейс Ethernet (сетевой кабель повышенной категории 5 с длиной кабеля не более 100 м);

– диапазон номеров станций – от 1 до 125, внутренний адрес автоматически упорядочивается в последовательности подключения к сетевой шине;

– может быть подключено до 16 модулей ввода/вывода. Фактическое количество и конфигурация зависят от потребляемой мощности каждого модуля.

(A21-A22)GL10-4AD/GL10-4DA – модули аналогового ввода/вывода

GL10-4AD – модуль аналогового ввода

Основные характеристики:

– четыре входных канала;

– напряжение питания – 24 В;

– потребляемая внутренняя мощность – 5 В, 85 мА (типичное значение);

– входное сопротивление напряжения – > 1 кОм;

– сопротивление выборки тока – 250 Ом;

– диапазон входного тока – от 0 до 20 мА, от 4 до 20 мА, ±20 мА, ±10 В, от 0 до 10 В;

– время выборки – 1 мс;

– пределы напряжения – ±15 В;

– пределы тока – ±30 мА (переходный), ±24 мА (средний);

– системная программа обновляется через USB-интерфейс;

– защита от короткого замыкания на выходе.

GL10-4DA – модуль аналогового вывода

– четыре выходных канала;

– напряжение питания – 24 В;

– потребляемая внутренняя мощность – 5 В, 85 мА (типичное значение);

– выходное напряжение нагрузки – от 1 кОм до 1 МОм;
– диапазон выходного напряжения:
биполярный: ±5 В, ±10 В;
однополярный: +5 В, +10 В;
– диапазон выходного тока – от 4 до 20 мА, от 0 до 20 мА;
– защита от короткого замыкания на выходе;
– системная программа обновляется через USB-интерфейс.

Преобразователи частоты SV660N

Пропускная способность токовой петли SV660N 3 кГц позволяет двигателю следовать профилю движения с минимальной погрешностью.

Рис. 1.5. Преобразователи частоты SV660N

Высокая скорость передачи данных EtherCAT для устройства составляет 125 мкс.

В устройствах серии SV660N (рис. 1.5) используется высокопроизводительный процессор для высокоскоростной связи, обеспечивающий время цикла 125 мкс для всех режимов работы EtherCAT. Доступны семь режимов работы профиля EtherCAT CiA402 (CoE):

– режим положения профиля (PP);
– режим скорости профиля (PV);
– режим крутящего момента профиля (PT);
– режим самонаведения (HM);
– циклический синхронный позиционный режим (CSP);
– режим циклической синхронной скорости (CSV);
– режим циклического синхронного крутящего момента (CST).

23-битный последовательный однооборотный/многооборотный абсолютный энкодер с обратной связью высокого разрешения выдает

8 388 608 импульсов за один механический оборот. Информация о нескольких оборотах также может сохраняться при отключении питания, что позволяет избежать необходимости выполнять автонастройку частотного преобразователя при каждом включении питания.

Общие характеристики:

– управление по протоколу EtherCAT;

– режим управления: IGBT-ШИМ-управление, режим возбуждения синусоидальным током 220 В, однофазное;

– 23-разрядный абсолютный энкодер, который может использоваться в качестве инкрементного энкодера при отсутствии батареи;

– пыле-, влагозащищенность – IP20;

– пропускная способность контура скорости – 3 кГц;

– время плавного запуска – до 65 с (ускорение и замедление можно установить отдельно).

Сервомоторы MS1 (рис. 1.6) – серводвигатели последнего поколения, разработанные Inovance. Серводвигатели серии MS1 имеют диапазон мощности от 30 Вт до 7,5 кВт, а размеры фланцев варьируются от 25 до 180 мм. Серводвигатели указанной серии служат для обеспечения быстрого и точного контроля положения, скорости и крутящего момента в оборудовании автоматизации – полупроводниковых приборах, SMT-машинах, машинах для штамповки печатных плат, погрузочно-разгрузочных машинах, механических инструментах и трансмиссионных механизмах.

Рис. 1.6. Сервомотор MS1

Основные характеристики:

– напряжение – 220 В;

– мощность – 400 Вт;

– номинальный ток – 2,8 А;

– максимальное количество оборотов в минуту – 3000;

– режим работы – непрерывный;

– уровень вибрации – V15;

– сопротивление изоляции – 500 В постоянного тока, более 10 Мом;

– температура окружающей среды – от 0 до +40 °С;

– температура хранения – от –20 до +60° С (максимальная температура – +80 °С в течение 72 ч);

– режим возбуждения – постоянный магнитный;

– способ монтажа – фланец;

– уровень термостойкости – уровень F;

– класс защиты корпуса – IP67;

– влажность окружающей среды – 20 – 80 % (без конденсата);

– вибрация – ниже 49 м/с^2.

Панель визуализации IT7070

Общие характеристики (рис. 1.7):

– центральный процессор Cortex A8 600 МГц;

– оперативная память – 128 Мб DDR3;

– Flash-память – 128 Мб;

– слот для SD карты – 1;

– последовательный порт COM1 (RS422/RS485), COM2 (RS232), COM3 (RS485);

– Ethernet порт – 1;

– Mini USB type B порт – 1;

– USB type A порт – 1;

– входной ток – 250 мА;

– диагональ экрана – 7“;

– разрешение – 800×480;

– яркость – 350;

– цвет дисплея – True Color;

– тип отображения – LED;

– срок службы подсветки – 35 000 ч;

– рабочая температура – от –10 до 55 °С;

– рабочая влажность – 10 – 90 %;

– есть возможность написать скрипты на языке JavaScript;

– программное обеспечение бесплатное;

– визуализация может быть адаптирована, за счет стилизации можно накладывать различные стили на все компоненты. Стилистика может применяться сразу ко всему проекту;

– связь с контроллерами происходит по OPC UA, Modbus TCP и через последовательные интерфейсы.

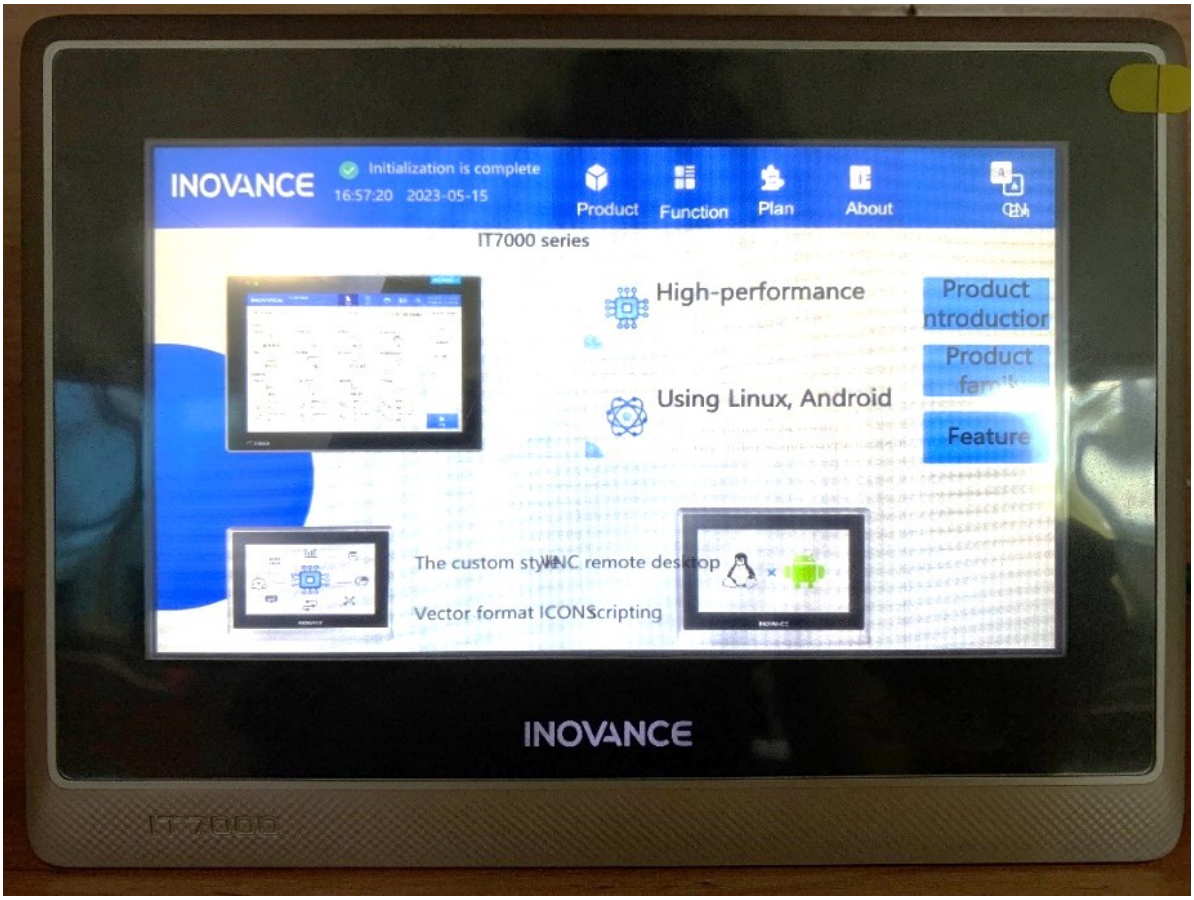


Рис. 1.7. Панель визуализации IT7070

Рассмотрим схему подключения всех устройств стенда и принципиальные электрические схемы (рис. 1.8 – 1.16).

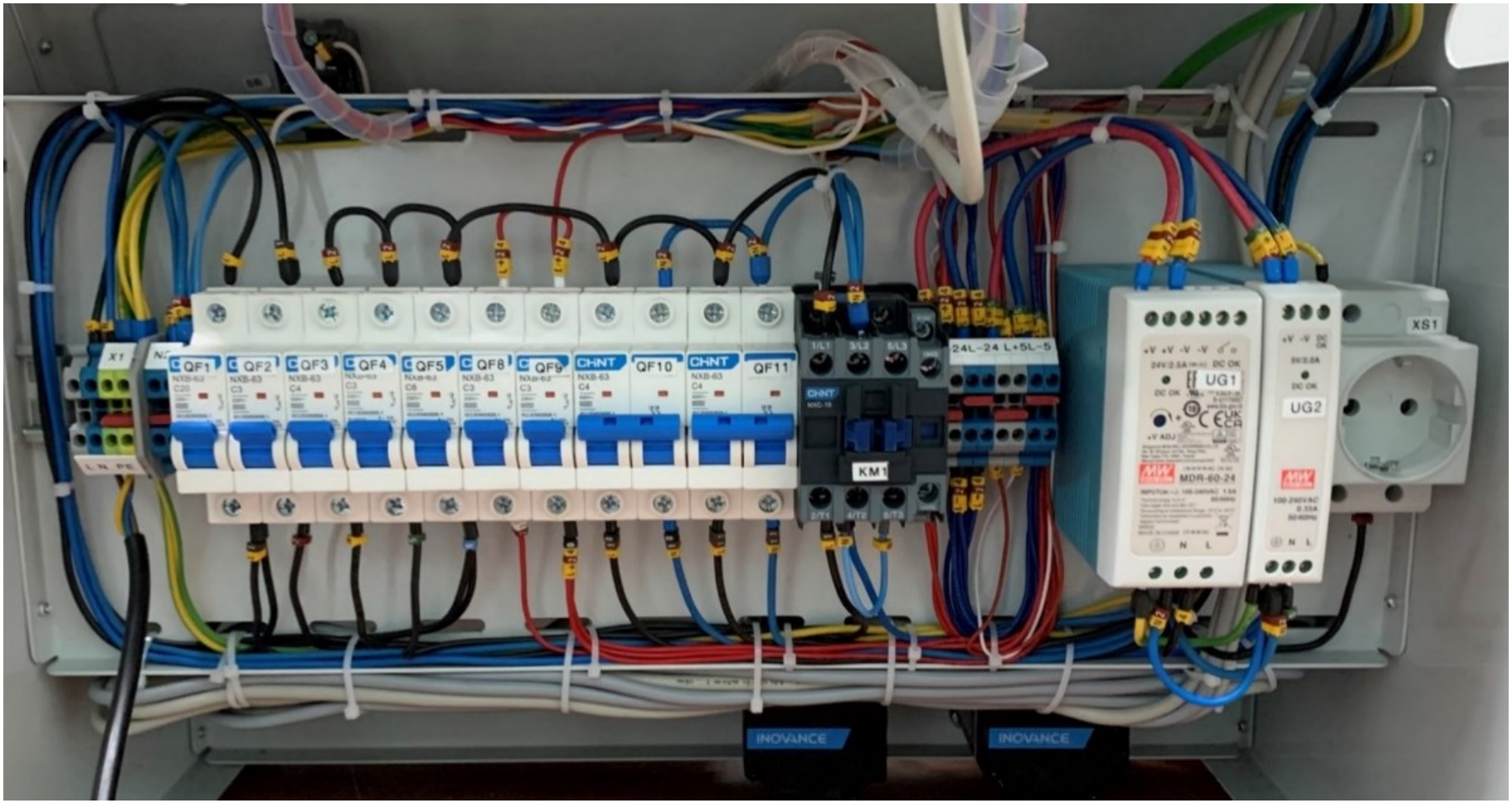


Рис. 1.8. Кабельное подключение периферии

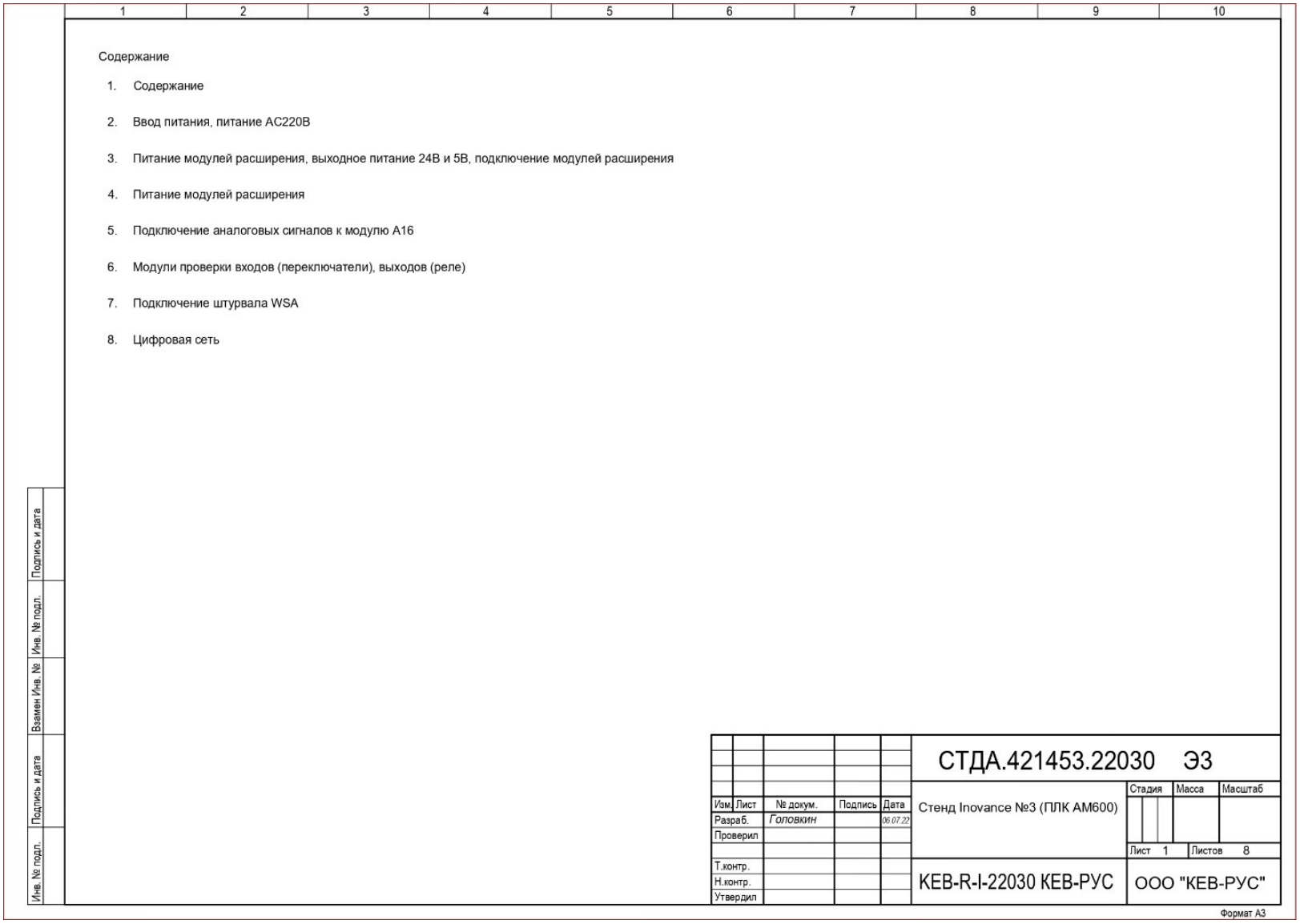


Рис. 1.9. Первая страница электрической принципиальной схемы

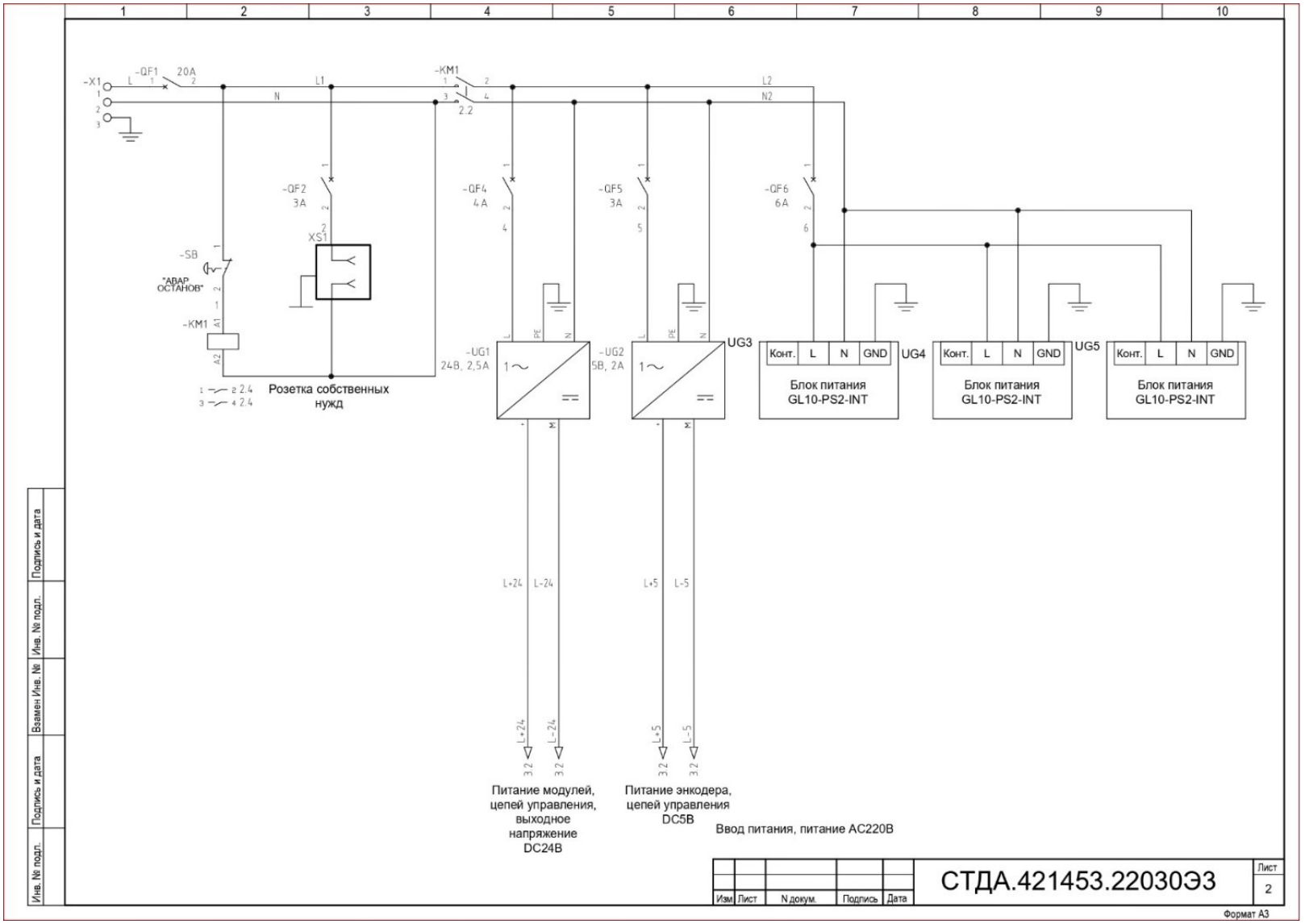


Рис. 1.10. Вторая страница электрической принципиальной схемы

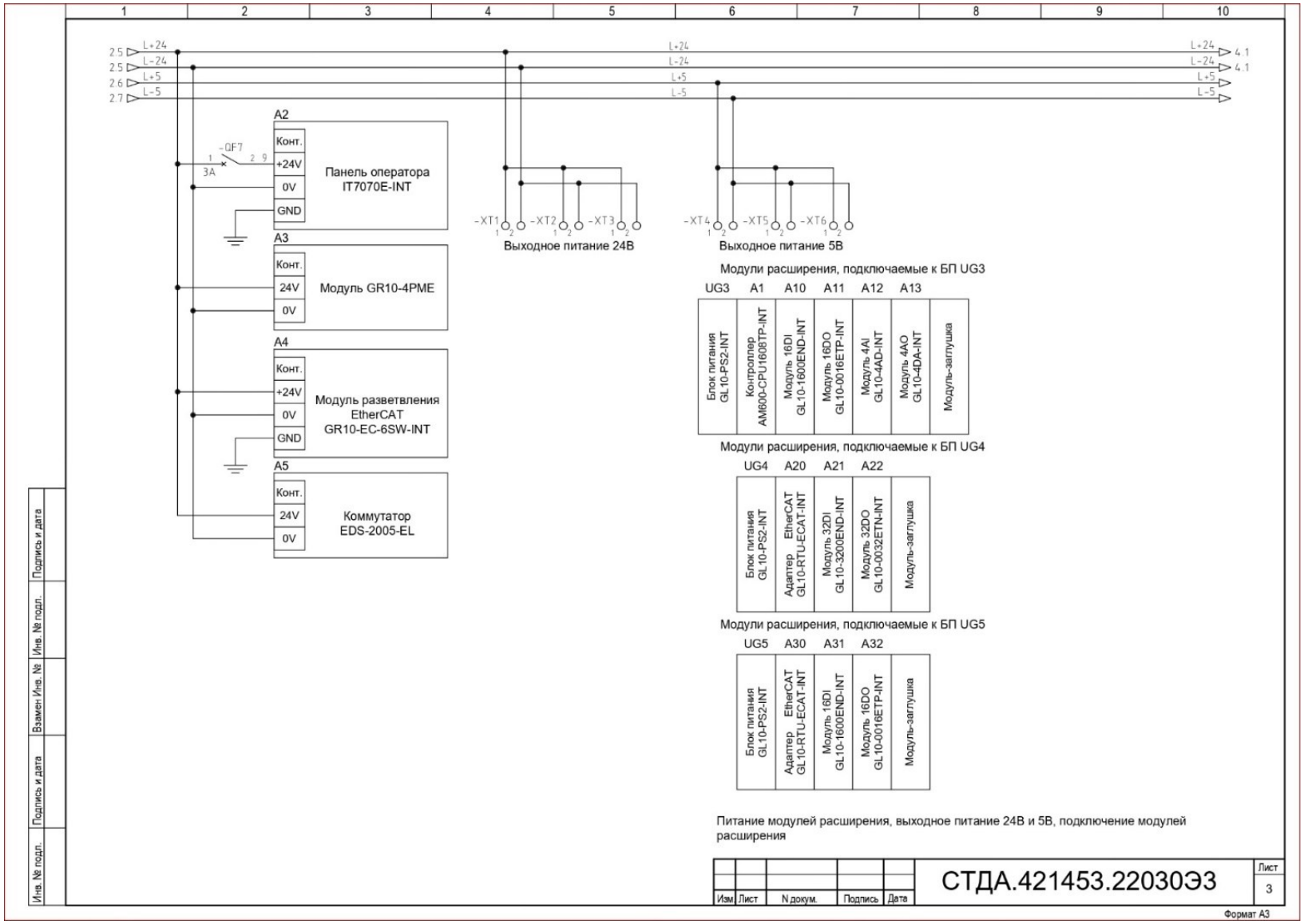


Рис. 1.11. Третья страница электрической принципиальной схемы

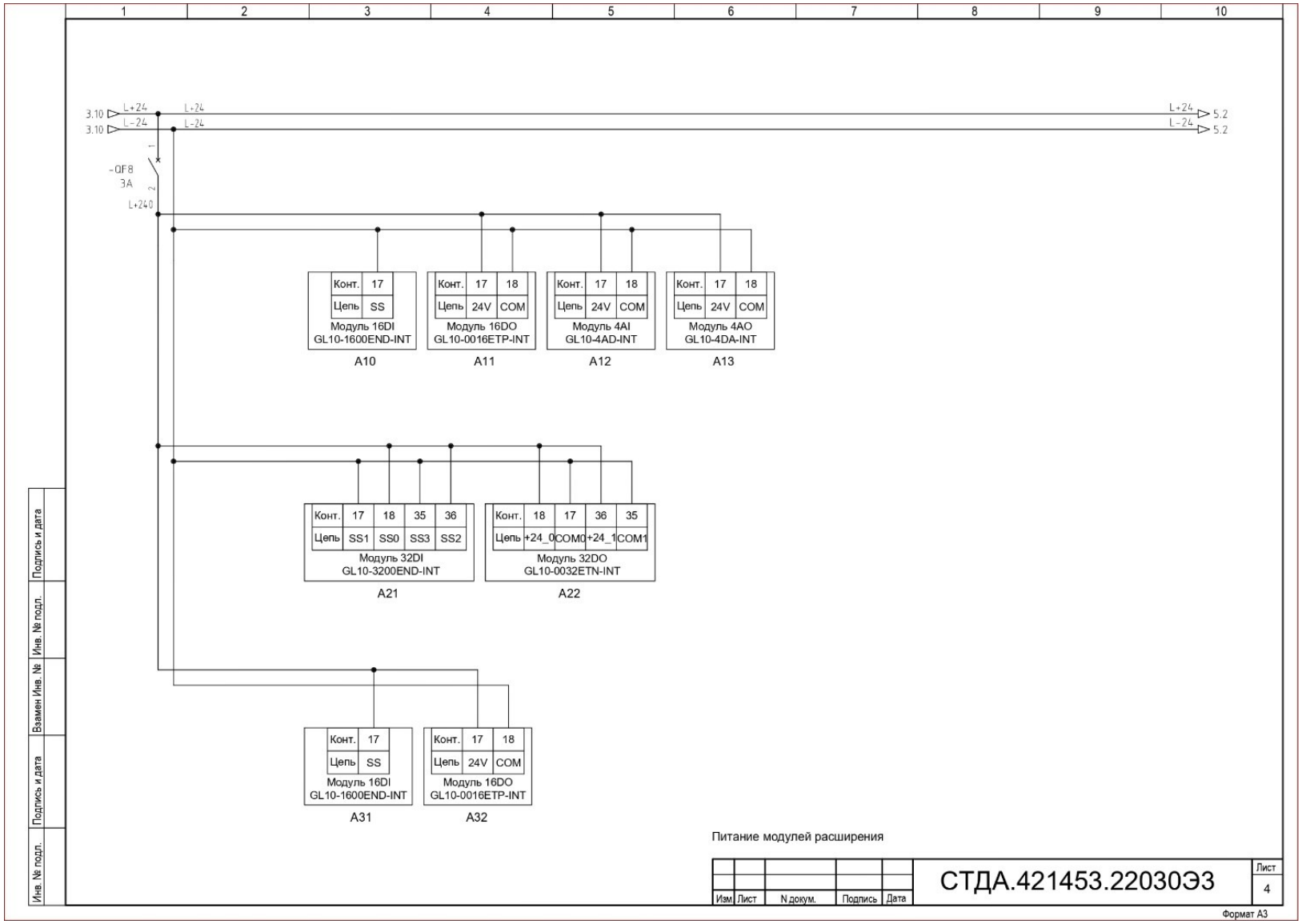


Рис. 1.12. Четвертая страница электрической принципиальной схемы

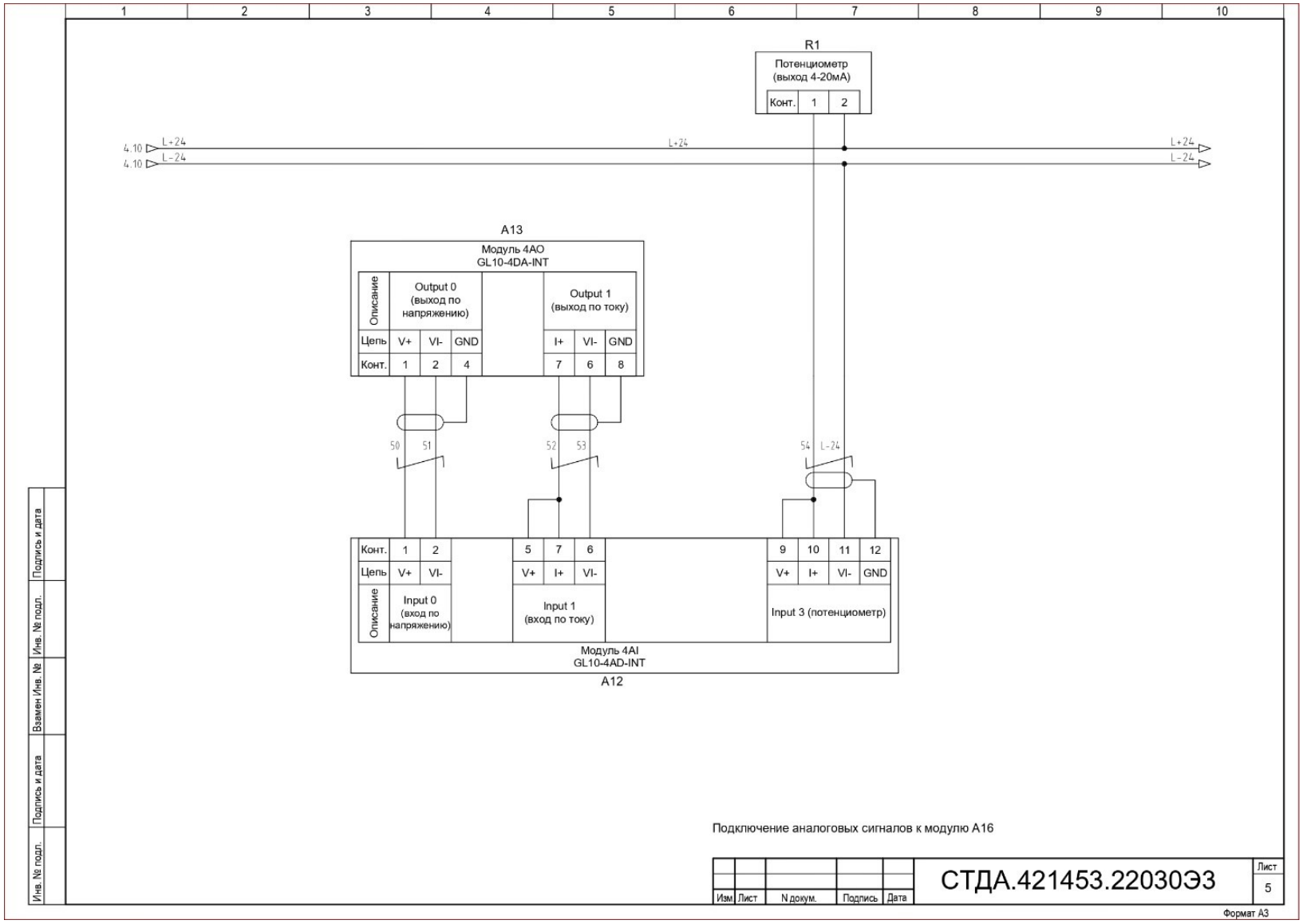


Рис. 1.13. Пятая страница электрической принципиальной схемы

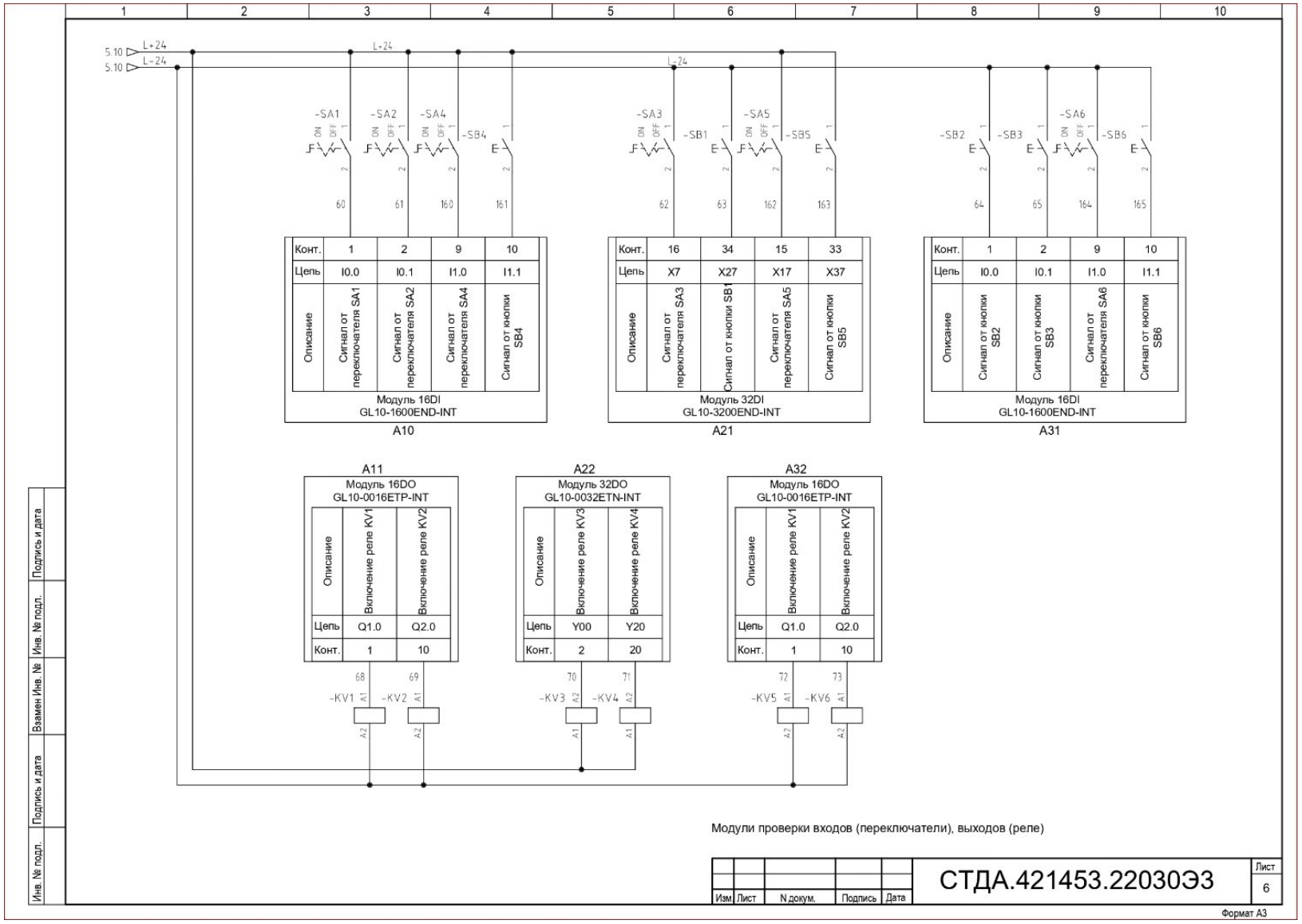
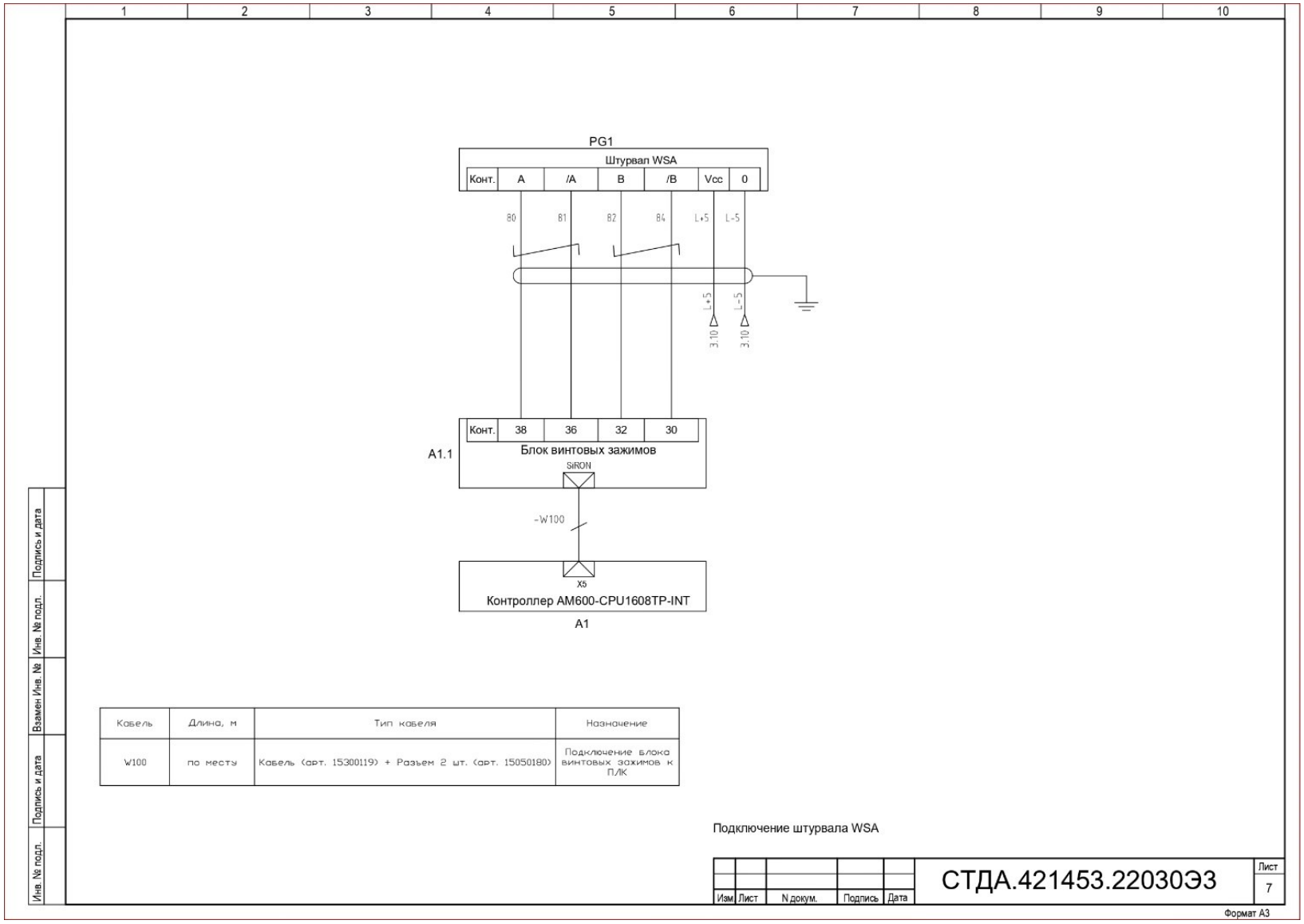


Рис. 1.14. Шестая страница электрической принципиальной схемы



Кабель	Длина, м	Тип кабеля	Назначение
W100	по месту	Кабель (арт. 15300119) + Разъем 2 шт. (арт. 15050180)	Подключение блока винтовых зажимов к ПЛК

Подключение штурвала WSA

Изм	Лист	N докум.	Подпись	Дата	СТДА.421453.22030Э3	Лист 7

Формат А3

Рис. 1.15. Седьмая страница электрической принципиальной схемы

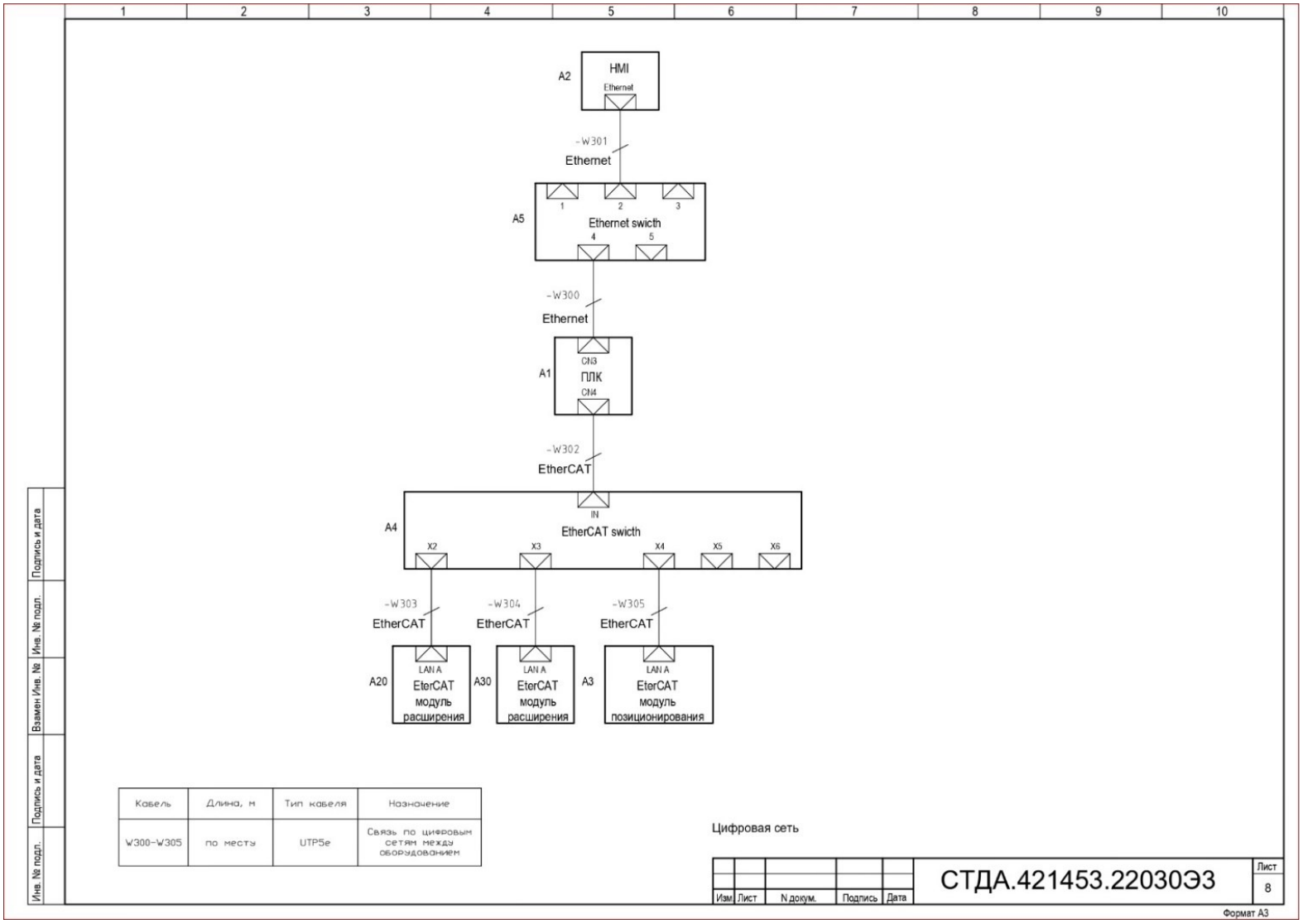


Рис. 1.16. Восьмая страница электрической принципиальной схемы

Как видно из рис. 1.10, питание 220 В в систему поступает на клемму X1. Затем необходимо включить 20 А автомат QF1 для замыкания электромагнитного контактора.

Аварийный выключатель SB находится в закрытом состоянии; если он будет активирован, то цепь будет разомкнута и питание дальше не пойдет.

Автомат QF2 отвечает за подачу питания на розетку собственных нужд.

Автоматы QF3, QF4, QF5 отвечают за включение блоков питания UG1, UG2, UG3 и UG4 соответственно.

После включения автоматов цепь питания идет от блоков питания и от цепи питания в соответствии с указанными стрелочками.

Номера после стрелочек указывают страницу и столбец, куда приходит питание. Например: после стрелочки написано 3.2 – это значит, что нужно смотреть на 3-ю страницу и 2-й столбец.

На рис. 1.11 показано подключение модулей расширения 24 и 5 В.

Автомат QF8 включает панель, модули разветвителей EtherCAT и EtherNET включаются, когда подается напряжение на блоки питания.

Напряжение 24 и 5 В приходит на клеммную колодку X11-X16. От них можно в будущем запитать датчики или модули.

Цифры перед стрелочками в первой колонке указывают, с какой страницы пришла цепь подключения.

На рис. 1.12 автомат QF9 заводит питание на клеммную колодку модулей дискретных входов/выходов и аналоговых входов/выходов.

На рис. 1.13 представлены подключение потенциометра к аналоговому модулю ввода и соединительные перемычки между вводом и выводом.

На рис. 1.14 показано подключение кнопок (SB2-SB4) и тумблеров (SA1-SA3) в дискретные входы и подключение дискретных выходов к реле.

На рис. 1.15 отражено подключение штурвала PG1 к блоку винтовых зажимов A1.1, а затем к высокоскоростным входам/выходам, расположенным на борту контроллера.

На рис. 1.16 показано подключение цифровой сети.

Для обмена данными между панелью A2 и контроллером A1 предусмотрен EtherNET разветвитель A5, между контроллером A1, сервоприводами UZ1/UZ2 и коплером A20 – разветвитель EtherCAT A4.

Контрольные вопросы

1. Что такое контроллер? Назовите функции и характеристики контроллера.

2. Расскажите про модули A21-A22, установленные на стенде. Какие функции они выполняют?

3. Для чего нужен EtherCAT разветвитель?

4. Какую функцию выполняют дискретные вводы/выводы?

5. Для чего предназначен сервомотор MS1? Каковы его основные характеристики?

6. Опишите функции и характеристики преобразователя частоты SV660N.

7. Опишите электрическую принципиальную схему.

2. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ КОНТРОЛЛЕРОВ INOVANCE

Действительно функциональным контроллер становится только тогда, когда создают и запускают программу по его использованию. Программу ПЛК Inovance разрабатывают в программном комплексе InoProShop.

Ключевые особенности InoProShop:

– языки программирования IEC 61131-3 (LD, FBD, SFC, ST, IL);

– объектно-ориентированное программирование;

– инструменты отладки;

– стандартные библиотеки МЭК.

Среда разработки основана на стандарте IEC 61131-3, который описывает языки программирования:

а) текстовые:

– Instruction List (IL) – список инструкций;

– Structured Text (ST) – структурированный текст;

б) графические:

– Sequential Function Chart (SFC) – последовательные функциональные схемы;

– Function Block Diagram (FBD) – функциональные блоковые диаграммы;

– Ladder Diagram (LD) – релейно-контактные схемы;

– Continuous Function Chart (CFC) – непрерывные функциональные схемы.

Создание и конфигурация шаблона проекта

Запустите InoProShop, создайте новый проект во вкладке «Файл» (рис. 2.1).

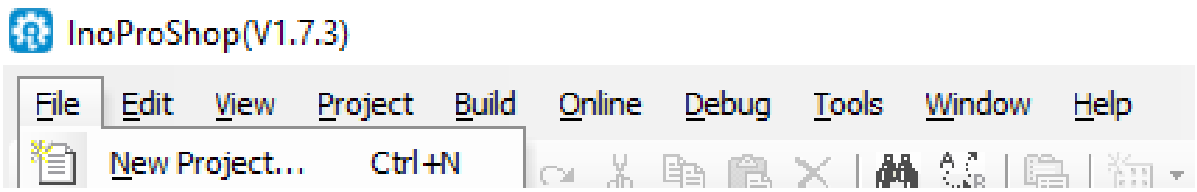


Рис. 2.1. Создание нового проекта

В диалоговом окне выберите шаблон «Standart PLC project», задайте атрибуты проекта («Имя», «Расположение») и нажмите «ОК» (рис. 2.2).

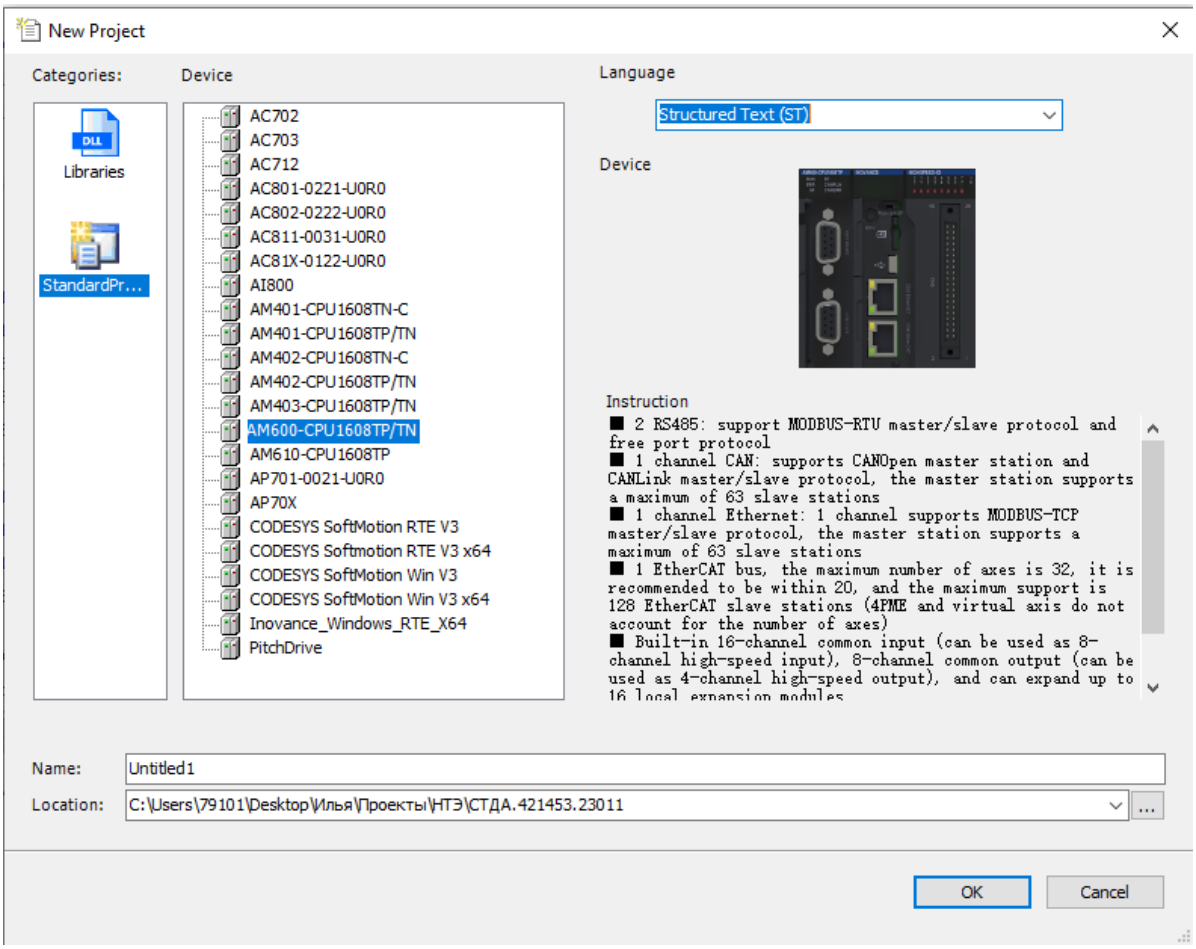


Рис. 2.2. Выбор шаблона

При создании проекта в соответствии с установленной конфигурацией формируется ряд объектов.

Программа работы ПЛК не зависит от конкретной модели контроллера. При программировании ПЛК созданная программа преобразуется в машинный код конкретного процессора. На этапе установления связи с контроллером программа должна взаимодействовать с конфигурацией конкретного ПЛК. Исходная информация о конфигурации ПЛК содержится в предварительных настройках целевой платформы контроллера. Настройки целевой платформы поставляются в виде

набора файлов, основным является Target-файл, который содержит информацию о ресурсах конкретного ПЛК, количестве/типе входов и выходов, интерфейсов, памяти, дополнительных устройств и т. д. Целевую платформу (контроллер) выбирают через вкладку «Устройство». В качестве исполнительного устройства может выступать как физический контроллер Inovance, так и виртуальный ПЛК.

Для ознакомления с возможностями среды используется виртуальный ПЛК. Это устройство реализует полнофункциональный ПЛК на персональном компьютере, позволяя работать с внешними устройствами, подключенными к Com-портам компьютера. В пункте «Устройство» укажите ПЛК AM600-CPU1608TP/TN.

Вкладка «Language» определяет язык реализации объекта «Программа PLC_PRG». PLC_PRG – это программа ПЛК, которая вызывается один раз за цикл управления.

CFC – графический язык программирования. Программный код представляет собой последовательность цепей, содержащих логическое или арифметическое выражение. Код выполняется по типу определенных шагов. Каждый шаг – это конкретные операции. Как только шаг выполнен – следует действие по передаче управления следующему шагу.

Программирование на языке CFC сводится к выбору библиотечных, готовых функциональных блоков, их позиционированию на экране, установке соединений между их входами и выходами, а также настройке параметров выбранных блоков.

ST (Structured Text) – это текстовый язык высокого уровня общего назначения, по синтаксису схожий с языком Pascal. Удобен для программ, включающих числовой анализ или сложные алгоритмы. Может использоваться в программах, в теле функции или функционального блока, а также для описания действия и перехода внутри элементов SFC.

Задав атрибуты проекта и кликнув по кнопке «OK», создают проект с указанными настройками.

Интерфейс InoProShop

Интерфейс InoProShop (рис. 2.3) представляет собой набор рабочих областей.

1. Панель управления представлена кнопками, разделами, дублирующими часто используемые команды среды.

2. Окно навигации проекта содержит структуру проекта: POU, типы данных, визуализации, ресурсы, библиотеки.

3. Рабочая область POU отображает редактор программной организационной единицы (POU), каждый программный компонент состоит из раздела объявлений и раздела кода. Для написания кода POU используется один из МЭК языков программирования – IL, ST, FBD, SFC, LD, CFC.

4. Панель инструментов содержит стандартные элементы программирования.

5. Панель сообщений аккумулирует данные компиляции, загрузки проекта.

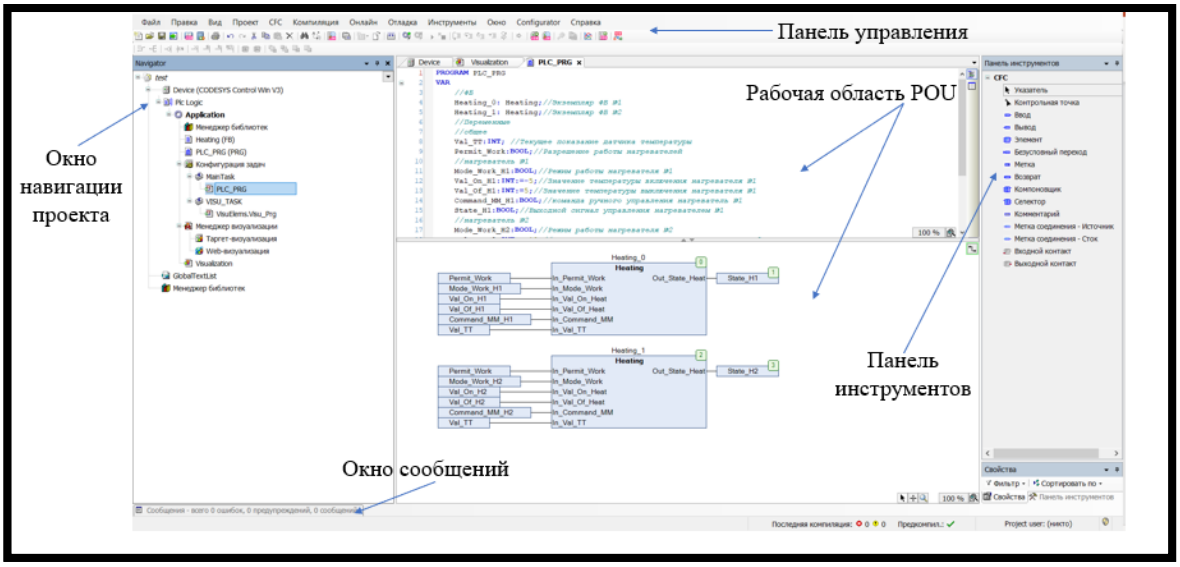


Рис. 2.3. Интерфейс программы InoProShop

Состав проекта

Проект представлен структурой (рис. 2.4), включающей в себя программируемое устройство (AM600-CPU1608TP/TN) и приложение (Application).

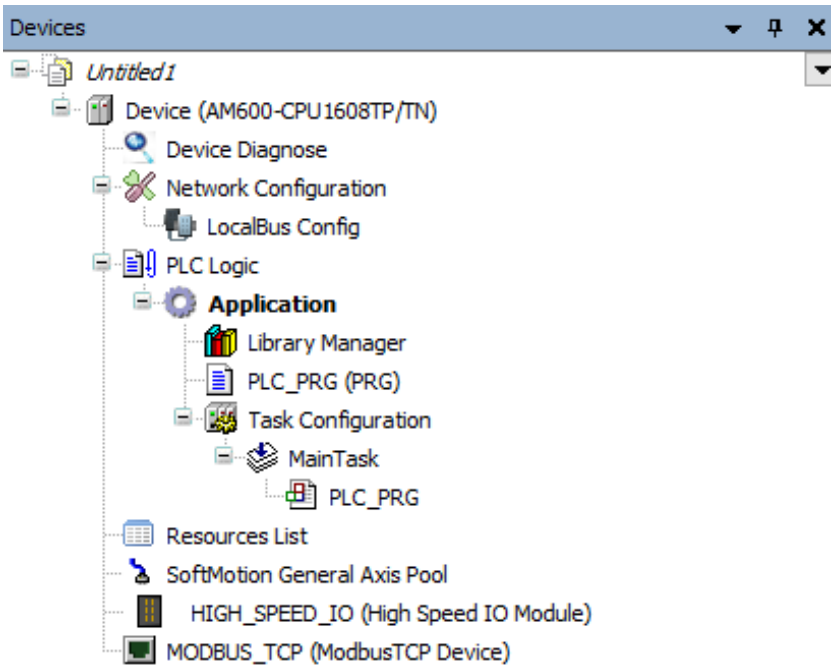


Рис. 2.4. Выбор структуры

«Application» представляет собой набор объектов, обеспечивающих работу ПЛК.

Объект «*Менеджер библиотек*» обеспечивает управление библиотеками.

Объект «*PLC_PRG*» – программная организационная единица (рис. 2.5), содержащая код программы. Двойной щелчок ЛКМ по объекту PLC_PRG открывает редактор программы, состоящий из двух областей. В области «Объявления» осуществляется инициализация переменных. В части «Тело программы» на основе инициализированных данных реализуется программный код на выбранном языке.

Рис. 2.5. Организационная программная единица

Объект *«Конфигурация задач»* служит для организации планирования вычислительного процесса ПЛК. Здесь формулируются задачи и настраиваются их параметры. При создании нового проекта неявно организуется задача, содержащая вызов одной программы PLC_PRG. Данная задача выполняется циклически.

Двойной щелчок по вложенному элементу «MainTask» (Задача) открывает атрибуты задачи.

Параметр «*Приоритет*» – система исполнения задач ПЛК построена по принципу вытесняющей многозадачности. Задача с высоким приоритетом вытесняет задачу с меньшим приоритетом. Нулевое значение соответствует наивысшему приоритету.

Параметр «*Тип*» определяет способ вызова задачи: через установленный интервал, свободный вызов, событие.

Параметр *«Сторожевой таймер»* обеспечивает защиту от зацикливания задачи. Если задача не возвращает управление дольше заданного времени, то «срабатывает» таймер и происходит вызов специального обработчика системного события.

В нижней части окна «MainTask» (рис. 2.6) располагается редактор, определяющий программы, вызываемые текущей задачей.

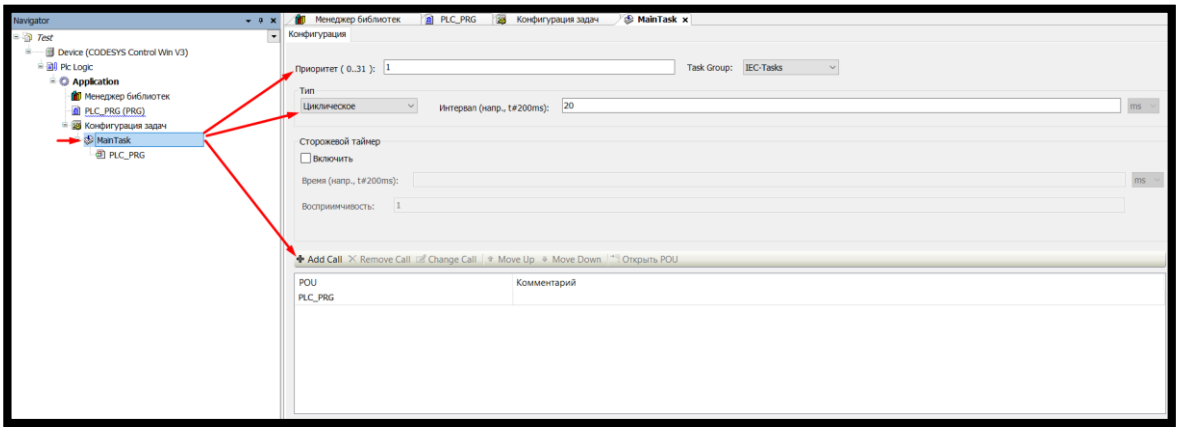


Рис. 2.6. MainTask

Объект «*Device Diagnose*» (рис. 2.7) обеспечивает диагностику ПЛК в онлайн-режиме. В нем отображаются ошибки и предупреждения всех устройств, подключенных к ПЛК, а также ошибки, допущенные при написании кода в программе.

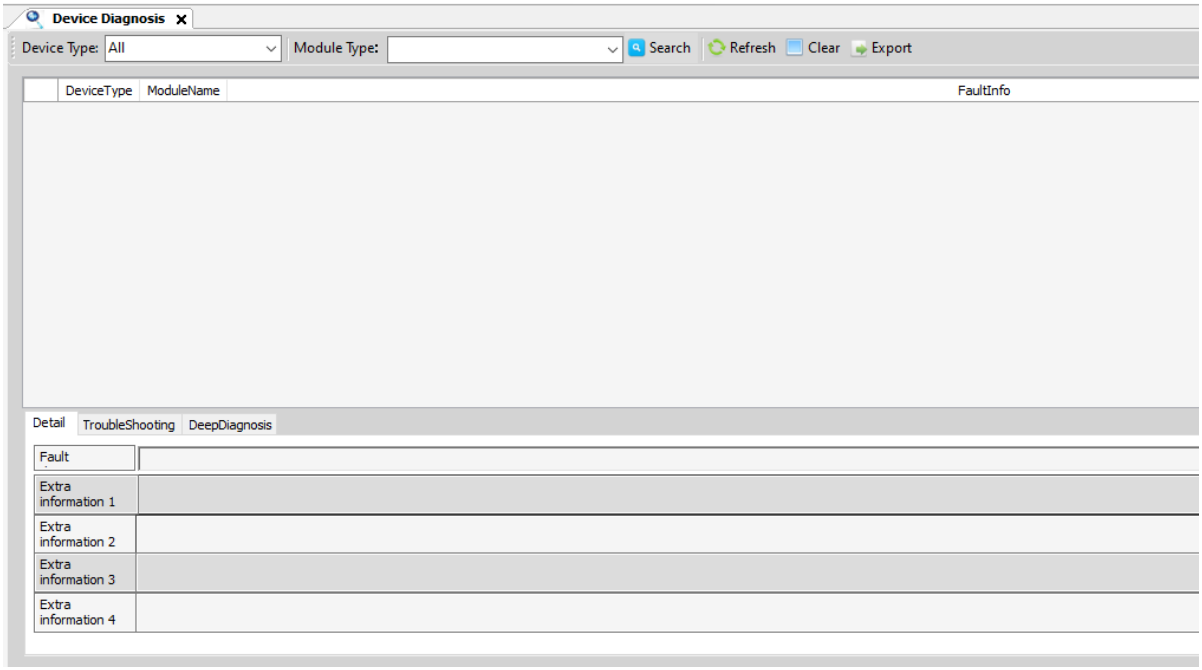


Рис. 2.7. Device Diagnose

Столбец *«Device Type»* отображает тип девайса, в котором есть ошибка или предупреждение, столбец *«Module Name»* – название модуля, столбец *«Fault Info»* – информацию по текущей ошибке или предупреждению.

Объект «*Network Configuration*» (рис. 2.8) обеспечивает выбор протокола для передачи информации внешним устройствам.

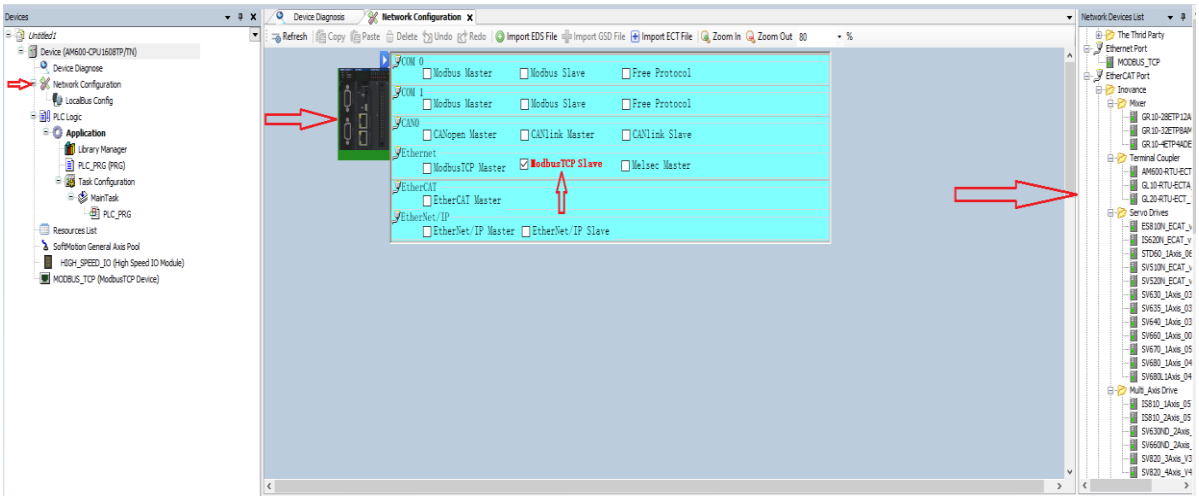


Рис. 2.8. Network Configuration

При нажатии на ПЛК во всплывающем окне отображаются различные протоколы для передачи информации внешним устройствам. По умолчанию в проекте создается связь по протоколу Modbus TCP.

В правой части окна находятся все удаленные устройства Inovance. Двойным щелчком ЛКМ их можно добавить в проект для задания параметров и программирования в проекте.

Объект «*Local Config*» необходим для выбора устройств, которые будут подключаться непосредственно на шину контроллера. В контроллере AM600-CPU1608TP/TN на локальную шину можно подключать до 16 модулей ввода/вывода.

Все модули ввода/вывода от Inovance представлены в правой части экрана (рис. 2.9). Двойным щелчком ЛКМ можно добавить модуль в проект. После этого добавленный модуль отобразится в дереве проекта.

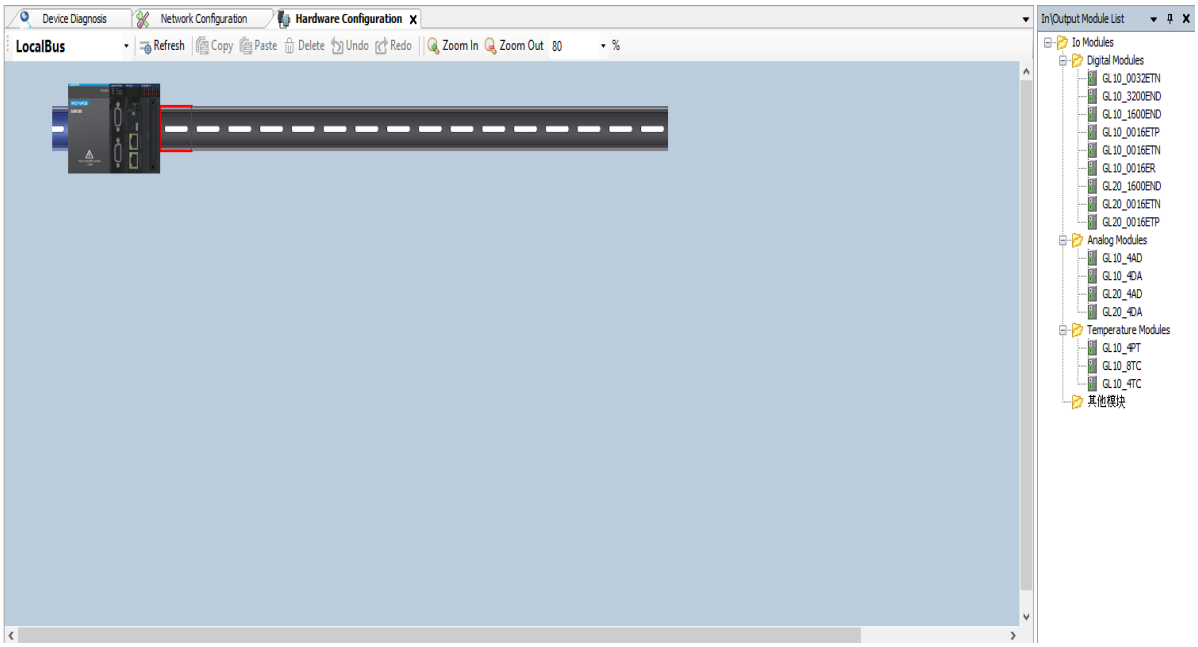


Рис. 2.9. LocalBus

Объект «*Resources List*» (рис. 2.10) показывает, насколько заполнена память контроллера существующим кодом и переменными, которые используются в нем.

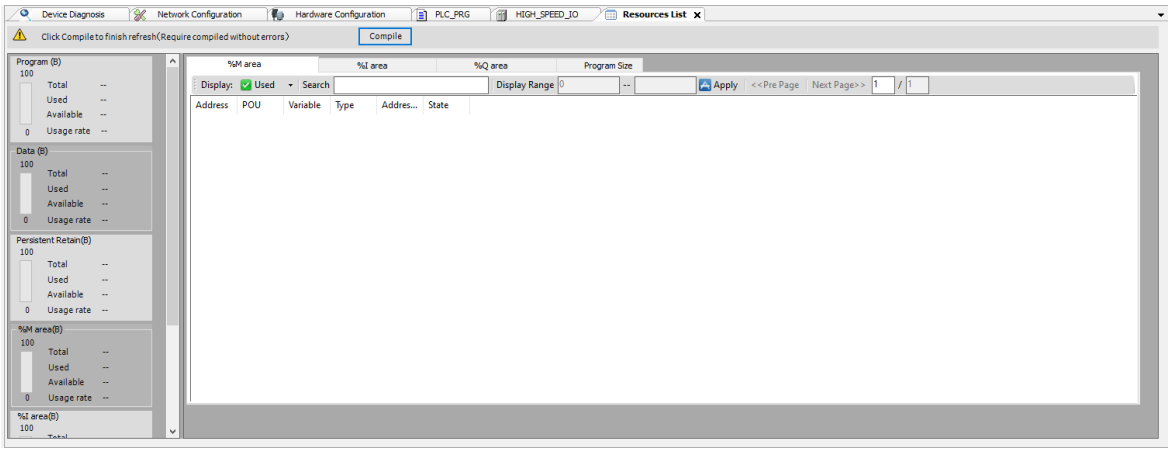


Рис. 2.10. Resources List

Строчка «*Program*» отображает размер программы, строчка «*Data*» – заполнение оперативной памяти, строчка «*Persistent Retain*» – заполнение энергонезависимых переменных, строчка «*M area, I area, Q area*» – размер переменных, используемых для связи с внешними устройствами.

Объект «*SoftMotion General Axis pool*» позволяет добавить в проект виртуальную ось или энкодер с использованием интерфейса SoftMotion.

Интерфейс привода SoftMotion – это стандартизированный интерфейс, который используется для подключения, настройки и адресации аппаратного обеспечения привода в рамках программы IEC. Подключая различное оборудование к одному интерфейсу, можно легко обмениваться информацией в ходе технологического процесса и повторно использовать программы IEC. Интерфейс подключает приводы к схеме ввода/вывода и отвечает за обновление и передачу требуемых данных о движении в систему управления приводом.

Интерфейс привода включает следующие компоненты:

1) описание устройств SoftMotion для их представления в дереве устройств;

2) библиотеки, на которые даны ссылки в описании устройства, – они расширяют или перегружают базовые функциональные блоки AXIS_REF_SM3 в соответствии с требованиями конкретных типов приводов;

3) библиотеки, содержащие функциональные блоки для ациклического чтения и записи данных для переноса стандартных функций драйвера полевой шины.

Для добавления оси необходимо нажать ПКМ на объект и выбрать «Add Device».

Во всплывающем окне можно выбрать для добавления в проект (рис. 2.11):

– виртуальный энкодер;

– виртуальную ось для управления по позиции;

– виртуальную ось для векторного управления по скорости.

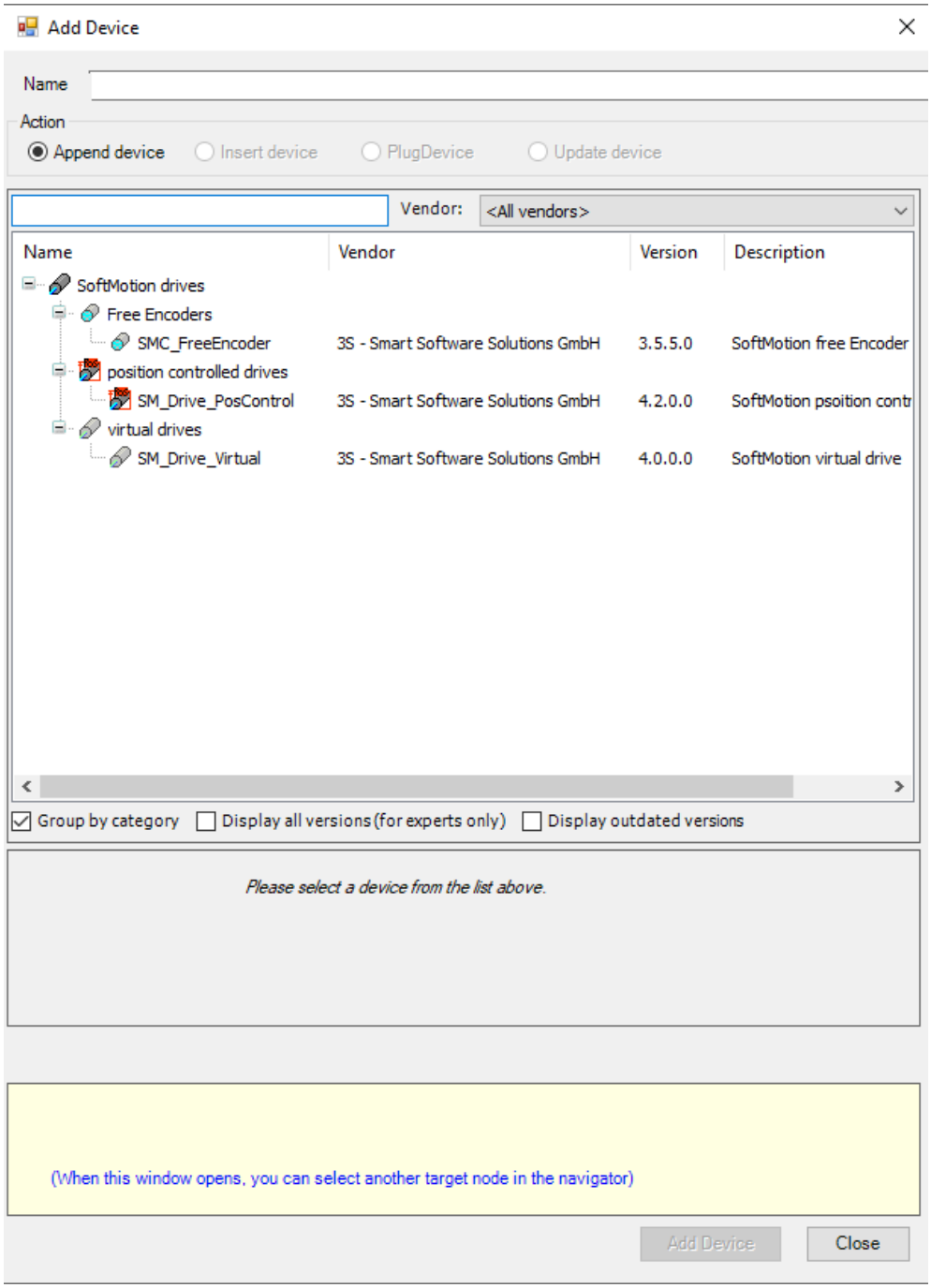


Рис. 2.11. Add Device

Объект «*HIGH_SPEED_IO*» (рис. 2.12) представляет собой конфигуратор высокоскоростных входов/выходов.

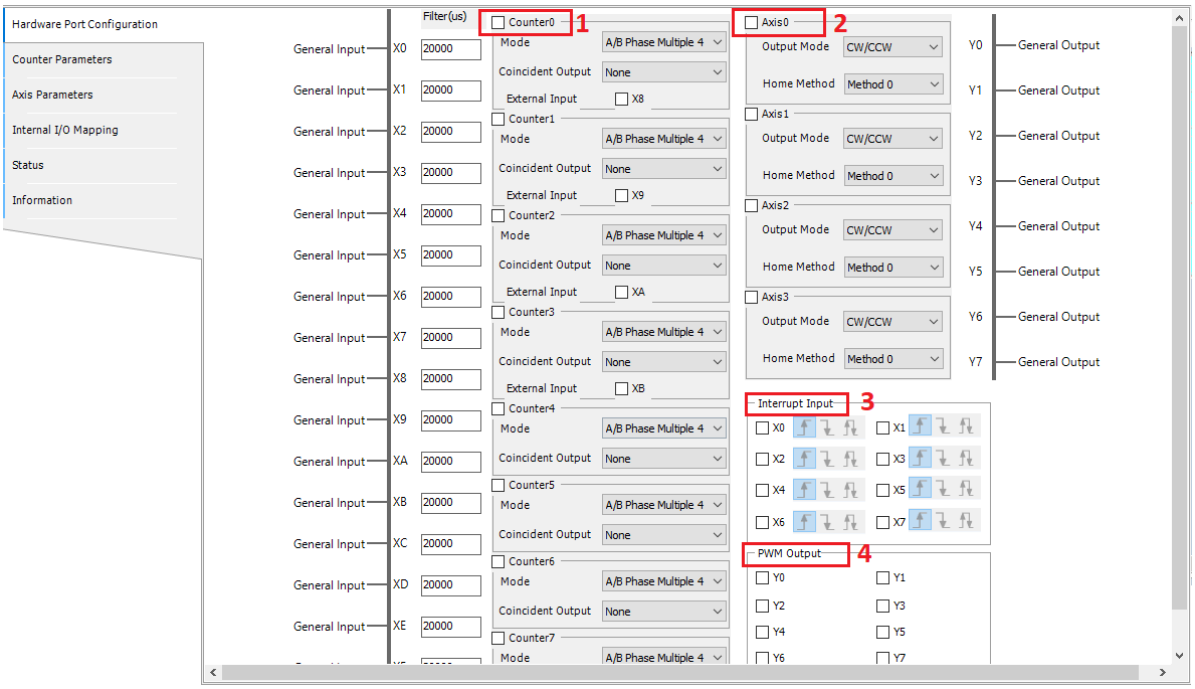


Рис. 2.12. Выбор дополнительных элементов

На основном экране необходимо выбрать, как будут использоваться высокоскоростные входы/выходы:

1. «Counter» – добавление счетчика (рис. 2.13). После выставления галочки в окне «Counter parameters» открывается возможность параметризировать счетчик.

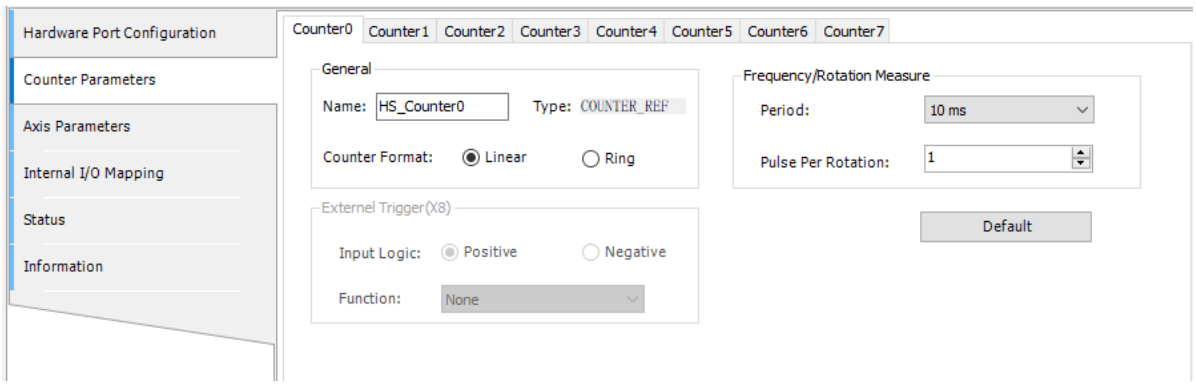


Рис. 2.13. Counter

2. «Axis» – добавление оси (рис. 2.14). После выставления галочки в окне «Axis parameters» открывается возможность задавать параметры оси. При подключении к высокоскоростным входам/выходам будет осуществляться импульсное управление приводом.

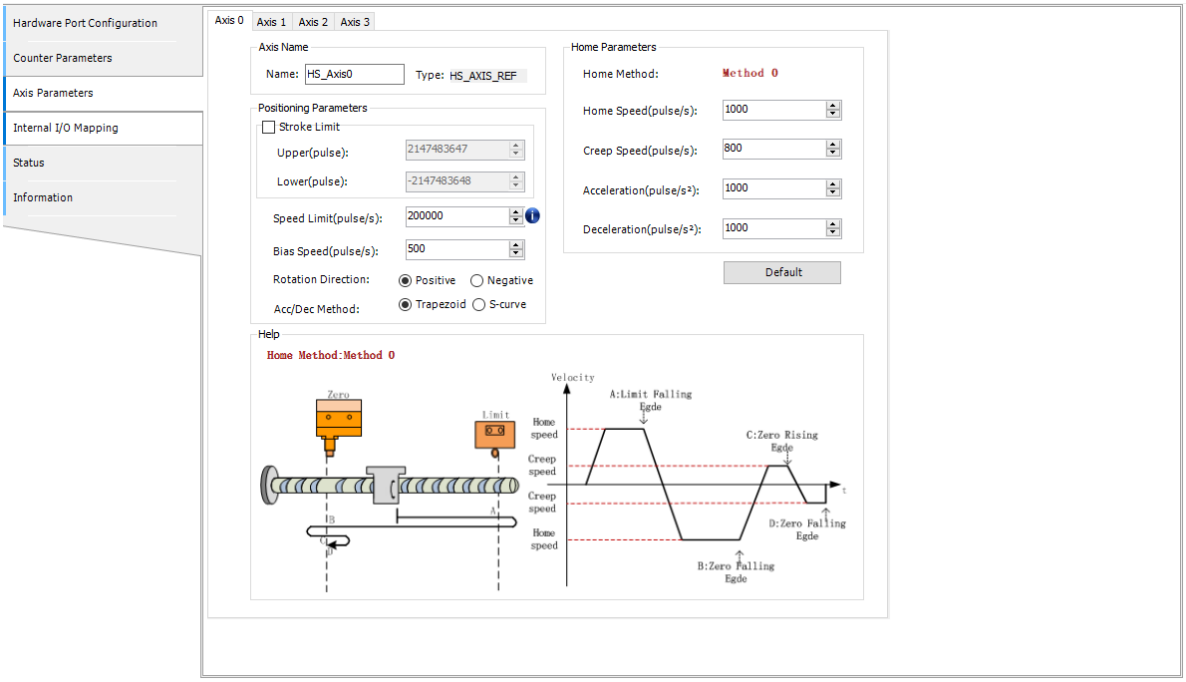


Рис. 2.14. Axis

Максимальное количество осей, подключенных к высокоскоростным входам/выходам, – четыре.

3. «Input/Output» – высокоскоростные дискретные входы/выходы (рис. 2.15). При выставлении галочки во вкладке «Internal I/O Mapping» можно добавлять программные переменные для считывания значений.

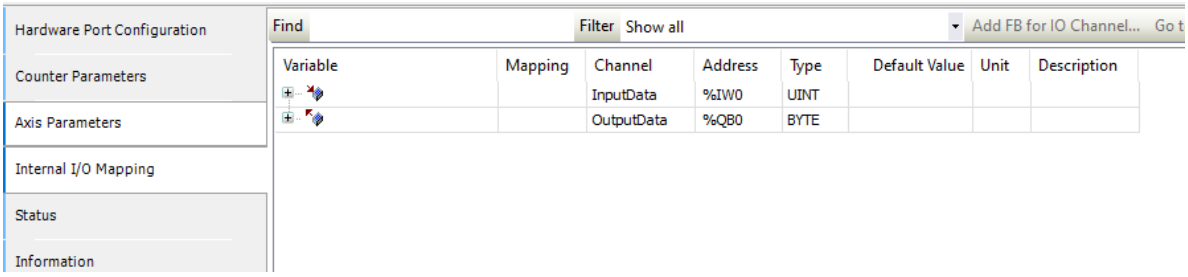


Рис. 2.15. Input/Output

Инструменты, расположенные сверху программы, представлены на рис. 2.16:

– Создание нового проекта;

– Открыть существующий проект;

– Сохранить проект;

– Распечатать (программный код, название и т. д.);

– Проверка правильности подключения модулей и удаленных устройств;

– Компиляция проекта. Проверка правильности написания программного кода, подключения устройств и др.;

– Загрузка проекта в ПЛК и запуск программы.

Рис. 2.16. Инструменты основной панели

Для включения режима симуляции (рис. 2.17), чтобы компьютер выступал вместо ПЛК, необходимо нажать вкладку «Online» и выбрать строку «Simulation».

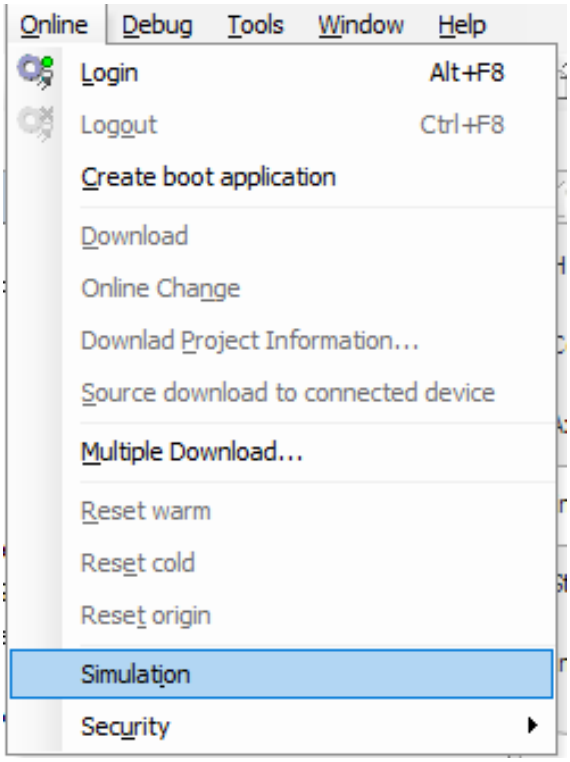


Рис. 2.17. Вкладка «Online»

Контрольные вопросы

1. Какие языки программирования представлены в контроллере?
2. Что такое рабочая область POU?
3. Из каких объектов состоит структура проекта?
4. Что такое PLC_PRG? Для чего он используется?
5. Что такое MainTask? Какие действия он выполняет?
6. С помощью какого объекта необходимо выбирать протоколы для связи с удаленными устройствами?
7. Что такое SoftMotion? Для чего он нужен?
8. Какое действие выполняют высокоскоростные входы/выходы? Какие устройства можно подключить к ним?

3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ПАНЕЛЕЙ INOVANCE

Программное обеспечение для панелей Inovance – основной инструмент для создания проектов и организации связи в ПЛК.

InoTouchPad – мощный визуальный редактор проектов для операторских панелей Inovance (рис. 3.1).

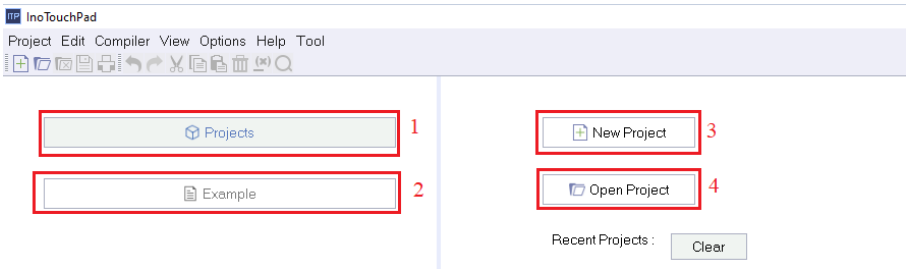


Рис. 3.1. Приветственное окно InoTouchPad

Двойной щелчок ЛКМ по иконке InoTouchPad открывает основной экран программы, где отображены:

1) проекты, которые в данный момент открыты;

2) примеры проектов от разработчика (как работать с кнопками, как добавлять элементы на экран и др.);

3) вкладка «Создать новый проект»;

4) вкладка «Открыть существующий проект».

После нажатия на иконку «Создать новый проект» на экране появляется всплывающее окно со всеми панелями от фирмы Inovance (рис. 3.2).

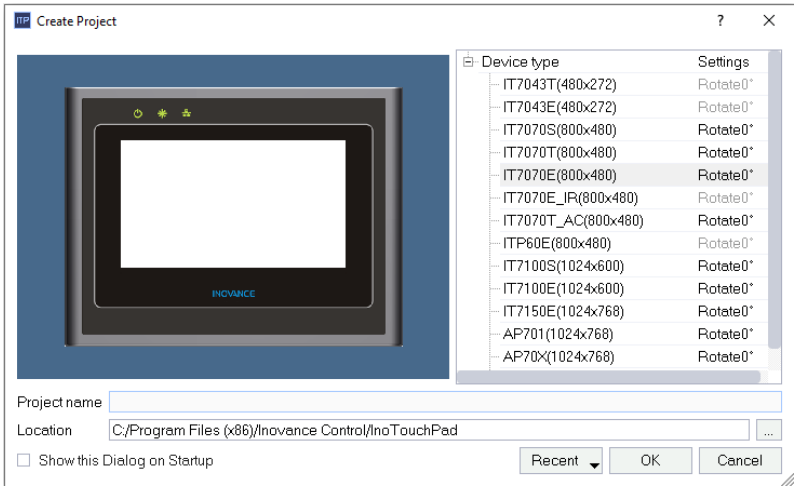


Рис. 3.2. Список панелей

Панели делятся на три категории:

7 дюймов – IT7070E

10 дюймов – IT7100E

15 дюймов – IT7150E

На учебном стенде установлена 7-дюймовая панель. Необходимо выбрать из списка панель IT7070E, задать название проекта и путь, куда будет сохранен проект, и нажать кнопку «ОК».

После введённой информации откроется проект с установленными по умолчанию различными инструментами (рис. 3.3).

1. Инструменты для работы с объектами, расположенными на экране, а также проверки на правильность, выбора языка, сохранения, открытия проекта.

2. Дерево проекта – на нем отображены экраны, настройки проекта, аварии и предупреждения, рецепты и другие процессы, которыми можно оперировать в проекте.

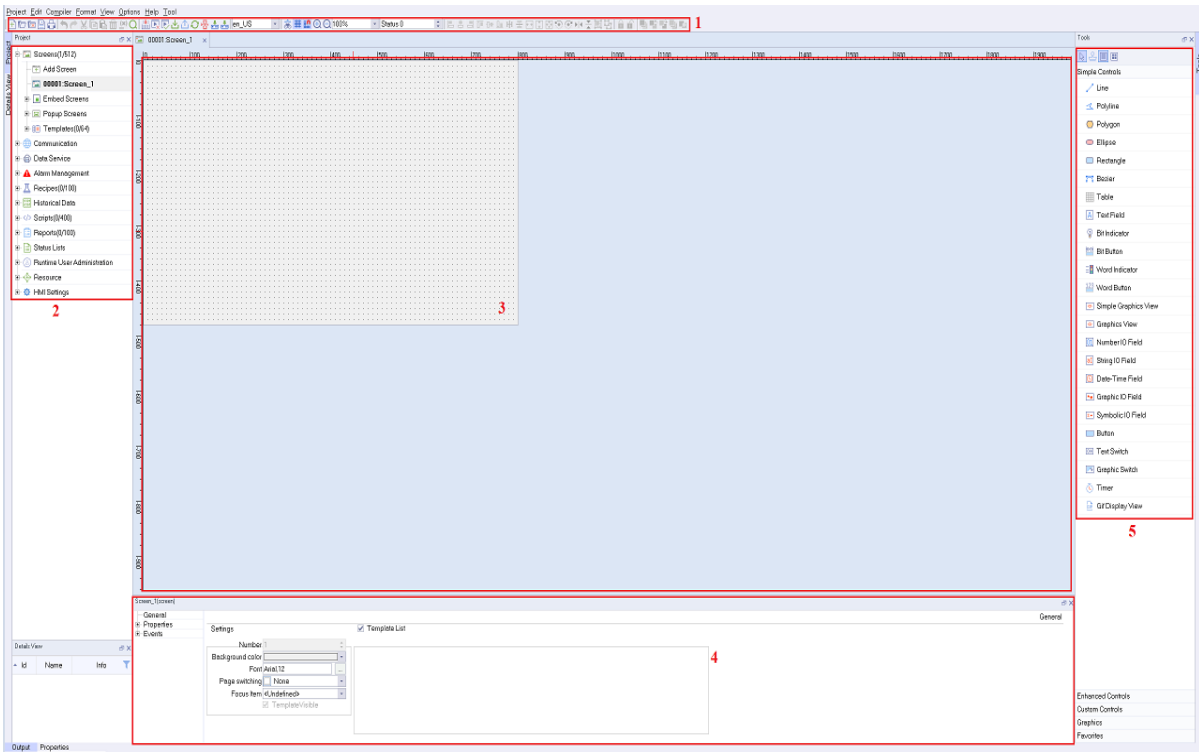


Рис. 3.3. Общий вид проекта

3. Область проекта – на области отображен экран панели в соотношении 1:1. На осях абсцисс и ординат обозначены пиксели панели.

4. Характеристики добавленных объектов. У каждого объекта есть свои уникальные свойства, которые можно параметризировать.

5. Окно для добавления объектов на экран (рис. 3.4). Для того чтобы добавить объект, необходимо ЛКМ зажать и перенести на экран выбранный объект.

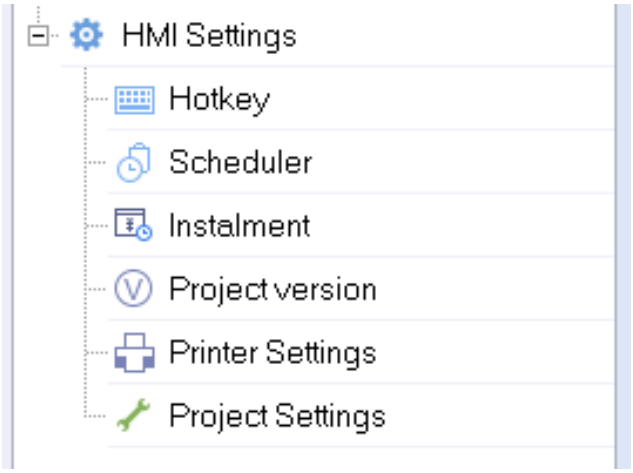


Рис. 3.4. Настройки InoTouchPad

Рассмотрим базовые настройки проекта. Они состоят из следующих элементов:

1. HotKey – бинд клавиши клавиатуры для активации какого-либо действия. Используется, если к панели подключена клавиатура.

2. Scheduler – счетчик. Применяется для активации какого-либо события или скрипта. Скрипт – маленькая программа, написанная на языке JavaScript. С помощью скрипта можно управлять перемещением объектов, изменением цвета и др.

3. Project Version – кнопка необходима для сохранения текущей версии программы. Данная опция позволяет сделать до 15 сохранений проекта, включая все его элементы. Если пользователь захочет вернуть все к какой-либо версии проекта, ему достаточно будет нажать одну кнопку.

4. Printer Settings – настройки принтера. К панели можно подключить принтер для печати различных событий и аварий.

5. Project Settings – основные параметры запуска.

Далее проанализированы расширенные настройки проекта (рис. 3.5):

1. HMI Settings – поле, отображающее тип панели, активного пользователя, стартовое окно, выбранный язык и стиль, примененный к проекту.

2. Screen Saver – поле, отображающее время ухода панели в спящий режим и на стартовый экран.

3. Security Settings – защита паролем. Если установить данную опцию, то при загрузке или выгрузке проекта из панели будет требоваться пароль.

4. Other Settings – настройки резистивного экрана, отображения курсора, звук и др.

5. Alarm Settings – настройки аварий (рис. 3.6). Отображение аварий, звук при появлении предупреждения, всплывающее окно с отображением аварий и др.

6. IP Settings – выставление IP панели.

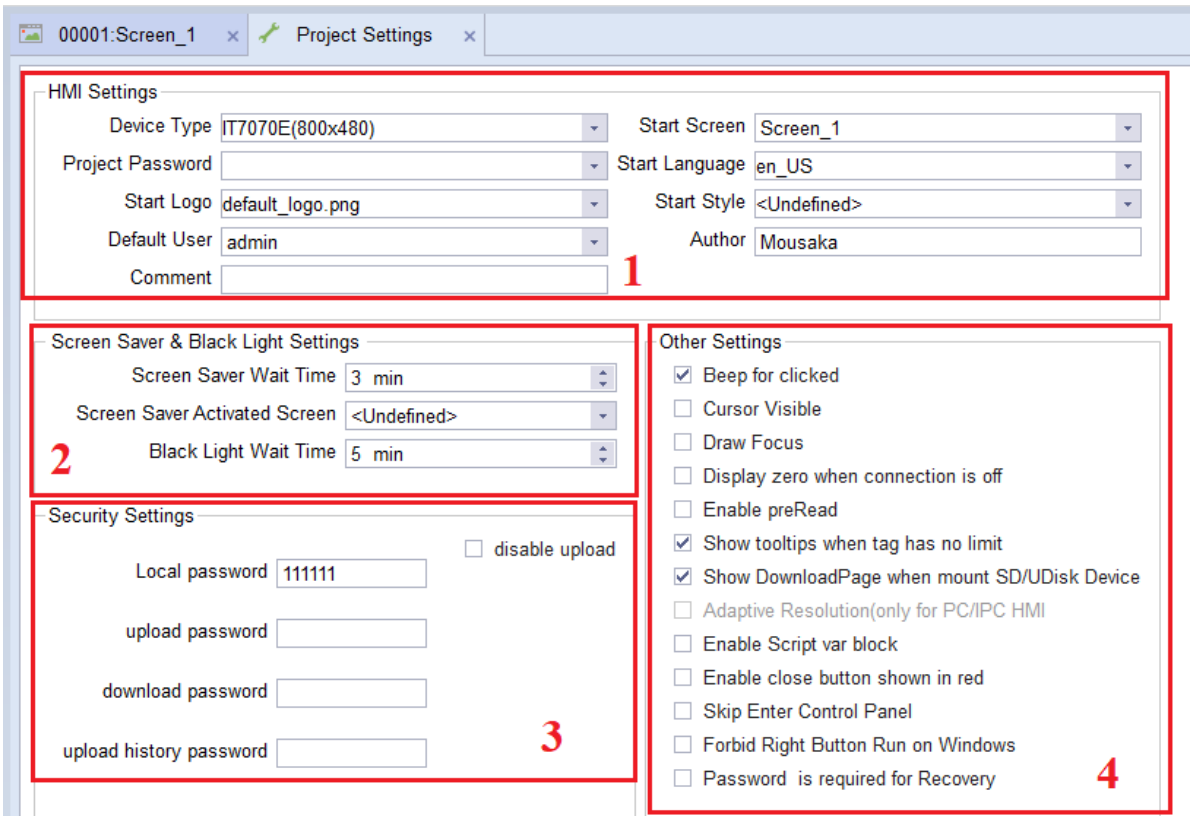


Рис. 3.5. Настройки проекта (расширенная версия)

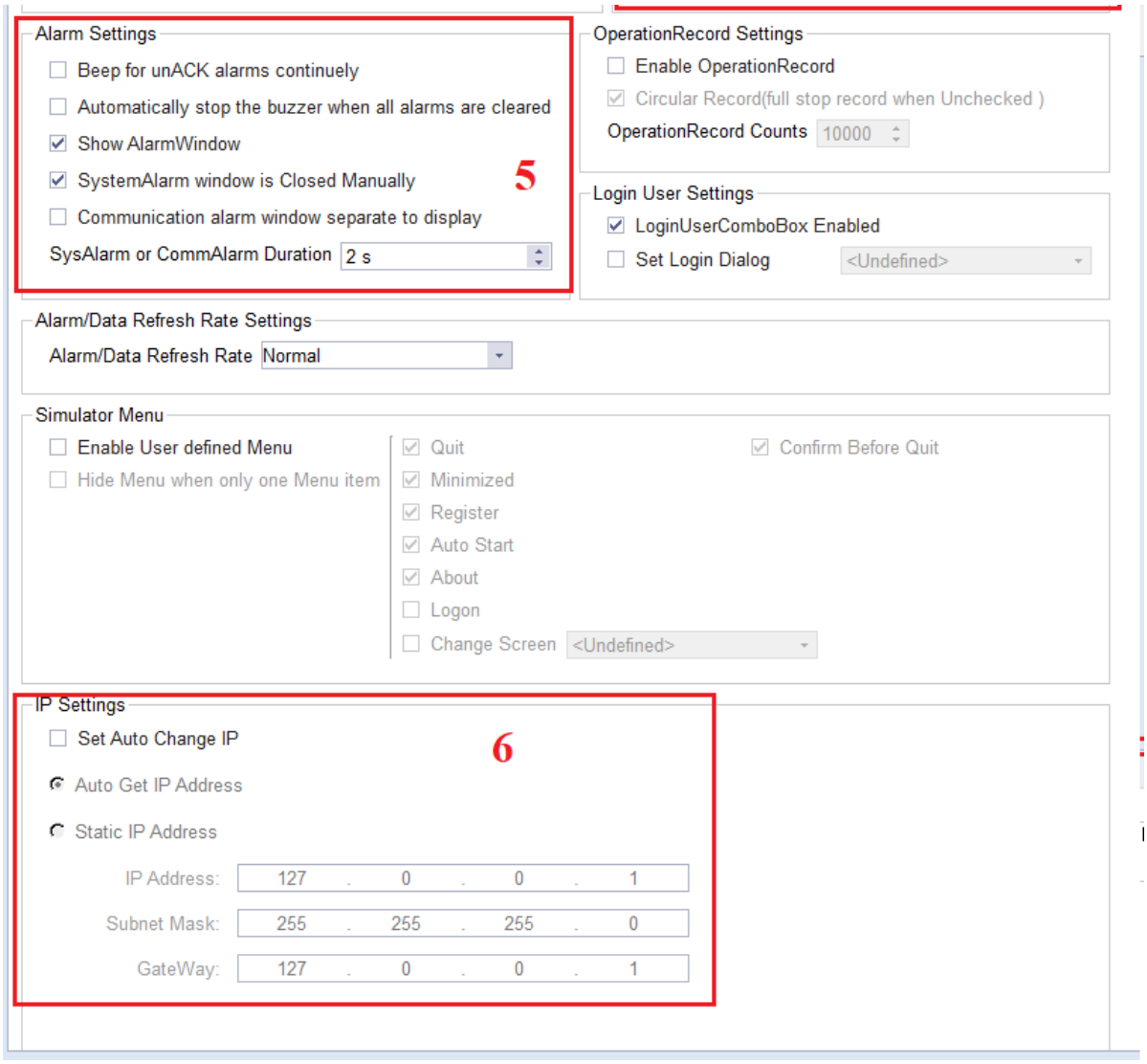


Рис. 3.6. Alarm Settings и IP Settings

Добавление экранов происходит в дереве проекта во вкладке «Screens» (рис. 3.7):

1. Screens – экраны, которые занимают всю область панели.

2. Embed Screens – экраны, которые буду открываться при достижении какого-либо события.

3. Popup Screens – всплывающие экраны. Маленькие окна, которые можно открывать с помощью кнопок, событий и т. д.

В проект можно добавить до 512 экранов.

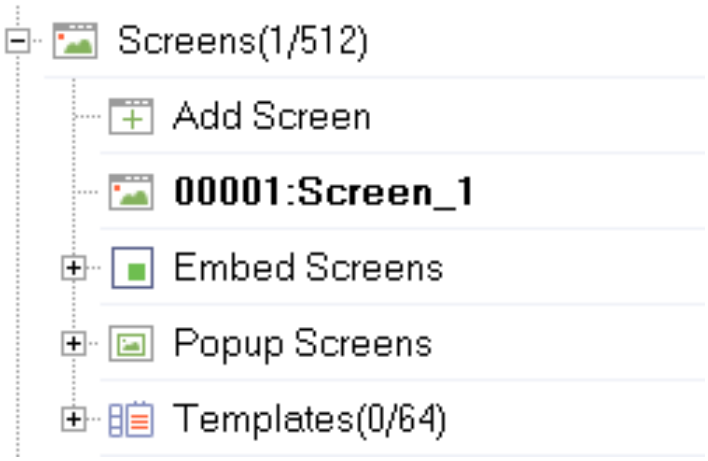


Рис. 3.7. Screens

Вкладка «Communication» (рис. 3.8) необходима для организации связи. Здесь настраиваются протоколы по передаче данных между панелью и внешними устройствами, а также во вкладке «Tag_Group» создаются теги для дальнейшего использования их в проекте. Практически каждый добавленный на экран элемент требует привязки тега.

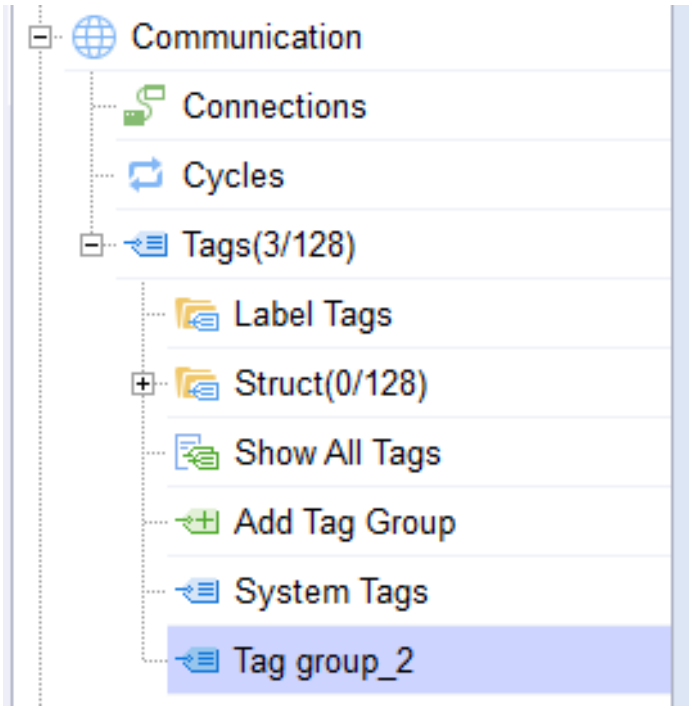


Рис. 3.8. Communication

Для добавления тега (рис. 3.9) необходимо нажать на «плюс», расположенный в верхнем левом углу вкладки.

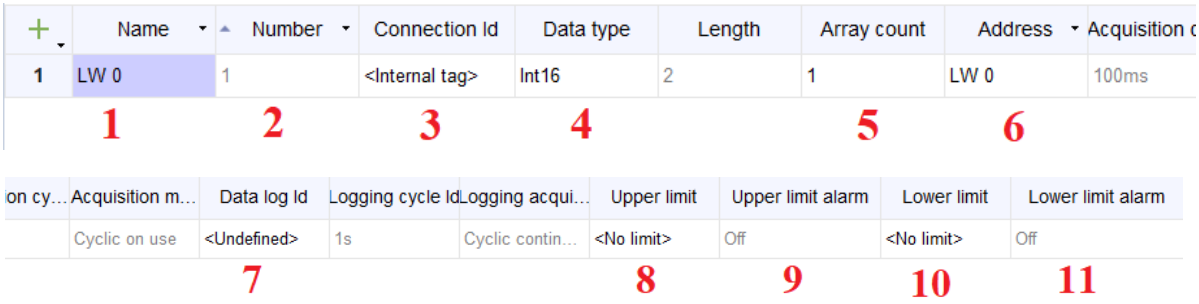


Рис. 3.9. Конфигурация тега

После добавления тег следует сконфигурировать:

1. «Name» – название тега.
2. «Number» – номер тега.
3. «Connection id» – отображение протокола передачи данных, взаимодействующего с внешними устройствами.
4. «Data type» – тип переменной.
5. «Array count» – какую область пространства будет занимать тег (по умолчанию – 2 байта).
6. «Address» – адрес тега. Адреса бывают:

LW – не энергонезависимая память, сохраняется в оперативной памяти. После выключения панели данные стираются.

RW – энергонезависимая память. После выключения панели данные, которые были в RW памяти, сохраняются.

7. «Data log id» – выбор памяти для сохранения (локальная память, SD карта, USB).
8. «Upper limit» – верхняя граница тега.
9. «Upper limit alarm» – верхняя граница тега для активации аварии.
10. «Lower limit» – нижняя граница тега.
11. «Lower limit alarm» – нижняя граница тега для активации аварии.

Вкладка «Data Service» представлена на рис. 3.10.

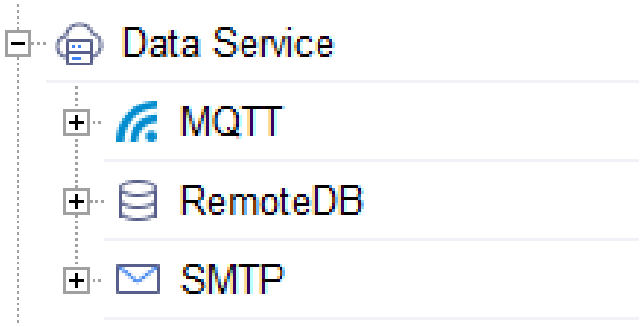


Рис. 3.10. Data Service

Для передачи данных в облачный сервис используют облачные технологии, через которые можно подключаться к панели через Wi-Fi (рис. 3.11).

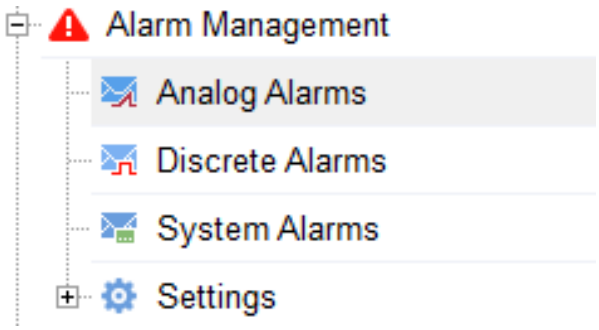


Рис. 3.11. Alarm Management

Во вкладке «Alarm Management» создаются аварии и предупреждения, которые будут участвовать в проекте (рис. 3.12).

	Text	Number	Class	Trigger tag	Trigger mode	Limit
1	analog_1	1	Errors	<Undefined>	>	<No limit>

Рис. 3.12. Аналоговые аварии

Основные параметры для конфигурации:

Text – название аварии

Number – номер аварии

Class – выбор типа сообщения (аварии, предупреждения, системные ошибки)

Trigger tag – тег, по которому будет активироваться сообщение

Trigger mode – выбор, по какому из вариантов будет соблюдаться условие (<,>, = и т. д.)

Limit – выбор границы активации сообщения (может быть как тегом, так и числом)

С помощью вкладки «Recipe» в проект добавляют рецепты (рис. 3.13).

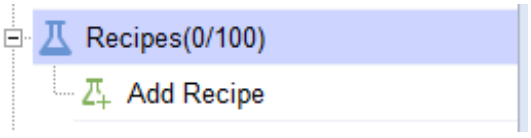


Рис. 3.13. Recipe

Рецепт – это набор параметров и их значений, дающий информацию, необходимую для производства продукта или управления процессом. Могут быть созданы различные определения рецептов, например колечек и печенья. Определение рецептов печенья может содержать много различных рецептов (печенье с шоколадной крошкой, сахарное печенье и т. д.).

Благодаря вкладке «Historical Data» создаются области памяти для сохранения переменных, аварий и предупреждений (рис. 3.14).

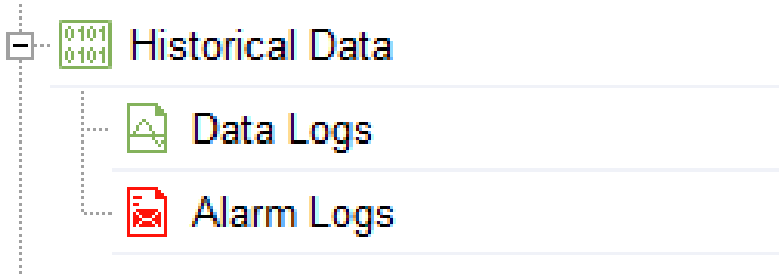


Рис. 3.14. Historical Data

Строка памяти состоит из названия, номера, количества записей, которые будут сохранены в файл (максимум на локальную память можно записать 100 000 килобит, на SD карту – до 5 000 000 килобит за один файл, максимальное количество файлов для записи – 64), пути, куда будут записываться данные.

Вкладка «Scripts» (рис. 3.15) добавляет в проект скрипт, который можно использовать для активации событий, записи различных текстов в поля ввода/вывода, управления стилями объектов и т. д.

Скрипт пишется на языке программирования JavaScript.

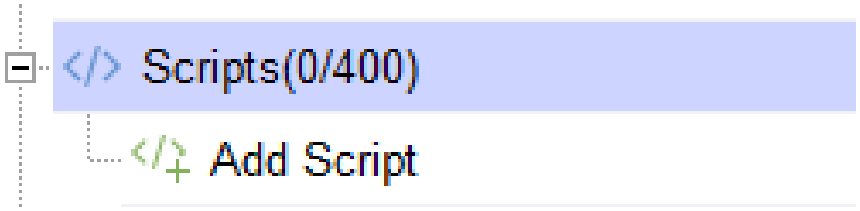


Рис. 3.15. Scripts

Вкладка «User Administration» (рис. 3.16) необходима для составления уровня доступа в проекте.

Уровень доступа – это набор разрешенных операций (например, загрузка файлов, редактирование страниц и другие), привязанный к пользователям и группам пользователей. Уровни доступа применяются в системе управления пользователями, определяются администратором, который может создавать или изменять их.

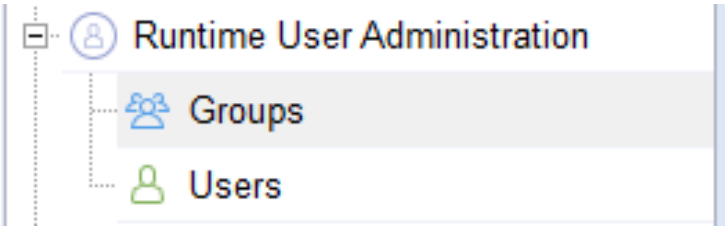


Рис. 3.16. User Administration

Контрольные вопросы

1. Какие вкладки входят в дерево проекта?
2. Каковы основные настройки проекта?
3. Что такое тег?
4. Что такое PopUp и как им управлять?
5. Для чего нужны рецепты в проекте?
6. Что такое Scheduler?

4. ПРАКТИКУМ

Лабораторная работа № 1

ЗАПУСК ПРОЕКТА И ПЕРЕДАЧА ДАННЫХ В ПЛК

Цель работы: освоить способы подключения панели оператора к контроллеру Inovance.

Задания

1. Создать оригинальный интерфейс для панели оператора.
2. Связать панель оператора с контроллером по OPC UA Server.
3. Связать панель оператора с контроллером по Modbus TCP Protocol.
4. Связать панель оператора с контроллером по Modbus TCP Protocol HMI Slave.

Ход работы

Создание интерфейса панели оператора

Для примера работы проекта создадим два экрана, шаблон и кнопки для переключения (рис. 4.1).

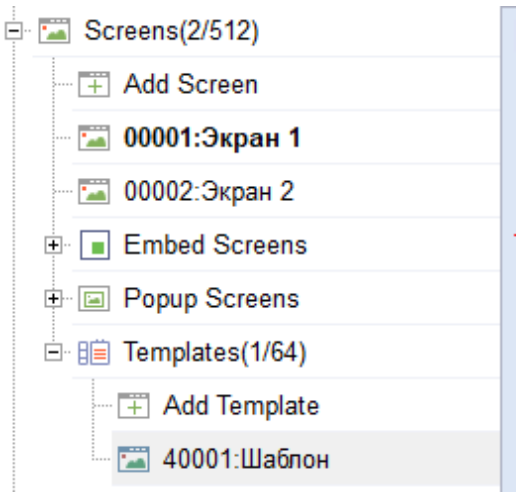


Рис. 4.1. Создание экранов

После добавления шаблона необходимо выбрать, где он будет отображаться (рис. 4.2).

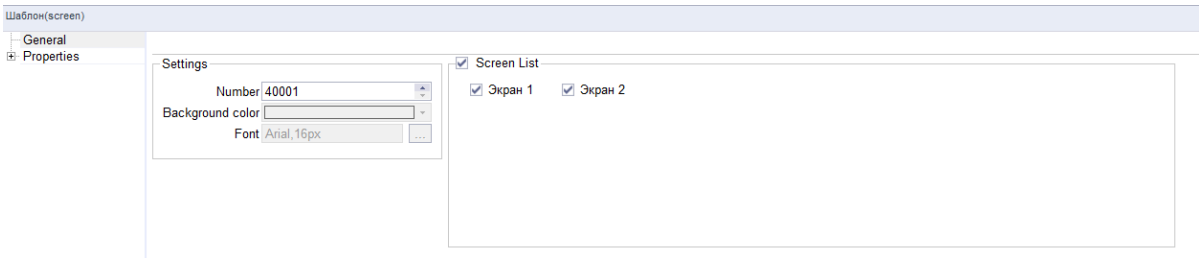


Рис. 4.2. Выбор места для отображения

В основных настройках выберите созданные экраны. Это значит, что созданный шаблон с двумя кнопками будет отображаться на всех экранах.

Для добавления кнопок на экран воспользуйтесь инструментами в правой части экрана. Выберите инструмент Button (рис. 4.3) и перенесите его на шаблон.

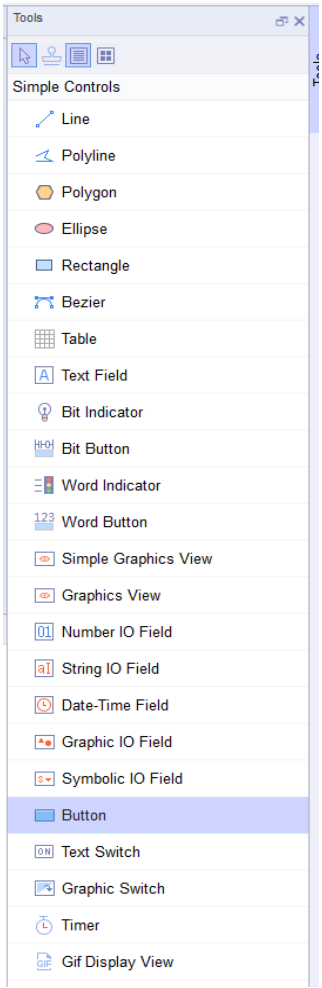


Рис. 4.3. Инструмент Button

Если нажать на элемент на экране «Кнопка», то в нижней части экрана появится поле со списком действий (рис. 4.4).

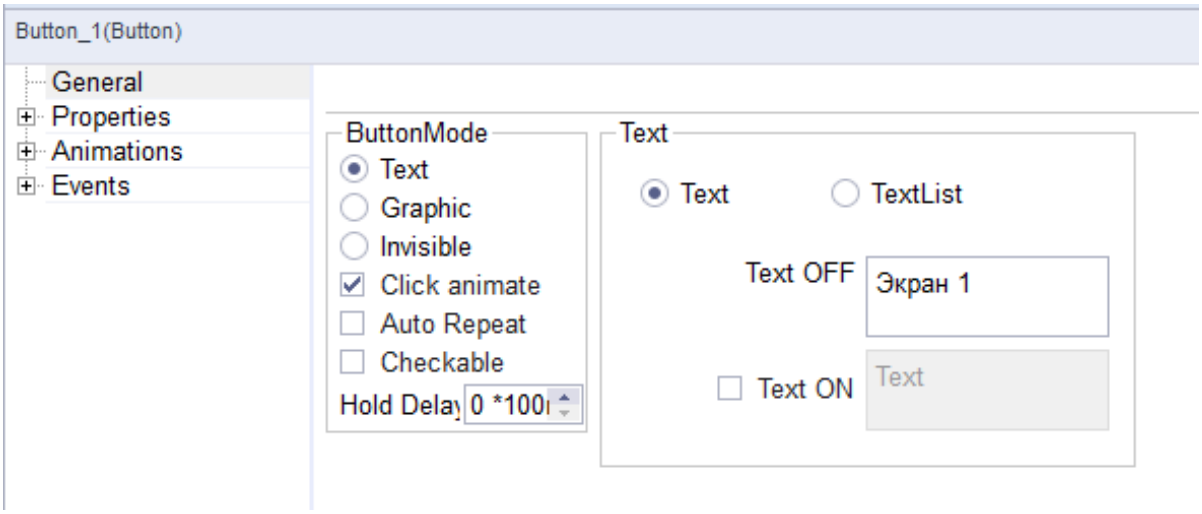


Рис. 4.4. Инструмент Button, вкладка General

Вкладка General у каждого элемента уникальная.

Вкладка Properties (рис. 4.5) служит для изменения цвета элемента, положения, стиля, уровня доступа, прозрачности.

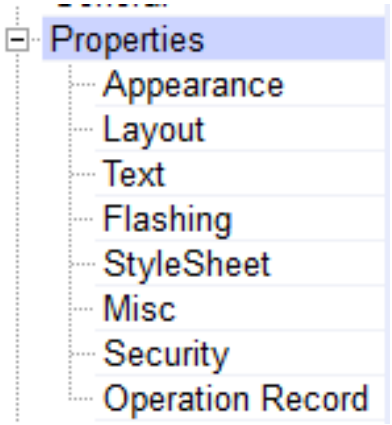


Рис. 4.5. Вкладка Properties

Вкладка Animation (рис. 4.6) служит для различной анимации, активируемой по тегу, например: если добавить тег и задать, что при целочисленном значении от 0 до 100 кнопка будет красной, а как только значение перейдет в 101, кнопка станет зеленой.

Enable objects – активация или деактивация элемента на восприятие касаний.

Visibility – видимость элемента, активация происходит по тегу.

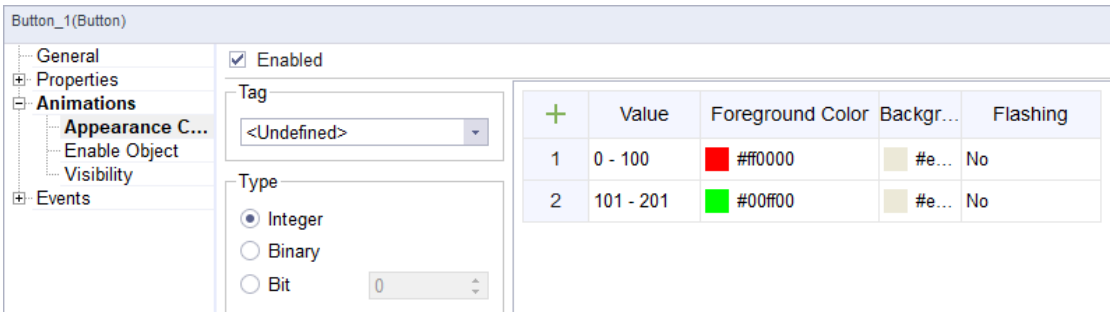


Рис. 4.6. Вкладка Animation

Вкладка Events (рис. 4.7) – события, которые будут активироваться при нажатии на кнопку.

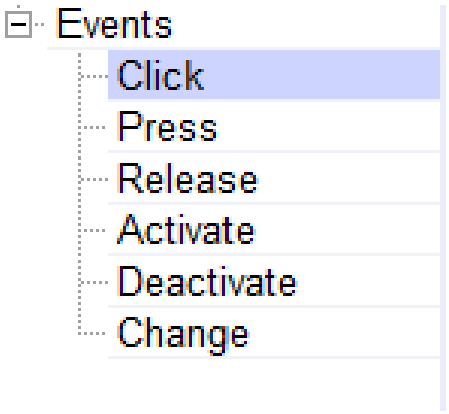


Рис. 4.7. Вкладка Events

Необходимо выбрать, на что будет срабатывать событие: на щелчок по кнопке, когда кнопка зажата, по изменению или по отжатию кнопки.

Выберите режим «По щелчку» (рис. 4.8), после чего откроется список с выбором действий:

Calculation – задает целочисленные значения в тег, инвертирует число.

Edit bits – задает биты в тег, инвертирует биты.

Screens – вкладка для управления экранами: переключение, открытие, закрытие, показ предыдущего экрана.

User administration – вкладка уровня доступа. Функция сохраняет название, пароль, группу текущего пользователя в тег, а также открывает окно для ввода логина и пароля.

Hmi Date Time – установка формата и даты на внутренних часах.

Settings – настройки панели (спящий режим, смена языка и т. д.).

Communication – выставление протокола, задание IP адреса и т. д.

Data Service – облачные технологии.

Print – настройка принтера.

Alarms – управление предупреждениями и авариями.

Logs – выбор, куда будут сохраняться теги и аварии.

Recipes – работа с рецептами, отправка в ПЛК, запоминание в панель и др.

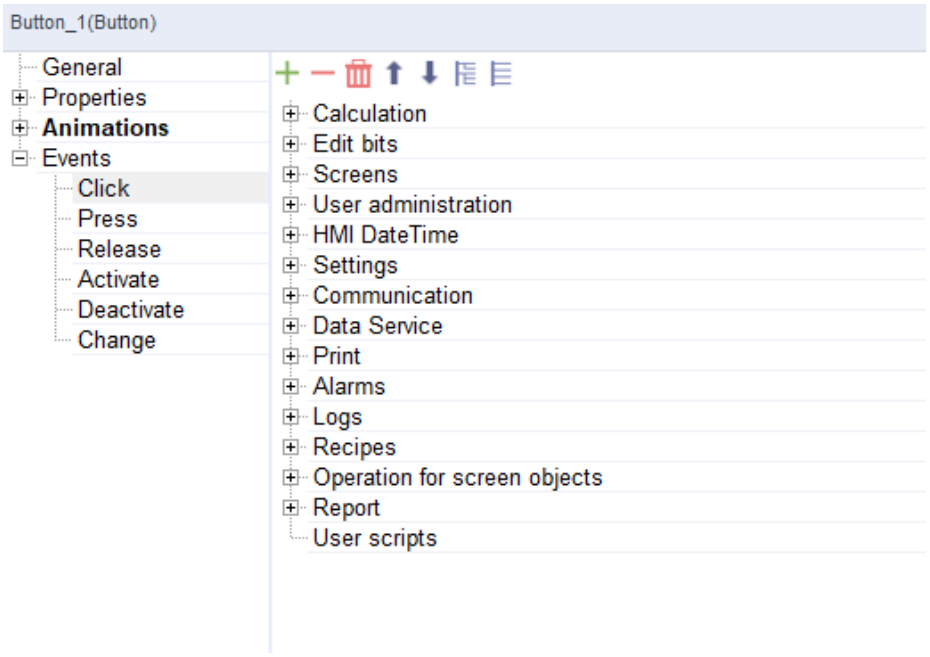


Рис. 4.8. Вкладка Events – Click

После того как рассмотрены настройки, необходимо на основном поле задать текст (рис. 4.9) и поменять цвет кнопки.

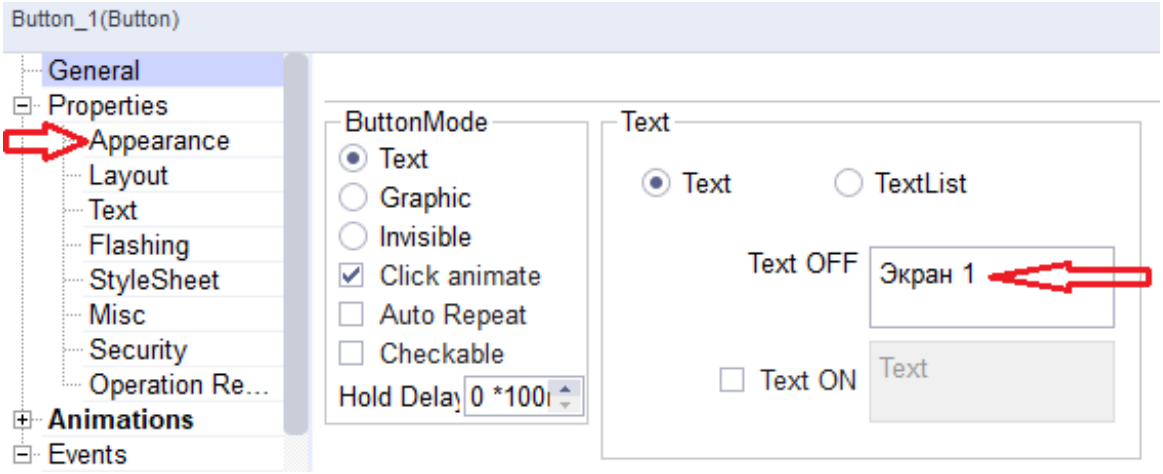


Рис. 4.9. Смена названия

Далее зайдите в Events и выберите Screens. Для переключения понадобится команда Activate Screen (рис. 4.10). Выберите первый экран для первой кнопки. Те же самые действия следует повторить для второго экрана.

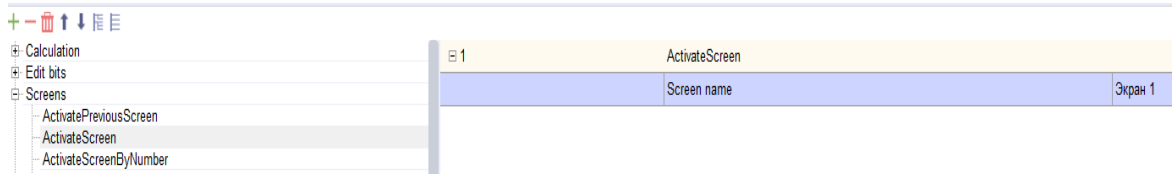


Рис. 4.10. Команда Activate Screen

Создайте две битовые кнопки и измените цвета у поля ввода/вывода. На второй экран добавьте Number io field и Bit button.

Главные настройки для битовой кнопки представлены на рис. 4.11.

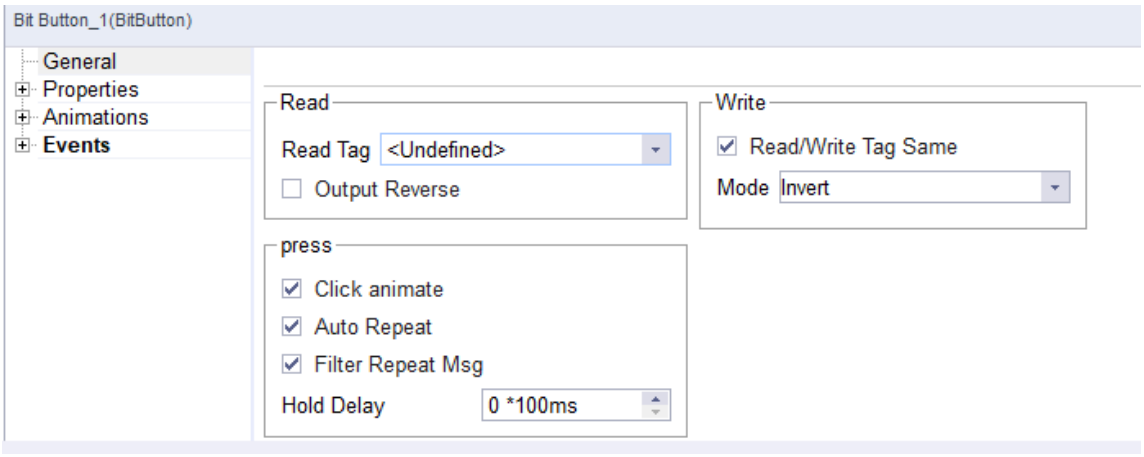


Рис. 4.11. Пример работы с 1-битовыми кнопками

Создайте тег (рис. 4.12) и в событиях кнопки задайте «Установить число в тег».

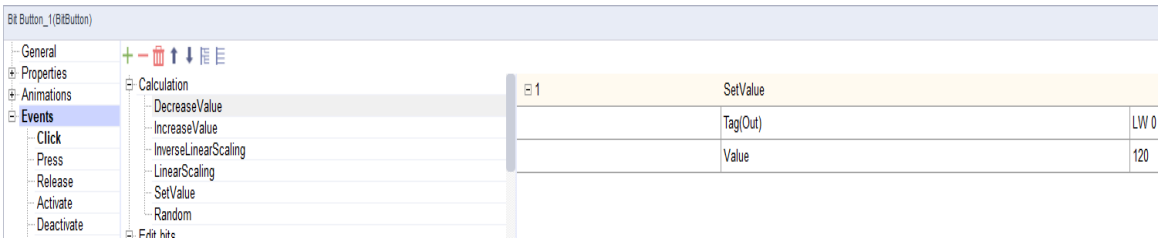


Рис. 4.12. Создание тега и установление значений кнопки

Далее выберите экран ввода/вывода, затем основные настройки и поле для изменения цвета.

Создайте первую битовую кнопку со значением 120 для активации красного цвета, затем – вторую кнопку и задайте значение 10 для активации зеленого цвета (рис. 4.13).

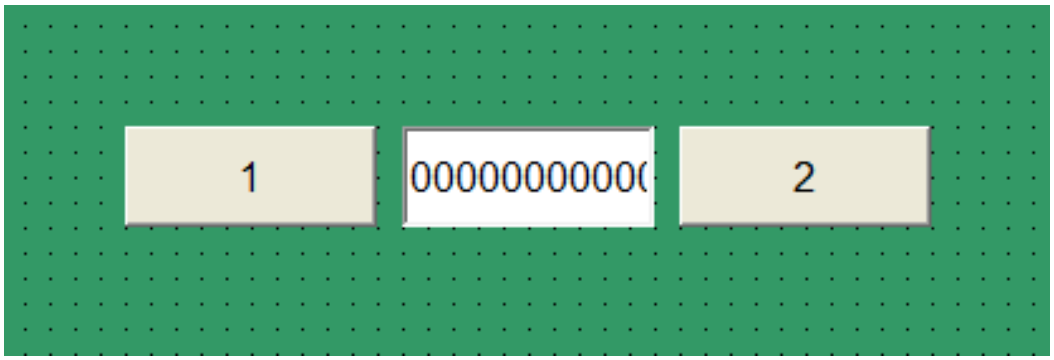


Рис. 4.13. Пример визуализации кнопки на экране

Для запуска проекта необходимо проверить правильность заполнения полей. Воспользуйтесь инструментами сверху (рис. 4.14).

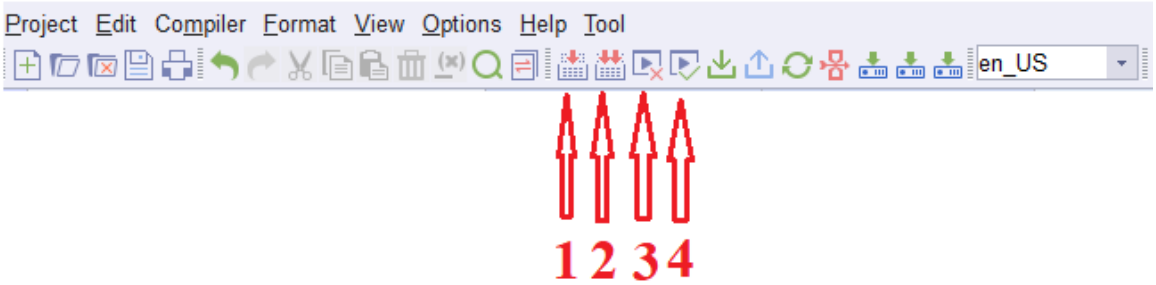


Рис. 4.14. Симуляция работы/проверка проекта

На рис. 4.14 цифрами обозначены следующие инструменты:

1) проверка всего проекта;

2) быстрая проверка;

3) запуск проекта в режиме офлайн (на компьютере будет отображаться панель с графикой, без соединения);

4) запуск в режиме онлайн (если панель подключена к контроллеру, то компьютер заменяет ее и все соединения идут с компьютера на ПЛК).

Нажмите на офлайн-запуск.

На мониторе отобразилась панель. Для переключения по экранам следует нажать «Экран 1» или «Экран 2» (рис. 4.15).

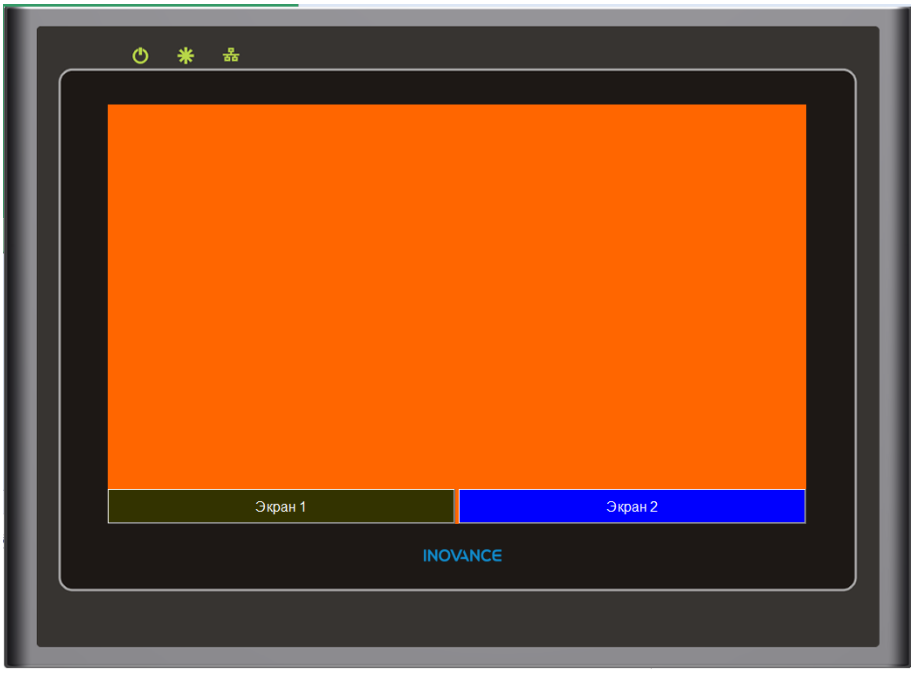


Рис. 4.15. Панель «Экран 1»

На втором экране, если нажать на кнопку 1, поле ввода/вывода поменяет цвет (рис. 4.16).

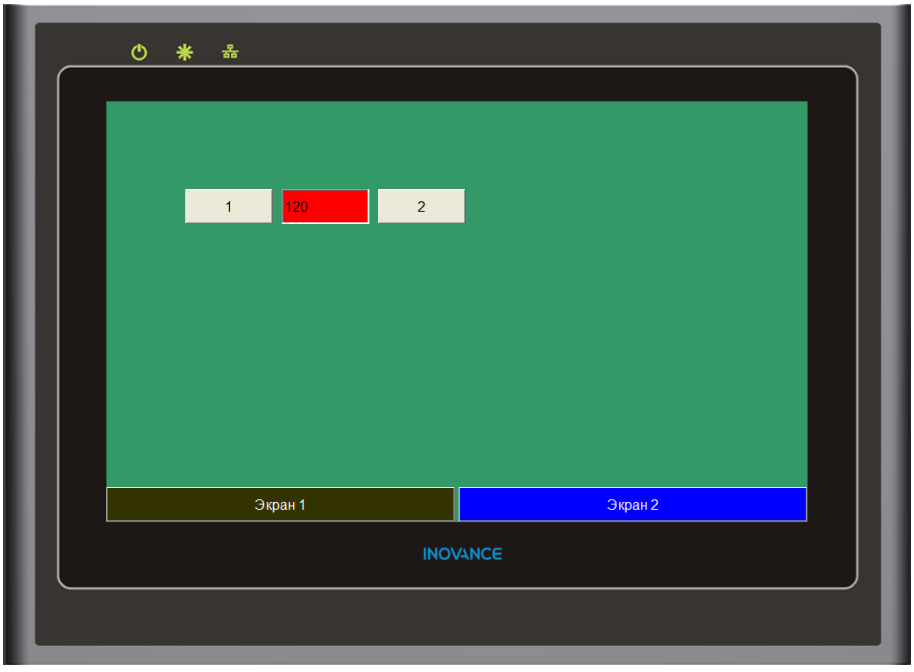


Рис. 4.16. Панель «Экран 2»

Для того чтобы закрыть визуализацию, щелкните ПКМ по экрану и нажмите Quit. Данные с ПЛК передаются на панель.

Организация связи контроллера с панелью визуализации IT7000 по OPC UA Server

В InoProShop зайдите во вкладку Device → PLC Logic → Application → Add Object –→ Symbol Configuration (рис. 4.17).

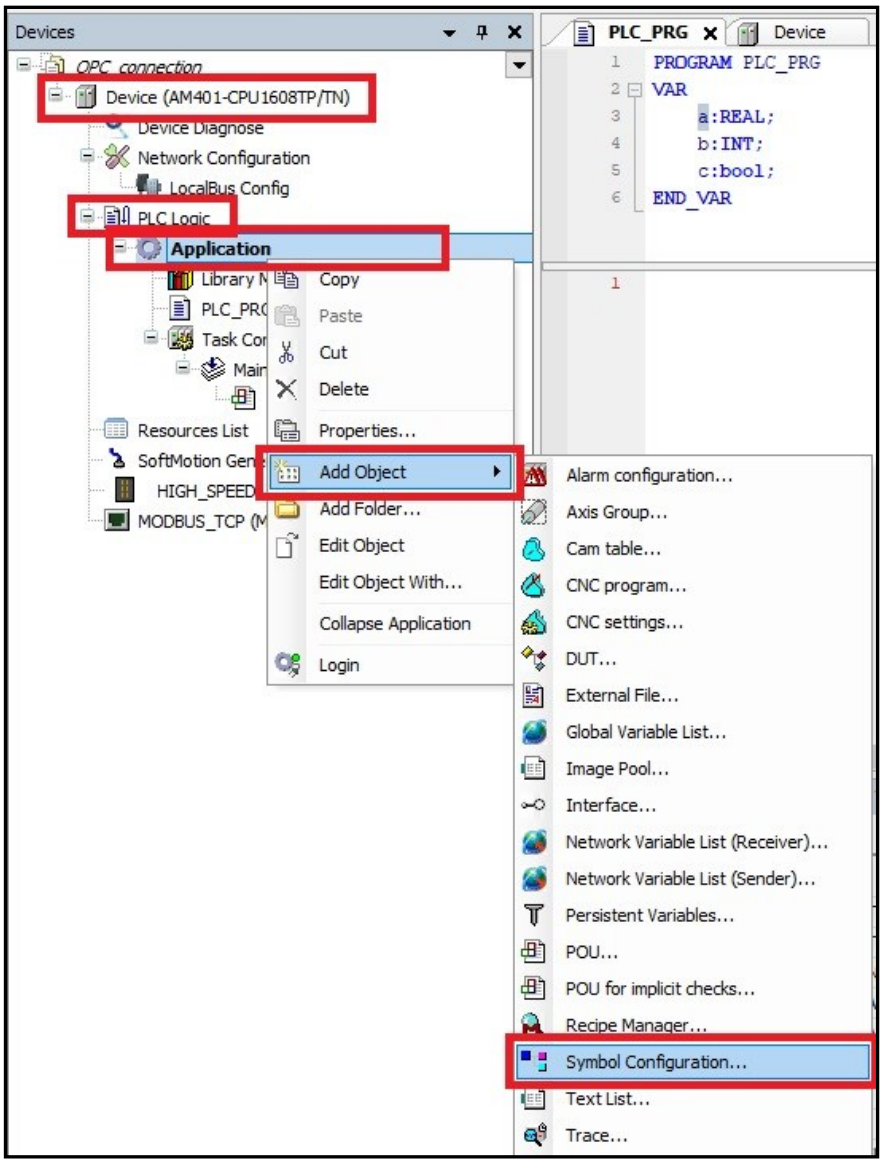


Рис. 4.17. Вход в Symbol Configuration

В появившемся окне выберите Support OPC UA features и Optimized Layout (рис. 4.18).

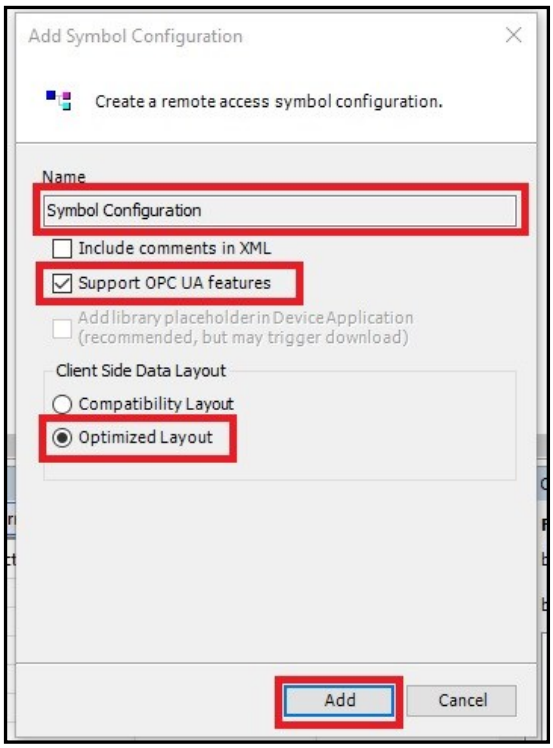


Рис. 4.18. Support OPC UA features и Optimized Layout

Далее в окне объявления переменных создайте теги, например переменные a:REAL, b:INT, c:bool, объявленные в PLC_PRG.

В Symbol Configuration нажмите Build и выберите созданные переменные (рис. 4.19).

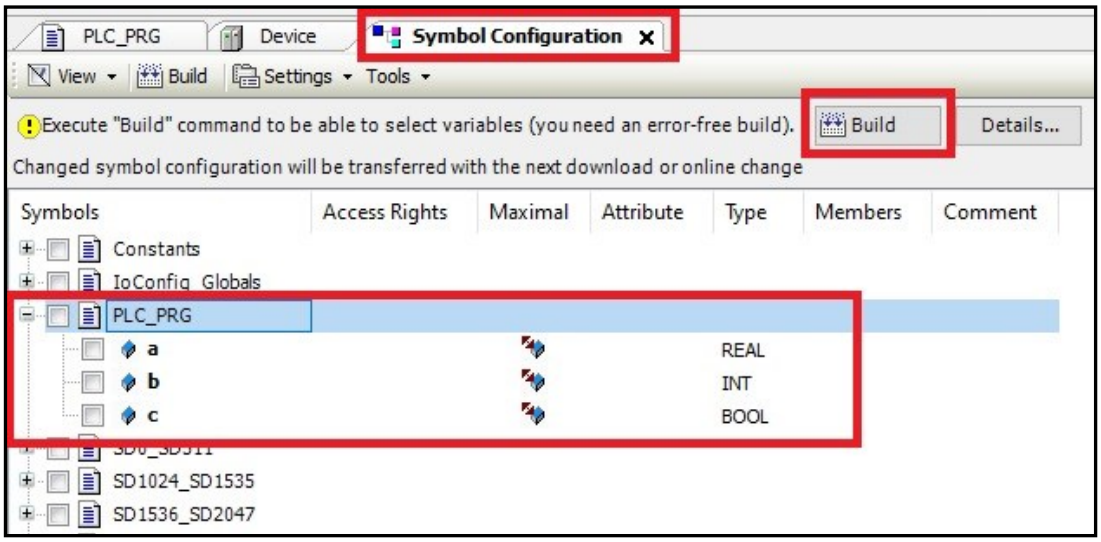


Рис. 4.19. Выбор созданных переменных

Загрузите программу в контроллер и запустите ее.

В **InoTouchPad пропишите соединение:**

Зайдите во вкладку Project → Communication → Connections (рис. 4.20).

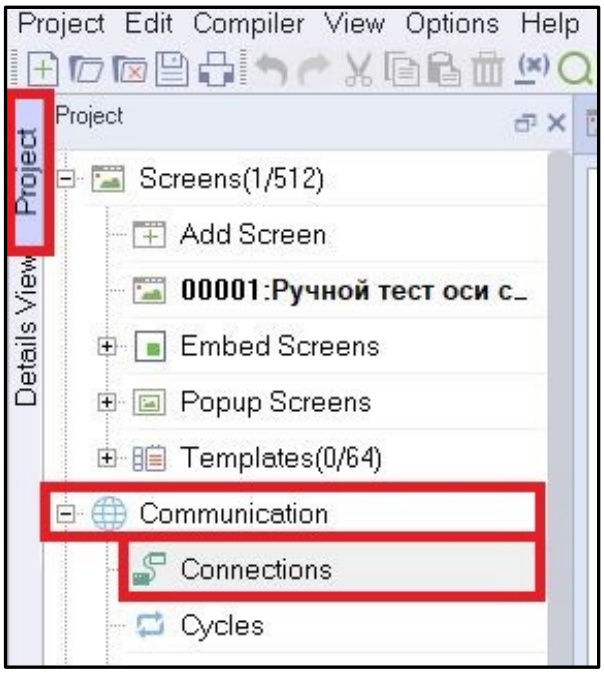


Рис. 4.20. Project → Communication → Connections

Создайте новое соединение, нажав на кнопку + .

Задайте имя, например “AM_OPC”.

В ячейке Communication protocol выберите OPC → OPC UA Client (рис. 4.21).

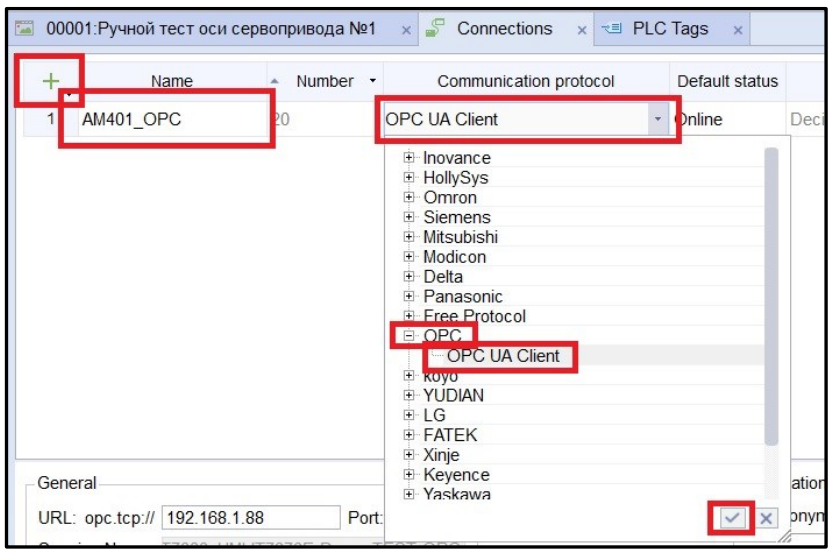


Рис. 4.21. OPC → OPC UA Client

Далее в General в URL:opc.tcp.// (рис. 4.22) пропишите IP контроллера. Для этого нажмите Browser Tags, при этом контроллер должен быть подключен к ПЛК.

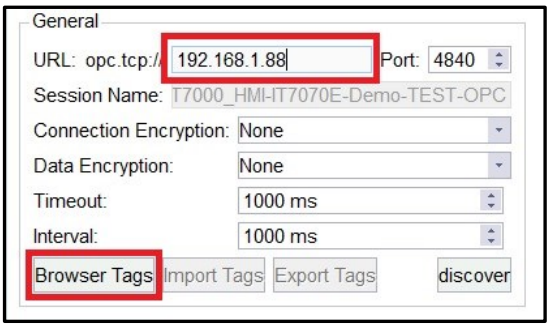


Рис. 4.22. Работа в разделе General

В открывшемся окне Root → Objects → DeviceSet → Inovance-ARM-Linux → Resources → Application → Programs выберите нужные переменные, например a[Float], b[Int16], c[Boolean], объявленные в PLC_PRG, и нажмите Add Tag (рис. 4.23).

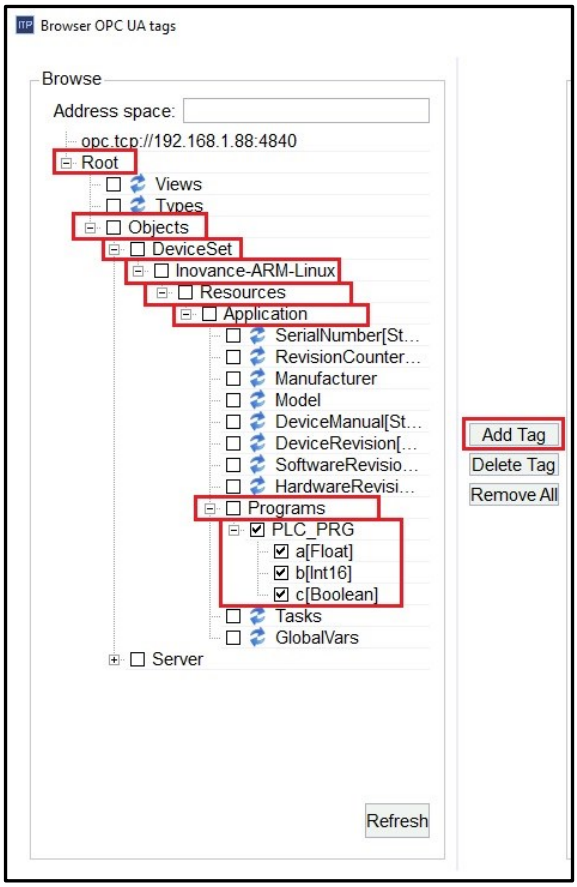


Рис. 4.23. Выбор необходимых переменных

Выберите Create PLC Tags to group, а также нужную группу тегов, в которую требуется записать данные переменные (рис. 4.24).

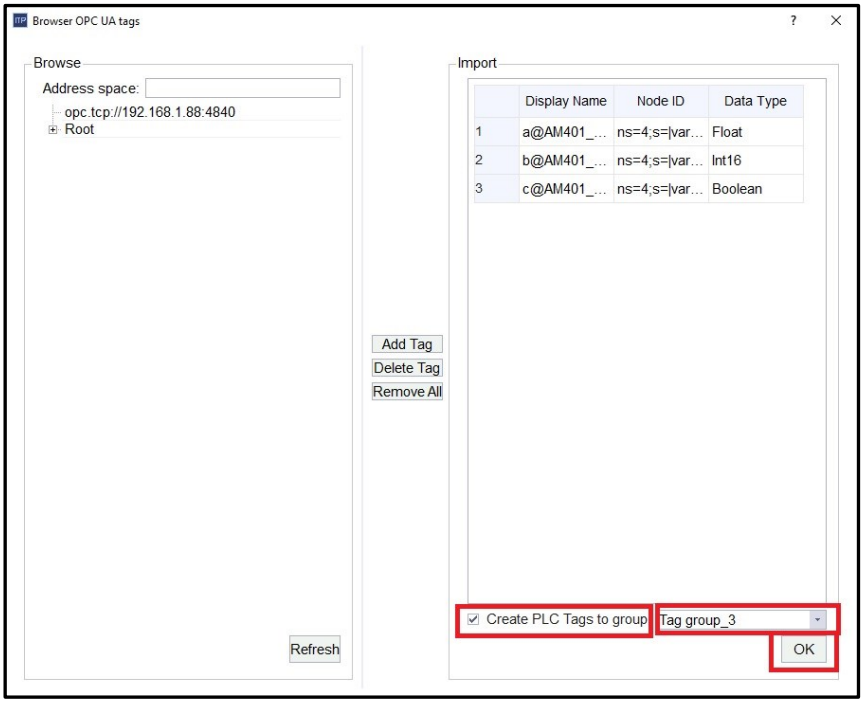


Рис. 4.24. Запись данных переменных

Организация связи контроллера с панелью визуализации IT7000 по Modbus TCP Protocol

В InoProShop зайдите во вкладку Device → Network Configuration и кликните ЛКМ на модуль. В появившемся окне выберите Ethernet → ModbusTCP Slave (4.25).

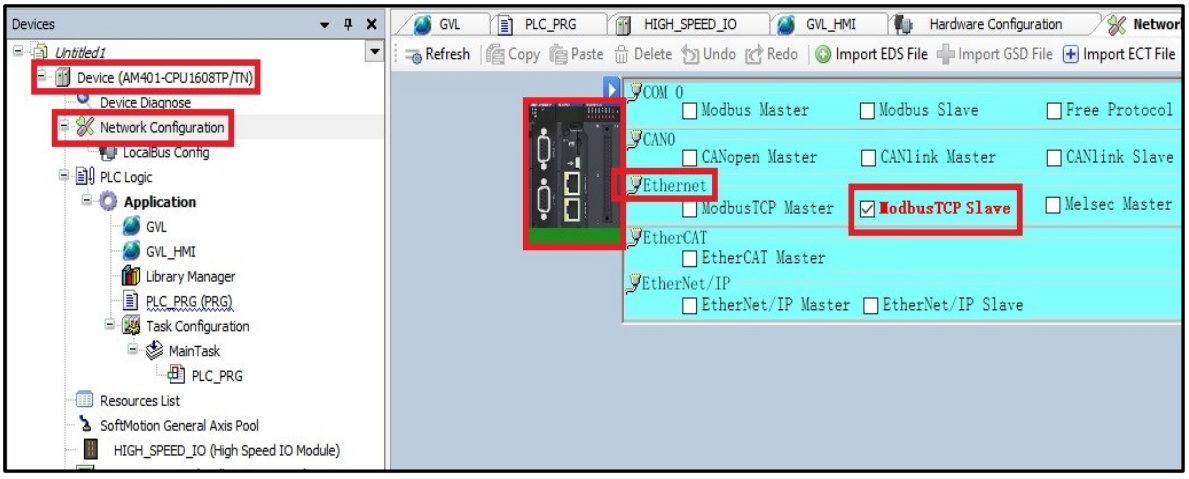


Рис. 4.25. Вход в ModbusTCP Slave

Далее зайдите в Device → MODBUS_TCP (ModbusTCP Device) → ModbusTCP Slave Configuration → Slave Port (рис. 4.26) и установите порт (порт должен быть таким же, как и в панели).

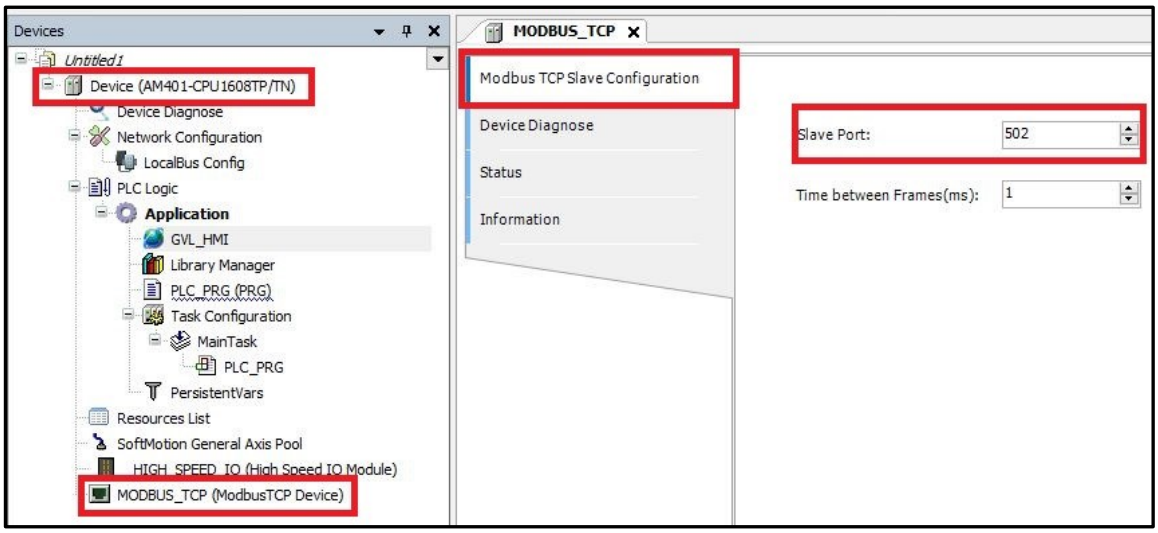


Рис. 4.26. Установка порта

Пропишите переменные с их адресами (рис. 4.27).

При определении адреса используются определенные символьные строки для выражения положения и размера памяти (табл. 4.1).

Синтаксис:

%<memory range prefix><size prefix><number|.number|.number....>

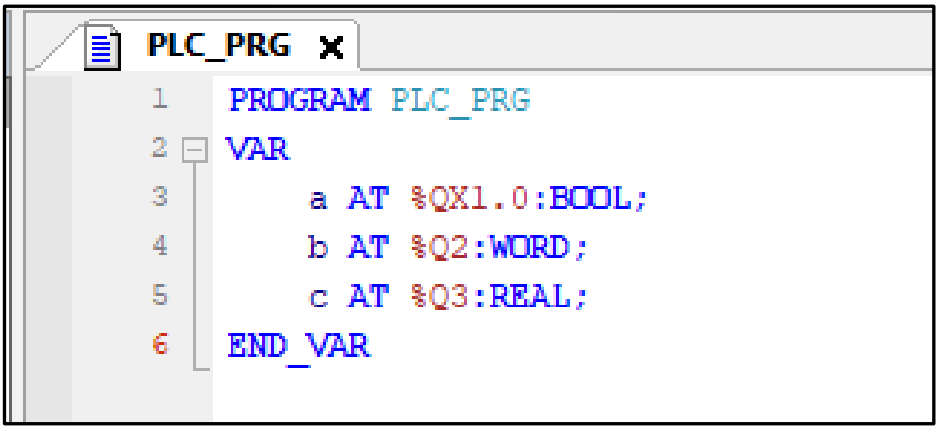


Рис. 4.27. Запись переменных

Поддерживаются следующие префиксы диапазона памяти:

I – диапазон входной памяти (физические входы от входных дисков, датчиков).

Q – диапазон выходной памяти (физические выходы для выходных дисков, исполнительных механизмов).

M – диапазон памяти флага.

Поддерживаются следующие префиксы размеров:

X – (одиночный) бит.

Никакой – (одиночный) бит.

B – BYTE (8 бит).

W – WORD (16 бит).

D – DWORD (32 бита).

Примеры программирования с использованием приведенных выше префиксов приведены в табл. 4.1.

Таблица 4.1

Примеры программирования

Фрагмент кода	Значение запрограммированной строки
%QX7.5 %Q7.5	Выходной бит 7,5
%IW215	Входное слово 215
%QB7	Выходной байт 7
%MD48	Двойное слово в позиции памяти 48 в памяти флага
%IW2.5.7.1	Интерпретация зависит от текущей конфигурации контроллера
ivar AT %IW0: WORD;	Пример объявления переменной с адресом

В объявлении переменных код присваивает переменную проекта определенному входному адресу, выходному адресу или адресу памяти контроллера, настроенного в дереве устройств.

Синтаксис:

<identifier> AT <address>:<data type>;

Это объявление (AT) позволяет дать информативное название адресу. Можно внести любые необходимые изменения для входных или выходных сигналов только в одном месте, например в объявлении.

Если назначается переменная адреса, следует обратить внимание на следующее:

1. Нельзя выполнять запись в переменные, которые размещаются на входных данных, поскольку это приведет к ошибкам компиляции.

2. Объявления AT можно выполнять только для локальных и глобальных переменных, а не для входных и выходных переменных POU.

3. Объявления AT нельзя использовать в списках постоянных переменных.

4. Если для компонентов структуры или функционального блока используются объявления AT, то все экземпляры используют одну и ту же память. Это похоже на использование статических переменных в классических языках программирования, таких как C.

5. Компоновка памяти структур также зависит от целевой системы. Пример: D0 содержит B0 – B3, W0 содержит B0 и B1, W1 содержит B1 и B2, а W2 содержит B2 и B3. Для того чтобы избежать перекрытия, не используйте W1 или D1, D2, D3 для адресации.

Если адрес одного бита не определен явно, CODESYS выделяет логические значения в байтах.

Пример: изменение значения влияет на диапазон varbool1 AT %QW0QX0.0QX0.7

В **InoTouchPad пропишите соединение:**

Зайдите во вкладку Project → Communication → Connections.

Создайте новое соединение, нажав на кнопку + .

Задайте имя, например “Modbus TCP_AM401”.

В ячейке Communication protocol выберите Inovance → AM600 Series → AM600 Modbus TCP Protocol (рис. 4.28).

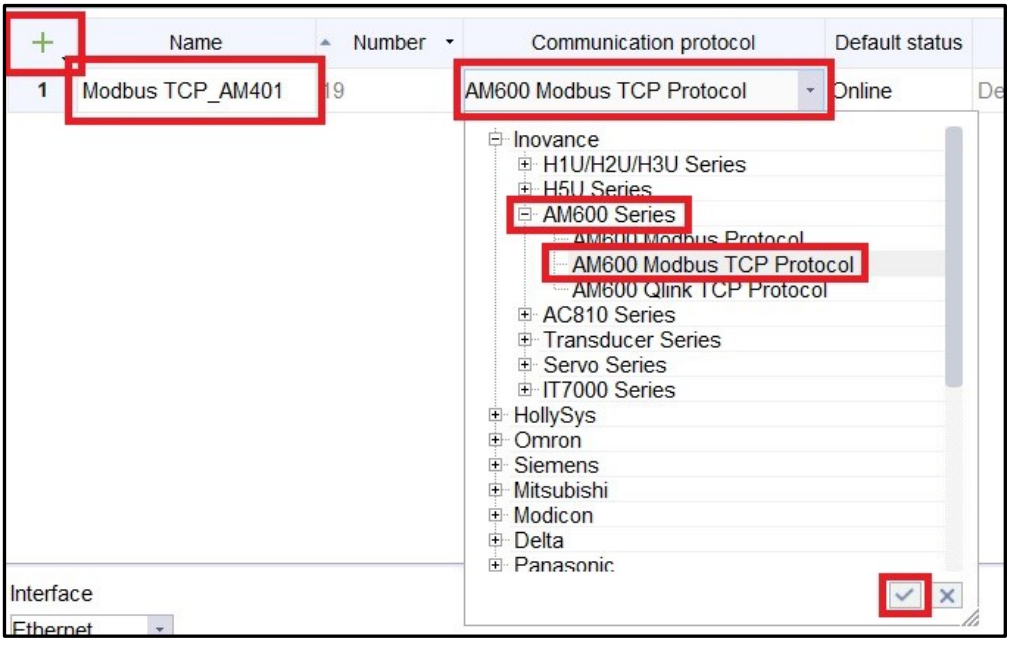


Рис. 4.28. Установка AM600 Modbus TCP Protocol

В Slave Device установите IP Address контроллера Port, Slave address и Address Interval(words) (рис. 4.29).

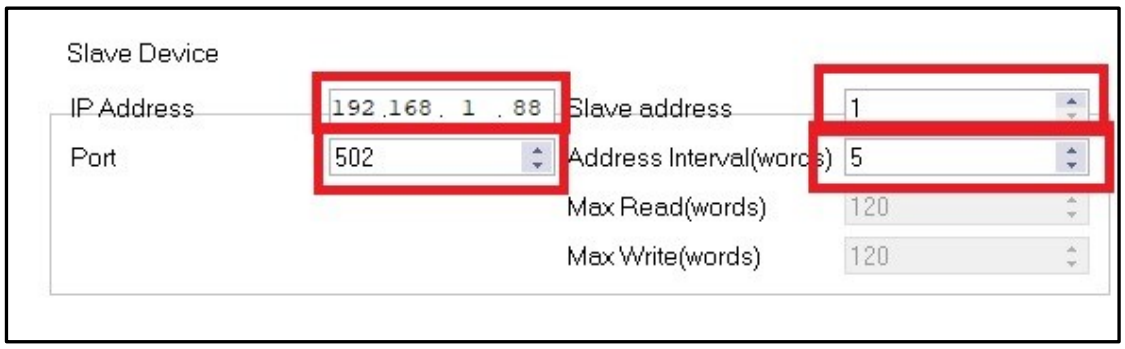


Рис. 4.29. Указание адреса контроллера

Далее перейдите во вкладку Project → Communication → Tags → → Add Tag Group, затем – в созданную группу тегов (рис. 4.30).

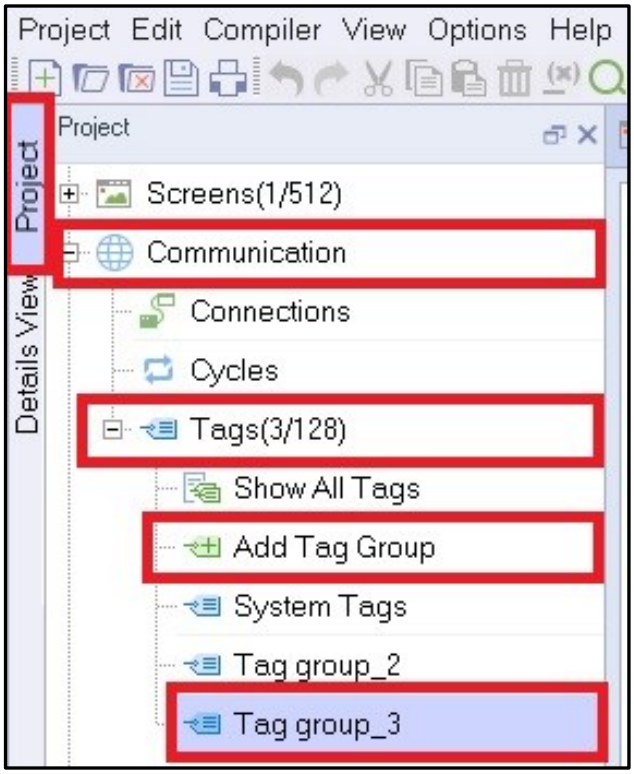


Рис. 4.30. Выбор необходимого тега

Создайте переменную, нажав на кнопку + .

Задайте имя переменной (для удобства пользования имя переменной в панели должно совпадать с именем переменной в контроллере).

Выберите в ячейке Connections Id созданное соединение (рис. 4.31).

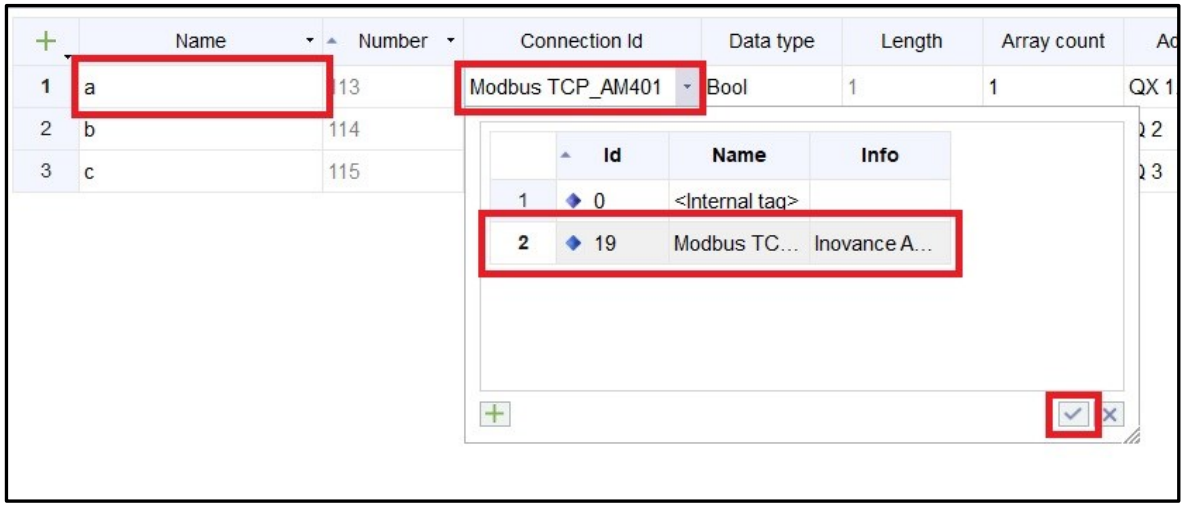


Рис. 4.31. Выбор созданного соединения

В ячейке Data type выберите тип переменной (должен совпадать с типом переменной в контроллере), в ячейке Address – Area и Address (рис. 4.32) (должен совпадать с адресом в контроллере).

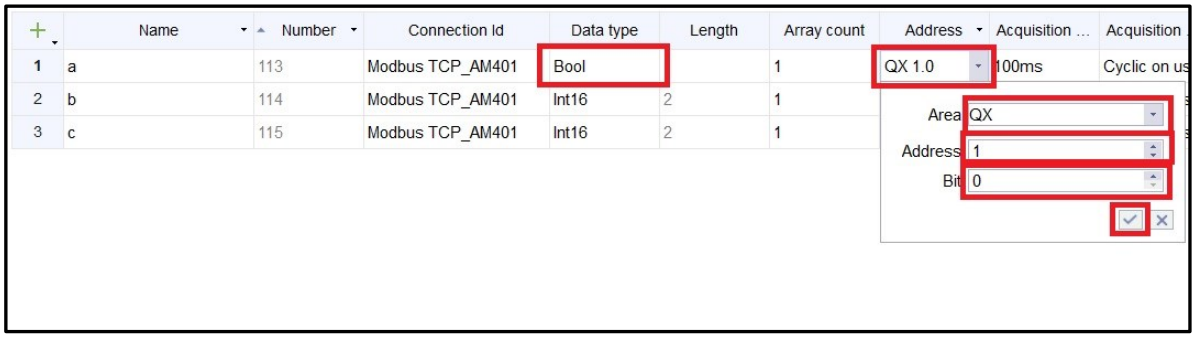


Рис. 4.32. Выбор Area и Address

При написании программы следует учитывать, что у каждого типа данных – свои варианты написания области и адреса.

Организация связи контроллера с панелью визуализации IT7000 по Modbus TCP Protocol HMI Slave

В InoProShop зайдите во вкладку Device → Network Configuration и кликните ЛКМ на модуль. В появившемся окне выберите Ethernet → → ModbusTCP Master (рис. 4.33).

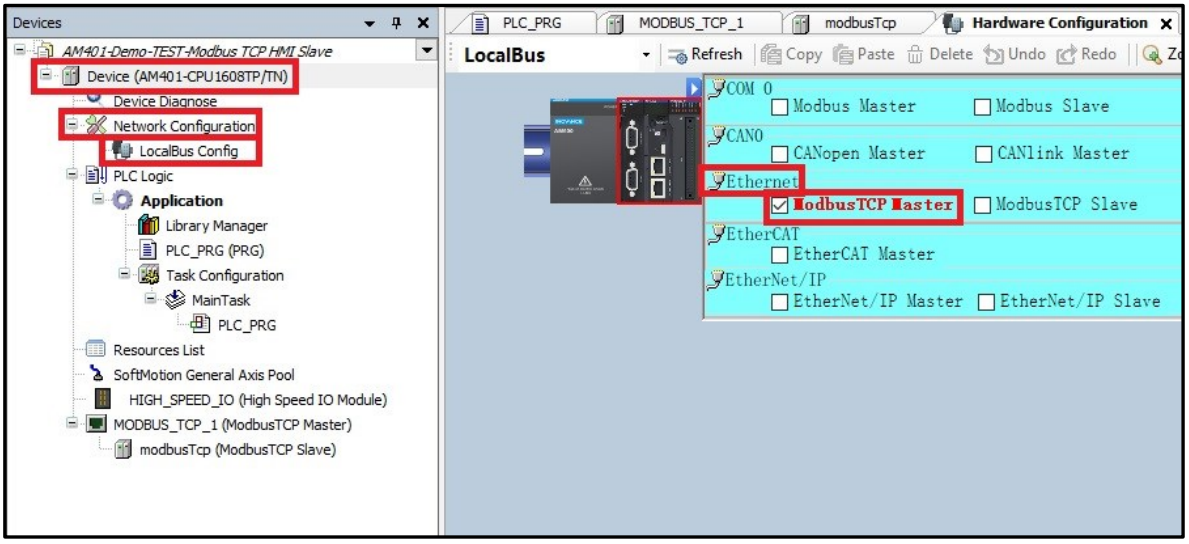


Рис. 4.33. Выбор ModbusTCP Master

Далее зайдите в Device → Network Configuration → Network Devices List → Ethernet Port → MODBUS_TCP (рис. 4.34).

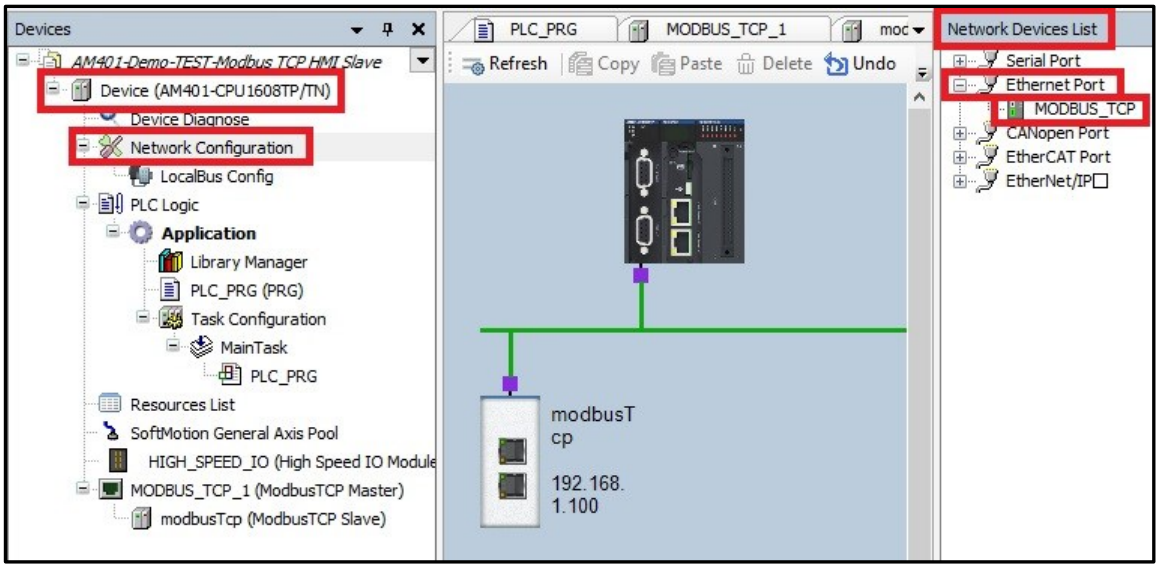


Рис. 4.34. Вход в MODBUS_TCP

Далее зайдите в Device → MODBUS_TCP (ModbusTCP Device) → → modbusTcp (ModbusTCP Slave) → ModbusTCP Slave Configuration. Окна для ввода настроек.

Slave IP Address – установите IP адрес панели.

Slave Port – установите порт (порт должен быть таким же, как и в панели).

Unit ID [0..255] – установите адрес в сети Modbus (адрес должен быть таким же, как и в панели).

Slave Enable Variable: SM – задайте переменную включения (рис. 4.35).

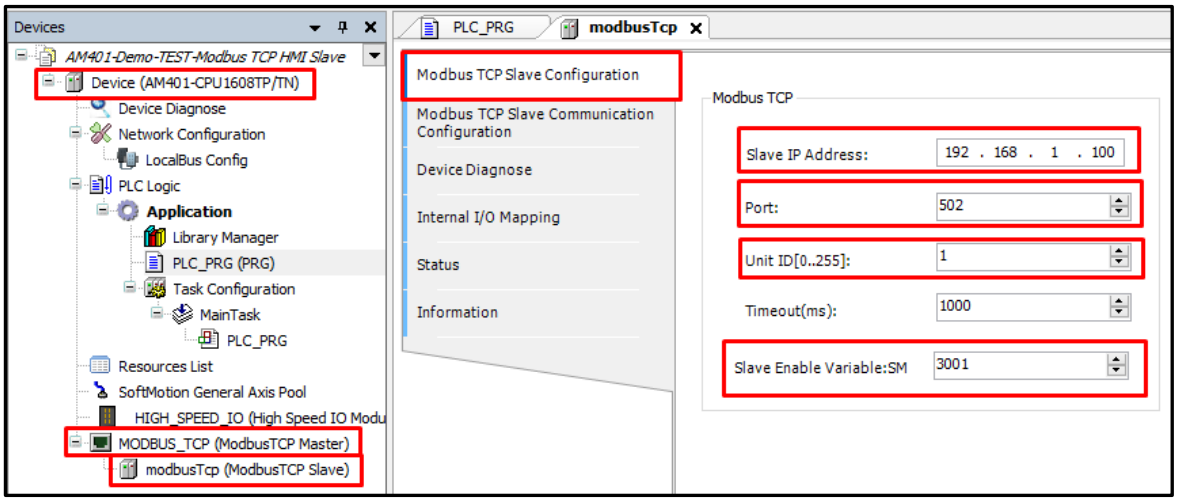


Рис. 4.35. Ввод переменной включения

Зайдите в Device → MODBUS_TCP (ModbusTCP Device) → modbusTcp (ModbusTCP Slave) → Modbus TCP Slave Communication Configuration (рис. 4.36) → Add (для того чтобы не переводить размер области памяти из десятичной системы счисления в шестнадцатеричную, можно поставить галочку Use decimal offset).

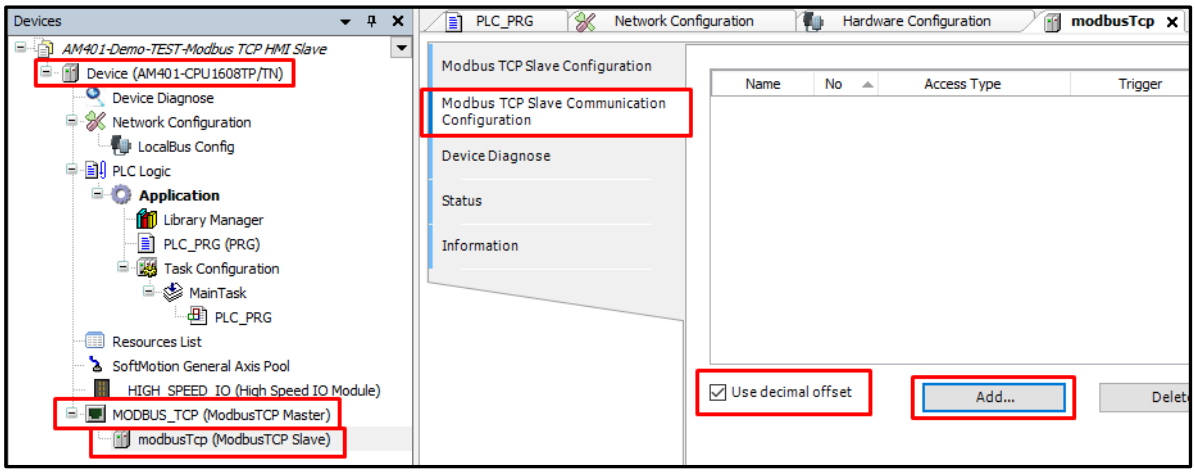


Рис. 4.36. Modbus TCP Slave Communication Configuration

В открывшемся окне задайте название канала, например Channel 01.

Access Type – выбор типа доступа: чтение или запись переменных (табл. 4.2).

Таблица 4.2

Функция чтения или записи

Код функции	Функция	Название	Тип значения	Тип доступа
(Function Code 01)	Чтение DO	Read Coils	Дискетное	Чтение
(Function Code 02)	Чтение DI	Read Discrete Inputs	Дискетное	»
(Function Code 03)	Чтение AO	Read Holding Registers	16-битное	»
(Function Code 04)	Чтение AI	Read Input Registers	16-битное	»
(Function Code 05)	Запись одного DO	Write Single Coil	Дискетное	»
(Function Code 06)	Запись одного AO	Write Single Register	16-битное	»
(Function Code 15)	Запись нескольких DO	Write Multiple Coils	Дискетное	»
(Function Code 16)	Запись нескольких AO	Write Multiple Registers	16-битное	»

Offset – смещение области памяти (табл. 4.3) (необходимо для того, чтобы в одну и ту же область памяти не попали разные переменные).

Таблица 4.3

Функции смещения области памяти переменных

Переменные в числовом виде	Переменные в буквенном виде
Чтение/запись переменных 0x/1x (0 – 12000)*	Соответствует чтение/запись переменной LB (0 – 11999)
Чтение/запись переменных 3x/4x/5x (0 – 10000)	Соответствует чтение/запись переменной LW (0 – 8999)
Чтение/запись переменных 3x/4x/5x (10000 – 65535)	Соответствует чтение/запись переменной RW (0 – 55535)

* Цифры в скобках – это условный диапазон, в котором может быть сохранена переменная.

Объявите переменную SM в PLC_PRG в соответствии с заданным значением в ModbusTCP Slave Configuration (рис. 4.37, 4.38). С помощью окна Length можно задать количество переменных.

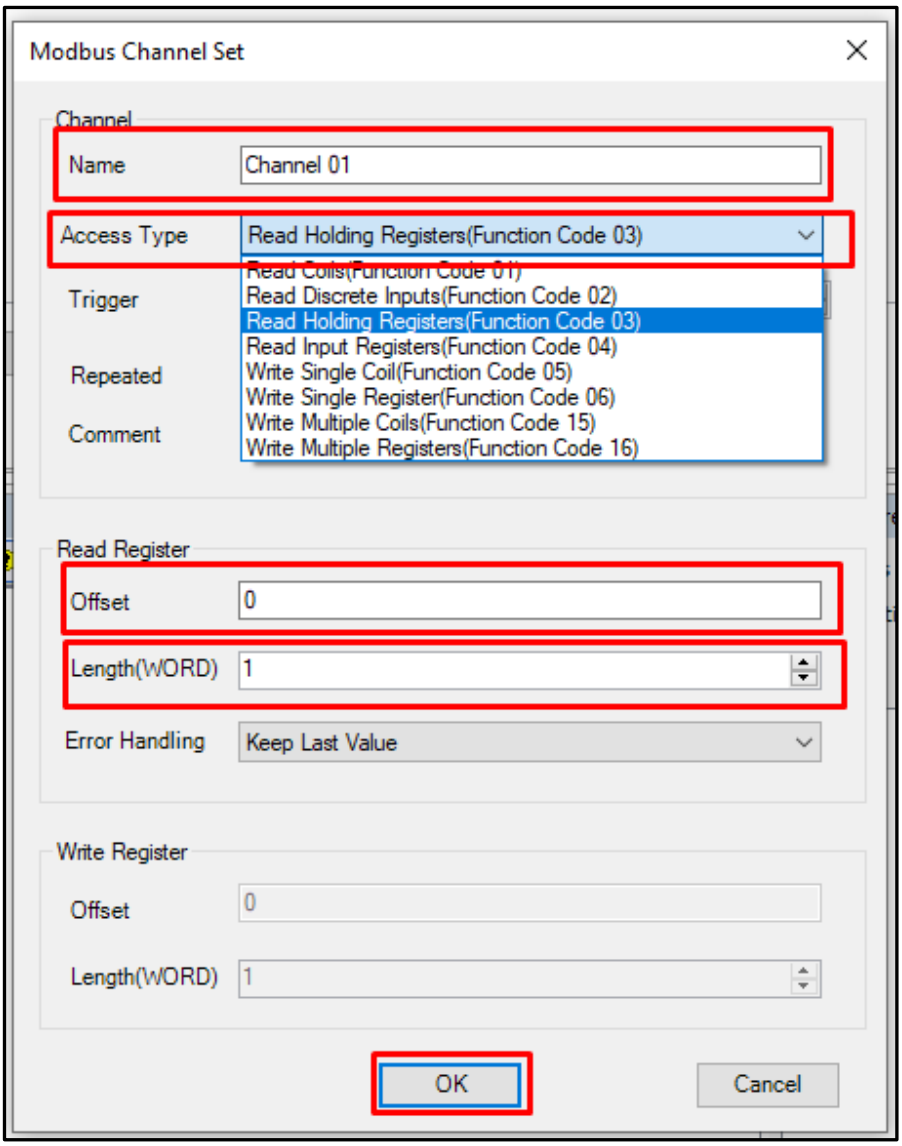


Рис. 4.37. Обновление переменной SM в PLC_PRG (шаг 1)

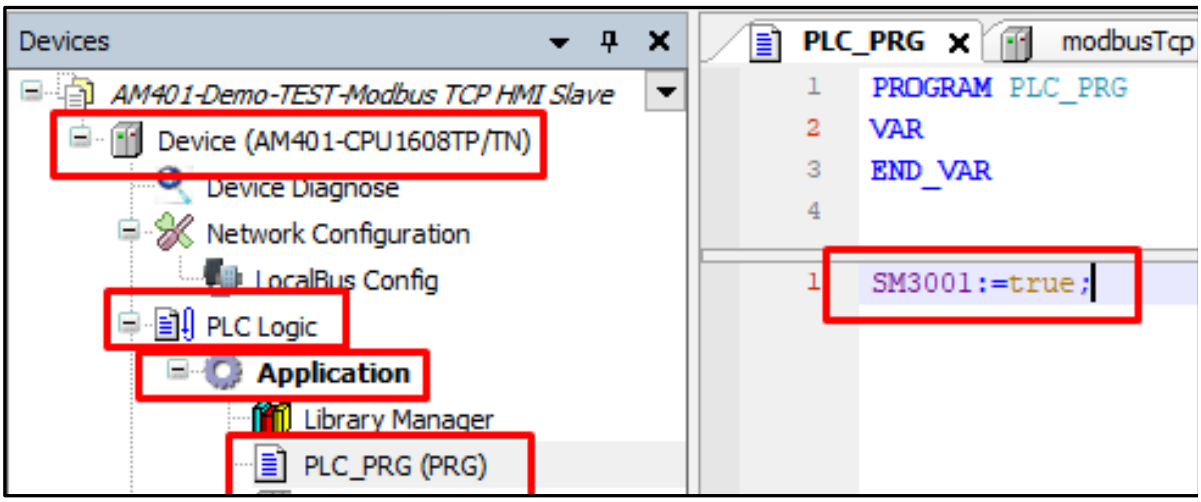


Рис. 4.38. Обновление переменной SM в PLC_PRG (шаг 2)

В **InoTouchPad пропишите соединение:**

Зайдите во вкладку Project → Communication → Connections.

Создайте новое соединение, нажав на кнопку + .

Задайте имя, например “Modbus TCP Slave_AM401”.

В ячейке Communication protocol выберите Inovance → IT7000 Series → Modbus TCP Protocol Slave (рис. 4.39).

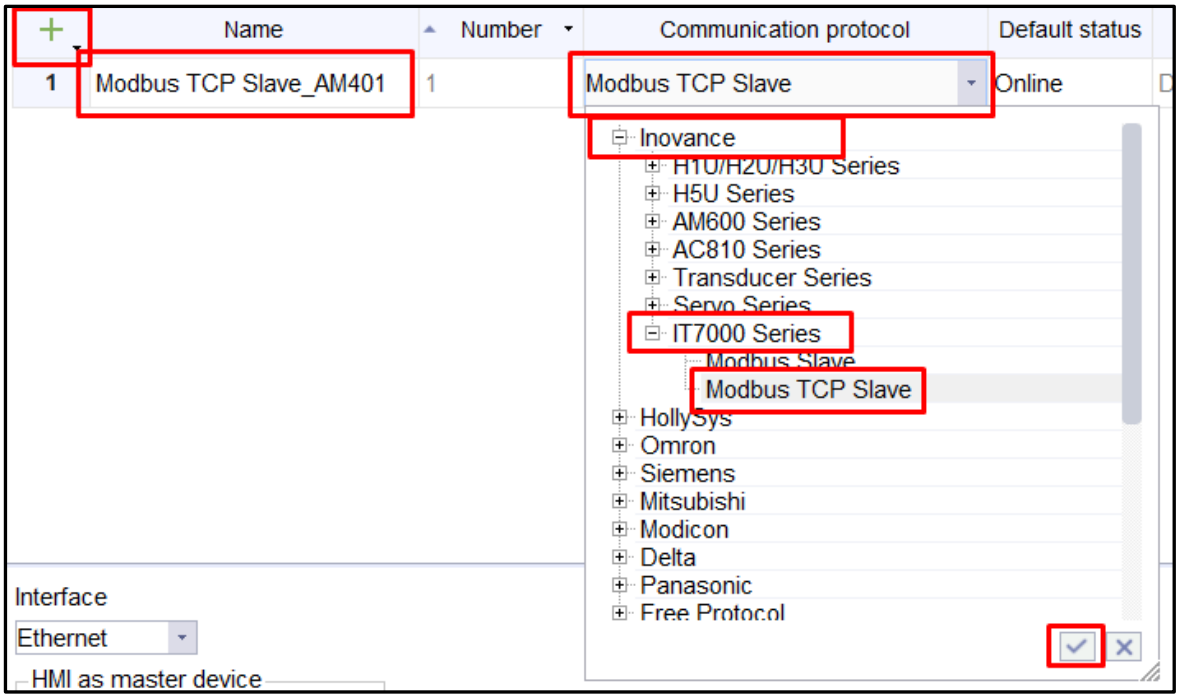


Рис. 4.39. Inovance → IT7000 Series → Modbus TCP Protocol Slave

В Slave Device установите Slave address и Address Interval(words) (рис. 4.40).

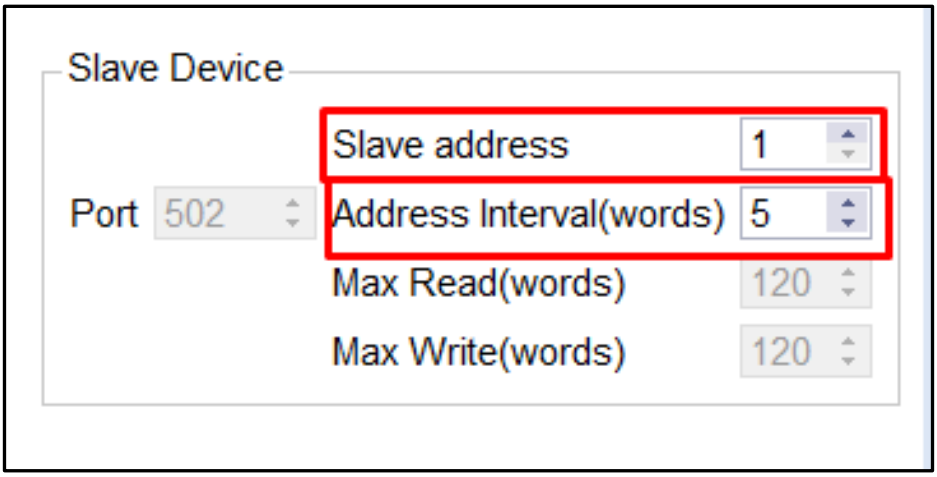


Рис. 4.40. Установка Slave address и Address Interval

Затем перейдите во вкладку Project → Communication → Tags → → Add Tag Group и в созданную группу тегов.

Создайте переменную, нажав на кнопку .

Задайте имя переменной (для удобства пользования имя переменной в панели должно совпадать с именем переменной в контроллере).

В ячейке Data type выберите тип переменной. Она должна совпадать с типом переменной в контроллере.

В ячейке Address выберите Area и Address – они должны совпадать с областью памяти в контроллере (рис. 4.41).

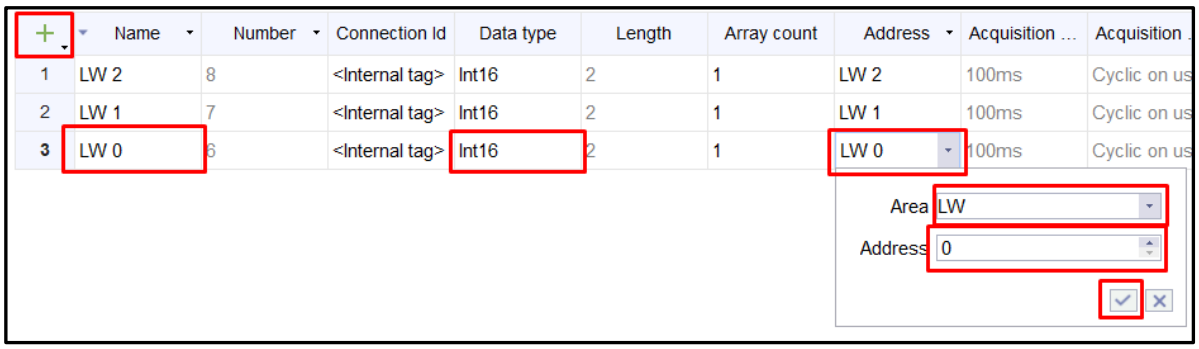


Рис. 4.41. Выбор Area и Address

При написании программы следует учитывать, что у каждого типа данных – свои варианты написания области и адреса.

Содержание работы

1. Цель работы.
2. Задания.
3. Ход работы.
4. Выводы.
5. Контрольные вопросы.

Средства, используемые при выполнении лабораторной работы

1. Методические указания к выполнению лабораторных работ.
2. Данные, предоставленные преподавателем во время занятия.
3. При проведении анализа допускается использование глобальной сети Интернет.

Контрольные вопросы

1. Какие способы связи панели оператора с контроллером вам известны?
2. Как связать панель оператора с контроллером по OPC UA Server?
3. Как связать панель оператора с контроллером по Modbus TCP Protocol?
4. Как связать панель оператора с контроллером по Modbus TCP Protocol HMI Slave?
5. Какие различия существуют между рассмотренными способами подключения панели оператора и контроллера?

Лабораторная работа № 2

РАБОТА С ДИСКРЕТНЫМИ ВХОДАМИ/ВЫХОДАМИ

Цель работы: освоить способы подключения и работы с модулями при помощи дискретных входов/выходов.

Задания

1. Подключиться к предоставленным модулям.
2. Написать программу для включения и выключения лампочек.
3. Протестировать написанный код на учебном стенде Inovance.

Ход работы

Разработка и создание кода для управления лампами с помощью дискретных входов/выходов

В дереве проекта во вкладке LocalBus Config необходимо добавить локальные модули, которые физически установлены на стенде (рис. 4.42). Первый модуль после контроллера A10 – GL10-1600END, второй A11 – GL10-0016ETP.

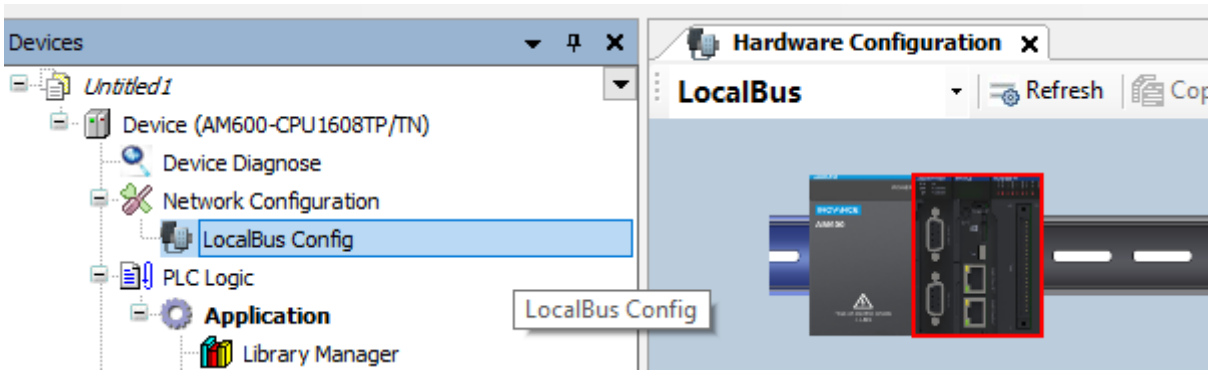


Рис. 4.42. Подключение к модулю

Для добавления модулей воспользуйтесь вкладкой справа со всеми модулями (рис. 4.43).

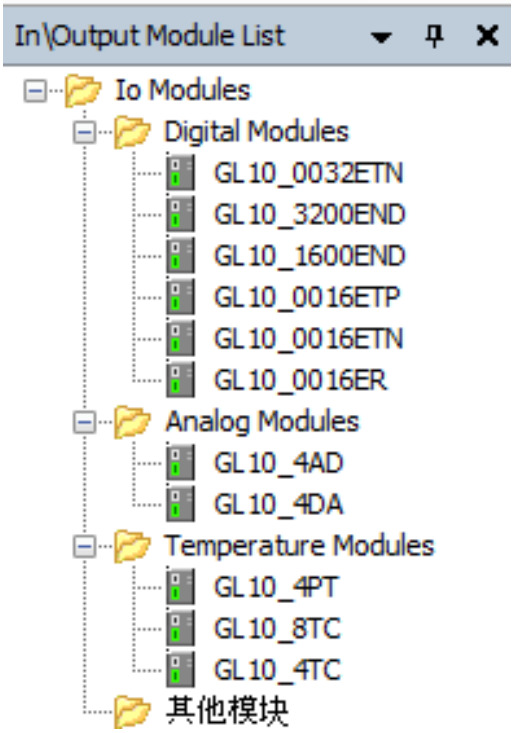


Рис. 4.43. Выбор и добавление модуля

После добавления модулей дерево проекта обновится (рис. 4.44).

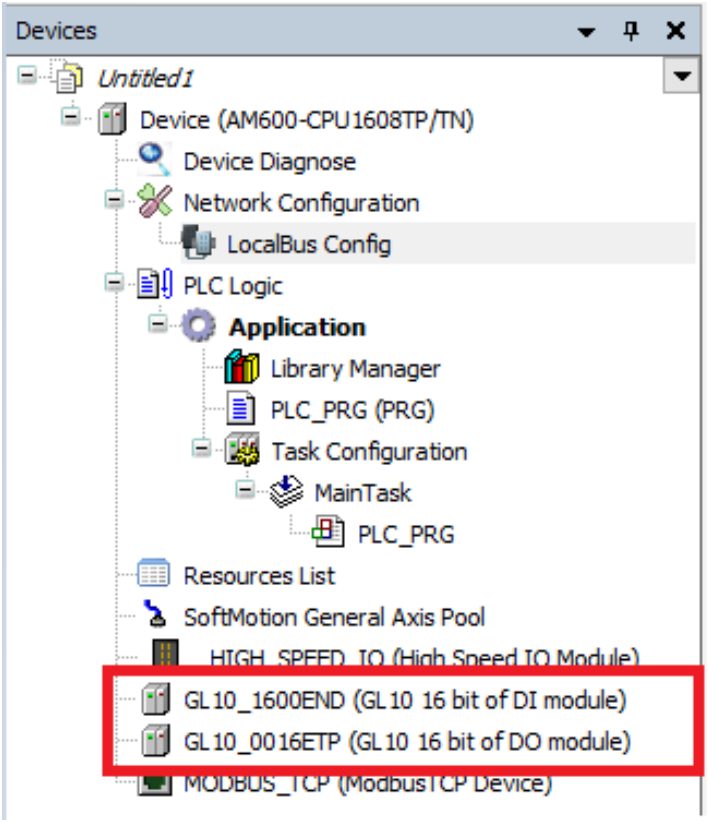
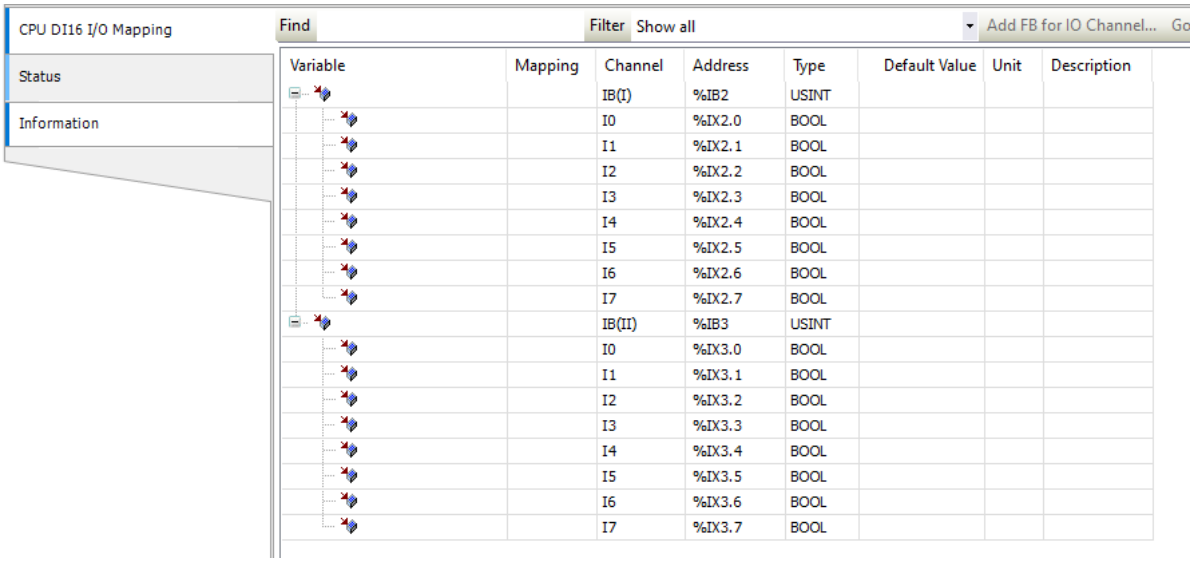


Рис. 4.44. Дерево проекта после выбора модуля

После нажатия ЛКМ на модуль дискретных входов на экране появляется конфигурация битов. Так как модуль рассчитан на 16 дискретных входов, то он поделен на два канала по 8 бит. Во вкладке Mapping можно добавлять внутренние программные переменные в модуль для отслеживания состояния дискретных входов. Переменные могут быть добавлены как на каждый бит в канале, так и на весь канал. Переменные должны быть такого же типа, как указано в колонке Type (рис. 4.45).



Variable	Mapping	Channel	Address	Type	Default Value	Unit	Description
		IB(I)	%IB2	USINT			
		I0	%IX2.0	BOOL			
		I1	%IX2.1	BOOL			
		I2	%IX2.2	BOOL			
		I3	%IX2.3	BOOL			
		I4	%IX2.4	BOOL			
		I5	%IX2.5	BOOL			
		I6	%IX2.6	BOOL			
		I7	%IX2.7	BOOL			
		IB(II)	%IB3	USINT			
		I0	%IX3.0	BOOL			
		I1	%IX3.1	BOOL			
		I2	%IX3.2	BOOL			
		I3	%IX3.3	BOOL			
		I4	%IX3.4	BOOL			
		I5	%IX3.5	BOOL			
		I6	%IX3.6	BOOL			
		I7	%IX3.7	BOOL			

Рис. 4.45. Написание переменных

Создайте переменные в основной программе в поле ввода параметров (рис. 4.46).

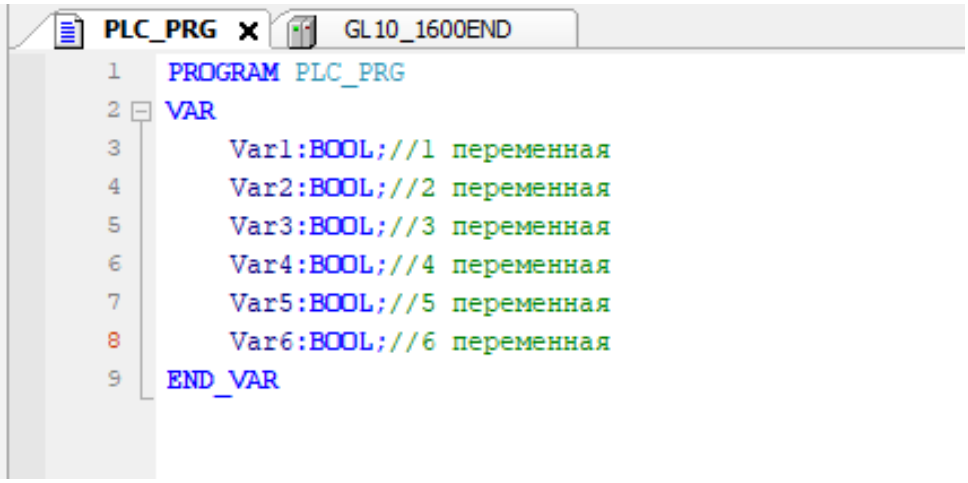


Рис. 4.46. Написание переменных в основной программе

По схеме (см. рис. 1.8 – 1.16) можно увидеть, что тумблеры и кнопки запрограммированы в битах от 0 до 2 на первом и втором каналах соответственно. Для добавления переменных необходимо нажать в поле Variable два раза ЛКМ (рис. 4.47).

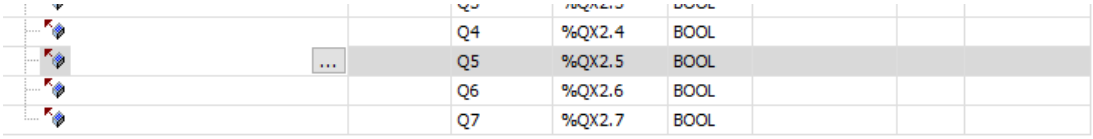


Рис. 4.47. Проверка подключения тумблеров и кнопок

После нажатия появится кнопка с тремя точками. Нажмите на нее. Во всплывающем окне откройте Application и раскройте программу – в ней видны все доступные переменные для выбора (рис. 4.48).

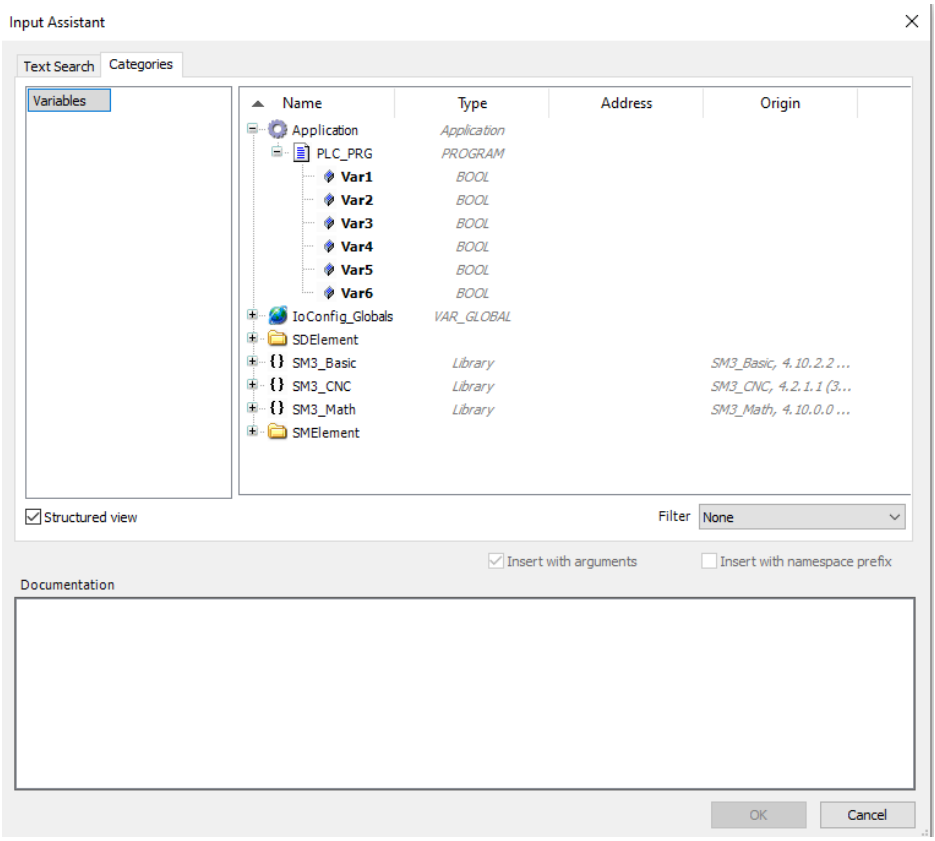


Рис. 4.48. Доступные переменные в программе

Выберите для каждого канала переменную. Все остальные переменные в данной вкладке относятся к системным переменным (рис. 4.49).

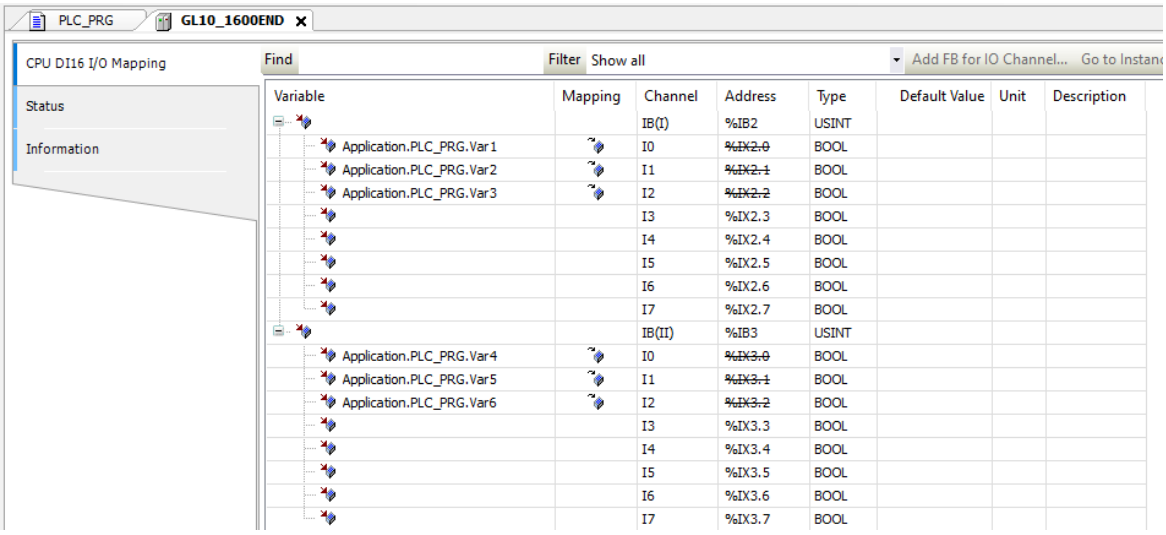


Рис. 4.49. Созданные и системные переменные

Те же самые операции выполните и для модуля дискретных выходов (рис. 4.50). На каждый канал от 0 до 2 бит физически подключены реле с лампочками.

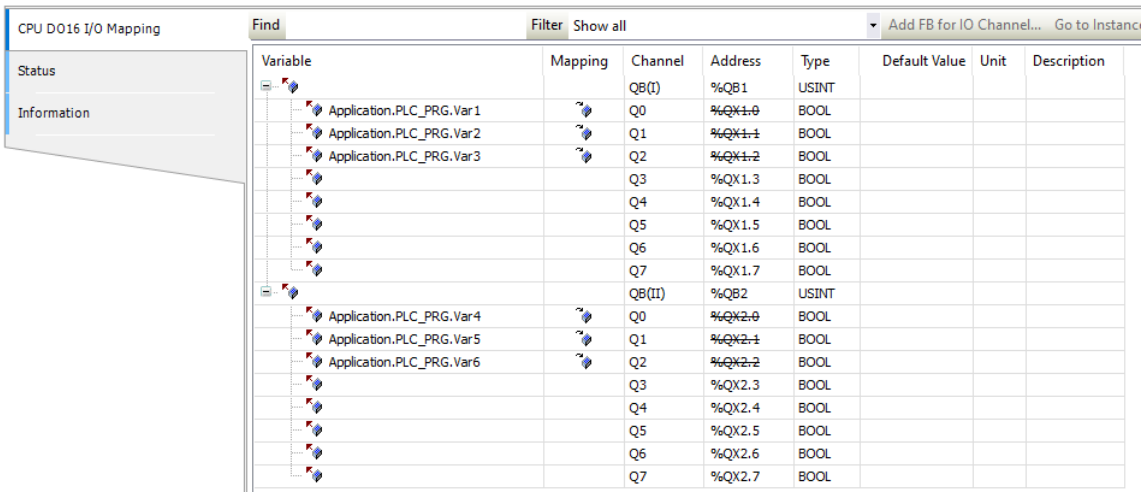


Рис. 4.50. Присвоение переменных для модуля дискретных выходов

Загрузите программу в контроллер (стенд) и нажмите Play для того, чтобы программа запустилась (рис. 4.51).

Рис. 4.51. Загрузка и запуск программы

После запуска программы включите и выключите тумблеры на стенде, нажмите на кнопки – на реле должны загореться лампочки.

Изучим математические функции CoDeSys.

Выйдите из программы, нажав шестеренки Logout, и зайдите в основную программу.

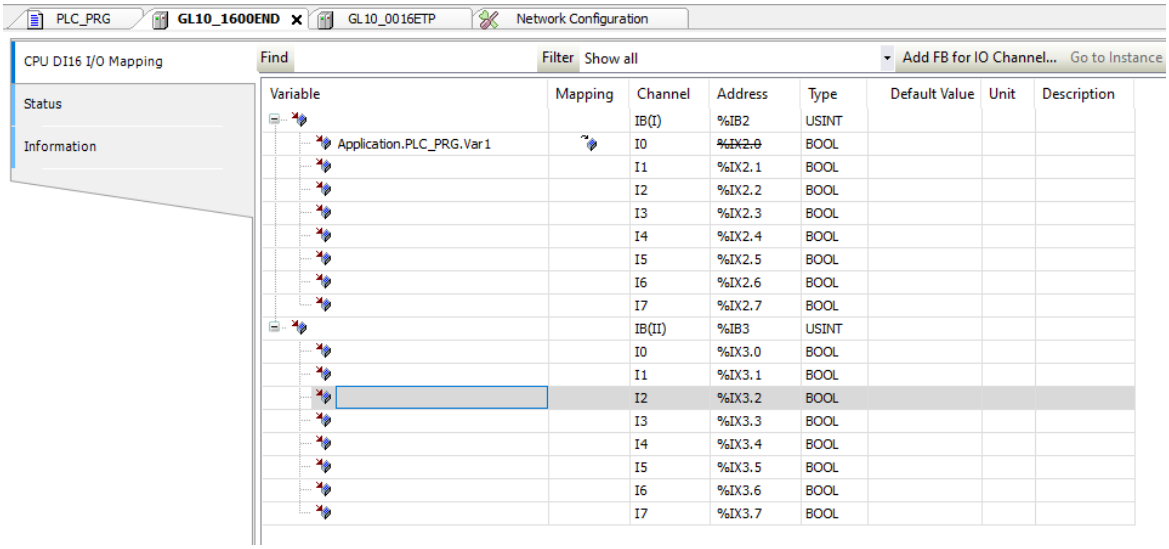
Запишите новые переменные (рис. 4.52).

PLC_PRG | GL10_1600END | GL10_0016ETP

```
1  PROGRAM PLC_PRG
2  VAR
3      Varlinput:BOOL;//1 переменная
4      Var2output:BOOL;//2 переменная
5      Var3output:BOOL;//3 переменная
6      Var4output:BOOL;//4 переменная
7      Var5output:BOOL;//5 переменная
8      Var6output:BOOL;//6 переменная
9      Var7output:BOOL;//7 переменная
10 END_VAR
```

Рис. 4.52. Запись новых переменных

Редактируйте значения дискретных входов (рис. 4.53, 4.54).



Variable	Mapping	Channel	Address	Type	Default Value	Unit	Description
		IB(I)	%IB2	USINT			
Application.PLC_PRG.Var1		I0	~~%IX2.0~~	BOOL			
		I1	%IX2.1	BOOL			
		I2	%IX2.2	BOOL			
		I3	%IX2.3	BOOL			
		I4	%IX2.4	BOOL			
		I5	%IX2.5	BOOL			
		I6	%IX2.6	BOOL			
		I7	%IX2.7	BOOL			
		IB(II)	%IB3	USINT			
		I0	%IX3.0	BOOL			
		I1	%IX3.1	BOOL			
		I2	%IX3.2	BOOL			
		I3	%IX3.3	BOOL			
		I4	%IX3.4	BOOL			
		I5	%IX3.5	BOOL			
		I6	%IX3.6	BOOL			
		I7	%IX3.7	BOOL			

Рис. 4.53. Поля ввода значений для дискретных входов

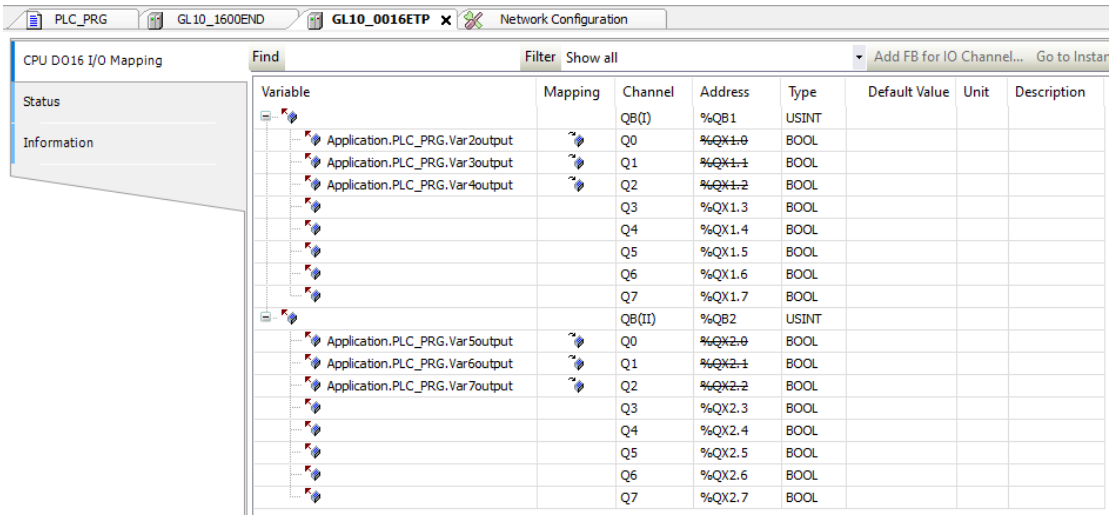


Рис. 4.54. Установленные значения для дискретных входов

Напишите условие для тумблера SA1. Если он будет включен, то загорятся первые три реле, если выключен, то вторые три реле.

Подсказкой для написания кода служит окно, расположенное справа от программы. В нем отражены все логические операции для написания кода. На данном этапе потребуется логическая операция If, которую можно вынести в тело программы, нажав на нее ЛКМ (рис. 4.55).

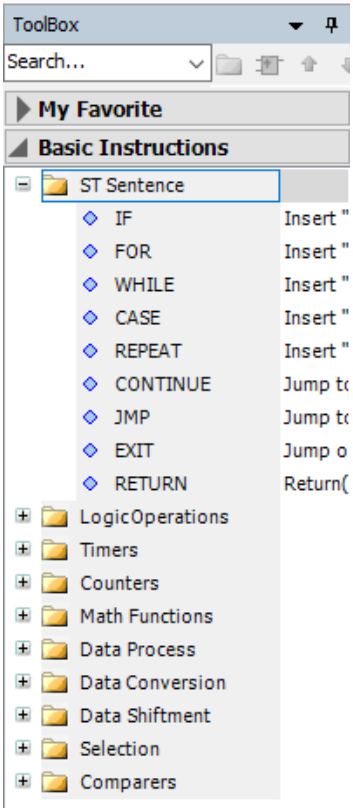


Рис. 4.55. ST Sentence

Загрузите проект в контроллер (рис. 4.56, 4.57) и запустите программу.

Включите/выключите тумблер SA1, должны загореться лампочки на реле.

Разберем логическую операцию NOT (Инверсия сигнала). Добавьте ее в код и запустите программу. Теперь, когда тумблер выключен, горят первые три реле, а когда включен – следующие три реле.

PLC_PRG | GL10_1600END | GL10_0016ETP

```
PROGRAM PLC_PRG
VAR
    Varlinput:BOOL;//1 переменная
    Var2output:BOOL;//2 переменная
    Var3output:BOOL;//3 переменная
    Var4output:BOOL;//4 переменная
    Var5output:BOOL;//5 переменная
    Var6output:BOOL;//6 переменная
    Var7output:BOOL;//7 переменная
END_VAR
```

```
IF Varlinput THEN //Если темблер включен
    Var2output:=TRUE;//Включаем реле 1
    Var3output:=TRUE;//Включаем реле 2
    Var4output:=TRUE;//Включаем реле 3
    Var5output:=FALSE;//Выключаем реле 4
    Var6output:=FALSE;//Выключаем реле 5
    Var7output:=FALSE;//Выключаем реле 6
ELSE//Иначе, если реле выключено
    Var2output:=FALSE;//Выключаем реле 1
    Var3output:=FALSE;//Выключаем реле 2
    Var4output:=FALSE;//Выключаем реле 3
    Var5output:=TRUE;//Включаем реле 4
    Var6output:=TRUE;//Включаем реле 5
    Var7output:=TRUE;//Включаем реле 6
END_IF
```

Рис. 4.56. Код с присвоенными значениями

```
IF NOT Varlinput THEN //Если темблер включен
    Var2output:=TRUE;//Включаем реле 1
    Var3output:=TRUE;//Включаем реле 2
    Var4output:=TRUE;//Включаем реле 3
    Var5output:=FALSE;//Выключаем реле 4
    Var6output:=FALSE;//Выключаем реле 5
    Var7output:=FALSE;//Выключаем реле 6
ELSE//Иначе, если реле выключено
    Var2output:=FALSE;//Выключаем реле 1
    Var3output:=FALSE;//Выключаем реле 2
    Var4output:=FALSE;//Выключаем реле 3
    Var5output:=TRUE;//Включаем реле 4
    Var6output:=TRUE;//Включаем реле 5
    Var7output:=TRUE;//Включаем реле 6
END_IF
```

Рис. 4.57. Код с присвоенными значениями с инверсией сигнала

Рассмотрим логическую операцию «Таймер» (TON) (рис. 4.58). Перетащите ее ЛКМ в проект «Таймер» и во всплывающем окне задайте имя таймера (рис. 4.59).

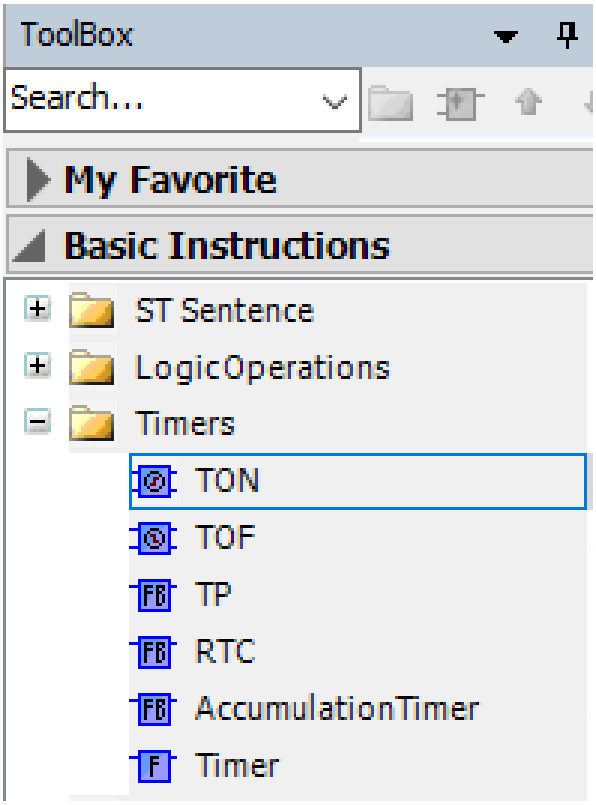


Рис. 4.58. Логическая операция «Таймер»

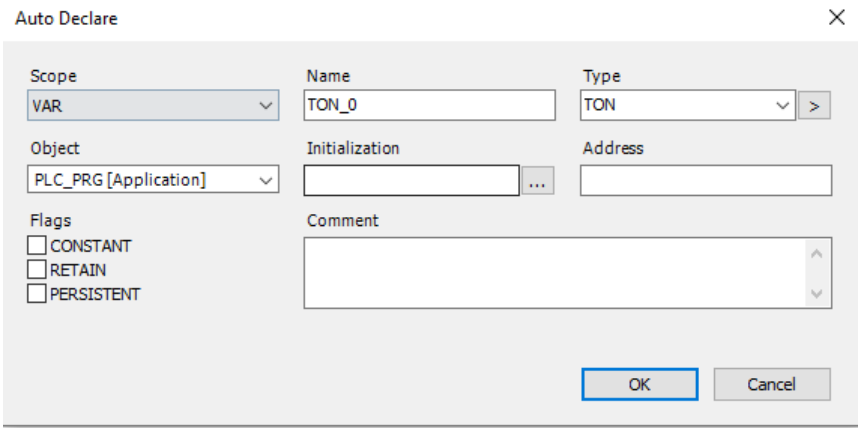


Рис. 4.59. Выбор значения таймера

После добавления логической операции TON в проект появится функциональный блок таймера со входами, которые отображаются «:=», и выходами « →». На каждый вход и выход можно добавить свою переменную и отслеживать статус выполнения функционального блока (рис. 4.60).

```
TON_0(IN:= , PT:= , Q=> , ET=> );
```

Рис. 4.60. Установка переменных TON

Для работы таймера принимаем условие: все реле должны включиться через 5 с. Для этого измените ранее написанный код (рис. 4.61).

```
TON_0(IN:=Varlinput, PT:=T#5S , Q=> , ET=> );
IF TON_0.Q THEN //Если таймер отсчитал время
    Var2output:=TRUE;//Включаем реле 1
    Var3output:=TRUE;//Включаем реле 2
    Var4output:=TRUE;//Включаем реле 3
    Var5output:=TRUE;//Включаем реле 4
    Var6output:=TRUE;//Включаем реле 5
    Var7output:=TRUE;//Включаем реле 6
ELSE//Иначе, если таймер не отсчитал время
    Var2output:=FALSE;//Выключаем реле 1
    Var3output:=FALSE;//Выключаем реле 2
    Var4output:=FALSE;//Выключаем реле 3
    Var5output:=FALSE;//Выключаем реле 4
    Var6output:=FALSE;//Выключаем реле 5
    Var7output:=FALSE;//Выключаем реле 6
END_IF
```

Рис. 4.61. Финальная версия кода

Значение IN необходимо для включения таймера. В нашем случае таймер будет включатся по тумблеру. Функции, которые можно задать в логической операции TON:

PT – переменная для установления времени, в течение которого будет работать таймер.

Q – битовая переменная (отображение статуса таймера), которая позволяет после отсчета таймера изменять значения «False» и «True».

ET – переменная, показывающая время на таймере в данный момент.

Содержание работы

1. Цель работы.
2. Задания.
3. Ход работы.
4. Выводы.
5. Контрольные вопросы.

Средства, используемые при выполнении лабораторной работы

1. Методические указания к выполнению лабораторных работ.
2. Данные, предоставленные преподавателем во время занятия.
3. При проведении анализа допускается использование глобальной сети Интернет.

Контрольные вопросы

1. Что такое LocalBus config в проекте?
2. Для чего нужен Mapping в модулях?
3. Как загрузить проект в контроллер?
4. Что такое логические операции в CoDeSys?
5. Что делает логическая операция NOT?
6. Для чего нужны входы и выходы функционального блока таймера (TON)?

Лабораторная работа № 3

ЗАПУСК ДВИГАТЕЛЯ ПО СКОРОСТИ

Цель работы: научиться программировать запуск двигателя по скорости.

Задания

1. Запрограммировать инициализацию привода.
2. Рассчитать коэффициенты редукции.
3. Запрограммировать включение/выключение привода по питанию.
4. Запрограммировать движение привода по скорости.
5. Протестировать написанный код на учебном стенде Inovance.

Ход работы

Написание кода для запуска двигателя по скорости

Для добавления протокола EtherCAT необходимо перейти во вкладку Network configuration (рис. 4.62) и поставить галочку напротив EtherCAT master.

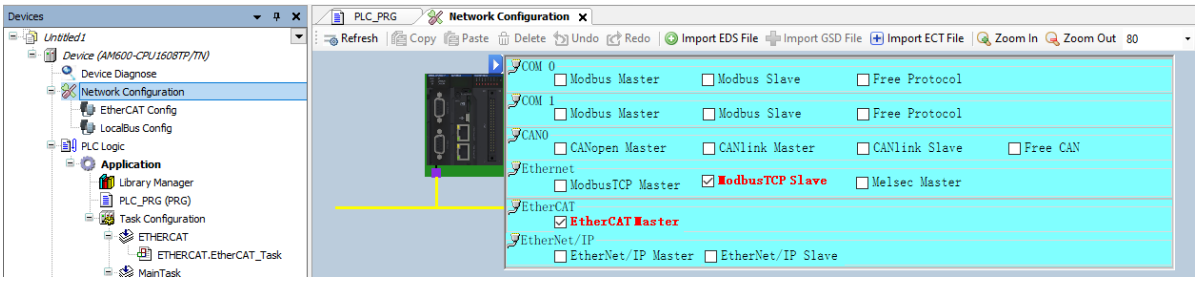


Рис. 4.62. Network configuration

В дереве проекта появятся новая задача EtherCAT и вкладка EtherCAT (рис. 4.63). Задача EtherCAT нужна для опроса всех устройств. Данный протокол является самым приоритетным в проекте.

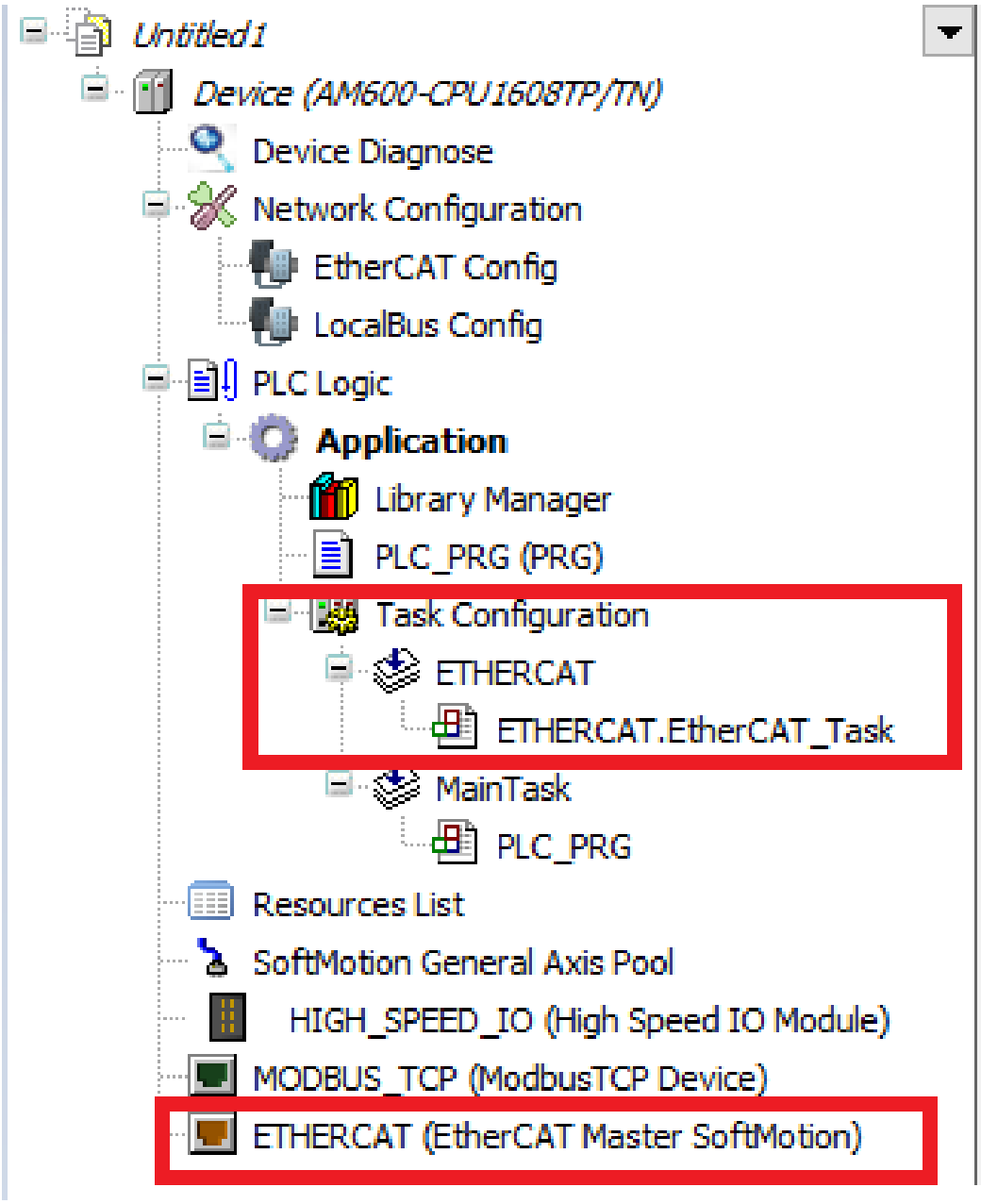


Рис. 4.63. EtherCAT

Для дальнейшей работы необходимо просканировать все устройства (рис. 4.64). Для этого нажмите ПКМ на вкладку EtherCAT и выберите «Сканировать устройства».

Если при сканировании возникнет ошибка, нужно загрузить проект в контроллер и выйти из него.

Если нет физического доступа к контроллеру для сканирования, можно вручную добавить приводы в проект. Во вкладке Network configuration в правой части необходимо добавить два привода SV660_1Axis.

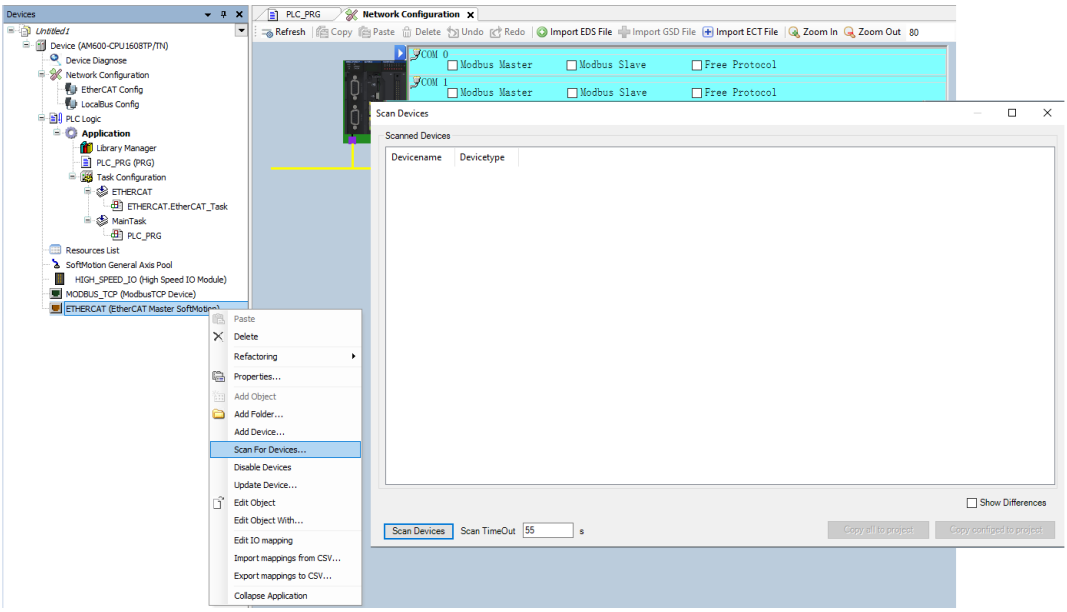


Рис. 4.64. Добавление приводов (сканирование)

После сканирования все подключенные устройства появятся во всплывающем окне (рис. 4.65). Нажмите кнопку Copy all to project.

Все устройства добавятся в дерево EtherCAT.

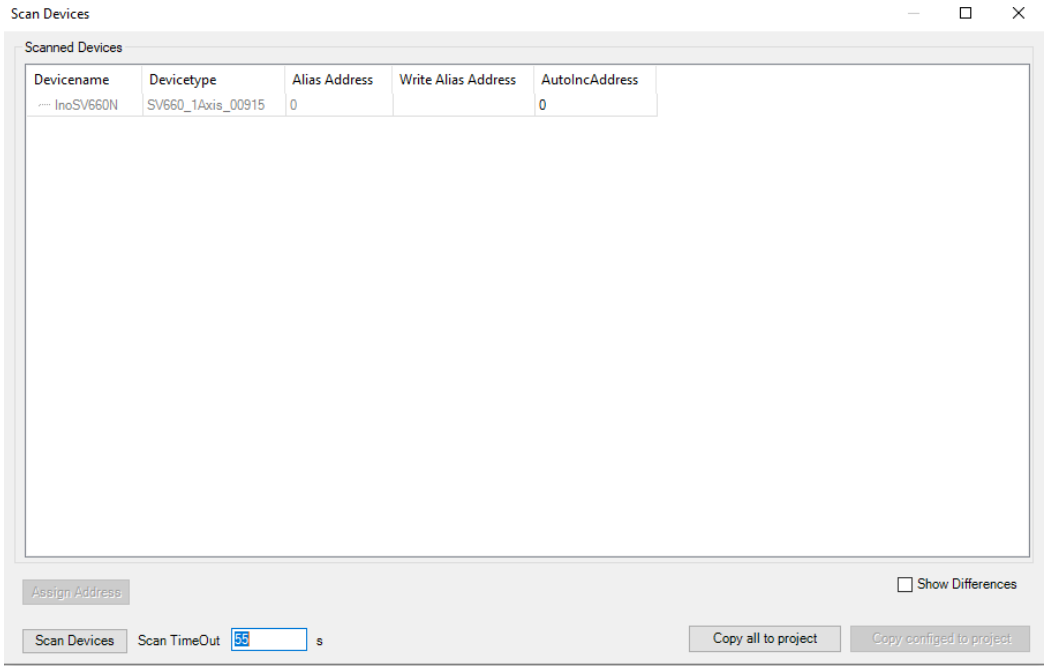


Рис. 4.65. Результат сканирования

Для управления сервоприводами SV660N необходимо разработанную (разрабатываемую) программу поместить в цикл задачи EtherCAT, как показано на рис. 4.66.

Для удобства дальнейшей работы с осями сервомоторов/сервоприводов переименуйте двигатель (рис. 4.66), например Drive.

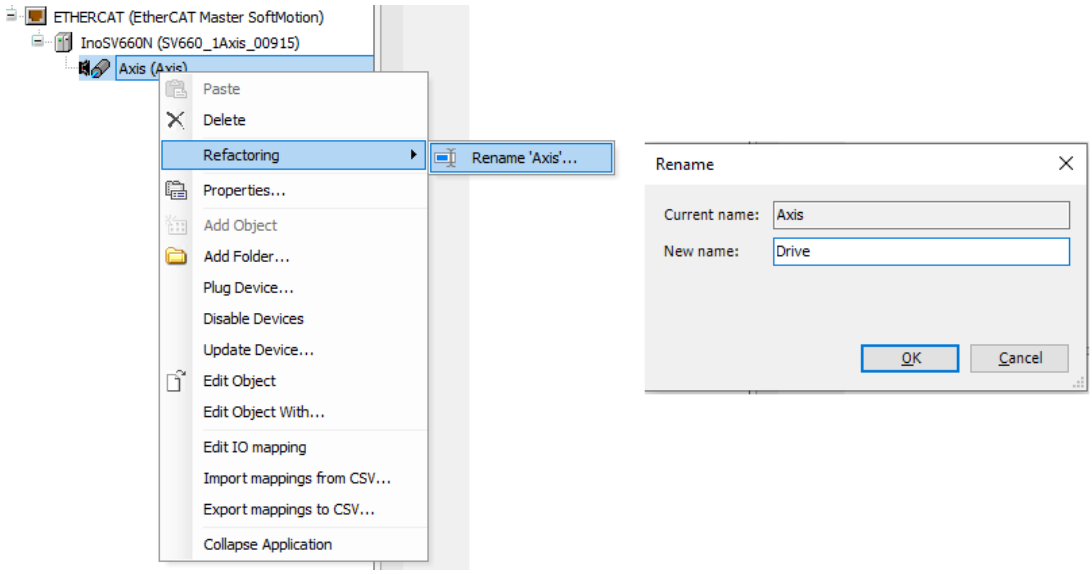


Рис. 4.66. Присвоение имени двигателю

Как только будет нажато подтверждение на переименование, на экране появится всплывающее окно с подтверждением изменения имени двигателя во всем проекте (рис. 4.67).

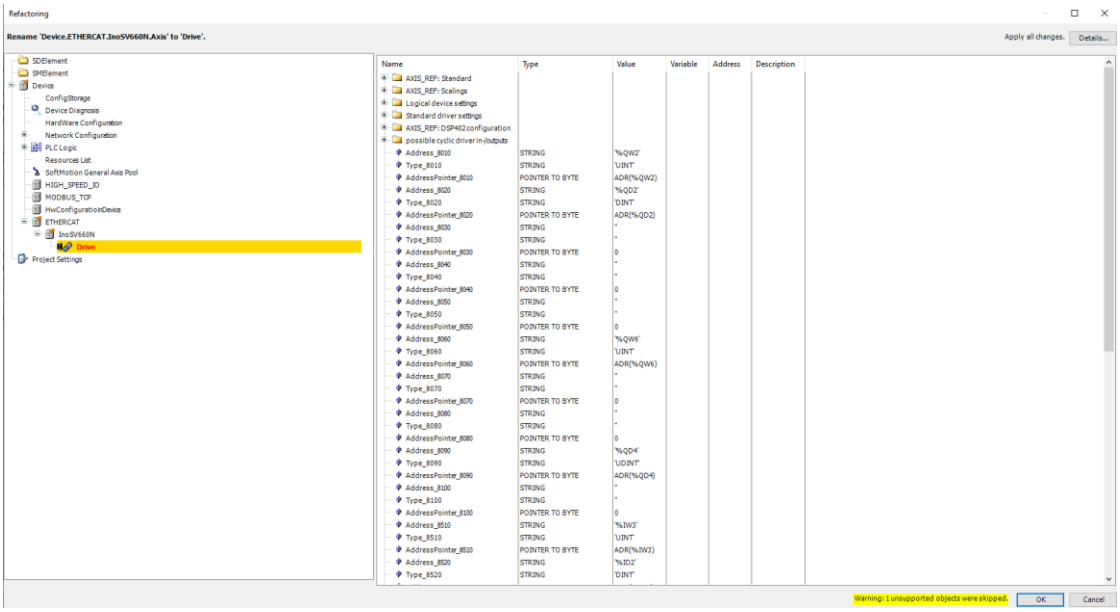


Рис. 4.67. Окно подтверждения имени двигателя

Приступим к изучению привода. С помощью двойного щелчка ЛКМ перейдите на настройку двигателя.

Во вкладке General отображены основные настройки двигателя (рис. 4.68):

1) выбор режима управления (Modulo – круговой/абсолютный; Finite – линейный/векторный);

2) активация лимитов для привода (скорость, дистанция и т. д.);

3) рампа разгона (по умолчанию выставлена трапеция) (рис. 4.69);

4) лимиты для системы с ЧПУ.

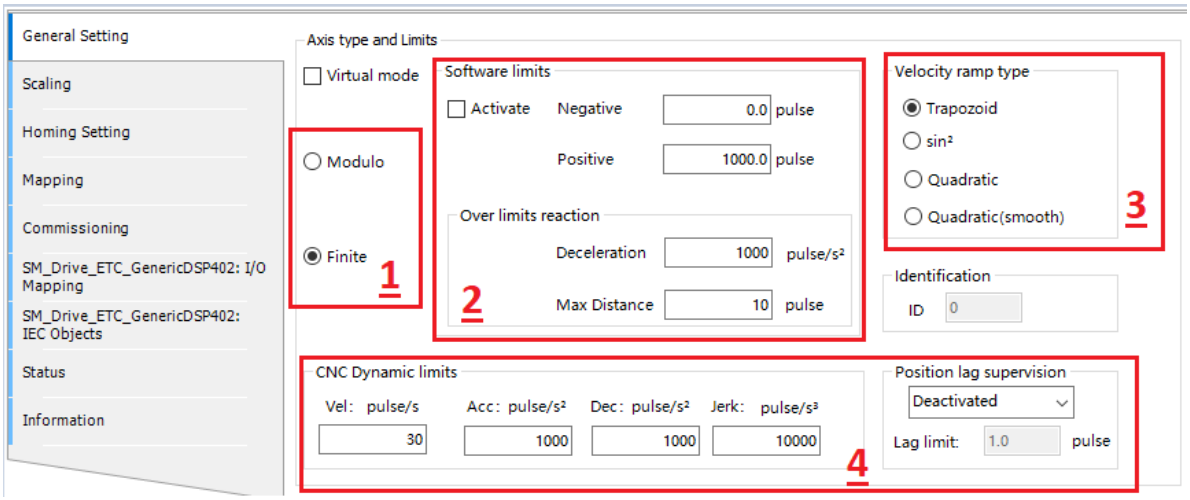


Рис. 4.68. Основные свойства двигателя

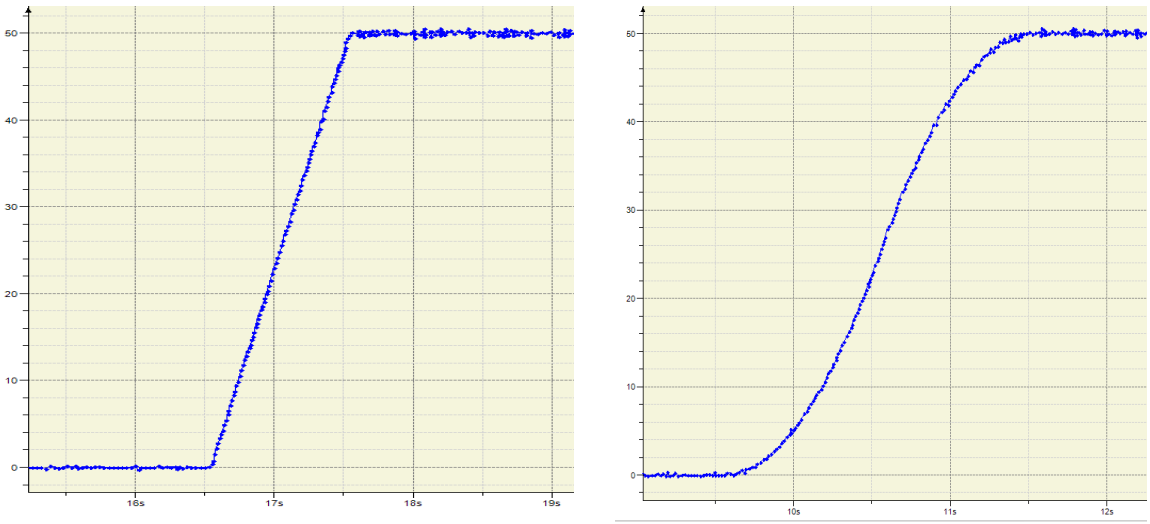


Рис. 4.69. Рампа разгона «трапеция» и «квадратичная»

Вкладка Scaling определяет (рис. 4.70):

1) выбор единиц измерения;

2) выставление разрешения энкодера за оборот;

3) выбор рабочего расстояния за один оборот двигателя в случае, если известен коэффициент редукции исполнительного вала с механизмом;

4) расчет коэффициента редукции (рис. 4.71).

Вкладка Homing Settings используется для выхода в ноль по энкодеру. Существуют различные вариации выхода в ноль по энкодеру: по z-сигналу, концевому выключателю и т. д.

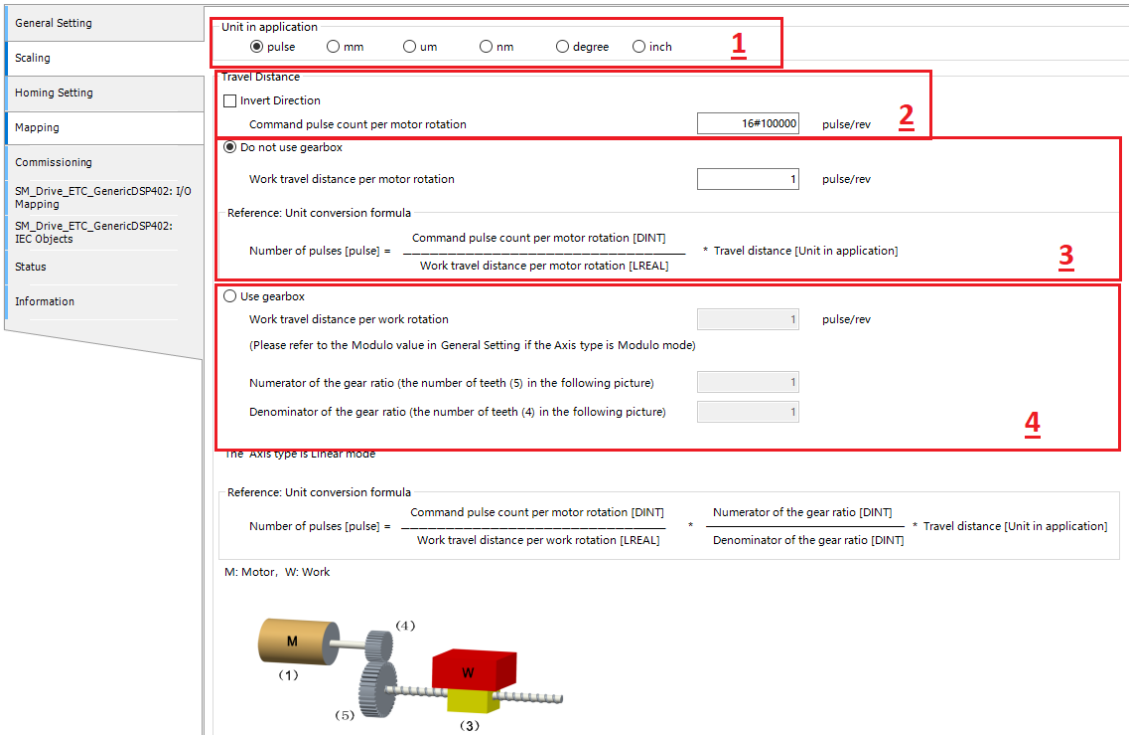


Рис. 4.70. Вкладка Scaling

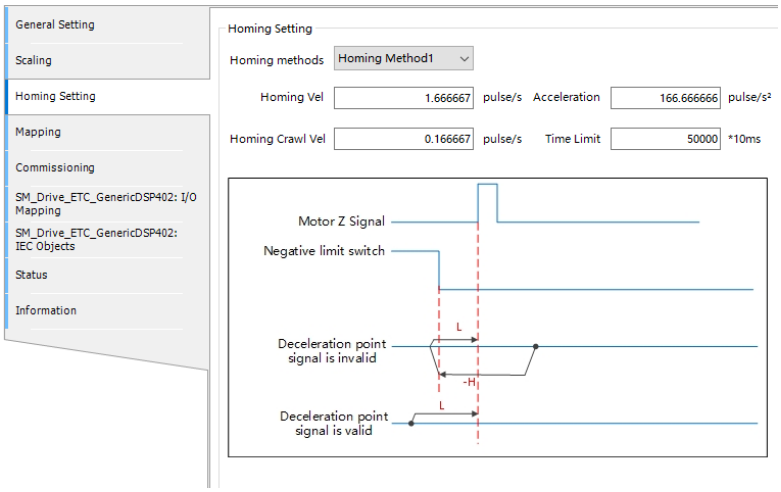


Рис. 4.71. Расчет коэффициента редукции

Для программирования выберите единицу измерения – миллиметры, мм (рис. 4.72).

Рис. 4.72. Выбор единицы измерения

Следует выставить разрешение энкодера и пройденное расстояние исполняющего механизма за оборот. На всех приводах SV660N стоят 23-битные энкодеры. Для этого в поле нужно ввести 800 000, как показано на рис. 4.73, иначе привод будет работать некорректно.

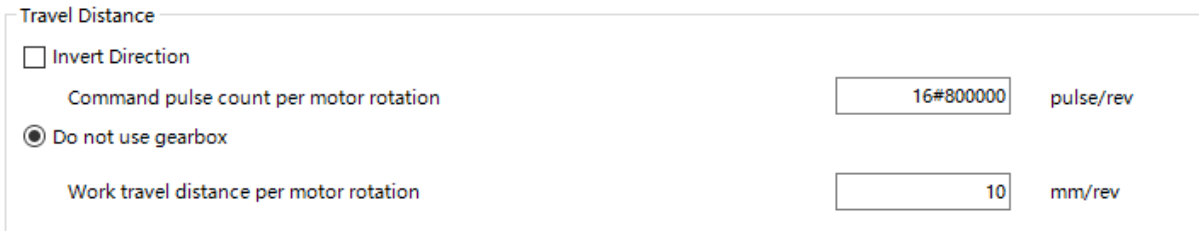


Рис. 4.73. Выбор разрешения энкодера

Вкладка самого привода состоит из вкладок, отраженных на рис. 4.74:

1) автоматическая нумерация и время вызова в программе;

2) PDO – параметры, которые можно изменять во время работы привода. На каждый параметр выделяется уникальный адрес (рис. 4.75);

3) SDO – параметры, которые записываются однократно при включении программы в контроллере (рис. 4.75).

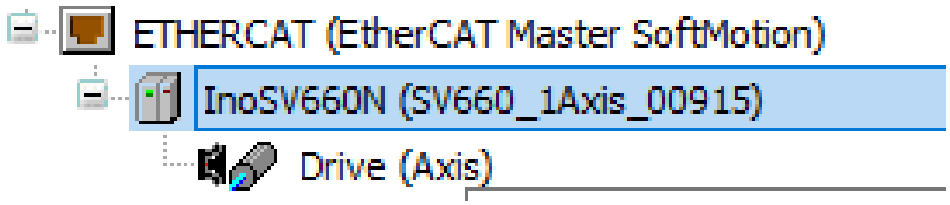


Рис. 4.74. Вкладка привода

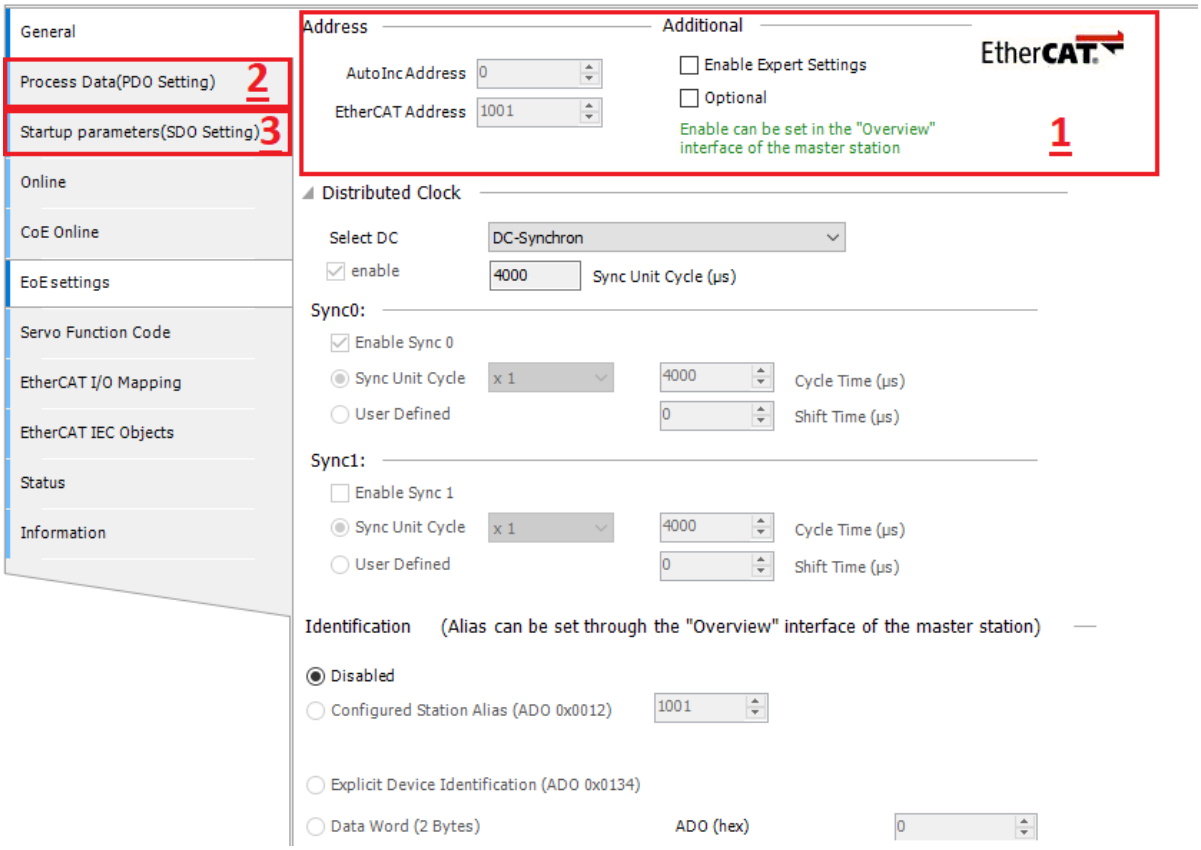


Рис. 4.75. PDO и SDO

Ниже представлены варианты программирования функционального кода и значения скорости (табл. 4.4).

Таблица 4.4

Программирование функционального кода

Порядковый номер	Скорость сервопривода, об./с
1	15
2	25
3	30
4	45
5	55
6	50
7	20
8	35
9	40
10	60

Программирование будет осуществляться через логическую операцию CASE.

Оператор CASE – это оператор множественного ветвления. В зависимости от некоторого условия может быть выполнено одно из многочисленных продолжений программы.

Для включения привода потребуется:

1) переменная, имеющая тип INT, для переключения шага;

2) функциональный блок MC_Power для включения питания привода;

3) функциональный блок MC_MoveVelocity для управления приводом по скорости;

4) функциональный блок MC_Halt для корректного останова.

Все функциональные блоки находятся во вкладке Motion Control (рис. 4.76, 4.77).

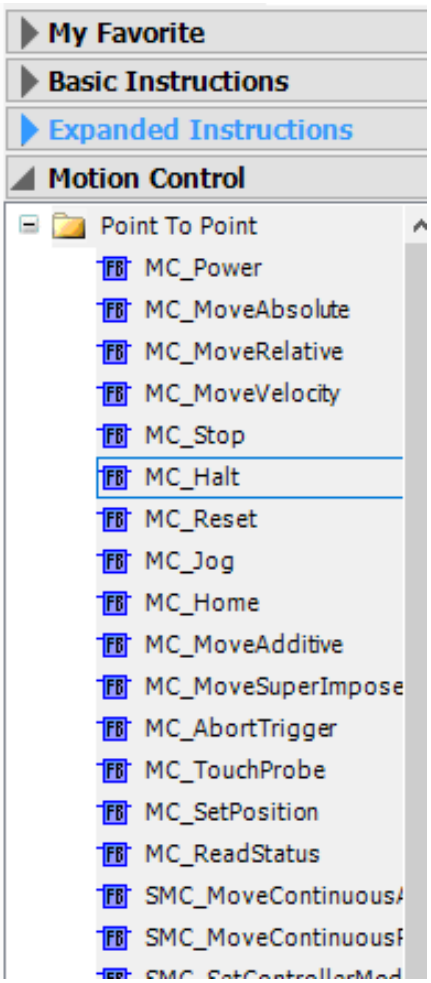


Рис. 4.76. Вкладка Motion Control

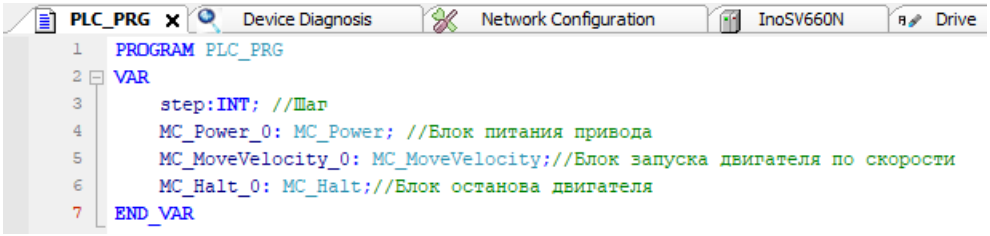


Рис. 4.77. Функциональные блоки

На нулевом шаге при включении будет инициализироваться привод (рис. 4.78). Без его загрузки дальнейшее движение будет невозможно. После того как привод загрузится на 100 %, перейдите на шаг 1 (step:=1).

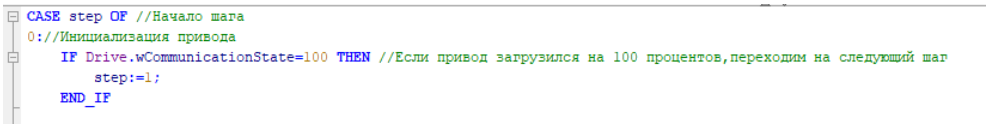


Рис. 4.78. Инициализация привода

На первом шаге включается привод по питанию (рис. 4.79). Функциональный блок включает питание, выключает тормоза, включает модуляцию у привода.

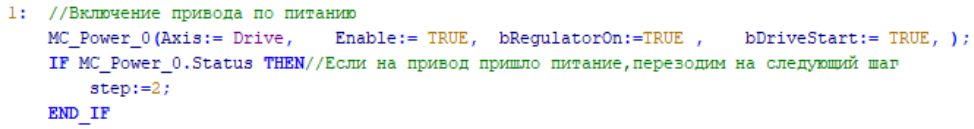


Рис. 4.79. Управление приводом по питанию

Выходное значение Status сигнализирует о том, что функциональный блок отработал. Далее происходит переход к шагу 2 (step:=2).

Второй шаг – управление приводом по скорости (рис. 4.80). Функциональный блок MC_MoveVelocity получает входные значения, а именно:

Axis – ось вращения;

Execute – разрешение на включение блока;

Velocity – скорость, с которой привод будет работать;

Acceleration – разгон (в единицах измерения);

Deceleration – торможение (в единицах измерения);

Jerk – толчок (в единицах измерения), влияет на плавность разгона рампы;

Direction – направление вращения вала двигателя.

```
2:  //Включение управления привода по скорости
    MC_MoveVelocity_0(Axis:= Drive, Execute:= TRUE, Velocity:= 50, Acceleration:=50 , Deceleration:=50 , Jerk:= 50, Direction:=positive ,);

3: //Останов привода
    MC_Halt_0(Axis:= Drive,Execute:= TRUE,Deceleration:= 50,Jerk:= 10);
END_CASE
```

Рис. 4.80. Управление приводом по скорости

Третий шаг – останов двигателя. Функциональный блок MC_Halt имеет следующие входные значения:

Axis – ось вращения;

Execute – разрешение на включение блока;

Deceleration – торможение (в единицах измерения);

Jerk – толчок (в единицах измерения), влияет на плавность разгона рампы.

После написания кода загрузите проект в контроллер и запустите программу.

В режиме онлайн-отслеживания можно открыть вкладку двигателя. Во вкладке появится дополнительная колонка с отслеживанием параметров и ошибок (рис. 4.81).

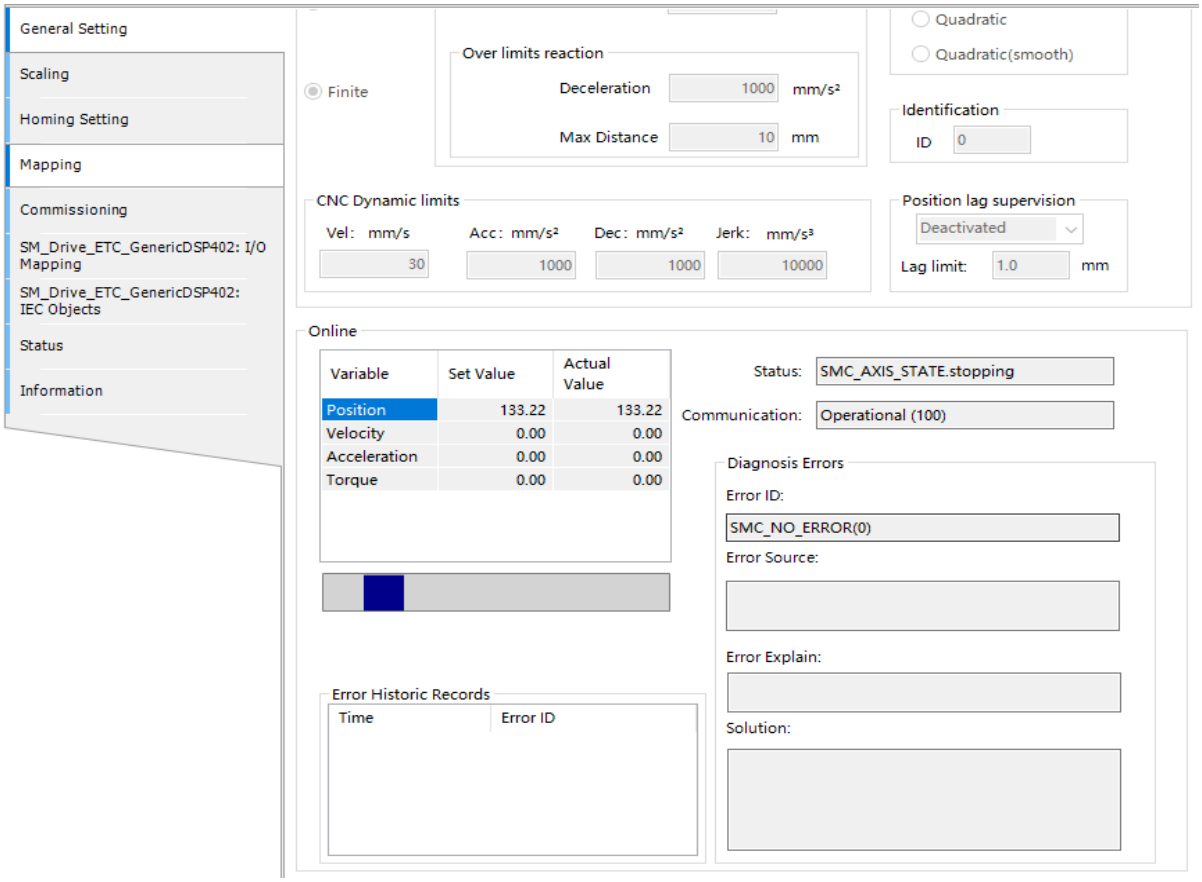


Рис. 4.81. Вкладка Mapping

Для того чтобы произвести останов двигателя, необходимо в режиме онлайн в столбце Prepared value выставить шаг 3 (рис. 4.82) и нажать CTRL+F7 для записи значений либо зайти сверху во вкладку Debug и нажать Write values.

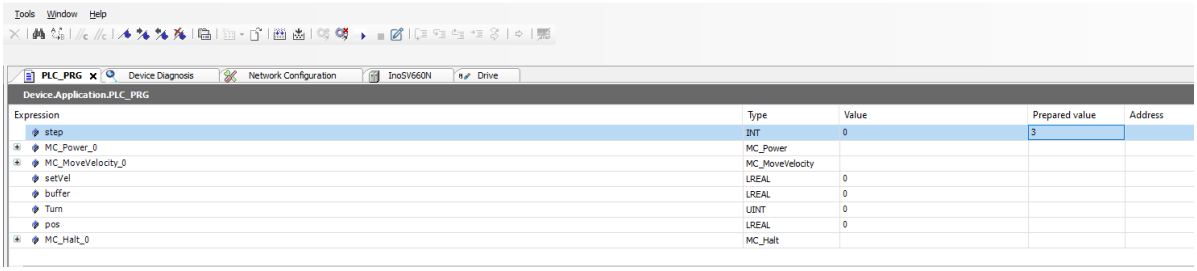


Рис. 4.82. Окно с общим видом запрограммированных действий

Аналогичные действия выполняют со вторым приводом. Для каждого привода нужны новые функциональные блоки управления и, соответственно, новый код.

Содержание работы

1. Цель работы.
2. Задания.
3. Ход работы.
4. Выводы.
5. Контрольные вопросы.

Средства, используемые при выполнении лабораторной работы

1. Методические указания к выполнению лабораторных работ.
2. Данные, предоставленные преподавателем во время занятия.
3. При проведении анализа допускается использование глобальной сети Интернет.

Контрольные вопросы

1. Для чего нужна задача EtherCAT?
2. Как просканировать все устройства?
3. Что нужно делать, если появилась ошибка при сканировании?
4. Чем отличается режим управления Modulo от Finite?
5. Чем различаются рампы разгона «трапеция» и «квадратичная»?
6. Какой энкодер стоит на двигателе?
7. Чем отличаются PDO параметры от SDO параметров?
8. Что такое оператор CASE?
9. Какие функциональные блоки использовались в программе, разработанной в ходе лабораторной работы?
10. Какие вводные параметры используются для функционального блока MC_MoveVelocity? Что они помогают выполнять?

Лабораторная работа № 4

СЕРВОПРИВОДЫ С АБСОЛЮТНЫМ УПРАВЛЕНИЕМ

Цель работы: освоить способ программирования сервоприводов с абсолютным управлением.

Задания

1. Запрограммировать два сервопривода.
2. Протестировать написанную программу на учебном стенде Inovance.

Ход работы

Программирование сервоприводов с абсолютным управлением

Создайте проект с приводами, придерживаясь той же последовательности, что и в лабораторной работе № 3. Выберите привод.

Выставьте управление по позиции (Modulo) и укажите значение Modulo value – 360 градусов (рис. 4.83).

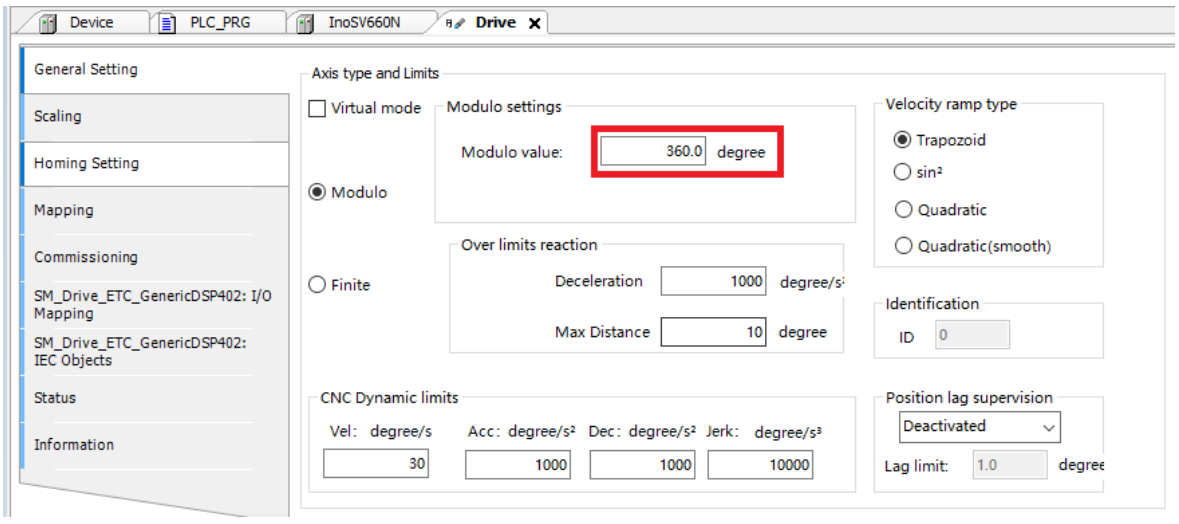


Рис. 4.83. Установка значения Modulo value

Во вкладке Scaling для программирования выберите единицы измерения – градусы (degree).

Необходимо выставить разрешение энкодера и количество градусов за оборот для исполняющего механизма (рис. 4.84). На всех приводах SV660N стоят 23-битные энкодеры. Для этого в поле следует ввести 800 000, как показано на рис. 4.84, иначе привод будет работать некорректно.

Рис. 4.84. Установка значения энкодера

Будем изменять положение сервопривода с помощью штурвала, который расположен на стенде. Для этого во вкладке высокоскоростных входов выберите счетчик «0» и выставьте название счетчика (рис. 4.85).

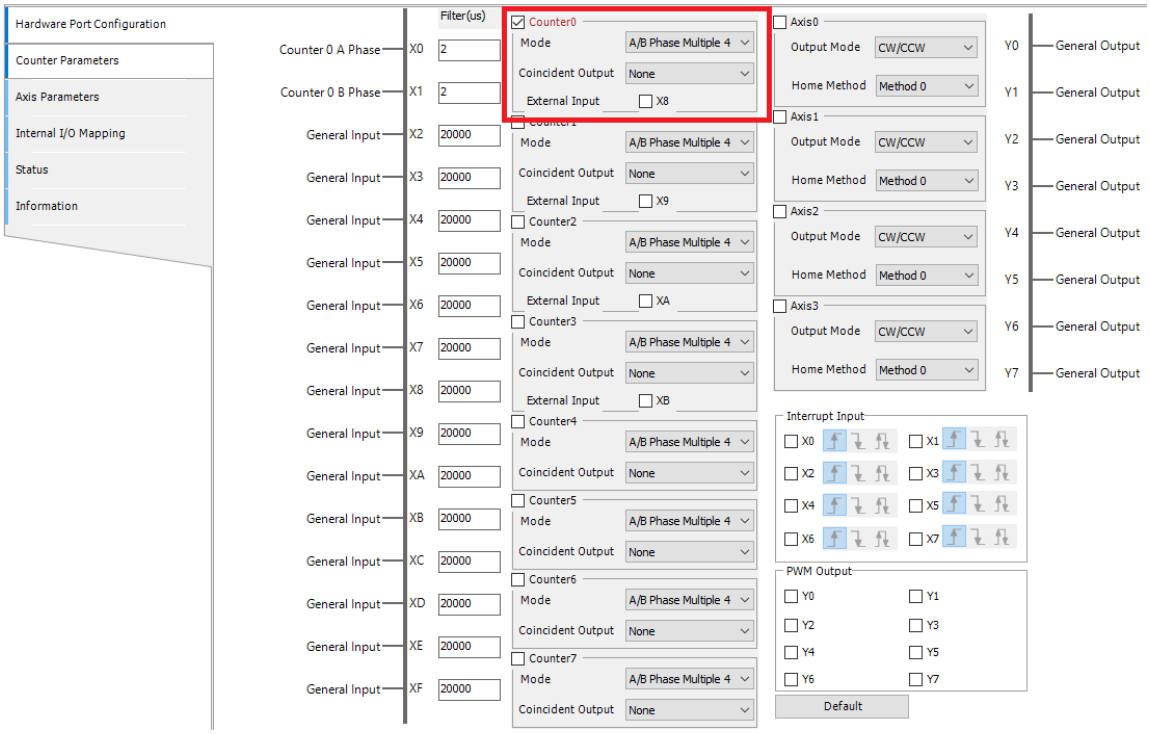


Рис. 4.85. Выбор счетчика

Во вкладке «Настройки счетчика» (рис. 4.86) выберите формат – линейный, период опроса – 10 мс.

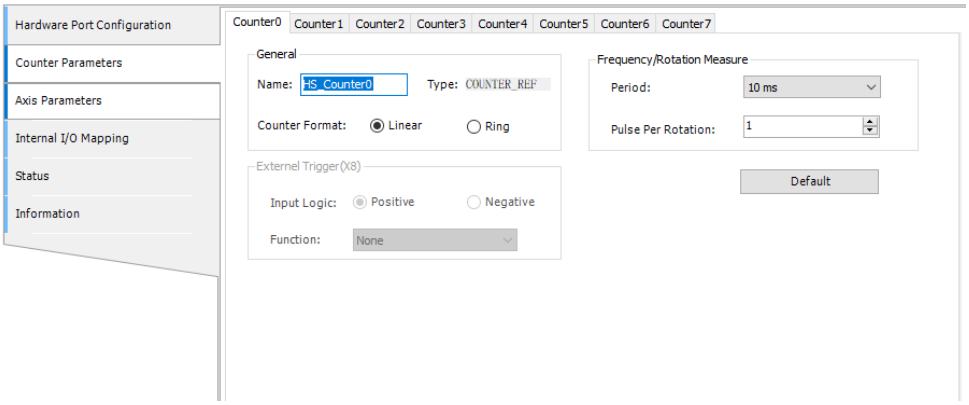


Рис. 4.86. Настройка счетчика

Ниже представлены варианты программирования функционального кода и значения скорости сервопривода (табл. 4.5).

Варианты:

Таблица 4.5

Программирование функционального кода

Порядковый номер	Скорость сервопривода, об./с
1	120
2	125
3	130
4	145
5	155
6	150
7	120
8	135
9	140
10	160

Программирование будет выполняться через логическую операцию CASE (рис. 4.87).

Для включения привода потребуется:

1) переменная, имеющая тип INT, для переключения шага;

2) функциональный блок MC_Power для включения питания привода;

3) функциональный блок MC_MoveAbsolute для управления приводом по положению;

4) функциональный блок MC_Halt для корректного останова;

5) функциональный блок HS_Counter для считывания значений со штурвала.

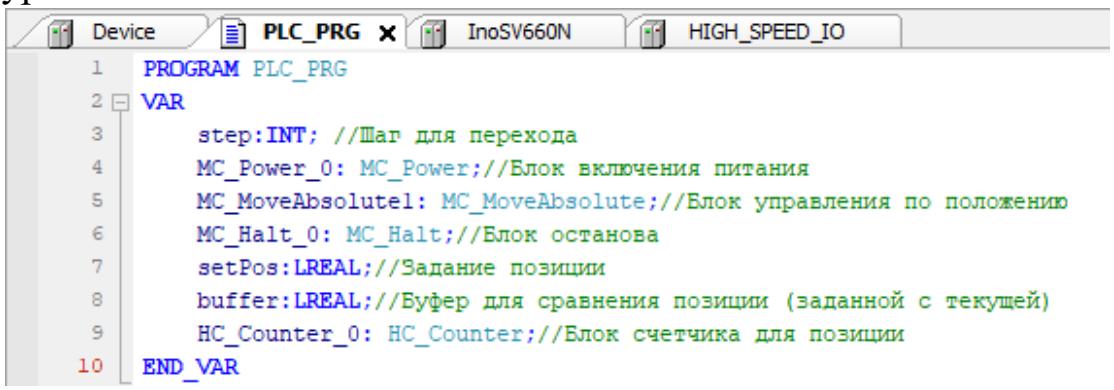


Рис. 4.87. Программирование включения привода

Далее выполните действия в соответствии с рис. 4.78 – 4.80. После чего задайте настройки в соответствии с рис. 4.88, 4.89.

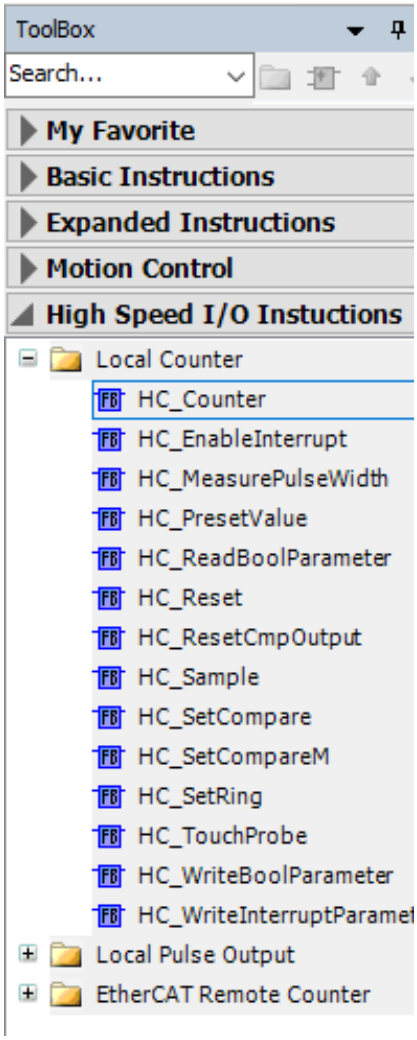


Рис. 4.88. Добавление

блока для штурвала

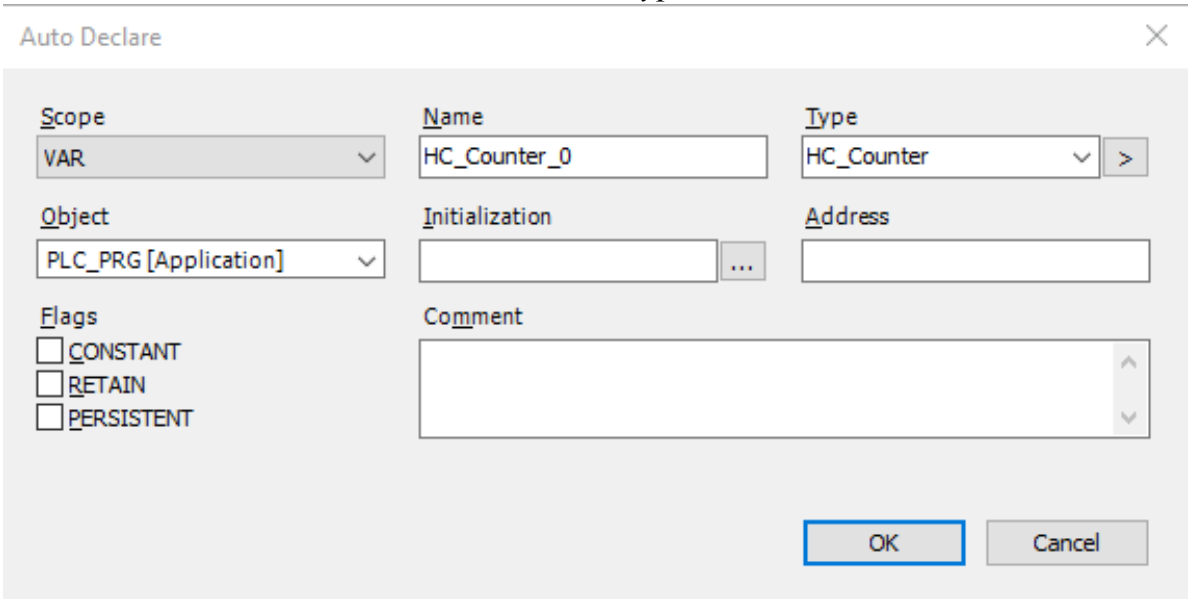


Рис. 4.89. Выставление значений в блоке

В блоке необходимо выставить входные и выходные параметры:

Counter – название счетчика из высокоскоростных входов;

Enable – включение блока (поставьте условие: когда на привод будет подано питание, тогда будет включаться блок);

Direction – направление вращения (когда направление пустое, может двигаться в любом направлении);

CounterValue – выходное значение счетчика. Значение градуса поворота штурвала, привязанного к сервоприводу. Задайте ранее созданную переменную «setPos» (рис. 4.90).

```
//блок счетчика
HC_Counter_0(Counter:=HS_Counter0,Enable:= MC_Power_0.Status,Direction:= ,CounterValue=> setPos);
```

Рис. 4.90. Команда «setPos»

Для изменения позиции необходимо выключить функциональный блок и включить его заново, задав новые значения.

Выполните следующие действия:

Создайте переменную-буфер (рис. 4.91) и сравните значения текущей позиции с заданной.

Для выключения и включения блока используйте входную переменную блока Execute.

После выполненных действий необходимо приравнять значения буфера к заданным значениям, чтобы программа до следующего изменения положения не заходила в код.

```
//Буффер для сравнения
IF buffer<>setPos THEN//Если значения не равны
    MC_MoveAbsolutel(Axis:=Drive,Execute:= FALSE,Position:= setPos,Velocity:= 500,Acceleration:= 500,Deceleration:=500,Jerk:= 500, Direction:=);//Выключаем блок
    MC_MoveAbsolutel(Axis:=Drive,Execute:= TRUE,Position:= setPos,Velocity:= 500,Acceleration:= 500,Deceleration:=500,Jerk:= 500, Direction:=);//Включаем блок с новыми значениями
    buffer:=setPos;//Прироваиваем буфер и текущее значение
END_IF

```

Рис. 4.91. Буфер

Далее выполните действия в соответствии с рис. 4.81 – 4.82.

Аналогичные действия выполняют со вторым приводом. Для каждого привода нужны новые функциональные блоки управления и, соответственно, новый код.

Содержание работы

1. Цель работы.
2. Задания.
3. Ход работы.
4. Выводы.
5. Контрольные вопросы.

Средства, используемые при выполнении лабораторной работы

1. Методические указания к выполнению лабораторных работ.
2. Данные, предоставленные преподавателем во время занятия.
3. При проведении анализа допускается использование глобальной сети Интернет.

Контрольные вопросы

1. Что такое Modulo в приводе?
2. Какое разрешение имеют энкодеры в приводах SV660N?
3. Из чего состоит функциональный блок HS_Counter?
4. Для чего служит переменная-буфер?

Лабораторная работа № 5

УПРАВЛЕНИЕ СЕРВОПРИВОДАМИ С ПОТЕНЦИОМЕТРА И ПАНЕЛИ

Цель работы: научиться программировать сервоприводы с потенциометра и панели.

Задания

1. Изучить потенциометр.
2. Создать интерфейс управления сервоприводами.
3. Запрограммировать управление сервоприводом с потенциометра.
4. Выполнить выданное преподавателем итоговое задание по примеру лабораторной работы № 2.

Ход работы

Разработка и создание реального проекта

Создайте проект и добавьте протокол EtherCAT во вкладку Network configuration, выполните действия, до этого указанные на рис. 4.62.

Если нет физического доступа к контроллеру для сканирования, можно вручную добавить приводы в проект. Во вкладке Network configuration в правой части необходимо добавить все устройства, которые установлены на стенде, в соответствии с инструкциями, представленными на рис. 4.64 – 4.65.

Все устройства добавятся в дерево EtherCAT. После сканирования дерево проекта должно выглядеть так, как на рис. 4.92.

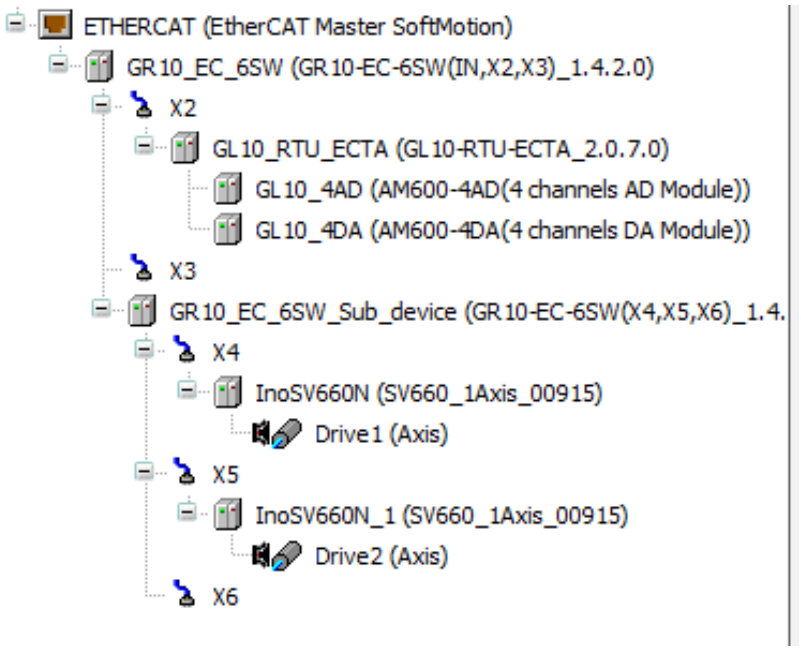


Рис. 4.92. Устройства в проекте

Для удобства работы с осями переименуйте двигатели, например: Drive1 и Drive2, в соответствии с рис. 4.66.

Перейдите к настройке привода, а затем двойным щелчком ЛКМ – к настройке двигателя.

Выставьте управление по скорости (Finite) (рис. 4.93).

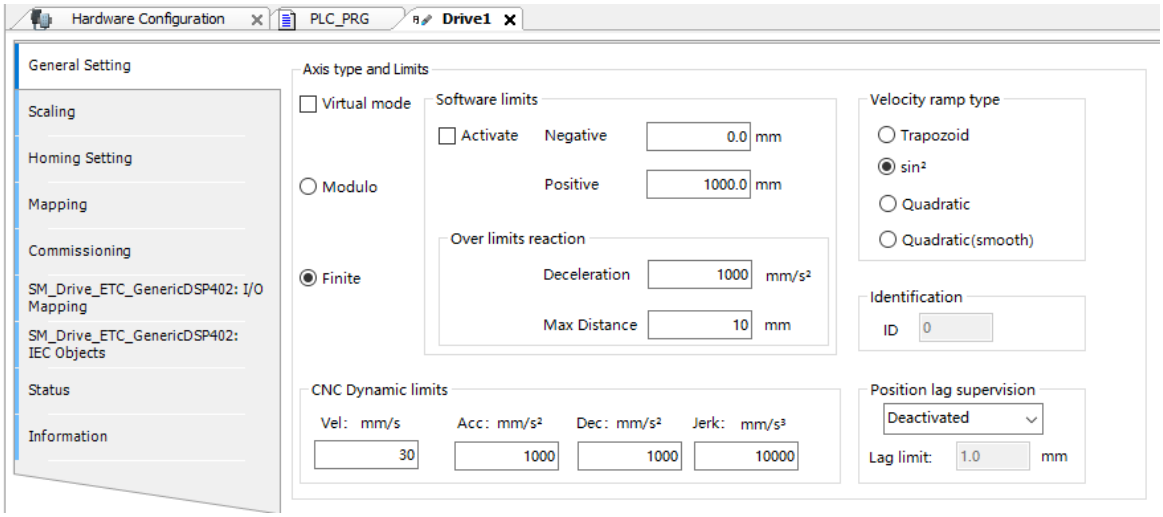


Рис. 4.93. Установка управления по скорости

Во вкладке Scaling для программирования выберите единицы измерения – мм.

Необходимо выставить разрешение энкодера в соответствии с рис. 4.73.

Затем перейдите во вкладку Homing setting и выберите метод выхода в ноль энкодера. Так как концевого выключателя нет, единственный метод, который можно выбрать, – это по z-сигналу. Этот метод имеет номера 33 и 34. Выберите метод 34 и выставьте скорость, ускорение и максимальное время нахождения нулевой точки (рис. 4.94).

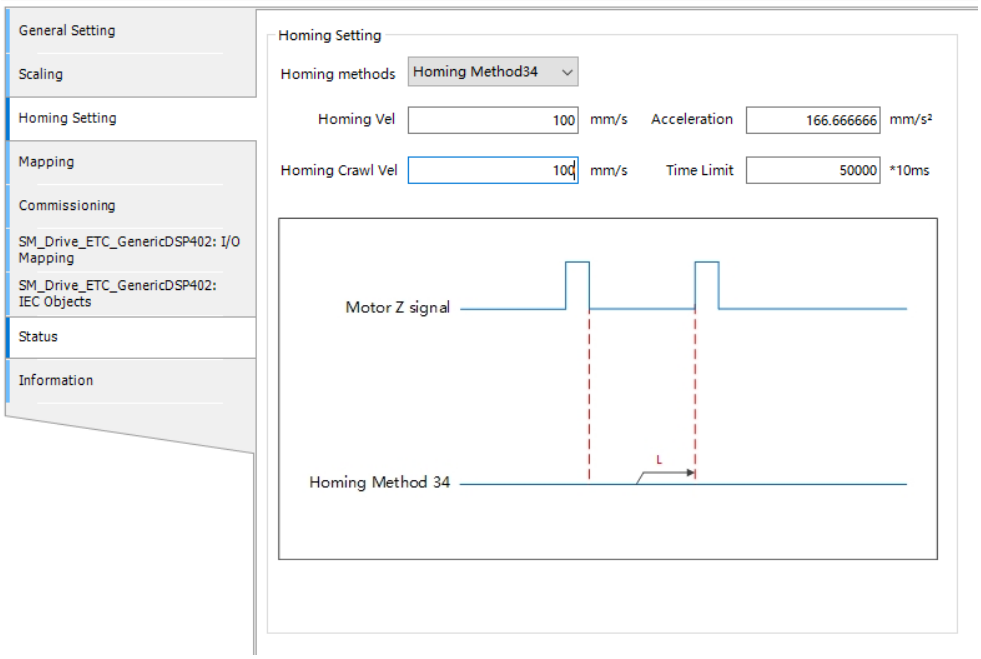


Рис. 4.94. Выбор метода

Переместите управляющую программу (рис. 4.95) в задачи EtherCAT.

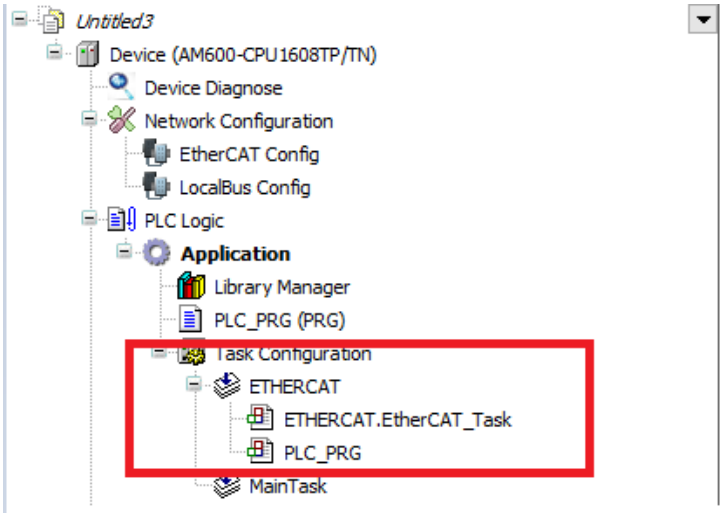


Рис. 4.95. Перемещение управляющей программы

Программирование функционального кода

Напишите код, по которому приводы будут включаться, управляться по скорости с помощью потенциометра, выходить в ноль, останавливаться, выключаться (рис. 4.96). Все это управление реализуется на панели.

Варианты для выполнения лабораторной работы представлены в табл. 4.6.

Таблица 4.6

Варианты заданий

Номер варианта	Значение скорости выхода в ноль	Рампа разгона привода
1	120	Трапеция
2	100	Синусоидальная
3	115	Квадратичная
4	90	»
5	105	Синусоидальная
6	110	Трапеция
7	120	Квадратичная
8	125	Синусоидальная
9	105	»
10	100	Трапеция

Hardware Configuration | PLC_PRG | Drive1 | Drive2 | GL10_4AD | GL10_RTU

```
PROGRAM PLC_PRG
VAR
//-----------------Программные переменные----------------
    BItStart:BOOL; //Команда на разрешения работы
    step:INT; //Шаг
    setVel:LREAL;//Задание скорости
    buffer:LREAL;//Буфер для сравнения
    InValueVel:INT;//Прием значения с аналогово входа
    Connection1:WORD;//Переменная для отслеживания статуса загрузки первого привода
    Connection2:WORD;//Переменная для отслеживания статуса загрузки второго привода
    Speed1:LREAL;//Переменная отслеживания скорости 1 привода
    Speed2:LREAL;//Переменная отслеживания скорости 2 привода
    Power:BOOL;//Переменная для отслеживания статуса включения питания
    Invelocity:BOOL;//Переменная для отслеживания статуса выхода на скорость
    Alarm:BOOL; //Авария приводов
//--------------Функциональные блоки--------------
    MC_Power_0: MC_Power; //Блок питания 1 привода
    MC_Power_1: MC_Power;//Блок питания 2 привода
    MC_MoveVelocity_0: MC_MoveVelocity;//Блок запуска 1 двигателя по скорости
    MC_MoveVelocity_1: MC_MoveVelocity;//Блок запуска 2 двигателя по скорости
    MC_Home_0: MC_Home;//Блок выхода в ноль 1 привода
    MC_Home_1: MC_Home;//Блок выхода в ноль 2 привода
    MC_Reset_0: MC_Reset;//Блок перезагрузки 1 привода
    MC_Reset_1: MC_Reset;//Блок перезагрузки 2 привода
    MC_Halt_0: MC_Halt;//Блок останова 1 привода
    MC_Halt_1: MC_Halt;//Блок останова 2 привода
END_VAR
```

Рис. 4.96. Объявление переменных

В программе используются следующие блоки:

MC_Power – блок включения питания;

MC_MoveVelocity – блок управления по скорости;

MC_Home – блок выхода в ноль;

MC_Reset – блок перезагрузки привода;

MC_Halt – блок корректного останова.

В теле программы создайте условие для активации кейса.

На нулевом шаге при включении будут инициализироваться приводы (рис. 4.97). Без их загрузки дальнейшее движение будет невозможно. После того как приводы загрузились на 100 %, перейдите на шаг 1.

```
//Кнопка разрешения работы
IF BItStart THEN//Если кнопка нажата то выполняется кейс

    CASE step OF
0:  //Инициализация приводов
    IF Drive1.wCommunicationState=100 AND Drive2.wCommunicationState=100 THEN
        step:=1;
    END_IF
```

Рис. 4.97. Инициализация приводов

Первый шаг – включение приводов по питанию (рис. 4.98). Функциональный блок включает питание, выключает тормоза, включает модуляцию привода.

```
1:  //Включение питания
    MC_Power_0(Axis:= Drive1,   Enable:= TRUE,  bRegulatorOn:=TRUE ,    bDriveStart:= TRUE, );
    MC_Power_1(Axis:= Drive2,   Enable:= TRUE,  bRegulatorOn:=TRUE ,    bDriveStart:= TRUE, );
```

Рис. 4.98. Включение приводов по питанию

Второй шаг – управление приводом по скорости (рис. 4.99).

```
2:  //Управление по скорости
    MC_MoveVelocity_0(Axis:= Drive1, Execute:= TRUE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 50, Direction:=positive);
    MC_MoveVelocity_1(Axis:= Drive2, Execute:= TRUE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 50, Direction:=positive);
```

Рис. 4.99. Включение приводов по скорости

Третий шаг – выход в ноль (рис. 4.100). Перед тем как вызвать блок выхода в ноль, необходимо остановить привод.

```
3:  //Выход в ноль
    MC_Halt_0(Axis:= Drive1,Execute:= TRUE,Deceleration:=50,Jerk:= 10);
    MC_Halt_1(Axis:= Drive2,Execute:= TRUE,Deceleration:=50,Jerk:= 10);
    IF MC_Halt_0.Done THEN
        MC_Home_0(Axis:= Drive1,Execute:= TRUE,Position:= 0);
        MC_Home_1(Axis:= Drive2,Execute:= TRUE,Position:= 0);
        IF MC_Home_0.Done AND MC_Home_1.Done THEN
            MC_Home_0(Axis:= Drive1,Execute:= FALSE);
            MC_Home_1(Axis:= Drive2,Execute:= FALSE);
            step:=1;
        END_IF

    END_IF
```

Рис. 4.100. Выход в ноль

Блок MC_Home имеет следующие вводные параметры:
Axis – ось, которая будет осуществлять выход в ноль;
Execute – разрешение на работу блока;
Position – позиция от заводского нуля.
Четвертый шаг – перезагрузка приводов (рис. 4.101).

```
4:  //Перезагрузка приводов
    MC_Reset_0(Axis:= Drive1, Execute:=TRUE);
    MC_Reset_1(Axis:= Drive2, Execute:=TRUE);
    IF MC_Reset_0.Done OR MC_Reset_1.Done OR MC_Reset_0.Error OR MC_Reset_1.Error THEN
        MC_Reset_0(Axis:= Drive1, Execute:=FALSE);
        MC_Reset_1(Axis:= Drive2, Execute:=FALSE);
        step:=1;
    END_IF
```

Рис. 4.101. Перезагрузка приводов

Для перезагрузки приводов необходим функциональный блок MC_Reset. Его входные параметры:
Axis – ось, которую блок будет перезагружать;
Execute – разрешение работы блока.
Пятый шаг – корректный останов приводов (рис. 4.102).

```
5:  //Корректный останов
    MC_Halt_0(Axis:= Drive1,Execute:= TRUE,Deceleration:=50,Jerk:= 10);
    MC_Halt_1(Axis:= Drive2,Execute:= TRUE,Deceleration:=50,Jerk:= 10);
    IF MC_Halt_0.Done AND MC_Halt_1.Done THEN
        step:=1;
    END_IF
```

Рис. 4.102. Программирование корректного останова

Шестой шаг – выключение приводов (рис. 4.103).

```
6: //Выключение питания
    MC_Power_0(Axis:= Drive1,   Enable:= TRUE,  bRegulatorOn:=FALSE ,   bDriveStart:= FALSE)
    MC_Power_1(Axis:= Drive2,   Enable:= TRUE,  bRegulatorOn:=FALSE ,   bDriveStart:= FALSE)

END_CASE

END_IF
```

Рис. 4.103. Программирование выключения приводов

Для выставления скорости с потенциометра необходимо добавить переменную в модуль аналогового входа (рис. 4.104).

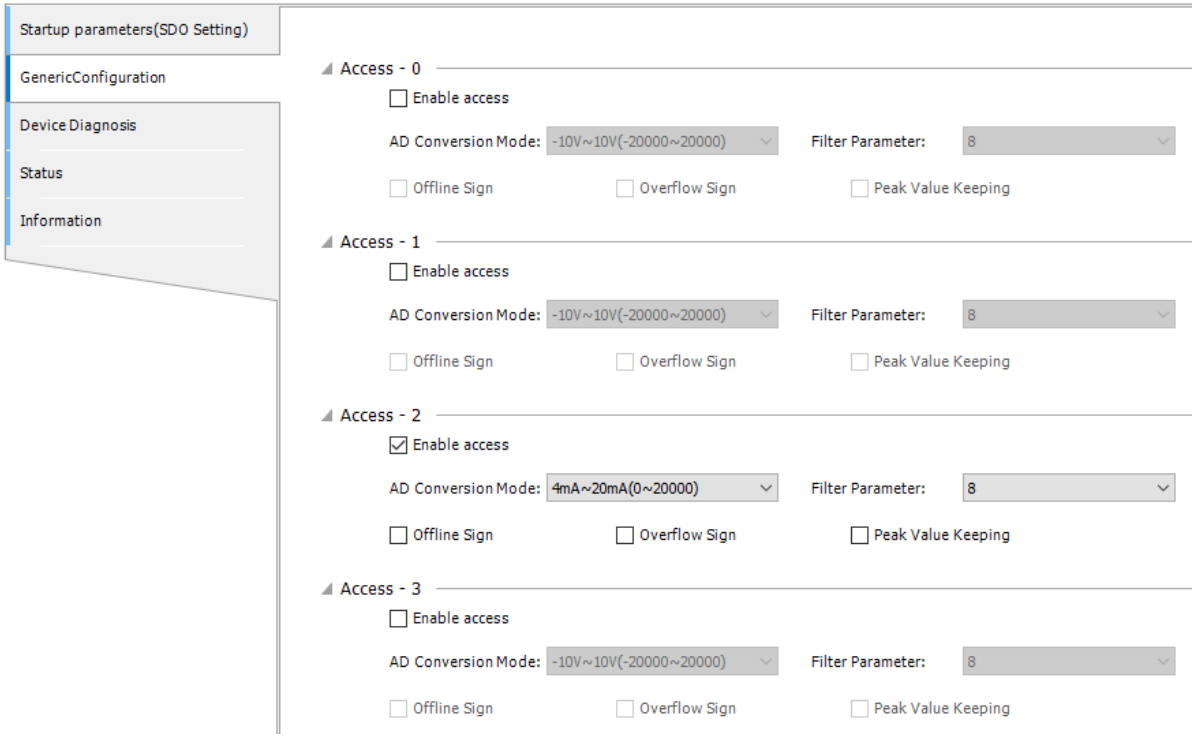


Рис. 4.104. Установка значений модуля

Зайдите в модуль и переместитесь во вкладку Generic Configuration.

Потенциометр по электрической схеме подключен ко второму каналу и имеет сигнал 4 – 20 мА, поэтому выставьте активацию второго канала и прием сигнала (4 – 20 мА).

Перейдите в Coupler (рис. 4.105), который соединяет все локальные устройства, подключенные к нему, и зайдите во вкладку Mapping для добавления переменной на канал аналогового входа.

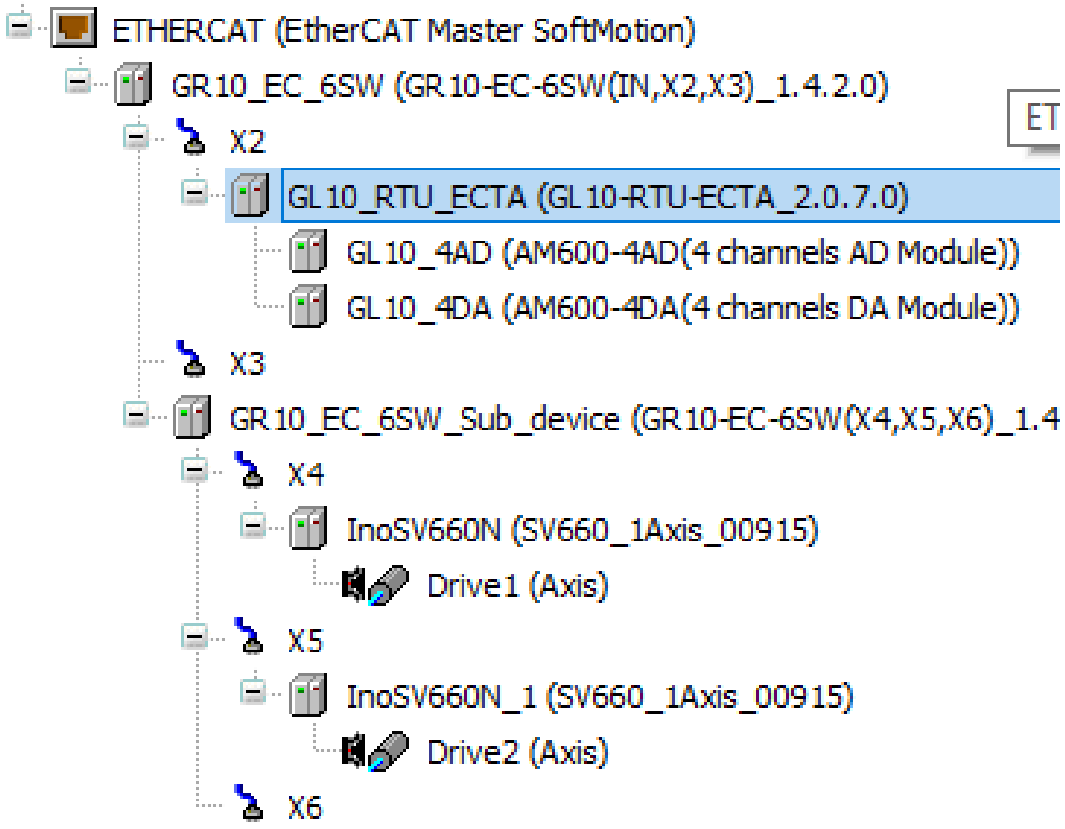


Рис. 4.105. Переход в коплер

Выберите второй канал, щелкнув в столбце Variable два раза ЛКМ. Нажмите на три точки для добавления переменной (рис. 4.106).

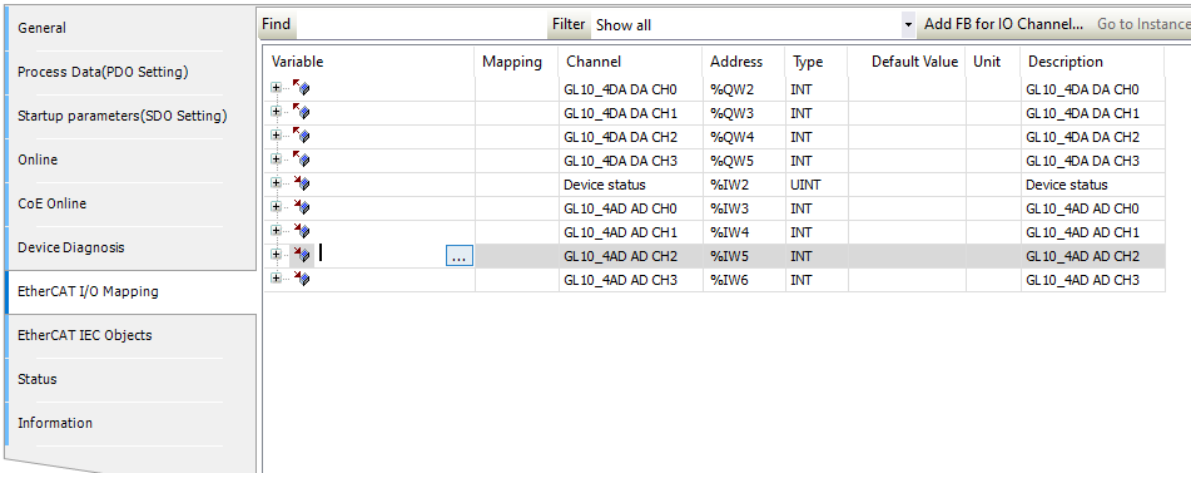


Рис. 4.106. Добавление переменной

Во всплывающем окне выберите ранее созданную переменную InValueVel (рис. 4.107, 4.108).

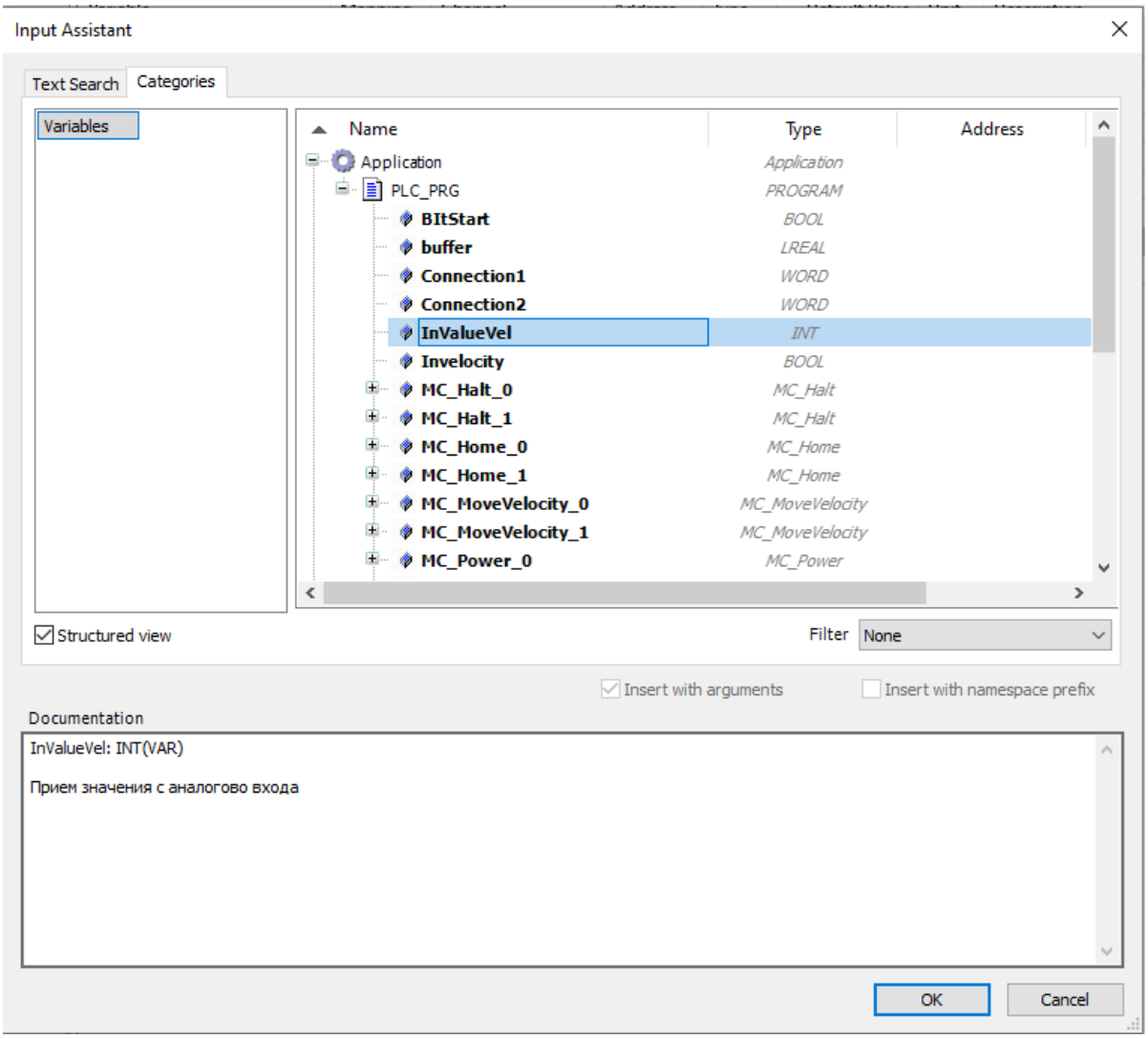


Рис. 4.107. Выбор созданной переменной

Variable	Mapping	Channel	Address	Type	Default Value	Unit	Description
		GL10_4DA DA CH0	%QW2	INT			GL10_4DA DA CH0
		GL10_4DA DA CH1	%QW3	INT			GL10_4DA DA CH1
		GL10_4DA DA CH2	%QW4	INT			GL10_4DA DA CH2
		GL10_4DA DA CH3	%QW5	INT			GL10_4DA DA CH3
		Device status	%IW2	UINT			Device status
		GL10_4AD AD CH0	%IW3	INT			GL10_4AD AD CH0
		GL10_4AD AD CH1	%IW4	INT			GL10_4AD AD CH1
Application.PLC_PRG.InValueVel		GL10_4AD AD CH2	%IW5	INT			GL10_4AD AD CH2
		GL10_4AD AD CH3	%IW6	INT			GL10_4AD AD CH3

Рис. 4.108. Привязка выбранной переменной

Так как диапазон принимающих устройств с аналогового входа варьируется от 0 до 20 000 (рис. 4.109), необходимо преобразовать эти значения в скорость в диапазоне 0 до 100.

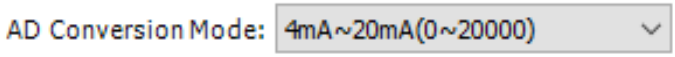


Рис. 4.109. Выбор диапазона

Для этого напишите формулу после программы. По условию включения приводов по питанию будет рассчитываться формула по преобразованию значений с аналогового входа в скорость (рис. 4.110).

```
56  //Преобразование входного диапазона с модуля в диапазон скорости
57  IF MC_Power_0.Status AND MC_Power_1.Status THEN
58      setVel:=((INT_TO_LREAL(InValueVel)-0)/(20000-0))*(100-0)+0;
59  END_IF
```

Рис. 4.110. Преобразование входного диапазона с модуля на диапазон скорости

Преобразование INT_TO_LREAL необходимо, потому что в блоках MC_MoveVelocity при задании скорости следует ввести значение типа Lreal, а на вводе аналогового входа типа INT.

Для изменения скорости следует выключать функциональный блок и включать его заново с новыми значениями.

Выполните следующие действия:

Создайте переменную-буфер и сравните значения текущей скорости с заданной.

Для выключения и включения блока используйте входную переменную блока Execute.

После выполненных действий приравняйте значения буфера к заданному значению, чтобы программа до следующего изменения положения не заходила в код (рис. 4.111).

Добавьте условие: если блок останова не работает, то отключить его.

Проконтролируйте систему переменных, которые в дальнейшем будут передаваться в панель.

После выполненных операций программа должна выглядеть как на рис. 4.112.

```
60  //Сравнение буфера текущей с заданной скоростью
61  IF buffer<>setVel THEN
62      MC_MoveVelocity_0(Axis:= Drive1, Execute:= FALSE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 100, Direction:=positive);
63      MC_MoveVelocity_1(Axis:= Drive2, Execute:= FALSE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 100, Direction:=positive);
64      MC_MoveVelocity_0(Axis:= Drive1, Execute:= TRUE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 100, Direction:= positive);
65      MC_MoveVelocity_1(Axis:= Drive2, Execute:= TRUE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 100, Direction:= positive);
66      buffer:=setVel;
67  END_IF
68  //Отключения блока останова, если он не используется
69  IF step<>5 AND step<>3 THEN
70      MC_Halt_0(Axis:=Drive1, Execute:=FALSE);
71      MC_Halt_1(Axis:=Drive2, Execute:=FALSE);
72  END_IF
74  //Переменные для контроля системы и управления
75  Speed1:=Drive1.fActVelocity;
76  Speed2:=drive2.fActVelocity;
77  Connection1:=Drive1.wCommunicationState;
78  Connection2:=Drive2.wCommunicationState;
79  Power:=MC_Power_0.Status AND MC_Power_1.Status;
80  Invelocity:=MC_MoveVelocity_0.InVelocity AND MC_MoveVelocity_1.InVelocity;
81  Alarm:=Drive1.nAxisState=SMC_AXIS_STATE.errorstop OR Drive2.nAxisState=SMC_AXIS_STATE.errorstop;
82
```

Рис. 4.111. Программирование буфера сравнения, отключения блока останова и переменных для контроля системы и управления

```
//Кнопка разрешения работы
IF BItStart THEN//Если кнопка нажата то выполняется кейс

    CASE step OF
0:  //Инициализация приводов
    IF Drive1.wCommunicationState=100 AND Drive2.wCommunicationState=100 THEN
        step:=1;
    END_IF

1:  //Включение питания
    MC_Power_0(Axis:= Drive1,   Enable:= TRUE,  bRegulatorOn:=TRUE ,    bDriveStart:= TRUE, );
    MC_Power_1(Axis:= Drive2,   Enable:= TRUE,  bRegulatorOn:=TRUE ,    bDriveStart:= TRUE, );

2:  //Управление по скорости
    MC_MoveVelocity_0(Axis:= Drive1, Execute:= TRUE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 50, Direction:=positive);
    MC_MoveVelocity_1(Axis:= Drive2, Execute:= TRUE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 50, Direction:=positive);


3:  //Выход в ноль
    MC_Halt_0(Axis:= Drive1,Execute:= TRUE,Deceleration:=50,Jerk:= 10);
    MC_Halt_1(Axis:= Drive2,Execute:= TRUE,Deceleration:=50,Jerk:= 10);
    IF MC_Halt_0.Done THEN
        MC_Home_0(Axis:= Drive1,Execute:= TRUE,Position:= 0);
        MC_Home_1(Axis:= Drive2,Execute:= TRUE,Position:= 0);
        IF MC_Home_0.Done AND MC_Home_1.Done THEN
            MC_Home_0(Axis:= Drive1,Execute:= FALSE);
            MC_Home_1(Axis:= Drive2,Execute:= FALSE);
            step:=1;
        END_IF

    END_IF

4:  //Перезагрузка приводов
    MC_Reset_0(Axis:= Drive1, Execute:=TRUE);
    MC_Reset_1(Axis:= Drive2, Execute:=TRUE);
    IF MC_Reset_0.Done OR MC_Reset_1.Done OR MC_Reset_0.Error OR MC_Reset_1.Error THEN
        MC_Reset_0(Axis:= Drive1, Execute:=FALSE);
        MC_Reset_1(Axis:= Drive2, Execute:=FALSE);
        step:=1;
    END_IF

5:  //Корректный останов
    MC_Halt_0(Axis:= Drive1,Execute:= TRUE,Deceleration:=50,Jerk:= 10);
    MC_Halt_1(Axis:= Drive2,Execute:= TRUE,Deceleration:=50,Jerk:= 10);
    IF MC_Halt_0.Done AND MC_Halt_1.Done THEN
        step:=1;
    END_IF

6: //Выключение питания
    MC_Power_0(Axis:= Drive1,   Enable:= TRUE,  bRegulatorOn:=FALSE ,    bDriveStart:= FALSE);
    MC_Power_1(Axis:= Drive2,   Enable:= TRUE,  bRegulatorOn:=FALSE ,    bDriveStart:= FALSE);

END_CASE
```

Рис. 4.112. Полная программа

```
END_IF
//Преобразование входного диапазона с модуля в диапазон скорости
IF MC_Power_0.Status AND MC_Power_1.Status THEN
    setVel:=((INT_TO_LREAL(InValueVel)-0)/(20000-0))*(100-0)+0;
END_IF
//Сравнение буфера текущей с заданной скоростью
IF buffer<>setVel THEN
    MC_MoveVelocity_0(Axis:= Drive1, Execute:= FALSE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 100, Direction:=positive);
    MC_MoveVelocity_1(Axis:= Drive2, Execute:= FALSE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 100, Direction:=positive);
    MC_MoveVelocity_0(Axis:= Drive1, Execute:= TRUE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 100, Direction:= positive);
    MC_MoveVelocity_1(Axis:= Drive2, Execute:= TRUE, Velocity:= setVel, Acceleration:=50 , Deceleration:=50 , Jerk:= 100, Direction:= positive);
    buffer:=setVel;
END_IF
//Отключения блока останова, если он не используется
IF step<>5 AND step<>3 THEN
    MC_Halt_0(Axis:=Drive1, Execute:=FALSE);
    MC_Halt_1(Axis:=Drive2, Execute:=FALSE);
END_IF

//Переменные для контроля системы и управления
Speed1:=Drive1.fActVelocity;
Speed2:=drive2.fActVelocity;
Connection1:=Drive1.wCommunicationState;
Connection2:=Drive2.wCommunicationState;
Power:=MC_Power_0.Status AND MC_Power_1.Status;
Invelocity:=MC_MoveVelocity_0.InVelocity AND MC_MoveVelocity_1.InVelocity;
Alarm:=Drive1.nAxisState=SMC_AXIS_STATE.errorstop OR Drive2.nAxisState=SMC_AXIS_STATE.errorstop;
```

Рис. 4.112. Окончание

Для передачи данных в панель необходимо создать Symbol Configuration во вкладке Application (рис. 4.113).

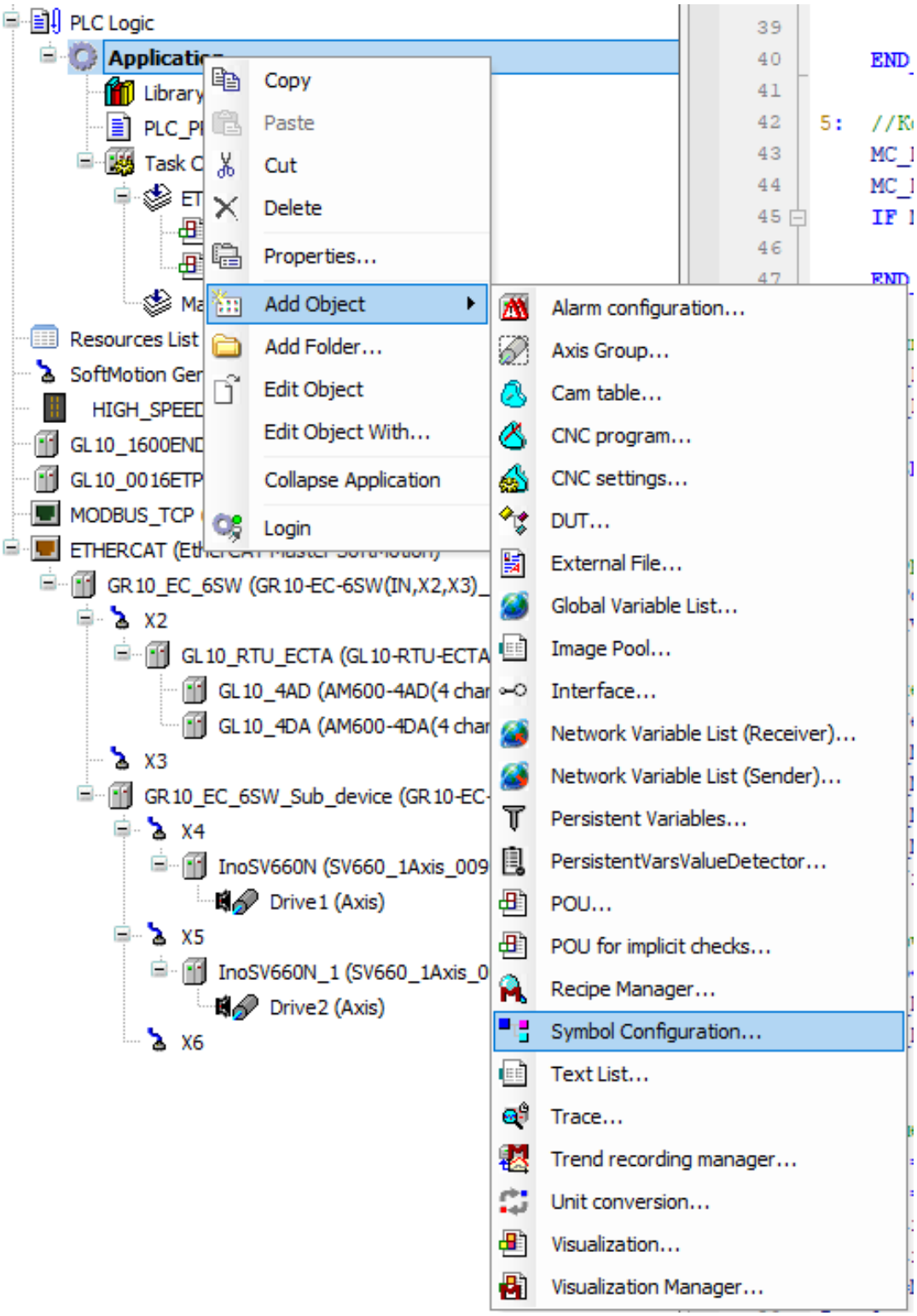


Рис. 4.113. Application – Symbol Configuration

Во всплывающем окне поставьте галочку «Поддерживать OPC» (рис. 4.114).

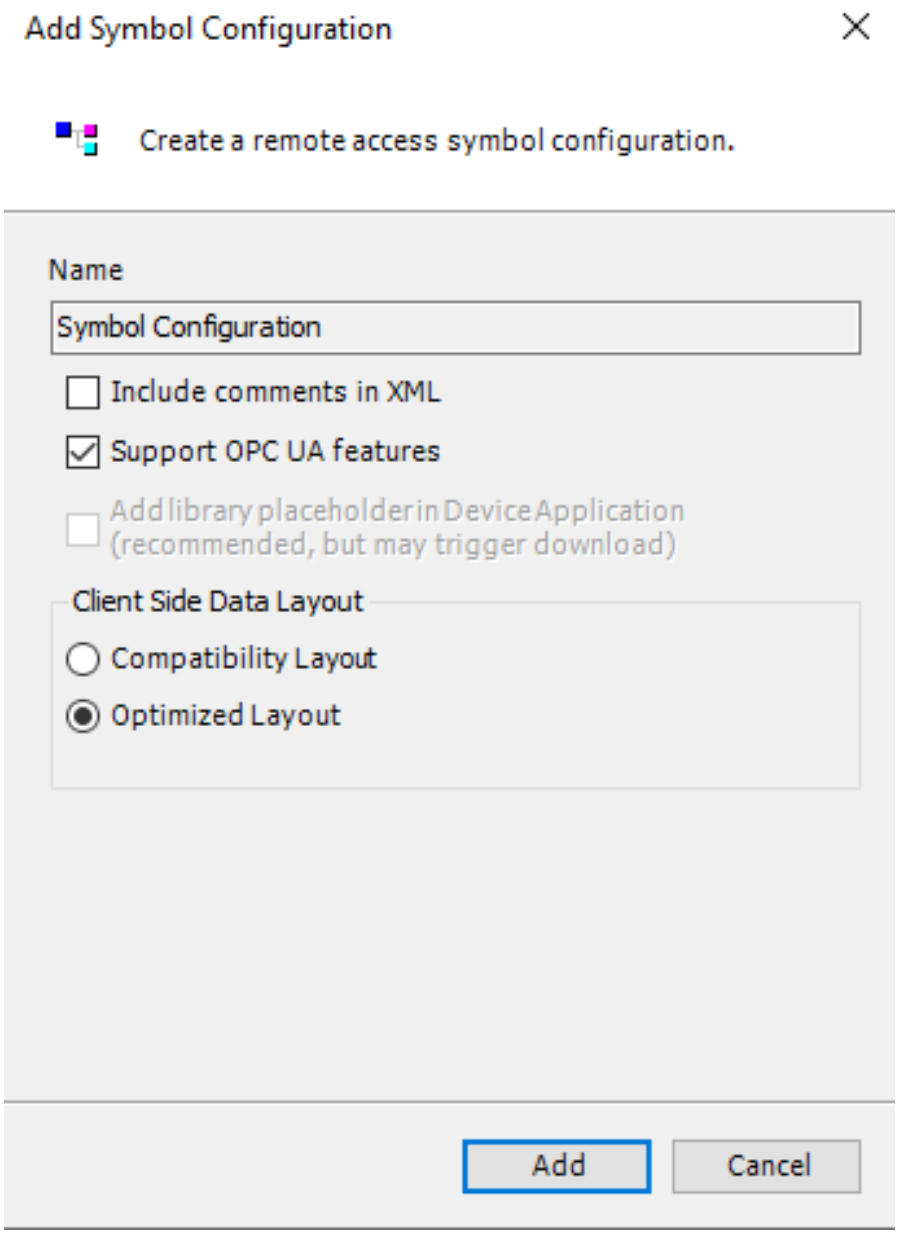


Рис. 4.114. Выбор OPC

Далее нажмите кнопку Build (рис. 4.115).

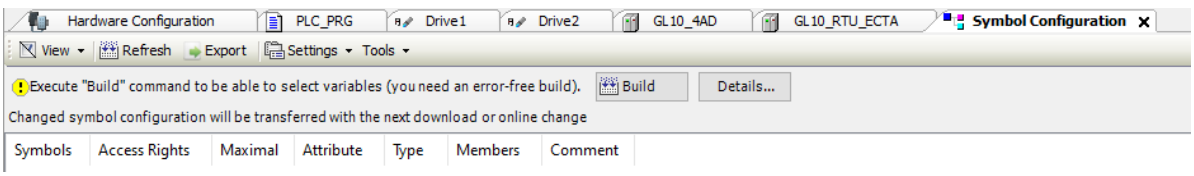


Рис. 4.115. Кнопка Build

Выберите переменные для передачи (рис. 4.116).

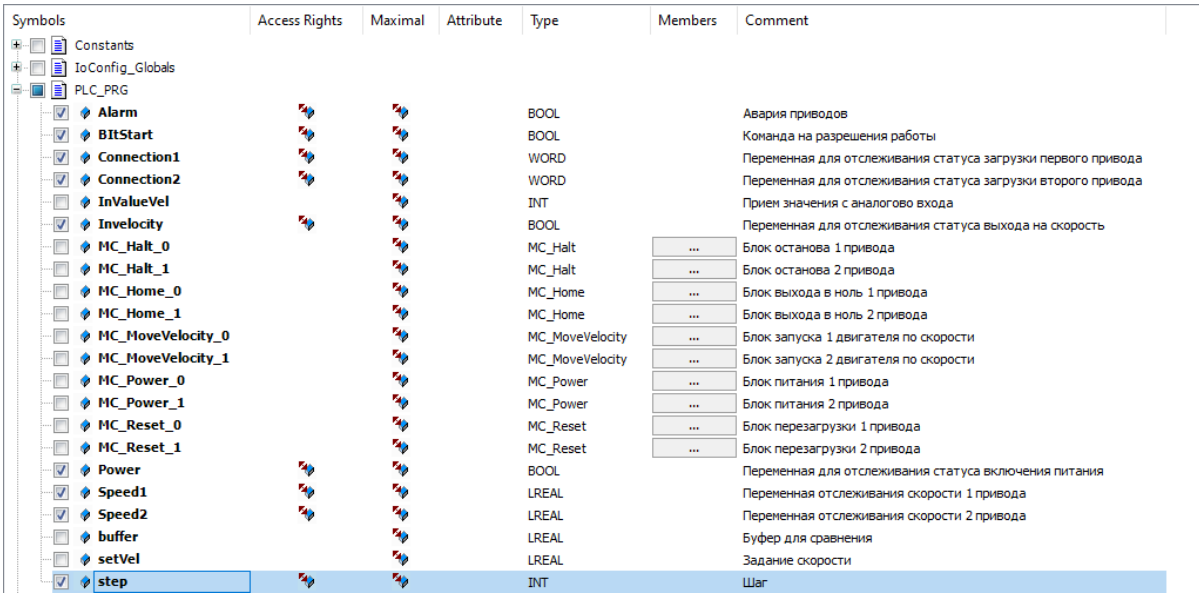


Рис. 4.116. Выбор переменных для передачи

После выбора обязательно нажмите на кнопку Build/Refresh (рис. 4.117) для того, чтобы контроллер запомнил все переменные для передачи.

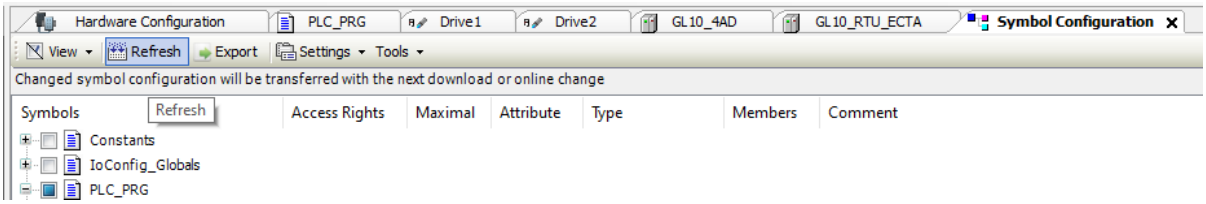


Рис. 4.117. Build/Refresh

Загрузите проект в контроллер и запустите программу.

Откройте среду для панелей InoTouchPad и создайте проект для IT7070E.

Зайдите во вкладку Connection, выберите EtherNET и добавьте новое соединение OPC UA Tags (рис. 4.118).

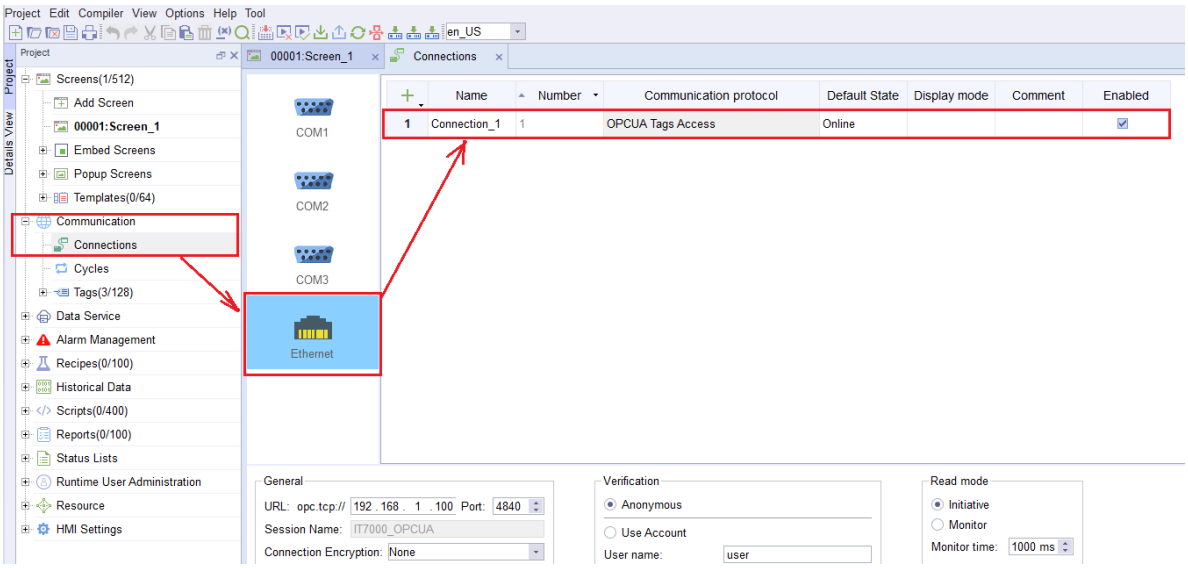


Рис. 4.118. Выбор нового соединения

Введите IP контроллера (рис. 4.119) и нажмите Browser Tags. Важно: при нажатии на Browser Tags программа в контроллере должна работать.

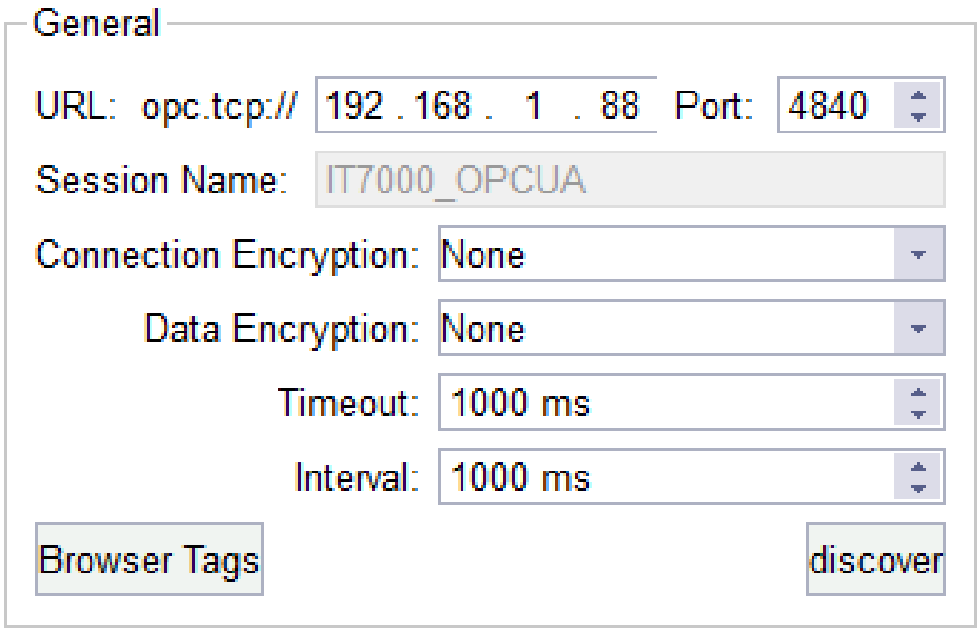


Рис. 4.119. Ввод IP

Раскройте список и отметьте переменные. Затем нажмите Add tags, поставьте галочку Create PLC tags to group и нажмите OK (рис. 4.120).

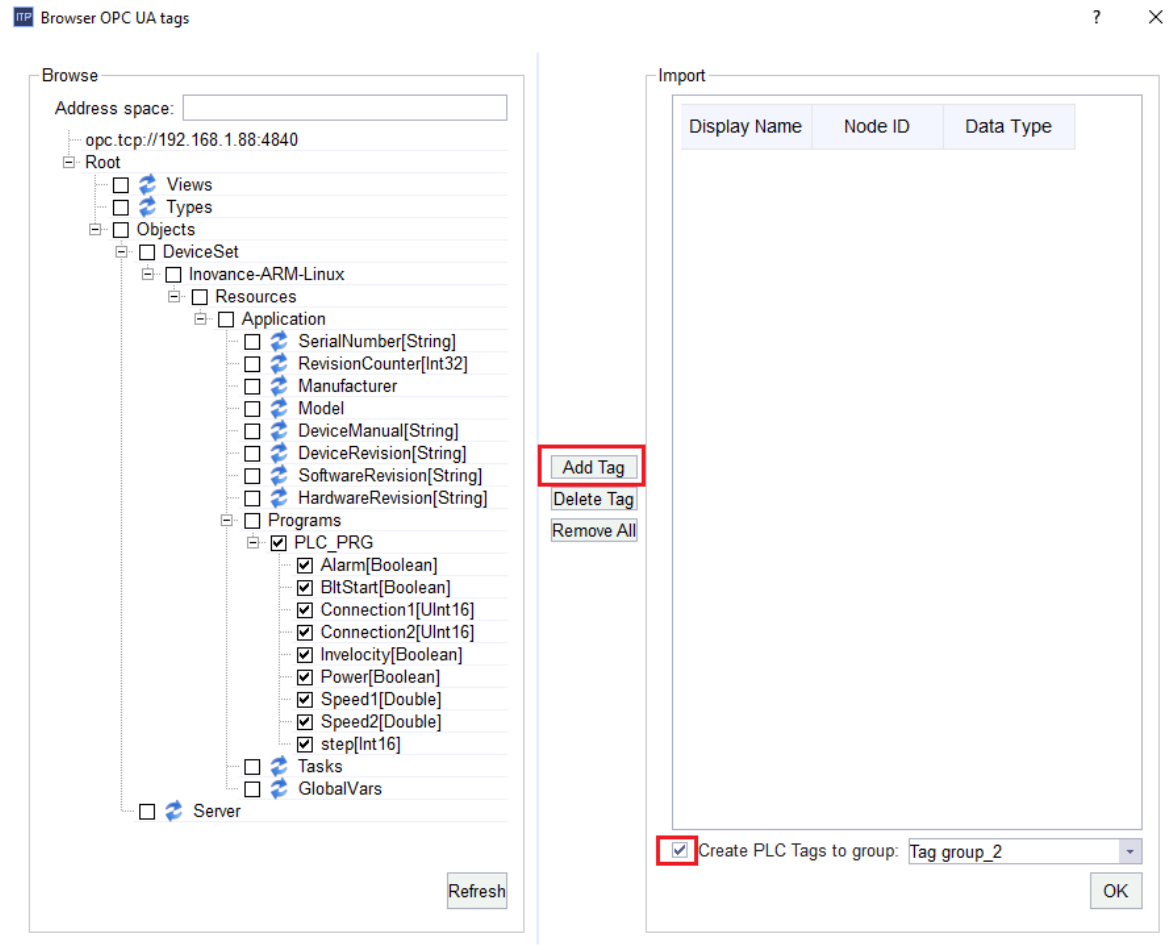


Рис. 4.120. Выбор переменных

Переменные добавились в группу тегов, которая находится во вкладке Communication – Tags (рис. 4.121). Название тега можно изменять.

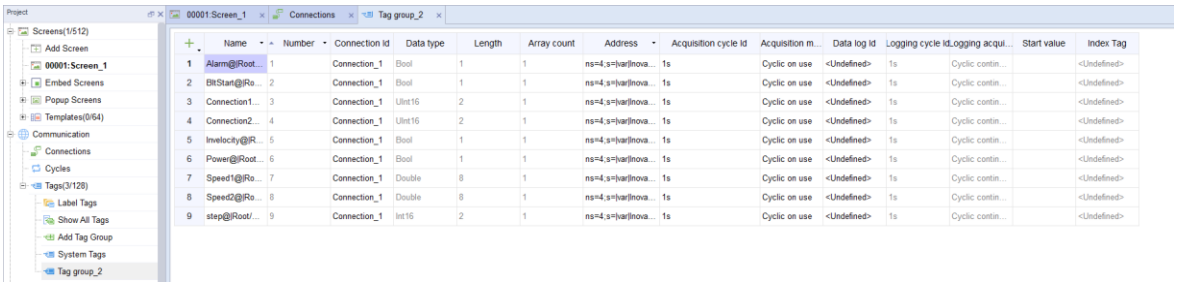


Рис. 4.121. Группа тегов

Перейдите на главный экран. Его фон можно сделать любого цвета.

Для редактирования управления и контроля приводов (рис. 4.122) понадобятся следующие инструменты:

Text Field – запись текста в поле;
Bit indicator и Bit button – индикатор и кнопка соответственно;
Number IO field – поле вывода значений.

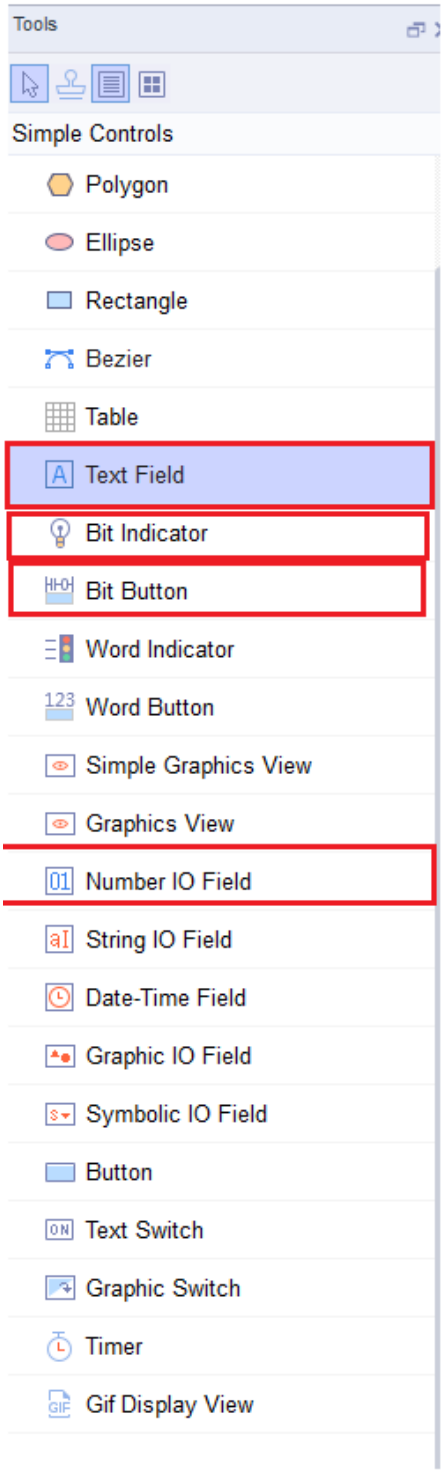


Рис. 4.122. Инструменты редактирования управления и контроля приводов

Конечный интерфейс проекта должен соответствовать изображенному на рис. 4.123.

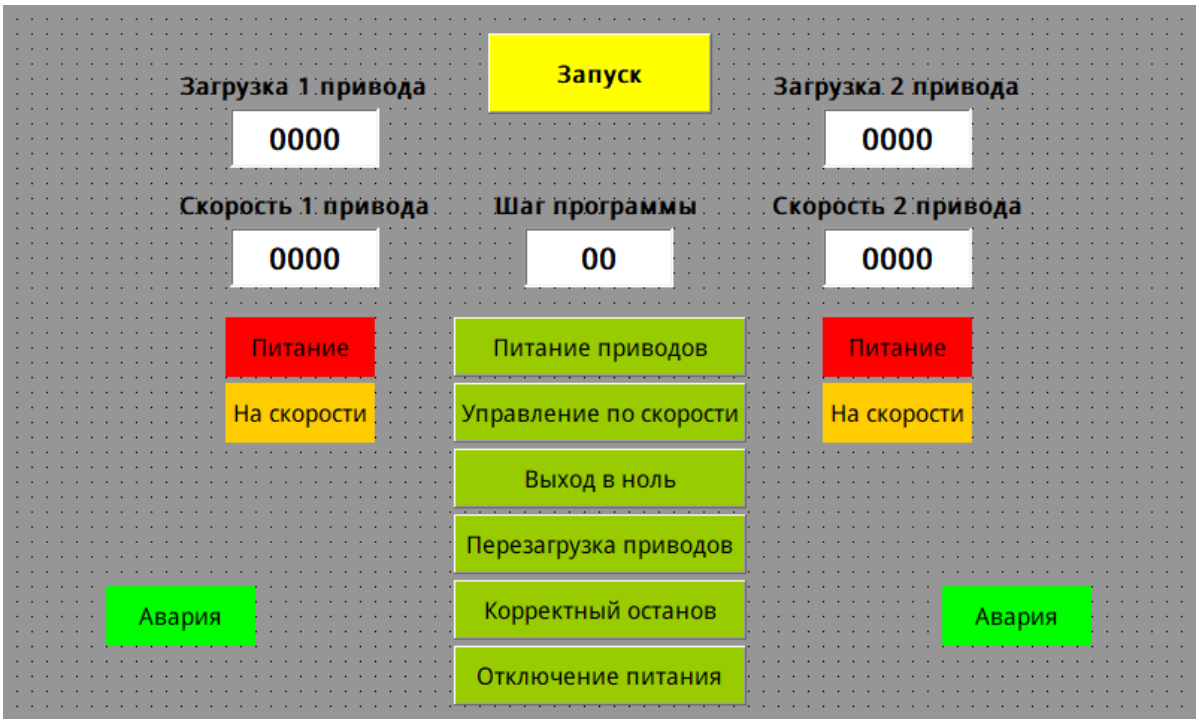


Рис. 4.123. Визуальное оформление программы

Кнопка «Запуск» имеет следующие настройки:

Выбор тега: Read tag – тег, который был передан с контроллера, выберите его и задайте Mode в положении Set (рис. 4.124).

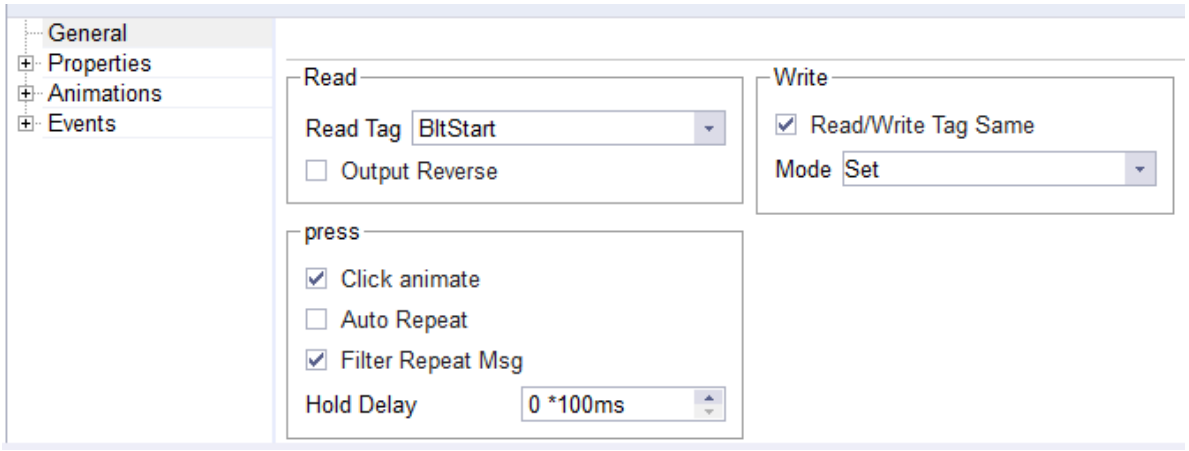


Рис. 4.124. Выбор тега и значения

Выставьте фон и подпись в строке Status: 0 – выключено, 1 – включено (рис. 4.125).

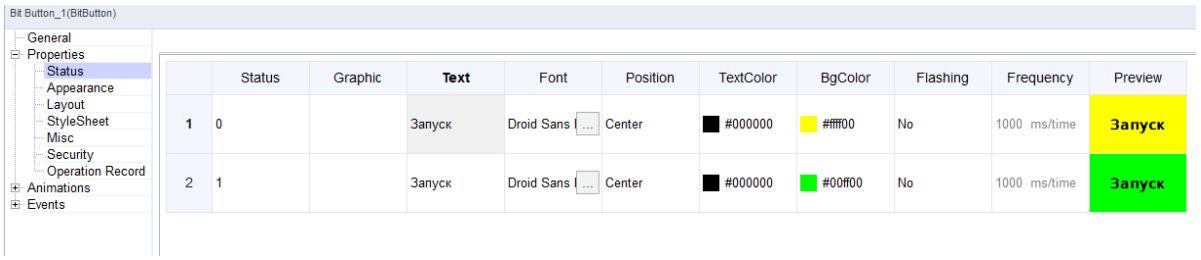


Рис. 4.125. Фон и статус

Далее программируются поля вывода шага программы, скорости привода и его загрузки.

В строке mode выставьте «Отображение» и размер символов – 4. Также измените текст внутри рамки на жирный в Properties – text (рис. 4.126).

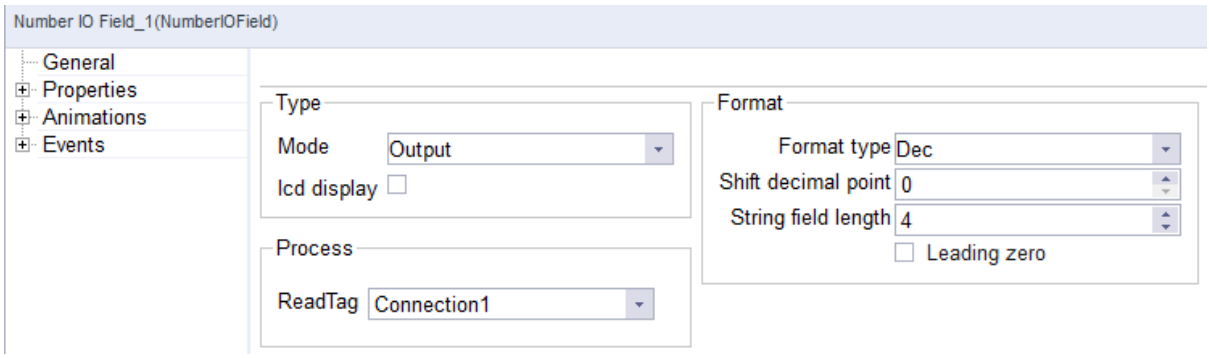


Рис. 4.126. Оформление отображения

Для каждого поля выставьте свой тег в Read tag:

1) для загрузки – Connection;
2) скорости – Speed;
3) шага – Step.

Далее сконфигурируйте кнопки управления (питание, управление, выход в ноль и т. д.).

В каждой кнопке необходимо выставить Mode – Momentary ON (рис. 4.127).

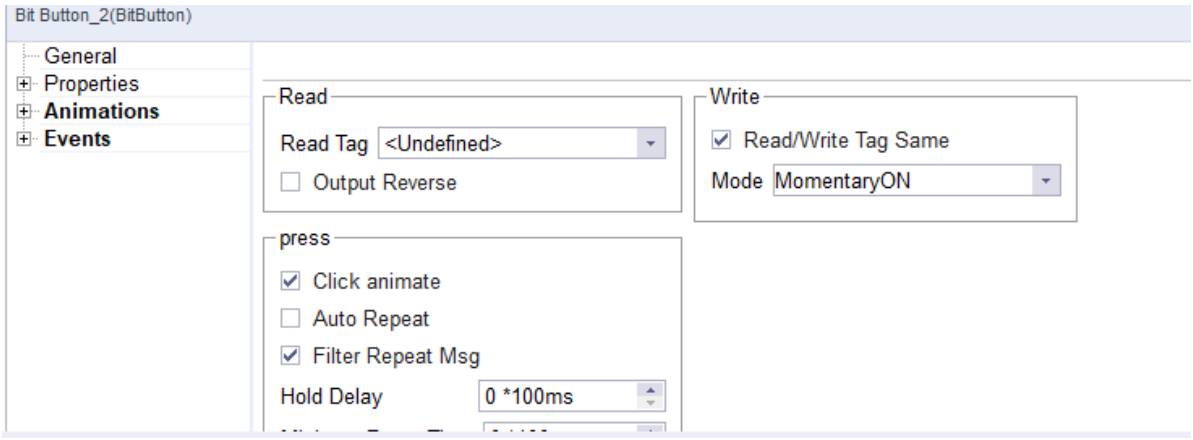


Рис. 4.127. Установка значения ON

Далее используйте для работы вкладку Animation. Выставьте тег загрузки и примите значение, при котором кнопка будет активна только когда загрузка привода достигнет 100 % (рис. 4.128).

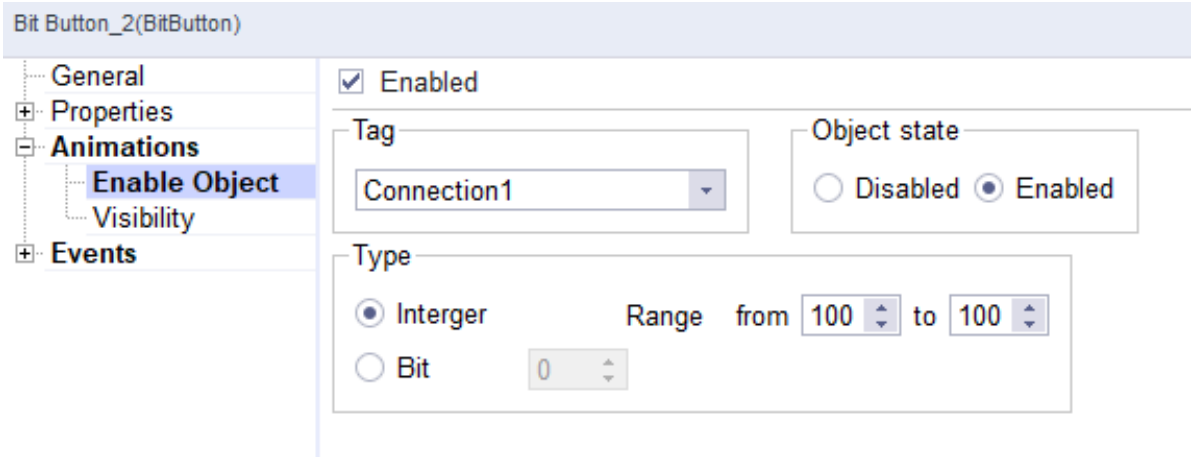


Рис. 4.128. Выставление тега для анимации

Во вкладке Events выберите Click и SetValue.

Для каждой программируемой кнопки прикрепите тег Step, значения в каждой кнопке будут разные. Данные значения символизируют шаги в программе.

Каждая запрограммированная кнопка имеет определенное значение, которое передается в ПЛК при нажатии на кнопку:

1) питание приводов – «1»;
2) управление по скорости – «2»;

3) выход в ноль – «3»;

4) перезагрузка приводов – «4»;

5) корректный останов – «5»;

6) отключение питания – «6».

Далее программируют индикатор отображения состояния (авария, питание, на скорости) (рис. 4.129).

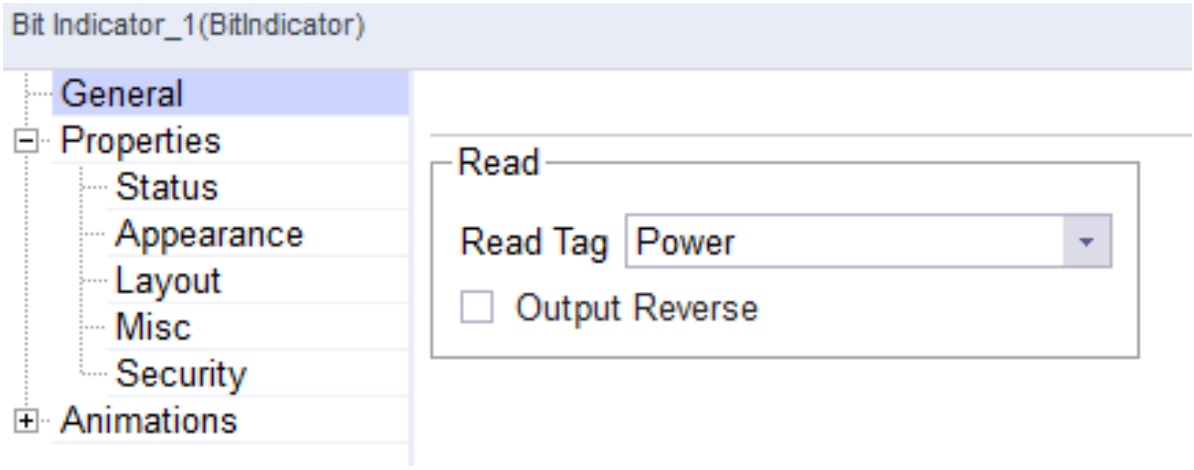


Рис. 4.129. Индикатор отображения состояния

Выставьте теги с ПЛК и в статусе отобразите цвет фона, который может изменяться в зависимости от статуса/состояния (рис. 4.130).

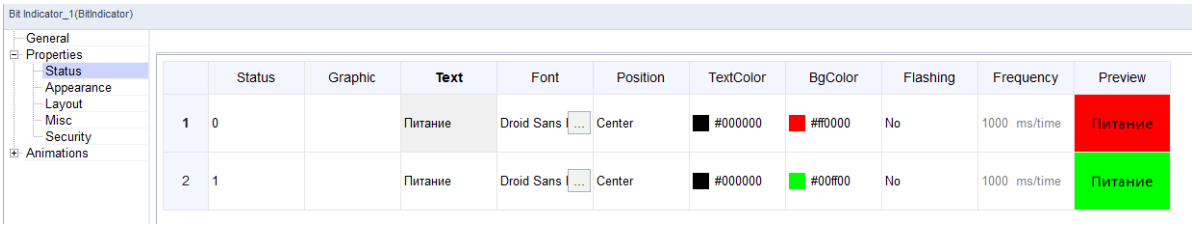


Рис. 4.130. Отображение питания

При работающем ПЛК загрузите проект в панель через инструменты или через вкладку Compiler.

Задайте IP панели и загрузите (рис. 4.131).

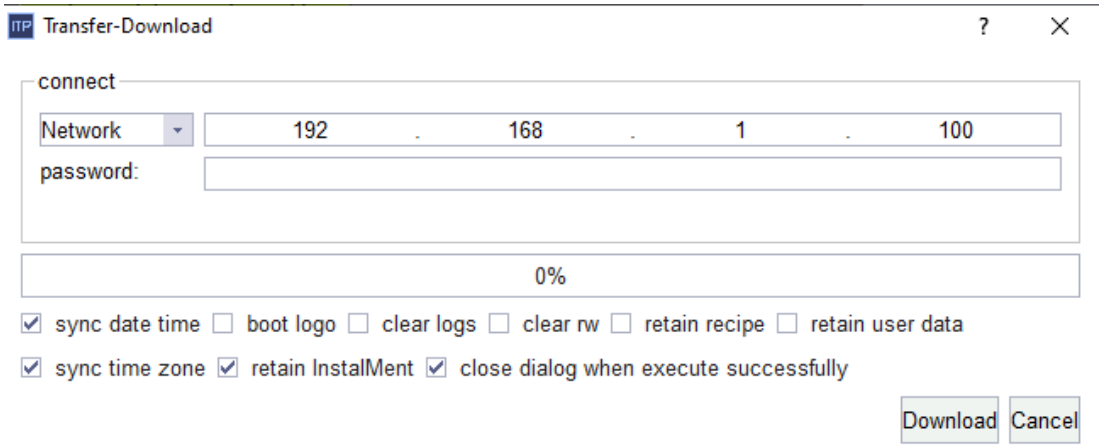


Рис. 4.131. Присвоение IP панели

После загрузки соединение с ПЛК будет автоматическим.

Содержание работы

1. Цель работы.
2. Задания.
3. Ход работы.
4. Выводы.
5. Контрольные вопросы.

Средства, используемые при выполнении лабораторной работы

1. Методические указания к выполнению лабораторных работ.
2. Данные, предоставленные преподавателем во время занятия.
3. При проведении анализа допускается использование глобальной сети Интернет.

Контрольные вопросы

1. Что делает функциональный блок MC_Reset?
2. Какой сигнал имеет потенциометр на стенде?
3. Для чего нужен Symbol Configuration?
4. Зачем в проекте была использована команда Enable Object?
5. Что делает функция SetValue?

ЗАКЛЮЧЕНИЕ

Компьютерная система управления представляет собой микроэлектронно-вычислительную машину, собирающую информацию от датчиков, которые, в свою очередь, отражают состояние объекта или процесса через аналого-цифровые преобразователи или отражают информацию на пульте управления и сохраняют ее на запоминающем устройстве.

Применение компьютерных систем управления в технических системах – разновидность современных информационных технологий, основанных на использовании средств вычислительной техники и направленных на решение задач управления техническими объектами.

Компьютерные системы управления позволяют добиться повышения производительности технологических процессов на предприятиях, а также повысить качество выпускаемой продукции.

Переход на компьютерные системы управления знаменует начало нового этапа в автоматизации. Наличие данных систем на производстве дает возможность отказаться от дорогостоящего импортного оборудования и одновременно с этим повысить собственную конкурентоспособность на мировом рынке.

РЕКОМЕНДАТЕЛЬНЫЙ БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Анашкин, А. С.* Техническое и программное обеспечение распределенных систем управления / А. С. Анашкин, Э. Д. Кадыров, В. Г. Харазов ; под ред. В. Г. Харазова. – СПб. : П-2, 2004. – 368 с. – ISBN 5-93893-274-2.

2. *Афонин, В. Л.* Интеллектуальные робототехнические системы / В. Л. Афонин, В. А. Макушкин. – М. : Интернет-университет информационных технологий – ИНТУИТ.ру, 2005. – 208 с. – ISBN 5-9556-0024-8.

3. *Денисов, М. С.* Системы числового программного управления : лаб. практикум / М. С. Денисов ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2021. – 112 с. – ISBN 978-5-9984-1412-1.

4. *Конюх, В. Л.* Компьютерная автоматизация производства : в 2 ч / В. Л. Конюх. – Кемерово : КузГТУ, 2003. – 118 с. – ISBN 5-89070-271-8 (ч. 1). – ISBN 5-89070-272-6 (ч. 2).

5. *Ловыгин, А. А.* Современный станок с ЧПУ и CAD/CAM система / А. А. Ловыгин, А. В. Васильев, С. Ю. Кривцов. – М. : Эльф ИПР, 2006. – 286 с. – ISBN 5-900891-60-7.

6. *Рассказчиков, Н. Г.* Компьютерные системы управления : учеб. пособие / Н. Г. Рассказчиков ; Владим. гос. ун-т. – Владимир : Изд-во Владим. гос. ун-та, 2010. – 155 с. – ISBN 978-5-9984-0017-9.

7. *Селевцов, Л. И.* Автоматизация технологических процессов : учебник / Л. И. Селевцов. – М. : Academia, 2019. – 160 с. – ISBN 978-5-4468-0615-7.

8. *Шалыгин, М. Г.* Автоматизация измерений, контроля и испытаний : учеб. пособие / М. Г. Шалыгин, Я. А. Вавилин. – СПб. : Лань, 2019. – 172 с. – ISBN 978-5-507-47370-0.

Учебное электронное издание

ДЕНИСОВ Максим Сергеевич
РУМЯНЦЕВ Илья Валерьевич
ЧЕБОТАРЕВ Петр Александрович

КОМПЬЮТЕРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Лабораторный практикум

Редактор Е. А. Лебедева

Технический редактор Ш. Ш. Амирсейидов

Компьютерная верстка Л. В. Макаровой

Корректор О. В. Балашова

Выпускающий редактор А. А. Амирсейидова

Системные требования: Intel от 1,3 ГГц; Windows XP/7/8/10; Adobe Reader; дисковод CD-ROM.

Тираж 9 экз.

Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых
Изд-во ВлГУ
rio.vlgu@yandex.ru

Институт машиностроения и автомобильного транспорта
кафедра автоматизации, мехатроники и робототехники
chebotarev@vlsu.ru