Содержание

Введение

Методические указания по программированию робота на языке Си относятся к лабораторному практикуму дисциплины «Программное обеспечение мехатронных и робототехнических систем». Практикум включает 4 лабораторные работы, выполняемых в среде Automation Studio, а также на учебном робототехническом комплексе (УРТК).

В методических указаниях приведены основные требования и методические рекомендации по выполнению лабораторных работ, которые обязательны для изучения. Проработку указаний по каждой лабораторной работе необходимо выполнять с изучением рекомендуемых к ней материалов.

Конечным итогом практикума является демонстрация студентом полученных знаний, умений и навыков по построению программного обеспечения робототехнических систем, реализации законов управления, а также отладки программного обеспечения.

Лабораторная работа №1

Отладка программного обеспечения роботехнических систем с использованием виртуального моделирования

Цель работы: получение навыков моделирования объекта управления в промышленных системах автоматического управления и создание функциональных блоков.

Задание: создать виртуальную систему управления (рис. 1 .1), включающую: модель объекта управления (рис. 1 .2), ПИ-регулятор (рис. 1 .3), сумматор и обратную связь. Передаточная функция объекта:

$$W = \frac{1}{k_e \cdot (T_м s + 1)}$$

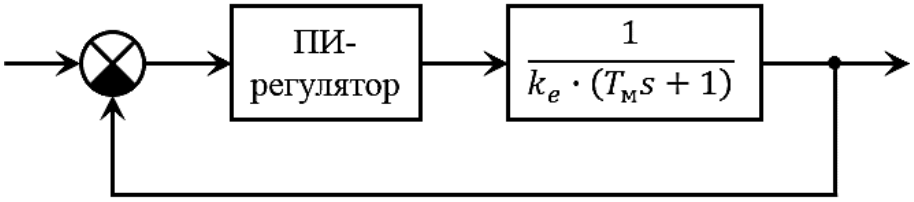

Рис. 1.1. Структура системы управления.

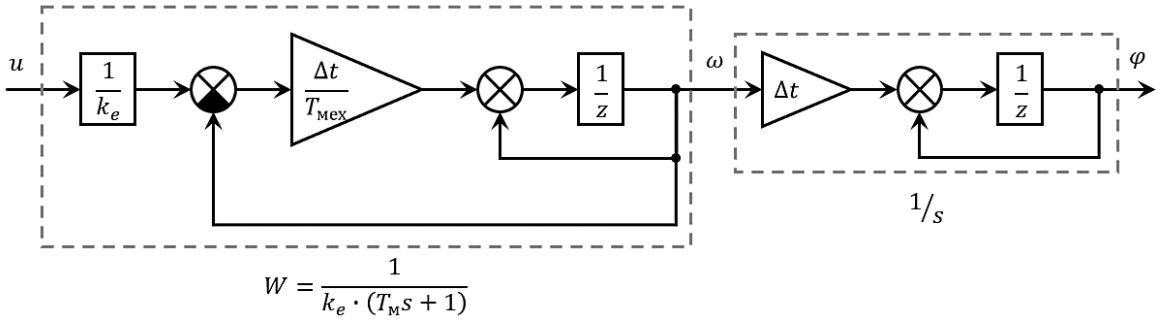

Рис. 1.2. Структура объекта управления.

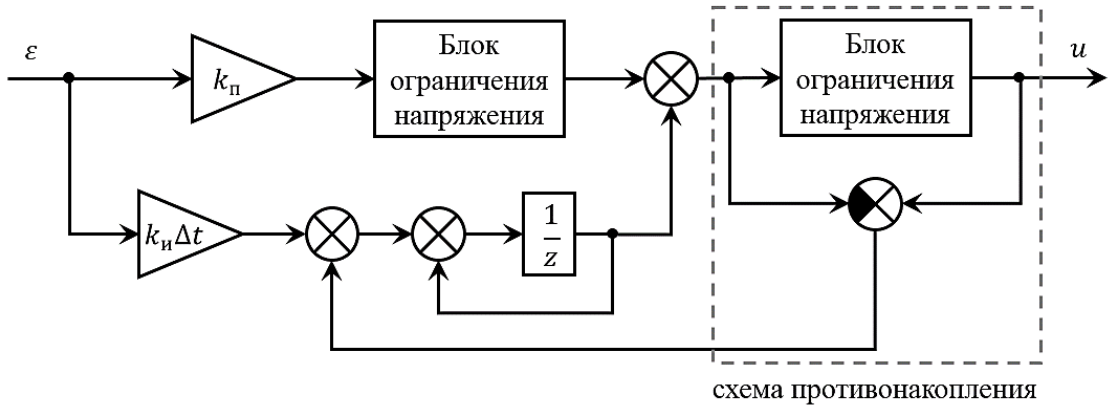

Рис. 1.3. Структура ПИ-регулятора.

Порядок выполнения:

1. Создать проект и добавить в него перечень объектов для реализации требуемых моделей;

1.1. Создать новый проект в среде Automation Studio без конфигурации оборудования (см. указания 1).

1.2. Создать в проекте объекты (см. указания 1).:

1.2.1. ANSI C Program;
1.2.2. ANSI C Library «MotorControl».
1.3. В библиотеке создать 3 функциональных блока (см. указания 3) и дать им имена:

1.3.1. «FB_Motor», в котором далее реализуем модель двигателя постоянного тока;
1.3.2. «FB_Regulator», выполняющий роль модели ПИ-регулятора.
1.3.3. «FB_Integrator», представляющим из себя модель интегрирующего звена.

2. Создание моделей объектов

2.1. Поскольку для создания функциональных блоков мотора и регулятора нам нужен интегратор, изначально разработаем его.

Таблица 1.1. Параметры функционального блока FB_Integrator.

Конфигурация	Имя	Тип данных	Описание
вход	in	REAL	вход интегрирующего звена
выход	out	REAL	выход интегрирующего звена
внутреннее состояние	dt	REAL	шаг расчета [с]

На структурных схемах блок интегратора представляет собой следующую структуру:

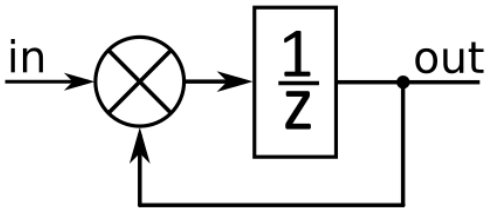

Рисунок 1.4. Структурная схема интегратора

Соответственно, логика работы интегратора заключается в накоплении сумм разностей входного и выходного значений в соответствии с шагом расчета. Это надо реализовать в программном коде данного функционального блока. Расчет значения на выходе данного функционального блока можно получить из передаточной функции интегратора путем Z-преобразования.

2.2. В структуру функционального блока «FB_Motor» ввести следующие поля, описанные в таблице 1.2.

Таблица 1.2. Параметры функционального блока FBMotor.

Конфигурация	Имя	Тип данных	Описание
вход	u	REAL	входное напряжение [В]
выход	w	REAL	частота вращения [об/мин]
выход	phi	REAL	положение [рад]
внутреннее состояние	integrator	FB_Integrator	интегратор
внутреннее состояние	Tm	REAL	электромеханическая постоянная времени [с]
внутреннее	ke	REAL	постоянная ЭДС двигателя

состояние			[В•мин/об]
внутреннее состояние	dt	REAL	шаг расчета [с]

2.3. Разработать программный код функционального блока «FB_Motor», использующий приведенные выше поля в соответствии с их описанием. Расчет значения на выходе блока происходит в соответствии со схемой ДПТ (рис. 1.2) или путем вывода из передаточной функции объекта разностного уравнения методом Z-преобразования.

2.4 В функциональном блоке «FB_Regulator» внести следующие поля, которые описаны в таблице 1.3.

Таблица 1.3. Параметры функционального блока FB_Regulator.

Конфигурация	Имя	Тип данных	Описание
вход	e	REAL	рассогласование между задающим воздействием и реальной скоростью вращения вала ДПТ [об/мин]
выход	u	REAL	напряжение, подаваемое на вход ДПТ [В]
внутреннее состояние	k_p	REAL	пропорциональный коэффициент регулятора
внутреннее состояние	k_i	REAL	интегральный коэффициент регулятора
внутреннее состояние	integrator	FB_Integrator	интегратор
внутреннее состояние	iyOld	REAL	хранение предыдущего значения схемы противонакопления
внутреннее состояние	max_abs_value	REAL	граница блока ограничения [В]
внутреннее состояние	dt	REAL	шаг расчета [с]

Написать программный код, использующий приведенные выше поля в соответствии с их описанием. Расчет значения на выходе блока происходит в соответствии со схемой ПИ-регулятора (рис. 1 .3). Схема противонакопления используется для ограничения напряжения, поступающего на двигатель, в целях обеспечения стабильности его работы.

Также в схеме присутствуют два ограничителя напряжения: после П-звена и выходного напряжения. Величина ограничения задается с помощью внутреннего состояния Umax.

Расчет коэффициентов регулятора производится с помощью метода обратной задачи динамики (ОЗД). Примем $T_{ж} = 1$.

$$W_p = \frac{1}{T_{ж} \cdot s \cdot W_o} \Longrightarrow W_p = \frac{T_m s + 1}{T_{ж} \cdot s \cdot \frac{1}{k_e}} = \frac{k_e(ms+1)}{T_{ж} \cdot s} = \frac{k_e \cdot T_m}{T_{ж}} + \frac{k_e}{T_{ж}} \cdot \frac{1}{s}$$

Следовательно, коэффициенты регулятора:

$$K_n = \frac{k_e \cdot T_m}{T_{ж}}; K_u = \frac{k_e}{T_{ж}}$$

3. Объединение объекта и регулятора в систему управления в основной программе с применением разработанных функциональных блоков;

3.1. В переменных(Variables) основной программы Main, создать следующие поля:

Таблица 1.4. Переменные основной программы.

Имя	Тип данных	Описание
fb_controller	FB_Controller	рассогласование между задающим воздействием и реальной скоростью вращения вала ДПТ [об/мин]
fb_motor	FB_Motor	напряжение, подаваемое на вход ДПТ [В]
Speed	REAL	уставка по скорости
Enable	BOOL	интегральный коэффициент регулятора

3.2. В основной программе, в части инициализации «Init», заполнить все постоянные (коэффициенты регуляторов, постоянные времени, граничные значения и шаги расчета) созданных объектов (fb_controller и fb_motor).

3.3. Написать в основном цикле программы реализацию системы управления (рис. 1 .1), имеющую на вход значение переменной speed и активирующейся при наличии логической «1» в переменной enable.

4. Написать программу, подающую на систему управления ступенчатое воздействие. Сравнить переходной процесс объекта управления без регулятора и с регулятором различных настройках коэффициентов регулятора.

4.1. Добавить в переменные основной программы второй мотор, указав в полях инициализации данные, аналогичные уже созданному ранее мотору. Добавить исполнение функционального блока второго мотора в основной цикл программы, подавая на его вход уставку speed.

4.2. Реализовать, используя вспомогательную переменную-счетчик counter ступенчатое воздействие на вход системы управления и второго двигателя.

4.3. Снять средством Trace графики:

- входное воздействие (скорость вращения);

- двигатель без регулятора;

- двигатель с регулятором.

Для этого во вкладке «Logical View» правой кнопкой мыши щелкаем на «Program» → «Open» → «Trace» (рисунок 1.5).

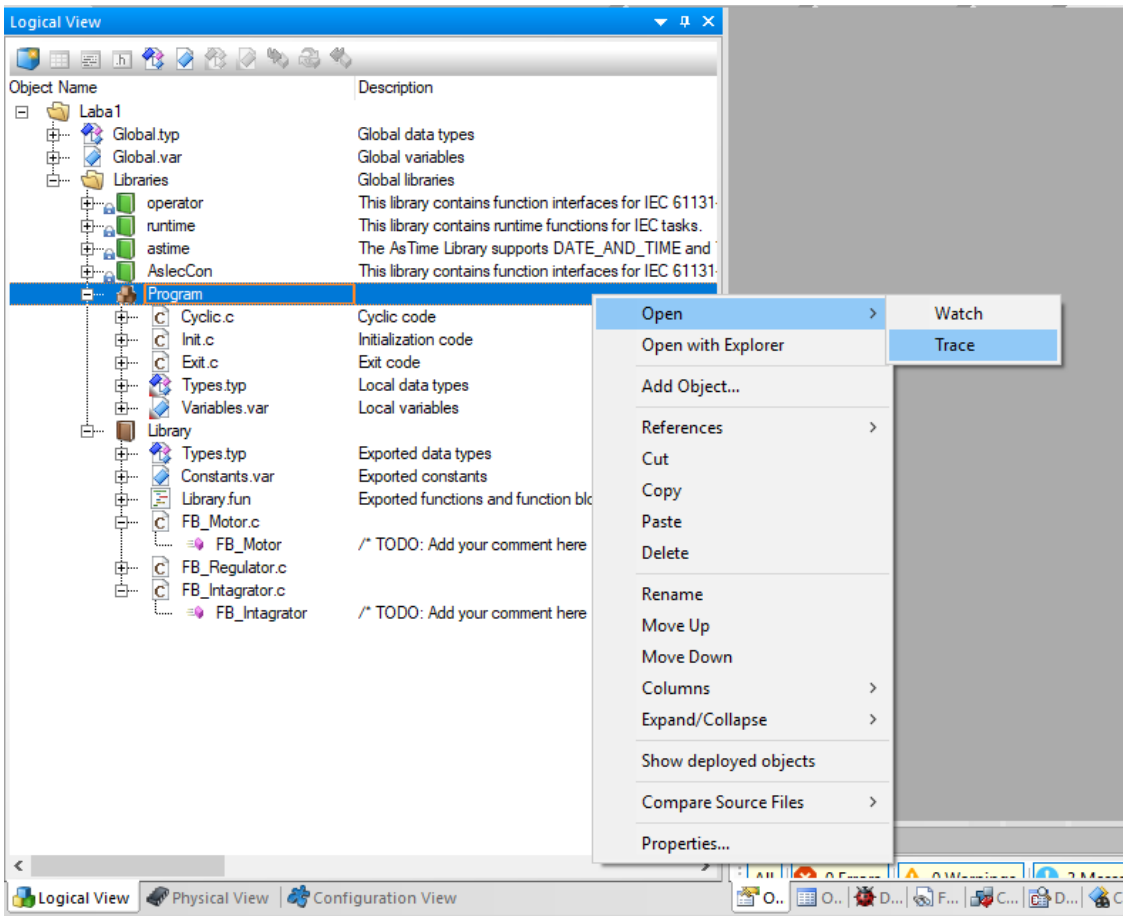

Рисунок 1.5. Выбор средства Trace.

В открывшейся вкладке вставляем конфигурацию Trace («Insert Trace configuration») и добавляем те переменные, графики изменения которых хотим получить («Insert a New Variable»). Последовательность действий в программной среде Automation Studio представлена на рисунке 1.6.

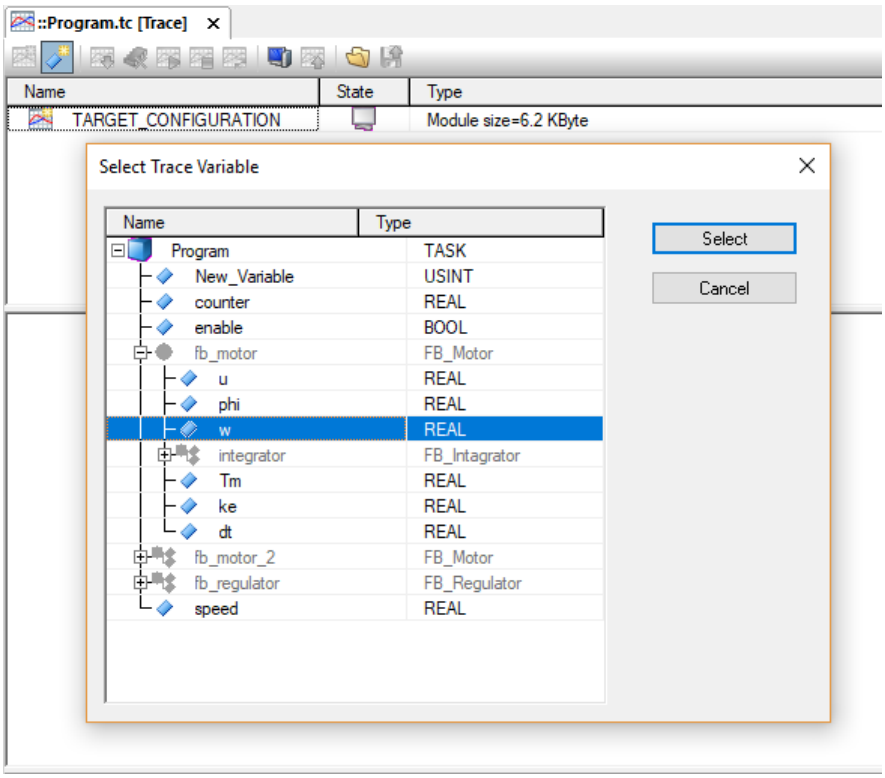

Рисунок 1.6. Создание новой конфиругации и ее переменных.

Для запуска Trace необходимо нажать кнопку «Install», а для остановки – «Stop» → «Show Target Data».

Пример требуемого графика приведен на рисунке 1.7.

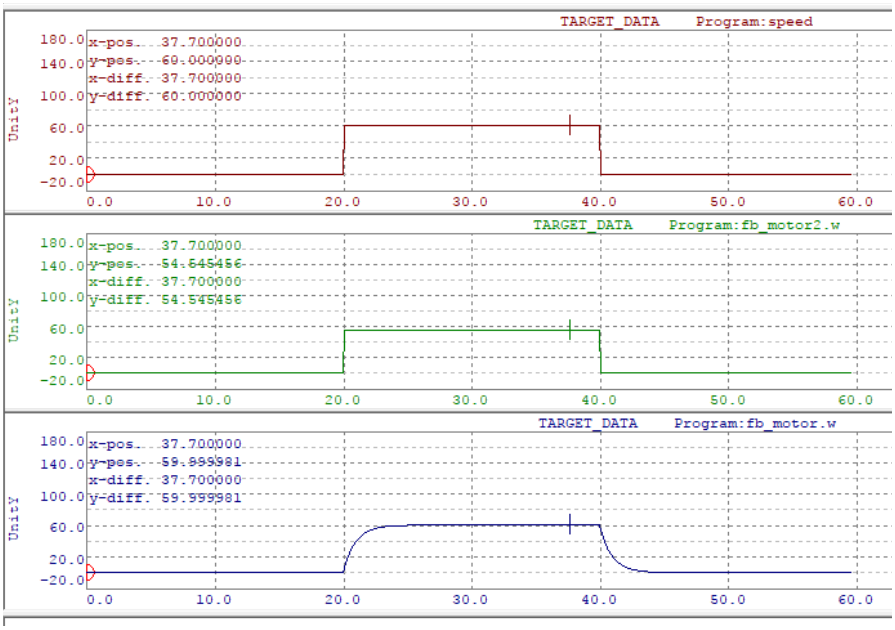

Рисунок 1.7. График уставки, скорости вращения двигателя без регулятора и скорости вращения двигателя с настроенным регулятором.

Содержание отчета:

1. Титульный лист.
2. Цель работы.
3. Задание.
4. Основная часть:
    - 4.1. описание действий и использованного аппарата.
    - 4.2. схема модели объекта управления и регулятора.
    - 4.3. структурная схема конечного автомата.
    - 4.4. описание состояний.
    - 4.5. описание функциональных блоков (назначение самого блока; входы и выходы, их назначение и работа).
    - 4.6. описание структуры программного обеспечения (например, в виде структурной схемы функциональных блоков).
    - 4.7. результаты экспериментальных исследований.
5. Выводы из лабораторной работы.
6. Приложение (листинг кода).

Контрольные вопросы:

1. Назовите основные достоинства применения виртуального моделирования исполнительных устройств при отладке программного обеспечение робототехнических и мехатронных систем.

2. Назовите на каком этапе разработки программного обеспечение осуществляется верификация программного обеспечения, запущенного на реальном контроллере, совместно с виртуальной моделью, окружающей среду.

3. Назовите последовательность необходимых действий для реализации закона управления, заданного в виде передаточной функции с одним входом и одним выходом.

4. Опишите методику расчета регулятора.

5. Что такое функциональный блок и для чего они используются?

Лабораторная работа №2

Программное обеспечение системы управления мехатронного модуля управления защитной дверью

Цель работы: получение навыков построения программного обеспечения промышленных систем управления на базе функциональных блоков и конечных автоматов.

Задание: разработать программное обеспечение системы автоматического управления приводом защитной двери. Схема механизма представлена на рисунке 2.1. Дверь оснащена асинхронным двигателем и четырьмя датчиками положения ($S_0 - S_3$), которые реагируют на пластину, обозначенную крестиком. Открытие и закрытие двери управляется тумблером.

При включении системы управления дверь должна двигаться в заданную сторону на небольшой скорости для определения своего местоположения. В этом режиме необходимо, чтобы индикаторы мерцали с частотой 1 Гц. После чего происходит переход в рабочий режим.

В рабочем режиме обеспечить максимально возможную скорость движения на отрезке s_1s_2, небольшую скорость на отрезках s_0s_1 и s_2s_3 (для обеспечения безопасности движения). Индикаторы должны показывать местоположение двери.

Система управления ворот должна быть выполнена в виде функционального блока, предполагающего повторное использование.

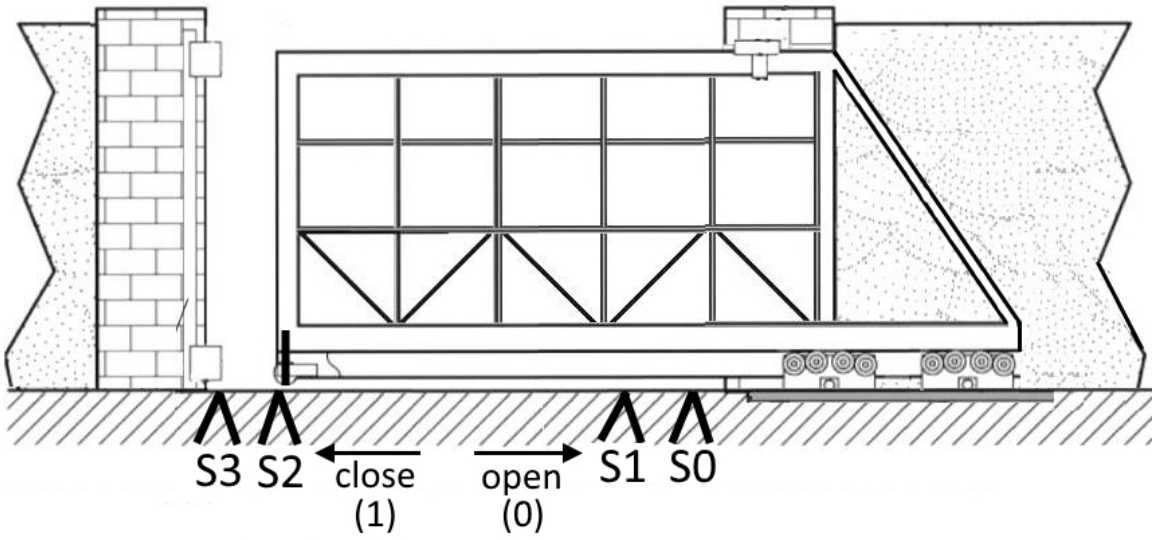

Рисунок 2.6. Устройство защитной двери.

Порядок выполнения:

1. Открыть исходную конфигурацию оборудования и проинициализировать частотный преобразователь.

1.1. Запустить B&R Automation Studio;

1.2. Открыть заархивированный проект (см. приложение) исходной конфигурации, хранящийся в директории “C:\projects\Blank lab\POMRS_2”, распаковав его в папку группы, находящуюся в “C:\projects”;

1.3. Открыть визуальную конфигурацию оборудования (приложение) и добавить к имеющимся модуль частотного преобразователя “8I64xxxxxxx.00x-1”;

1.4. Присоединить модуль шиной, аналогично рисунку 2 .7;

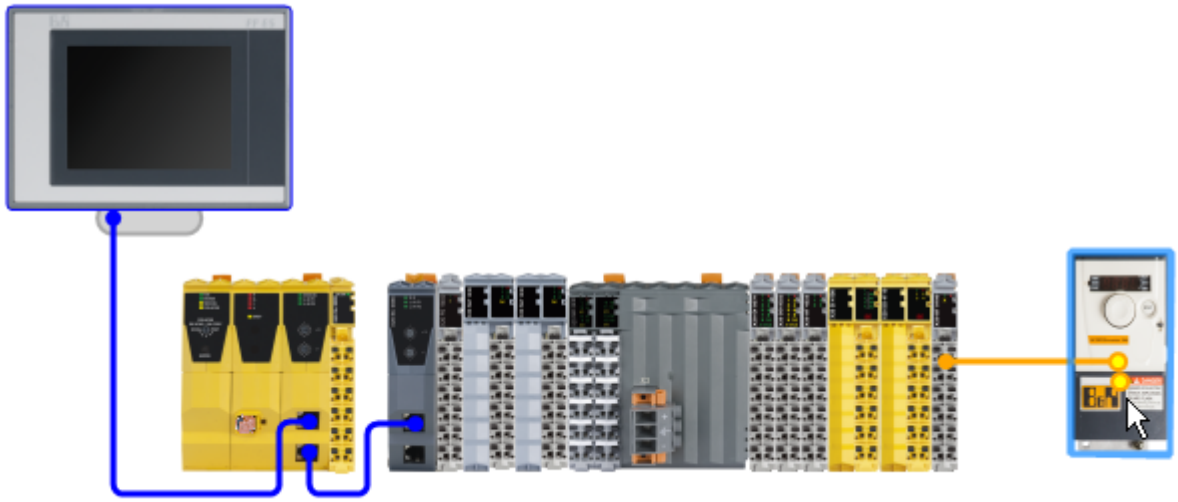

Рисунок 2.7. Конфигурация оборудования.

При этом, появится окно конфигурации подключаемого частотного преобразователя.

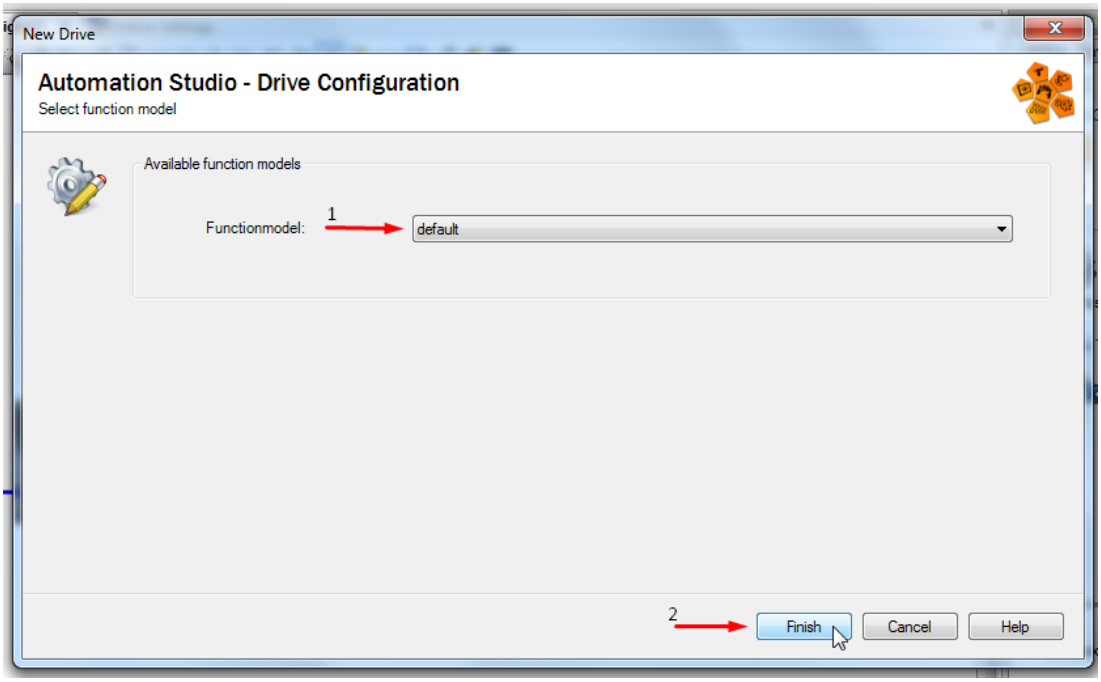

Рисунок 2.8. Конфигурация оборудования.

2. Сконфигурировать асинхронный двигатель, используя параметры, указанные на его корпусе.

2.1. Зададим необходимые параметры для асинхронного двигателя (табл. 2 .5). Они также указаны на самом двигателе или документации к нему.

Для этого необходимо в меню в меню Physical View выбрать частотный преобразователь и в контекстном меню выбрать пункт «Configuration» (рис. 2 .9);

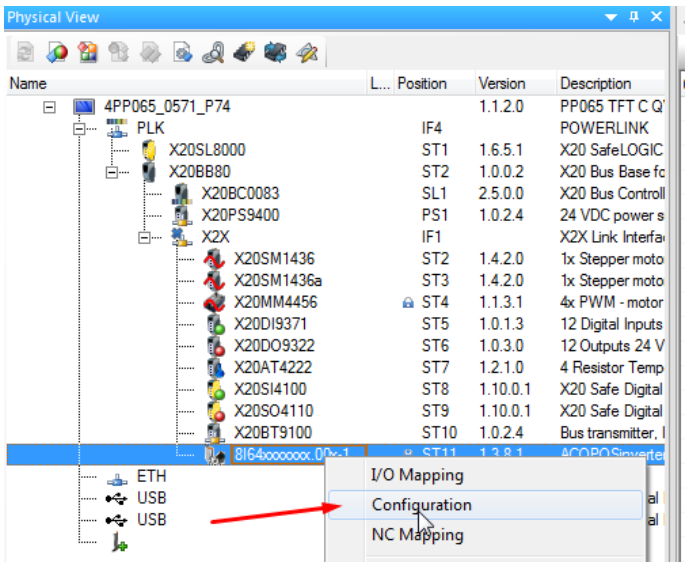

Рисунок 2.9. Открытие меню конфигурации частотного преобразователя.

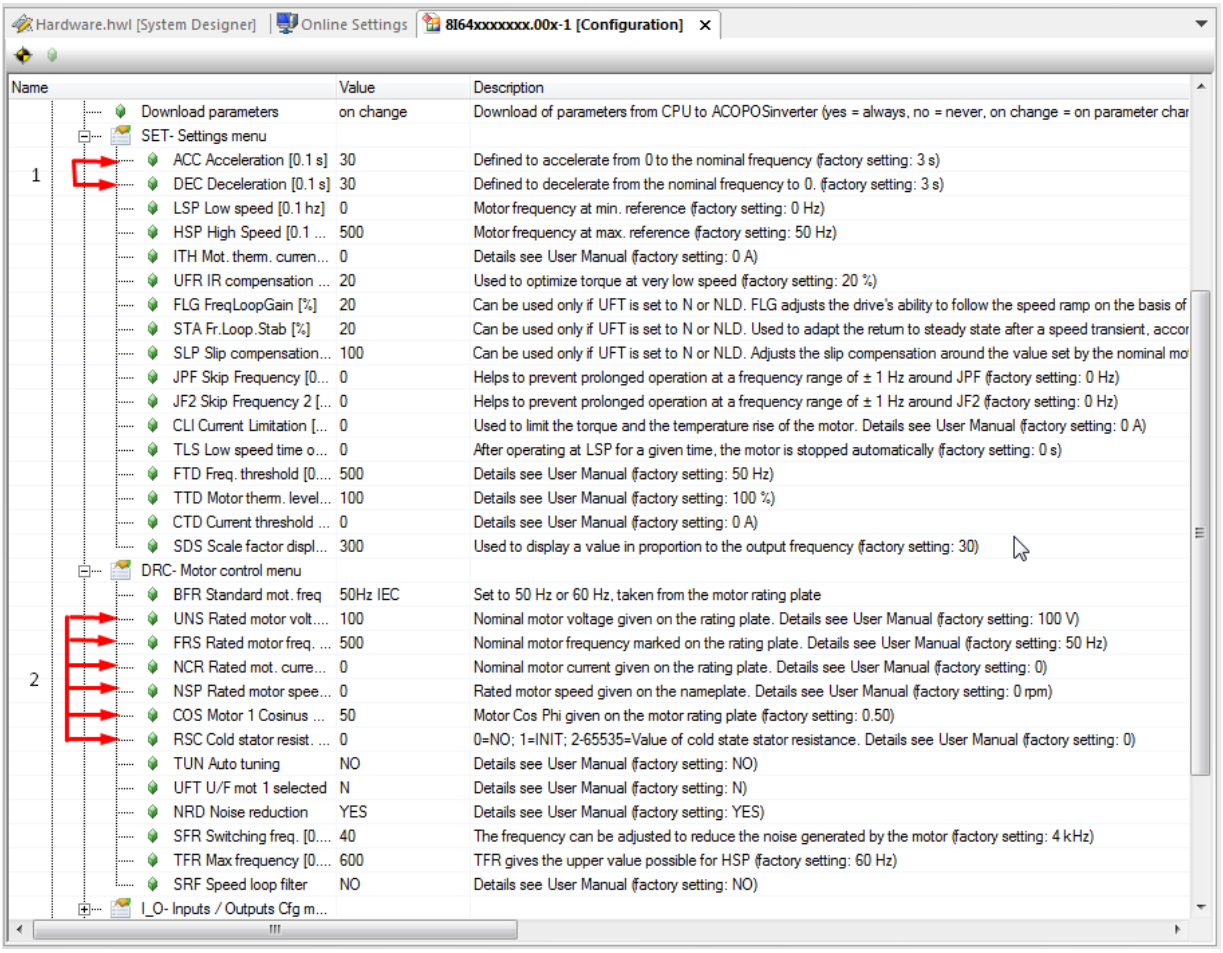

Рисунок 2.10. Пункты конфигурации, необходимые к изменению.

Таблица 2.5. Параметры асинхронного двигателя.

Наименование параметра	Название в конфигурационной таблице [eng.]	Заданное значение
Номинальное питающее напряжение двигателя [В]	UNS Rated Motor volt [V]	220
Номинальная питающая частота [0.1 Гц]	FRS Rated motor freq [0.1 Hz]	500
Сопротивление статора [мОм]	RSC Cold Stator resist [mOhm]	33000
Коэффициент мощности	COS Motor 1 Cosinus Phi [0.01]	64
Номинальная скорость вращения двигателя [об/мин]	NSP Rated motor speed [rpm]	890
Номинальный ток [0.1 А]	NCR Rated motor current [0.1 A]	10

3. Создать в проекте необходимые объекты.

3.1. Создать в проекте объекты (см. указания 1).:

3.1.1. ANSI C Program;
3.1.2. ANSI C Library « DriveLib».
3.2. В библиотеке «DriveLib» создать функциональные блоки:

«DriveStateMashine» – заготовка функционального блока управления частотным преобразователем. Структура ФБ должна иметь следующий набор переменных, указанный в таблице 2.6.

«DoorStateMashine» - функциональный блок, в котором будет реализована основная логика программы – задание направления вращения и скорости движения воротами в зависимости от их положения и требуемого направления вращения. Для упрощения описания состояний ворот необходимо создать новый тип данных (см. приложение): перечисление «DoorStates». Сами состояния, которые необходимо добавить приведены в таблице 2.3, а их связь представлена в функциональной схеме (рис. 2.6).

Таблица 2.6. Параметры функционального блока двигателя.

Конфигурация	Имя	Тип данных	Описание
вход	state	UINT	Состояние частотного преобразователя
вход	enable	BOOL	Сигнал работы функционального блока
выход	command	UINT	Команда, подаваемая на частотный преобразователь
выход	speed	INT	Заданная скорость

Таблица 2.3. Состояния ворот.

Состояние	Описание
ST_INIT	Инициализация параметров и ожидание включения частотного преобразователя
ST_UNKNOWN	Ворота в неизвестном положении
ST_OPEN	Ворота открыты
ST_CLOSE	Ворота закрыты
ST_ACC_POS	Ускорение ворот в сторону открытия
ST_ACC_NEG	Ускорение ворот в сторону закрытия
ST_POS	Движение к открытию
ST_NEG	Движение к закрытию
ST_DEC_POS	Замедление ворот в сторону открытия
ST_DEC_NEG	Замедление ворот в сторону закрытия

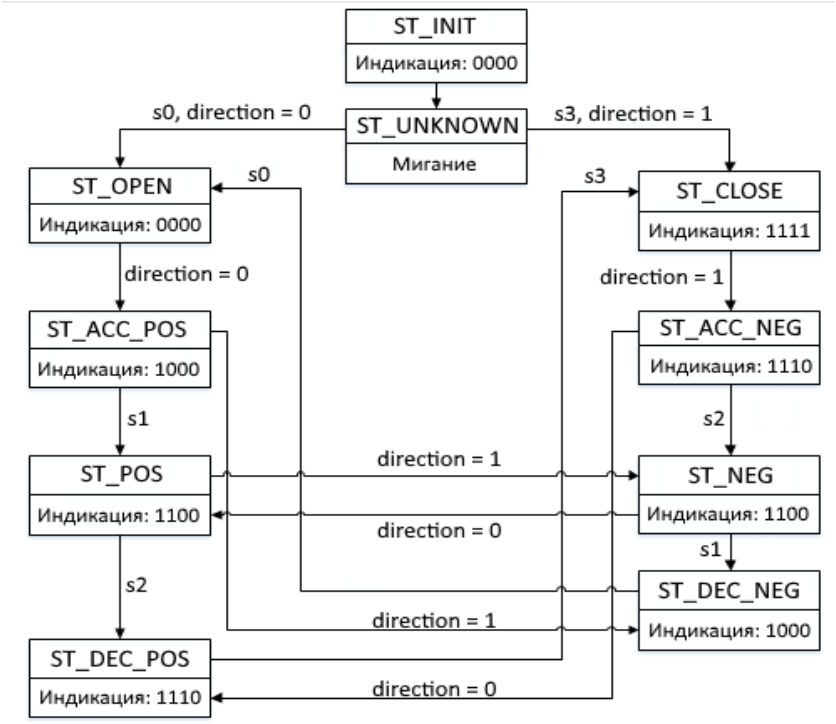

Рисунок 2.11. Блок-схема состояний ворот и возможных переходов между ними.

Теперь, имея новый необходимый тип данных, заполним структуру функционального блока:

Таблица 2.7. Структура функционального блока управления воротами.

Конфигурация	Имя	Тип данных	Описание
вход	state	UINT	Состояние частотного преобразователя
вход	sw1	BOOL	Сигнал концевого выключателя 1
вход	sw2	BOOL	Сигнал концевого выключателя 2
вход	sw3	BOOL	Сигнал концевого выключателя 3
вход	sw4	BOOL	Сигнал концевого выключателя 4
вход	direction	BOOL	Команда, подаваемая на частотный преобразователь
выход	speed	INT	Заданная скорость

«LedStateMashine» – машина состояний обработки светодиодных индикаторов.

Структура ФБ должна иметь следующий набор переменных:

Таблица 2.8. Структура функционального блока управления воротами.

Конфигурация	Имя	Тип данных	Описание
вход	state	UINT	Состояние частотного преобразователя
выход	led1	BOOL	Сигнал работы функционального блока
выход	led2	BOOL	Сигнал работы функционального блока
выход	led3	BOOL	Сигнал работы функционального блока

выход	led4	BOOL	Сигнал работы функционального блока
выход	timer	INT	Заданная скорость

4. Провести операцию мэпинга индикаторов, датчиков положения, частотного преобразователя.

Для совмещения работы аппаратной и программной частей, следует произвести мэпинг переменных. Подробно о этом написано в приложении, а конфигурация учебного стенда и соответствие входов/выходов каждого модуля приведена в приложении.

Соответствие входов модуля 8164xxxxxx.00x-1 и переменных приведены в таблице 4. Однако изначально нужные регистры представлены битовыми полями (рис. Рисунок 2 .12), из-за чего пришлось бы создавать множество переменных для работы с ними. Для преобразования битовых полей в привычные переменные, необходимо проделать следующие действия:

- В меню Physical View открыть выбрать частотный преобразователь и в контекстном меню выбрать пункт «Configuration» (рис. 2 .9);

- В меню Configuration изменить параметры аналогично рисунку 2 .13;

- Произвести мэпинг переменных согласно таблице 2.5.

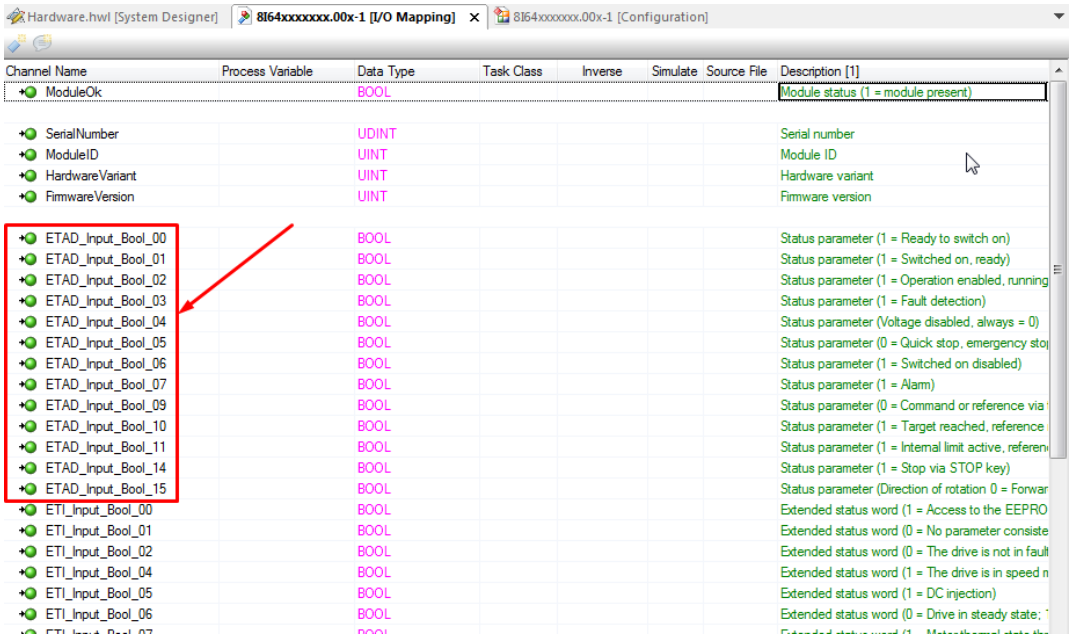

Рисунок 2.12. Изначальное представление регистров.

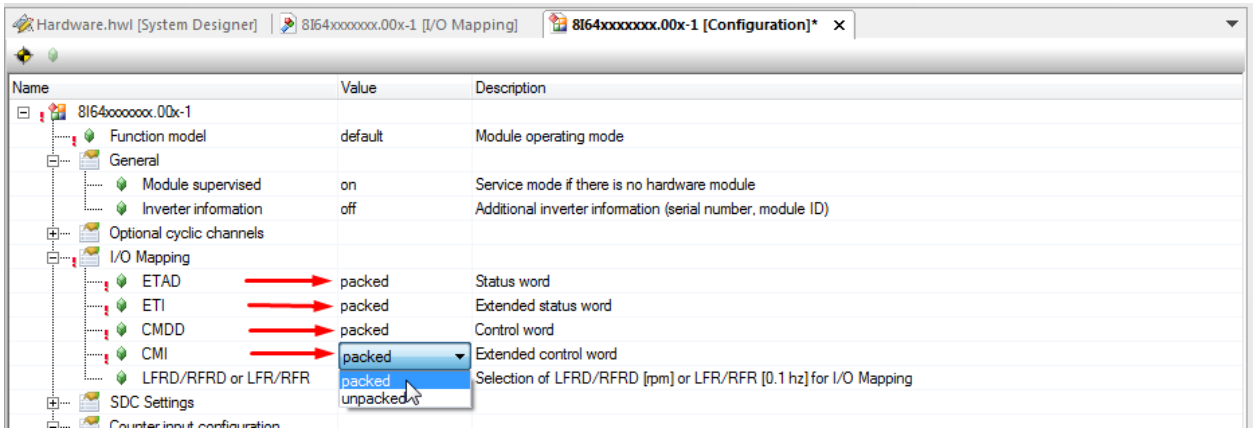

Рисунок 2.13. Изменение вида представления регистров.

Таблица 2.9. Мэпинг переменных для частотного преобразователя.

Наименование	Переменная	Описание
ETAD_Input	stateMachine.state	Текущее состояние частотного преобразователя
CMDD_Output	stateMachine.command	Команды на управление
LFRD_Output	stateMachine.speed	Заданное значение скорости

Таблица 2.10. Мэпинг переменных для датчиков положения

Имя канала	Используемая переменная
DigitalInput01	doorSM.direction
DigitalInput02	doorSM.s0
DigitalInput03	doorSM.s1
DigitalInput04	doorSM.s2
DigitalInput05	doorSM.s3

Для мэпинга индикаторов для модуля X20DO9322 (DO - Digital Output), соответствия каналов соответственно будут:

Таблица 2.7. Мэппинг переменных для индикаторов

Имя канала	Используемая переменная
DigitalOutput02	ledSM.s0
DigitalOutput03	ledSM.s1

DigitalOutput04	ledSM.s2
DigitalOutput05	ledSM.s3

5. Написать программный код логики управления дверьми.

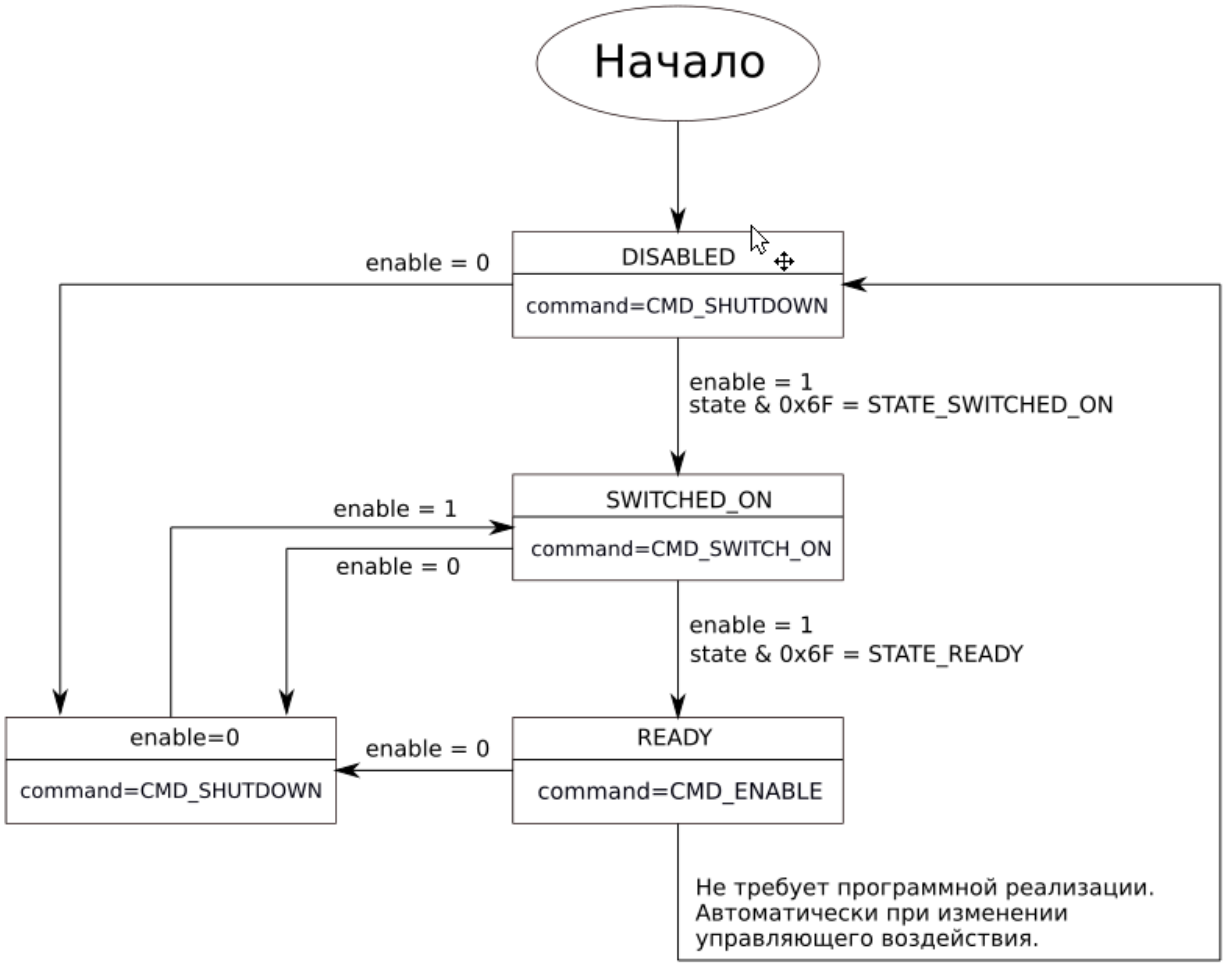

Рисунок 2.9.Блок-схема управления частотным преобразователем.

6. Провести эксперимент, подтверждающий работоспособность системы управления.

Эксперимент представляет из себя полный цикл управления дверью: от подачи питания до отработки команд на открытие и закрытие. Доказательством успешной работы является снятие показаний Trace по следующим величинам:

- Скорость вращения приводного мотора;

- Состояние датчиков;

- Состояние светодиодных индикаторов;

Содержание отчета:

1. Титульный лист
2. Цель работы
3. Задание
4. Основная часть:
    - 4.1. описание действий и использованного аппарата
    - 4.2. структурная схема конечного автомата
    - 4.3. описание состояний
    - 4.4. описание функциональных блоков (назначение самого блока; входы и выходы, их назначение и работа)
    - 4.5. описание структуры программного обеспечения (например, в виде структурной схемы функциональных блоков)
    - 4.6 результаты экспериментальных исследований
5. Выводы из лабораторной работы
6. Приложение (листинг кода)

Контрольные вопросы:

1. Поясните процедуру инициализации частотного преобразователя и перечислите её этапы.
2. Поясните каким образом задается скорость вращения асинхронного двигателя в рамках выполненной лабораторной работы.
3. Назовите, какая синтаксическая конструкция используется в языке Си для реализации конечного автомата.
4. Как реализовать мигание лампочки с частотой 1 Гц?
5. Что такое операция мэпинга?

Лабораторная работа №3

Программное обеспечение системы управления одной степенью робота УРТК

Цель работы: получение навыков создания программного обеспечения систем управления одной степенью подвижности учебного робота.

Задание: создать функциональный блок, основанный на библиотеке Acp10sdc, который будет осуществлять управление одной осью УРТК (см. рисунок 3.1). Включая обработку концевых датчиков, реферирование к датчику, расположенному со стороны мотора (после реферирования ось переходит в состояние «отключено»). Управление реализовать из теста или с кнопок стенда.

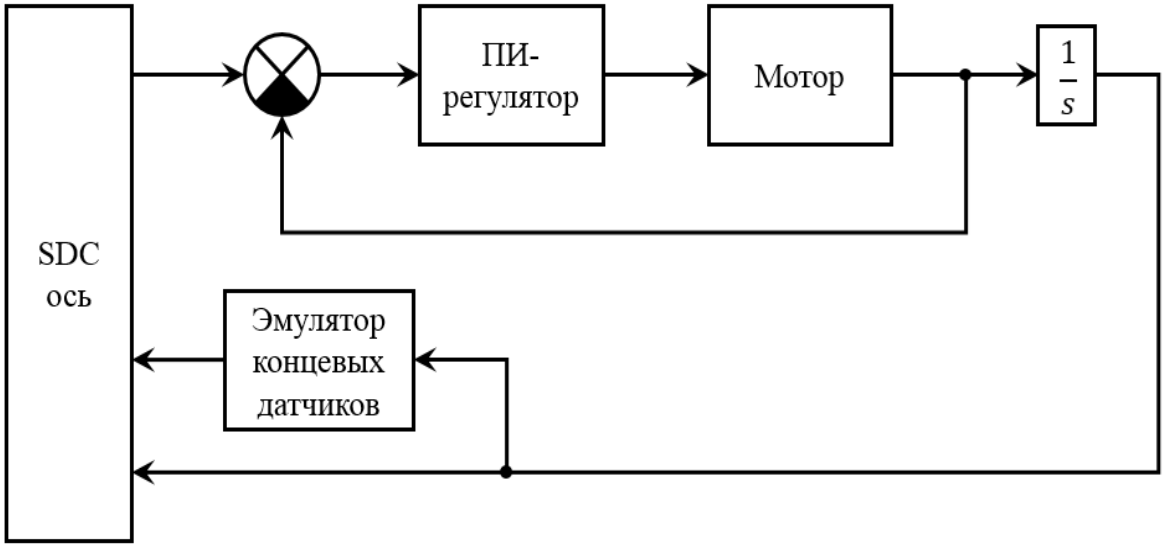

Рисунок 3.1. Структура системы управления одной степенью подвижности робота.

Порядок выполнения:

1. Создать проект.

При создании проекта необходимо руководствоваться наличием лабораторного стенда, на котором можно проводить тест. Если тестирование будет проводиться без него, рекомендуется при создании проекта добавлять в конфигурацию Standard PC, иначе - контроллер стенда 4PP065_0571_P74 (см. Указания по работе в среде Automation Studio).

Для создания симуляции проекта на ПК, необходимо добавить следующие модули и интерфейсы:

- 5LS182.6-1;

- X20BC0083;

- X20MM4456

- X20DI9371;

- X20D09322;

- 80VD100PS.C02X-01.

Конфигурация при работе с контроллером стенда аналогична конфигурации в лабораторной работе №2.

2. Добавить в проект библиотеки и конфигурационные таблицы для создания оси.

Для создания SDC-оси необходимо добавить библиотеки Acp10, а также создать в конфигурации NC-мэпинг. Это можно сделать как «вручную», так автоматически, путем добавления в проект любого motion-блока, к примеру, частотного преобразователя из лабораторной роботы №2.

После этого в проекте, помимо стандартного набора, появится следующий набор библиотек (см. рисунок 3.2):

-Acp10sdc;

-Acp10man;

-Acp10par;

-Acp10_MC;

-Acp10sim.

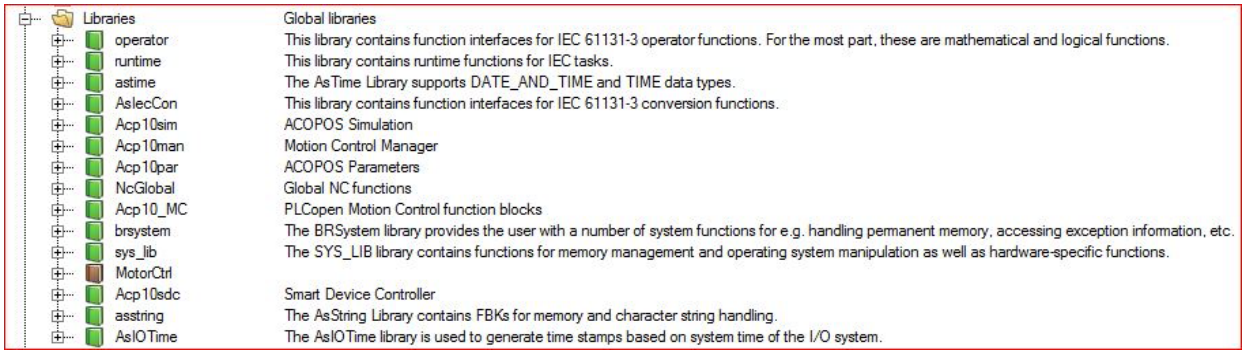

Рисунок 3.2. Библиотеки проекта.

Также необходимо добавить библиотеку AsIOTime.

В Global variables автоматически созданы переменные и структуры, необходимые для работы SDC. Для создаваемой в рамках данной лабораторной работы оси так же необходимы аналогичные структуры. Для этого скопируем их и переименуем только название оси, заменив его на Axis_X, поскольку в лабораторной работе №4 аналогично будут созданы «Axis_Y» и «Axis_Z» для

остальных осей УРТК. В итоге, набор глобальных переменных должен иметь следующий вид (см. таблицу 3.1 и рисунок 3.3):

Таблица 3.1. Глобальные переменные взаимодействия SDC-оси.

Имя	Тип	Описание
Axis_X	ACP10AXIS_typ	Инициализация SDC-оси в главной программе
Axis_X _HW	SdcHwCfg_typ	Переменная конфигурации аппаратных средств оси
Axis_X _DrvIf	SdcDrvIf16_typ	Переменная для контроля интерфейса привода
Axis_X _DiDoIf	SdcDiDoIf_typ	Переменная для контроля цифровых входов/выходов интерфейса привода
Axis_X _EncIf	SdcEncIf16_typ	Переменная для контроля датчиков двигателя

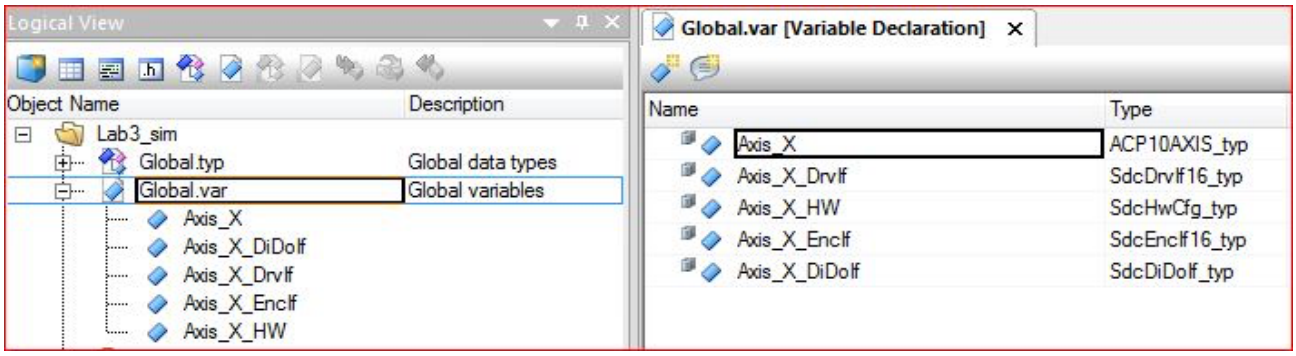

Рисунок 3.3. Глобальные переменные.

Для создания SDC-оси, если это не сделано автоматически, необходимо добавить в папку проекта, средством Toolbox, модуль инициализации оси «ACP10 Axis» (см. рисунок 3.4).

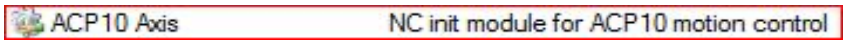

Рисунок 3.4 – Модуль инициализации оси.

Далее открываем его и в параметрах инициализации указываем реальные параметры оси УРТК (см. таблицу 3.2 и рисунок 3.5).

Таблица 3.2. Параметры инициализации SDC-оси.

Параметр	Значение	Описание
pos_hw_end	ncACTIV_HI	Состояние концевого датчика в положительном направлении
neg_hw_end	ncACTIV_HI	Состояние концевого датчика в отрицательном направлении
Units	3000	Количество юнитов на оборот [unit]
a1_pos	10000	Ускорение в положительном направлении [unit/sl]
a2_pos	10000	Замедление в положительном направлении [unit/sl]
a1_neg	10000	Ускорение в отрицательном направлении [unit/sl]
a2_neg	10000	Замедление в отрицательном направлении [unit/sl]
ds_stop	50000.0	Ошибка по положению ведущая к остановке [unit]
ds_warning	500.0	Ошибка по положению ведущая к предупреждению [unit]
Kv	10	Коэффициент П-регулятора положения [1/s]
v_switch	6000	Начальная скорость движения к концевому датчику [unit/s]
v_trigger	2000	Скорость движения вокруг концевого датчика [unit/s]
a	10000	Ускорение [unit/sl]
Mode	ncEND_SWITCH	Режим реферирования
edge_sw	ncNEGATIVE	Режим подъезда к датчику
trigg_dir	ncNEGATIVE	
fix_dir	ncOFF	

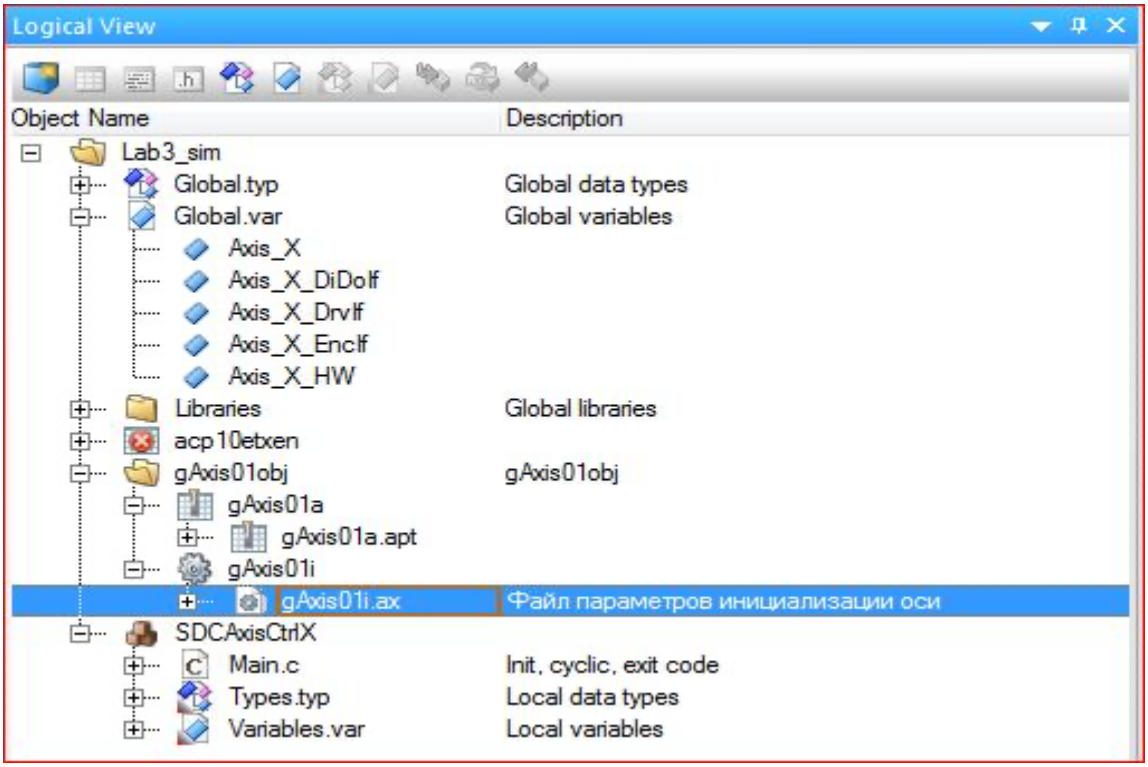

Рисунок 3.5. Файл параметров инициализации оси.

Далее, в случае, если таблица не была создана автоматически, средством Toolbox добавляем таблицу инициализации оси «ACOPOS Parameter table». В таблице указываем необходимые параметры (см. таблицу 3.3 и рисунок 3.6). Необходимые переменные могут быть добавлены посредством ввода ID, после чего пользователю остаётся указать свои значения.

Таблица 3.3. Параметры SDC-оси (AcpParTab.apt).

Имя	ID	Значение	Описание
SERVO_V_MAX_OUTPUT	64201	6500	Максимальная скорость вращения двигателя [unit/s]
SCALE_ENCOD_INCR	109	24	Число импульсов на оборот инкрементного датчика

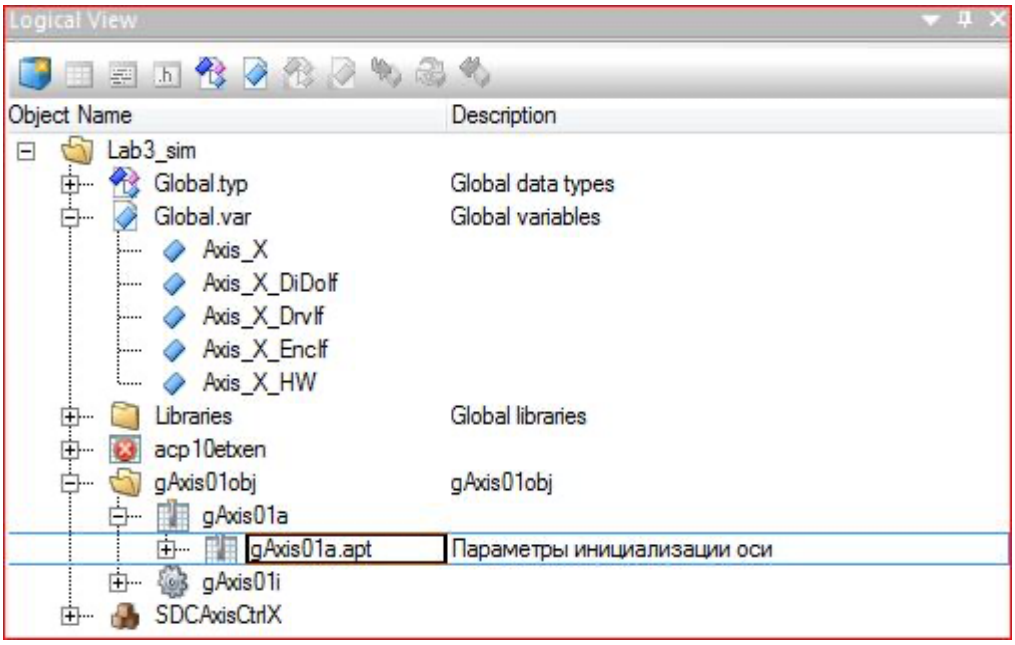

Рисунок 3.6. Параметры инициализации оси.

3. Мэпинг новой SDC оси.

Во вкладке «Configuration View» открываем конфигурацию «Real» и в каталоге «4PP065_0571_P74\Motion» средствами Toolbox – Object Catalog, выбрав фильтр «Motion», добавить в папку Motion файл «ACP10 NC Mapping Table» (см. рисунок 3.7).

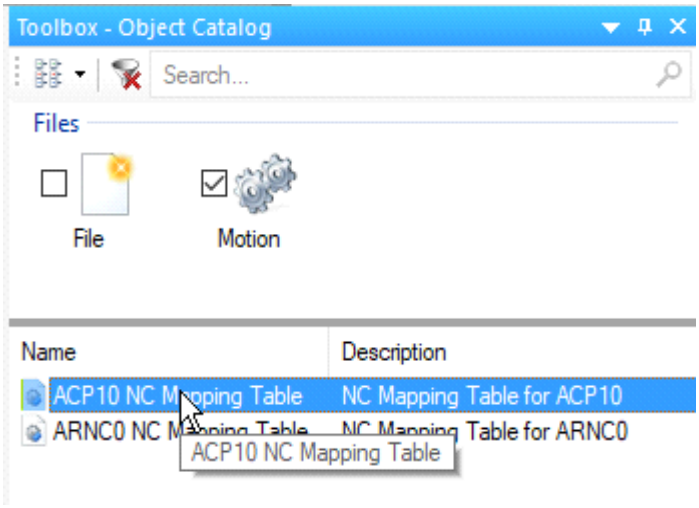

Рисунок 3.7. Создание новой таблицы мэппинга осей.

В созданную таблицу добавляем SDC ось, открываем новую таблицу (см. [1] на рисунке 3.8), вызываем контекстное меню правым щелчком мыши, создаем новую ось (см. [2] на рисунке 3.8).

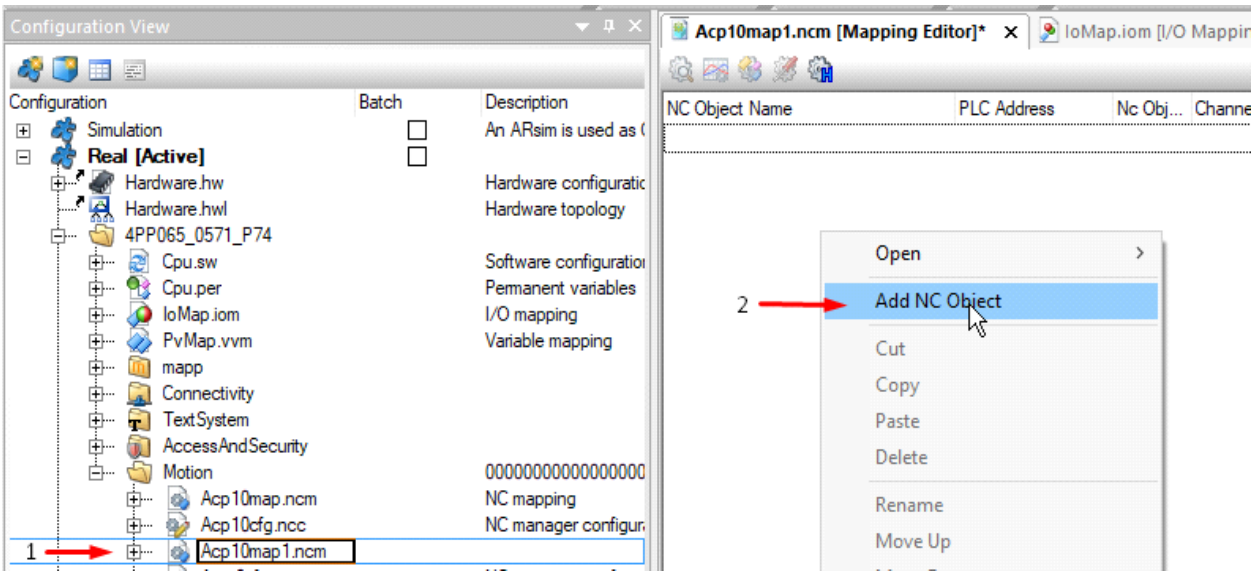

Рисунок 3.8. Добавление оси.

Далее необходимо заполнить параметры новой оси (см. таблицу 3.4 и рисунок 3.9).

Таблица 3.4. Параметры мэпинга создаваемой SDC-оси.

Имя NC объекта (оси)	PLC адрес	Тип NC объекта	Таблица инициализации	Таблица параметров ACOPOS
Axis_X	SDC_IF1.ST2	ncAXIS	gAxis01i	gAxis01a

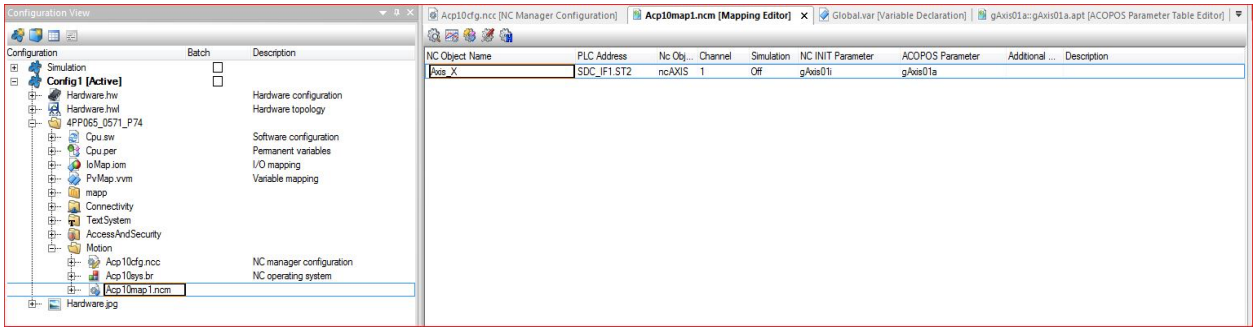

Рисунок 3.9. Параметры новой оси.

Используя модель регулятора и объекта управления из лабораторной работы №1, разработать управление одной осью робота.

Скопировав в данный проект библиотеку «MotorControl» из лабораторной работы №1, добавляем в нее функциональный блок «FB_Axis». Данный функциональный блок будет отвечать за определение входного воздействия на ось двигателя путем задания ШИМ сигнала. Параметры блока описаны в таблице 3.5.

Таблица 3.5. Параметры функционального блока FB_Axis.

Конфигурация	Имя	Тип данных	Описание
вход	reset_error	BOOL	Сброс ошибок мотора
вход	endswitch_a_reached	BOOL	Состояние начального концевого датчика
вход	endswitch_b_reached	BOOL	Состояние конечного концевого датчика
выход	reset_counter	BOOL	Сброс счетчика
выход	pwm_value	INT	Время импульса ШИМ
выход	counter	INT	Счетчик импульсов
выход	speed	REAL	Скорость вращения оси двигателя
внутреннее состояние	last_counter	INT	Хранение предыдущего значения counter в процессе расчета

4. Создание программы обработки одной оси с применением модернизированной библиотеки управления двигателем.

Для реализации этого необходимо создать ANSI C Program с названием «SDCAxisCtrlX» и добавить в нее необходимые переменные (см. таблицу 3.6).

Таблица 3.6. Переменные программы SDCAxisCtrlX.

Имя	Тип данных	Описание
axis_X	FB_Axis	Переменная оси
coil_powered	BOOL	Включение обмотки возбуждения
coli_pwm_value	INT	Время импульса ШИМ
fb_controller	FB_Controller	Переменная регулятора
pwm_period	UINT	Период ШИМ

Написать программу, реализующую алгоритм, представленный на рисунке 3.10. Цикл программы – 2мс.

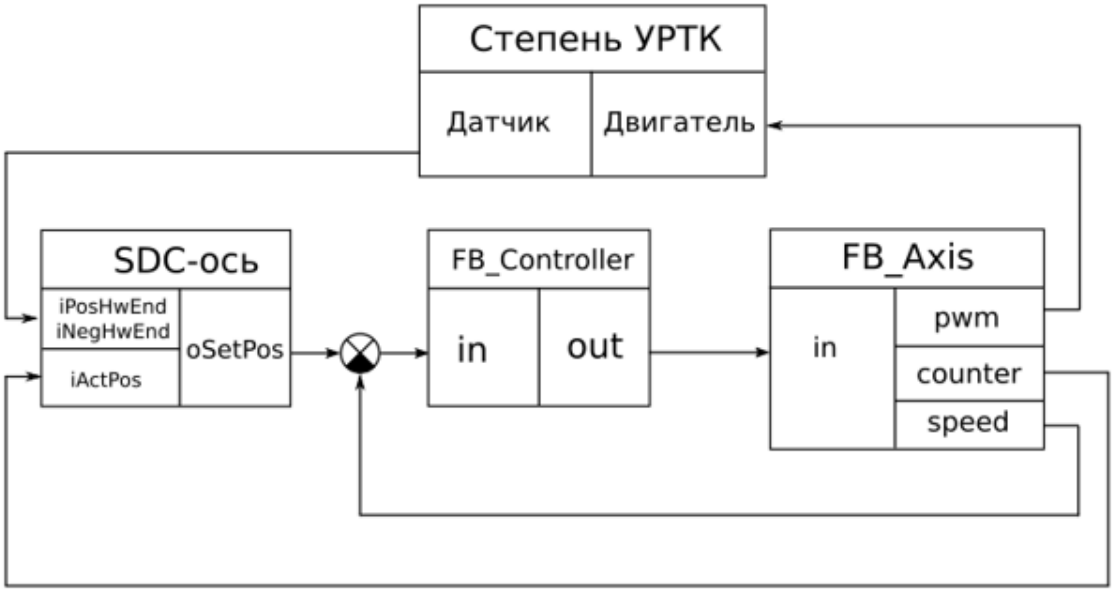

Рисунок 3.10. Структура программного обеспечения.

При реализации алгоритма управления важно выполнять перевод всех значений в одни единицы измерения. Так с OSetPos поступает скорость в Unit/s, помноженная на 32767 и деленная на количество меток на оборот, а с функционального блока FB_Axis на сумматор поступает скорость в мм/с. Следовательно, перед тем как проводить математические операции с данными значениями, их следует привести к единым единицам измерения – Unit/s. А также напряжение перед подачей на двигатель следует перевести из Unit/s в вольты.

Дополнительно, в части инициализации основной программы, необходимо:

- Установить параметры и входы готовности SDC-оси (листинг 1);

- Установить константы для FB_Control (аналогично ЛР №1).

Листинг 1

```
//Устанавливаем типы SDC модулей
Axis_X_HW.EncIf1_Typ = ncSDC_ENC16;
Axis_X_HW.DiDoIf_Typ = ncSDC_DIDO;
Axis_X_HW.DrvIf_Typ= ncSDC_DRVSERVO16;
//Устанавливаем имена переменных
strcpy(Axis_X_HW.EncIf1_Name, "Axis_X_EncIf");
strcpy(Axis_X_HW.DrvIf_Name, "Axis_X_DrvIf");
strcpy(Axis_X_HW.DiDoIf_Name,
"Axis_X_DiDoIf");
//Устанавливаем входы готовности и нормальной
работы
Axis_X_EncIf.iEncOK = 1;
```

```
    Axis_X_DrvIf.iDrvOK          =          1;
Axis_X_DrvIf.iStatusEnable       =          1;
Axis_X_DiDoIf.iDriveReady = 1;
```

В основном цикле программы необходимо также:

- Инкрементировать LiveCounter-ы SDC оси (см. листинг 2), чтобы ось не падала в ошибку, указать iActTime как время начала цикла;

- Включать и отключать обмотку возбуждения в соответствии со значением переменной выключателя.

Листинг 2

```
    Axis_X_EncIf.iLifeCnt++;

    Axis_X_DiDoIf.iLifeCntDriveEnable++;

    Axis_X_DiDoIf.iLifeCntDriveReady++;

    Axis_X_DiDoIf.iLifeCntNegHwEnd++;

    Axis_X_DiDoIf.iLifeCntPosHwEnd++;

    Axis_X_DiDoIf.iLifeCntReference++;

    Axis_X_DrvIf.iLifeCnt++;

    Axis_X_EncIf.iActTime=
(INT)AsIOTimeCyclicStart();
```

5. Провести симуляцию, подтверждающую работоспособность программного обеспечения.

Прежде чем тестировать разработанную программу на УРТК, выполним симуляцию, которая позволит выявить ошибки и отладить программное обеспечение.

Для обеспечения симуляции нам необходимо сымитировать следующие процессы: получение импульсов с рабочего устройства, наезд на концевые датчики.

За принятие импульсов с рабочего устройства отвечает переменная counter. В симуляции имитацией может послужить инкрементирование данной переменной на величину меток, которые теоретически были бы получены за один цикл программы.

Как уже упоминалось, для реферирования оси, необходимо будет сымитировать концевые датчики. Сделать это можно несколькими способами:

a. Осуществить программное изменение значений переменных endswitch_a_reached или endswitch_b_reached по достижении определённого значения переменной counter.

b. Производить изменение значений переменных endswitch_a_reached или endswitch_b_reached посредством опции monitor (см. рисунок 3.11).

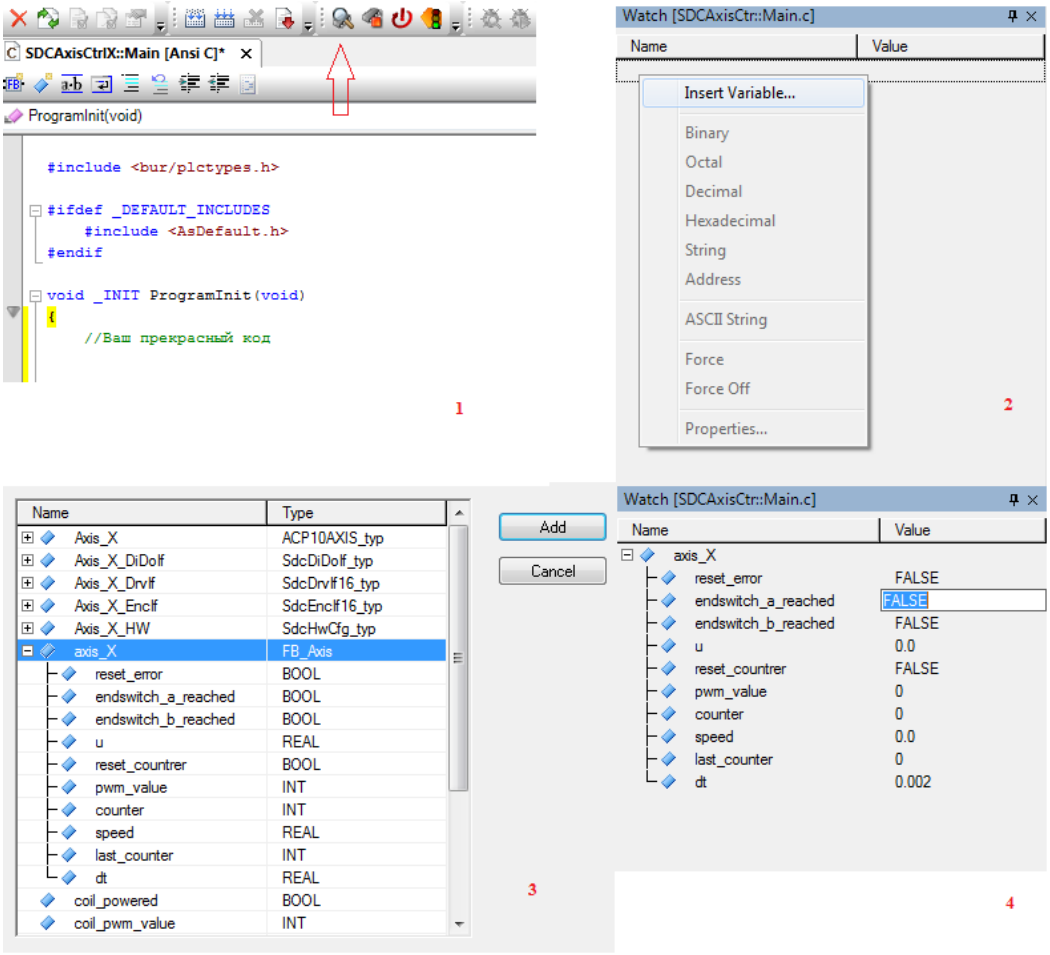

Рисунок 3.11. Изменение значений в режиме мониторинга.

c. Данный способ возможен только при работе с реальным стендом. Состоянием переменных можно управлять с кнопок стенда, как в лабораторной работе №2. Для этого необходимо выполнить мэпинг переменных (см. таблица 3.7)

Таблица 3.7. Мэпинг переменных для датчиков положения.

Имя канала	Используемая переменная
DigitalInput02	Axis_X.endswitch_a_reached

DigitalInput03	Axis_X.endswitch_b_reached

Выполним симуляцию средством «Test». Для этого во вкладке «Configuration View» в папке «Motion» открываем файл «Acpl0map.ncm». В открывшемся окне нажимаем на Axis_X правой кнопкой мыши и выбираем «Test» → «Parallel Mode». Далее выполняем инициализацию параметров и включаем контроллер (рисунок 3.12)

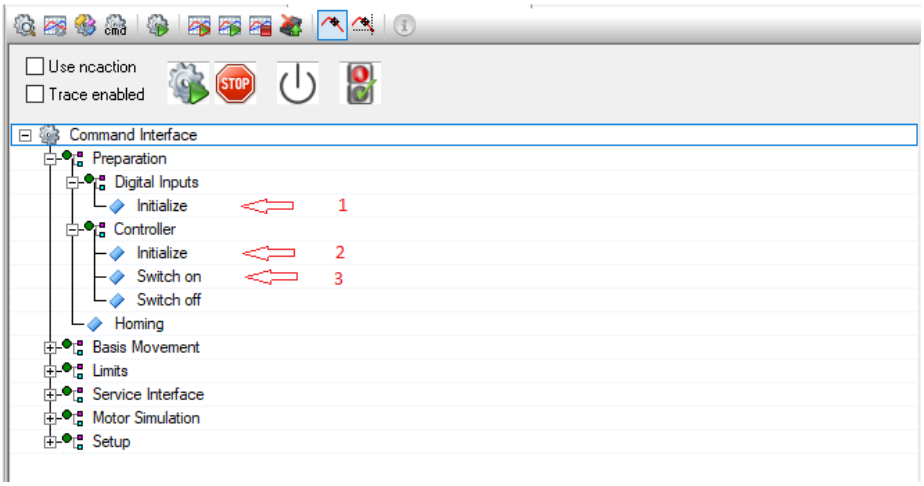

Рисунок 3.12. Инициализация и включение контроллера.

Если не возникло ошибок, то иконка в окне «Watch» загорится зеленым. Далее нажимаем «Homing». При этом значение переменной monitor.s будет меняться. При последовательном нажатие – отжатие – нажатие одной из кнопок стенда, ось отреферируется. При этом загорится иконка «Referensed». Изменения, которые должны произойти с иконками в окне «Watch», представлены на рисунке 3.13.

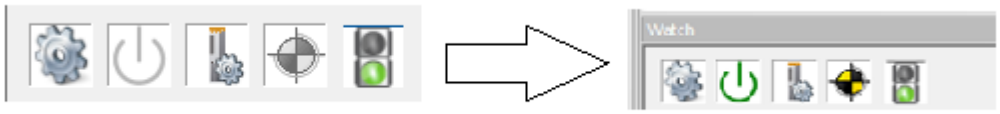

Рисунок 3.13. Окно «Watch» до и после реферирования оси.

При возникновении ошибок крайняя иконка справа загорится красным, нажав на нее можно просмотреть номер и значение ошибки. Если информации, содержащейся в данном окне, недостаточно, то можно воспользоваться логгером (Ctrl + L) или справкой (F1).

Снять графики можно средством «NC Trace» (рисунок 3.14).

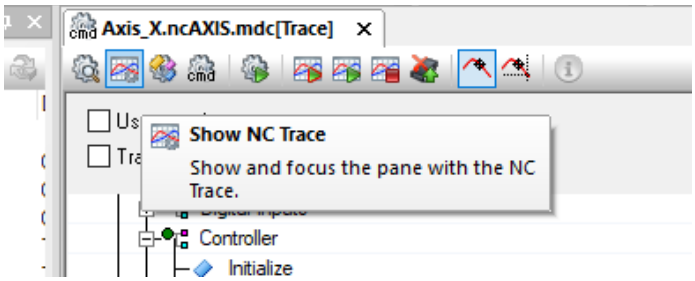

Рисунок 3.14. NC Trace.

Нажав правой кнопкой мыши в открывшемся окне «Trace» и открыв «Configuration» можно выбрать значения для записи и во вкладке «Timing» увеличить время снятия трейса (рисунок 3.15).

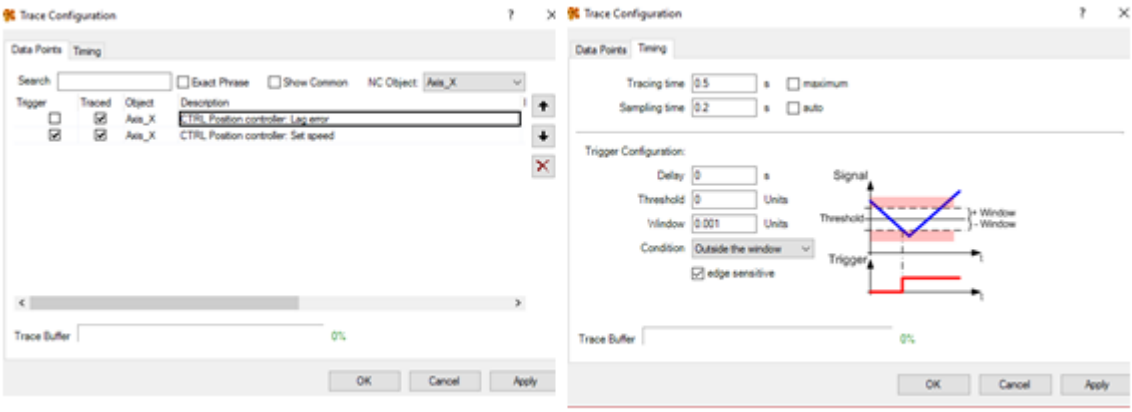

Рисунок 3.15. Trace Configuration.

Средством «Trace» записать графики перемещения оси по ParID:

- 111

- 112

- 114

- 113

- 462

6. Работа с реальным оборудованием. Мэпинг переменных в физических блоках.

Для работы с реальным оборудованием рекомендуется создать ещё одну конфигурацию (Configuration View → Add Configuration…). Также стоит изменить код, аккуратно убрав элементы имитации.

Примечание: также на случай непредвиденных ошибок рекомендуется замэпить кнопки для возможности управления осями при несрабатывании

концевых датчиков. Иначе будете ручками отгонять.

В таблицах 3.7 и 3.8 показан мэпинг параметров оси и питания двигателя для выполнения программного обеспечения на аппаратной модели.

Таблица 3.7. Мэпинг переменных SDC-оси (блок X20MM4456).

Наименование	Переменная процесса	Описание
PeriodDurationPWM	SDCAxisCtr:pwm_period	Период ШИМ
PulseWidthCurrentPWM01	SDCAxisCtr:axis_X.pwm_value	Время импульса ШИМ
ClearError01	SDCAxisCtr:axis_X.reset_error	Сброс ошибок мотора
ResetCounter01	SDCAxisCtr:axis_X.reset_counter	Сброс счетчика
Counter01	SDCAxisCtr:axis_X.counter	Счетчик импульсов
StatusInput03	SDCAxisCtr:axis_X.endswitch_a_reached	Состояние начального концевого датчика
StatusInput04	SDCAxisCtr:axis_X.endswitch_b_reached	Состояние конечного концевого датчика
PulseWidthCurrentPWM04	SDCAxisCtr:coil_pwm_value	Параметр определяющий значение максимальной величины ШИМ

Таблица 3.8. Мэпинг переменной питания обмотки возбуждения (в блокеX20DI9371).

Наименование	Переменная процесса	Описание

DigitalInput01	SDCAxisCtr:coil_powered	Питание двигателя

7. Провести эксперименты, подтверждающие работоспособность системы управления на реальной модели УРТК.

- Открыть ось средством «Test»;
- Произвести инициализацию контроллера оси;
- Выполнить операцию «Homing»;
- Средством «Trace» записать графики перемещения оси по ParID:

- 111

- 112

- 114

- 113

- 462

Дополнительные ParId перечислены в ACAPOS Parameter IDs Overview

Содержание отчета:

1. Титульный лист
2. Цель работы
3. Задание
4. Основная часть:
   - 4.1. описание действий и использованного аппарата
   - 4.2. структурная схема конечного автомата
   - 4.3. описание состояний
   - 4.4. описание функциональных блоков (назначение самого блока; входы и выходы, их назначение и работа)
   - 4.5. описание структуры программного обеспечения (например, в виде структурной схемы функциональных блоков)
   - 4.6. результаты экспериментальных исследований
5. Выводы из лабораторной работы
6. Приложение (листинг кода)

Контрольные вопросы:

1. Опишите программный интерфейс сервопривода постоянного тока, совместимый с технологией SDM.

2. Поясните каким образом осуществляется регулирование скорости и положения в рамках выполненной лабораторной работы.
3. Назовите, основные способы диагностики сервопривода Acopos.
4. Перечислите типовые перемещения сервопривода Acopos.

Лабораторная работа №4

Программное обеспечение системы управления робота УРТК с интегрированной системой ЧПУ

Цель работы: получение навыков управления тремя степенями робота УРТК с помощью интерфейса формирования траекторий (ЧПУ).

Задание: на основе библиотеки ARNC0 создать интерфейс формирования траекторий (траекторный задатчик), который позволит управлять всеми степенями подвижности робота УРТК (рис. 4.1). Создать программное обеспечение, которое позволит отрабатывать заданную траекторию по нажатию кнопки.

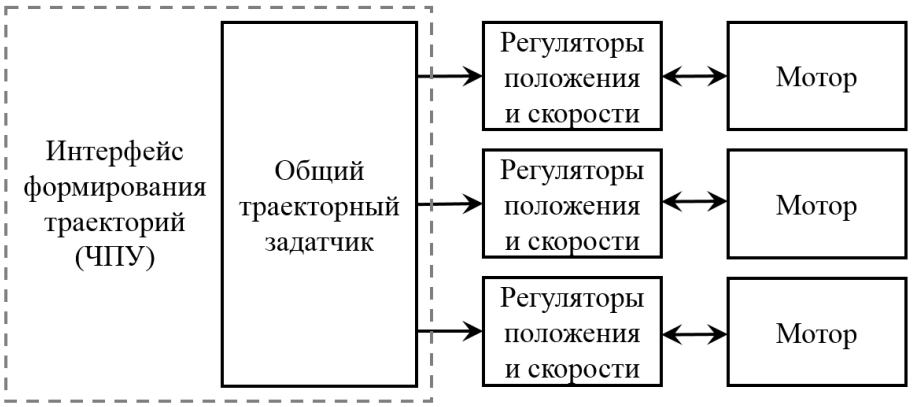

Рисунок 4.14. Управления роботом с помощью ЧПУ.

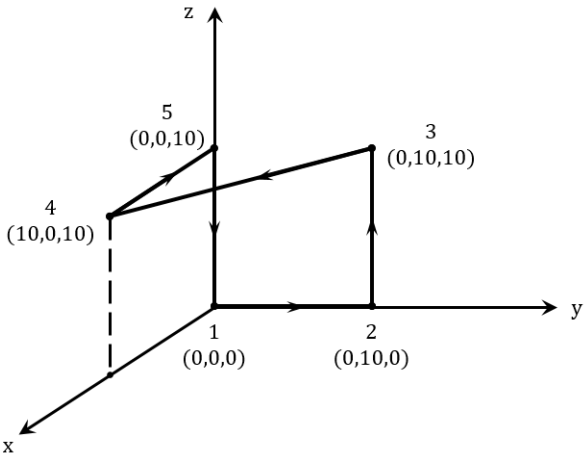

Рисунок 4.15. Заданная траектория движения.

Порядок выполнения:

1. Добавить в проект лабораторной работы №3 недостающие оси УРТК и сконфигурировать их на работу с ЧПУ.

1.1. Продублировать глобальные переменные и управляющие программы для каждой из новых осей, заменив соответственно в их названиях

«X» на «Y» и «Z», добавив так же новые программы в нужный Cyclic исполнения и выполнив мэпинг SDC-оси и физических входов-выходов.

1.2. Для разрешения осям работать с ЧПУ, в мэпинге каждой оси, в строке «Additional Data» добавить параметр «CNC_Enabled = 1».

Структура программного обеспечения представлена на рисунке 4.2.

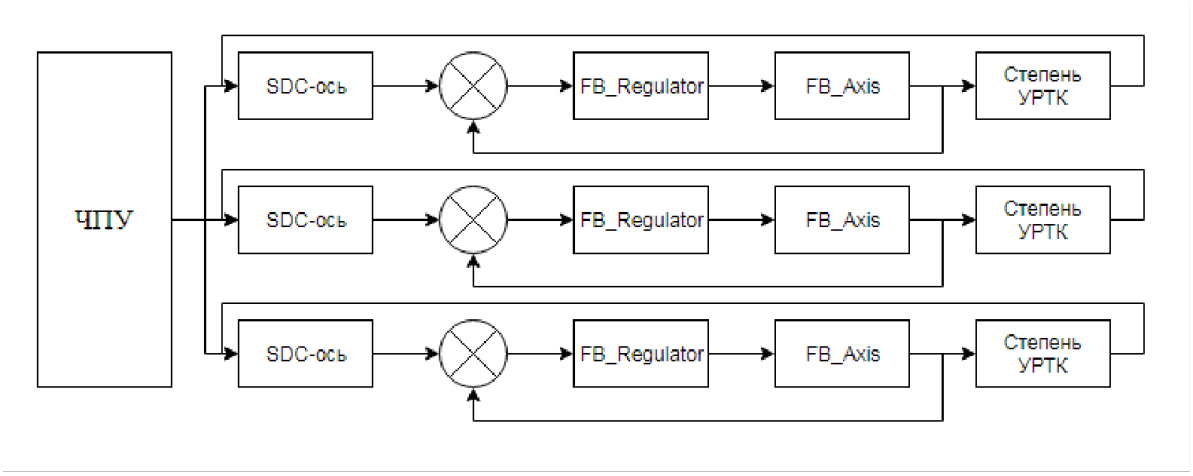

Рисунок 4.16. Структура программного обеспечения.

2. Подключение и конфигурация системы ЧПУ.

2.1. Подключение системы ЧПУ происходит путем добавления библиотек и управляющего цикла ЧПУ. Для этого необходимо, аналогично лабораторной работе №3 добавить любой motion блок в конфигурацию, в параметрах добавления оси которого поставить галочку «CNC».

По результатам данной операции, в проект добавится библиотека Arnc0man, мэпинг CNC (в каталоге «4PP065_0571_P74\Motion») и конфигурационные ЧПУ программы.

2.2. Для работы ЧПУ требуется создать и заполнить таблицу конфигурации системы «CNC Init». Создается она в меню Logical View, в папке проекта, средством «Toolbox – Object Catalog». В этом меню сразу можно добавить в проект новую программу «CNC Program», которая понадобится позже.

2.3. В полях таблицы инициализации указываем параметры, которые будут применяться к привязанным осям:

Таблица 4.11. Параметры инициализации SDC-оси (CNCInit).

Переменная	Состояние	Значение	Описание
v	limit	6500	Максимальная скорость движения по траектории [unit/s]
a_pos	limit	10000	Ускорение в положительном направлении [unit/sl]
a_neg	limit	10000	Ускорение в отрицательном направлении [unit/sl]
nc_object_name	asix→axis[0]	AxisX	Имя NC объекта
name	asix→axis[0]	X	Имя оси в программе ЧПУ
nc_object_name	asix→axis[1]	AxisY	Имя NC объекта
name	asix→axis[1]	Y	Имя оси в программе ЧПУ
nc_object_name	asix→axis[2]	AxisZ	Имя NC объекта
name	asix→axis[2]	Z	Имя оси в программе ЧПУ

2.4. Создаем глобальную структуру «CNCsys» типа «ARNC0CNC_typ», которая нужна для функционирования системы ЧПУ.

2.5. Производим мэпинг системы в файле «Arnc0map», находящемся каталоге «4PP065_0571_P74\Motion». Данные вносятся на место уже созданного объекта CNC1:

Таблица 4.12. Мэпинг системы ЧПУ.

Поле	Значение
NC Object Name	CNCsys
NC INIT Parameter	CNC Init

На этом создание системы ЧПУ закончено и можно приступить к созданию траектории для учебного робота.

3. Создание программного обеспечения для отработки траектории.

Движение по траектории задаем в программе ЧПУ (CNCProg). Программы ЧПУ пишутся на языке G-кодов. В данном случае используется команда линейной интерполяции (G01).

При исполнении функции G01, дается команда на абсолютное или относительное перемещение по указанным координатам и заданной скоростью в виде:

Номер команды	Команда	Координаты	Линейная скорость
N10	G01	X10000 Y50000	F1000
N20	G01	X0	
N30	M30		

Примечание: команда «M30» - флаг окончания программы.

3.1. Написать управляющую программу для отработки тракетории из задания лабораторной работы (рис. 4 .15)

Скорость движения оси выберем близкую к предельной для оси, а именно $6500\frac{units}{s}$, *однако,*поскольку скорость движения по траектории в данной команде задается в $\frac{units}{min}$ то она будет равна $6500\frac{units}{s} = 390000\frac{units}{min}$.

4. Провести эксперименты, подтверждающие работоспособность системы управления на реальной модели УРТК.

4.1. Загрузить проект на стенд (см. приложение)

4.2. Для подготовки к исполнению траектории, необходимо включить контроллеры и произвести операцию реферирования для всех осей (аналогично лабораторной работе №3).

4.3. Далее открыть мэпинг CNC и открыть для созданной системы ЧПУ средство отладки Test.

4.4. Далее следует выбрать из списка созданную ранее программу (стандартно «CncProg»), и, запустив программу одновременно с средством Trace, снять графики перемещения УРТК.

Содержание отчета:

1. Титульный лист
2. Цель работы
3. Задание

4. Основная часть (описание действий и использованного аппарата):
   4.1. структурная схема конечного автомата
   4.2. описание состояний
   4.3. описание функциональных блоков (назначение самого блока; входы и выходы, их назначение и работа)
   4.4. описание структуры программного обеспечения (например, в виде структурной схемы функциональных блоков)
5. Выводы из лабораторной работы
6. Приложение (листинг кода)

Контрольные вопросы:

1. Назовите возможные способы применения систем ЧПУ в автономных мобильных роботах, оснащенных манипулятором.
2. Опишите процедуру инициализации сервоприводов робота, необходимую для последующего управления ими при помощи системы ЧПУ.
3. Опишите способ реализации синхронного движения двух приводов с заданным соотношением их скоростей.
4. Назовите условия использования системы ЧПУ.
5. Что такое язык G-кодов?

Указания по работе в среде Automation Studio

1. Создание нового проекта.

В меню Automation Studio нажать File

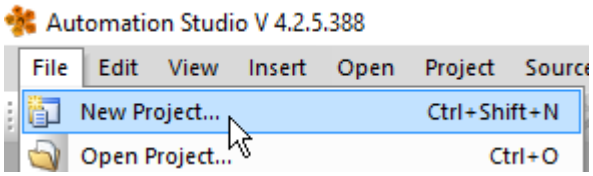

Рисунок 4.1.1. Создание нового проекта.

В раскрывшемся окне написать название проекта ([1] на рис. Рисунок 4. 1 .2) и выбрать директорию где он будет создан ([2] на рис Рисунок 4. 1 .2).

Если не требуется работать с аппаратными модулями B&R, а использовать AS только в качестве среды моделирования (как в лабораторной работе №1), то необходимо поставить флажок «Use Automation Runtime Simulation» ([3] на рис. Рисунок 4. 1 .2) и нажать на клавишу «Finish» ([4] на рис. Рисунок 4. 1 .2).

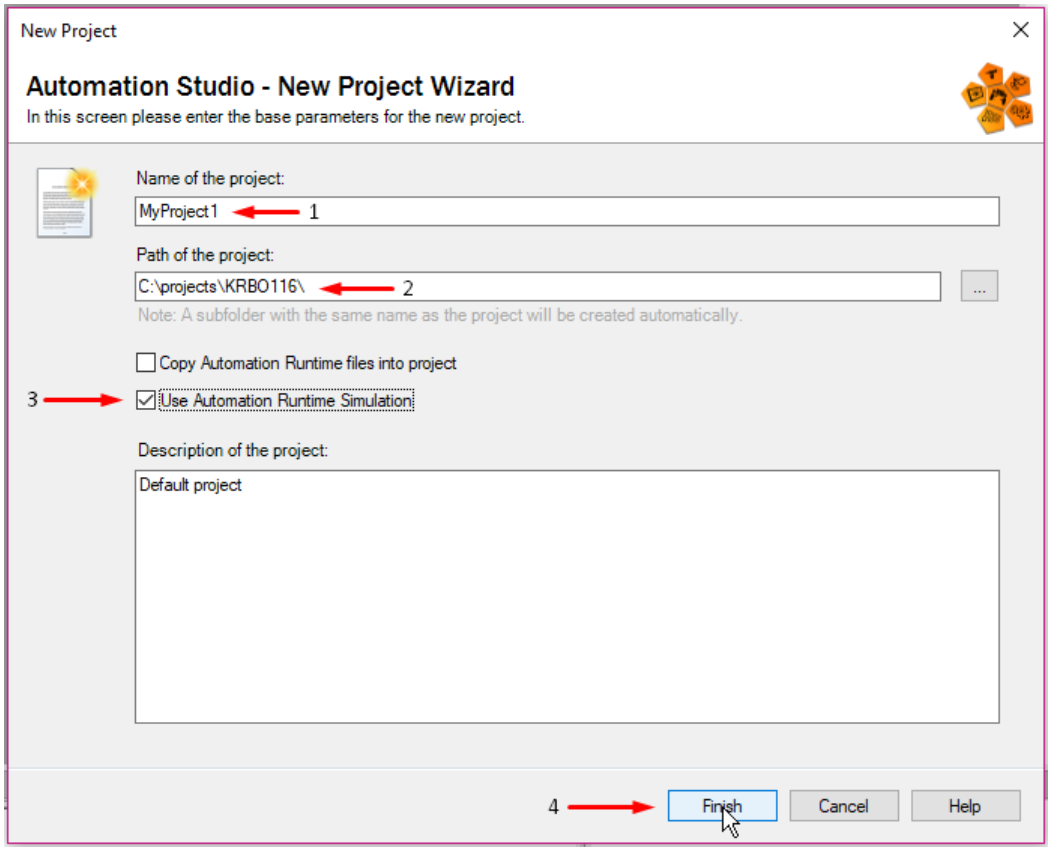

Рисунок 4.1.2. Создание нового проекта без конфигурации оборудования

Если требуется работа с реальной аппаратурой, то конфигурируем данное поле в соответствии с рисунком 4. 1 .3.

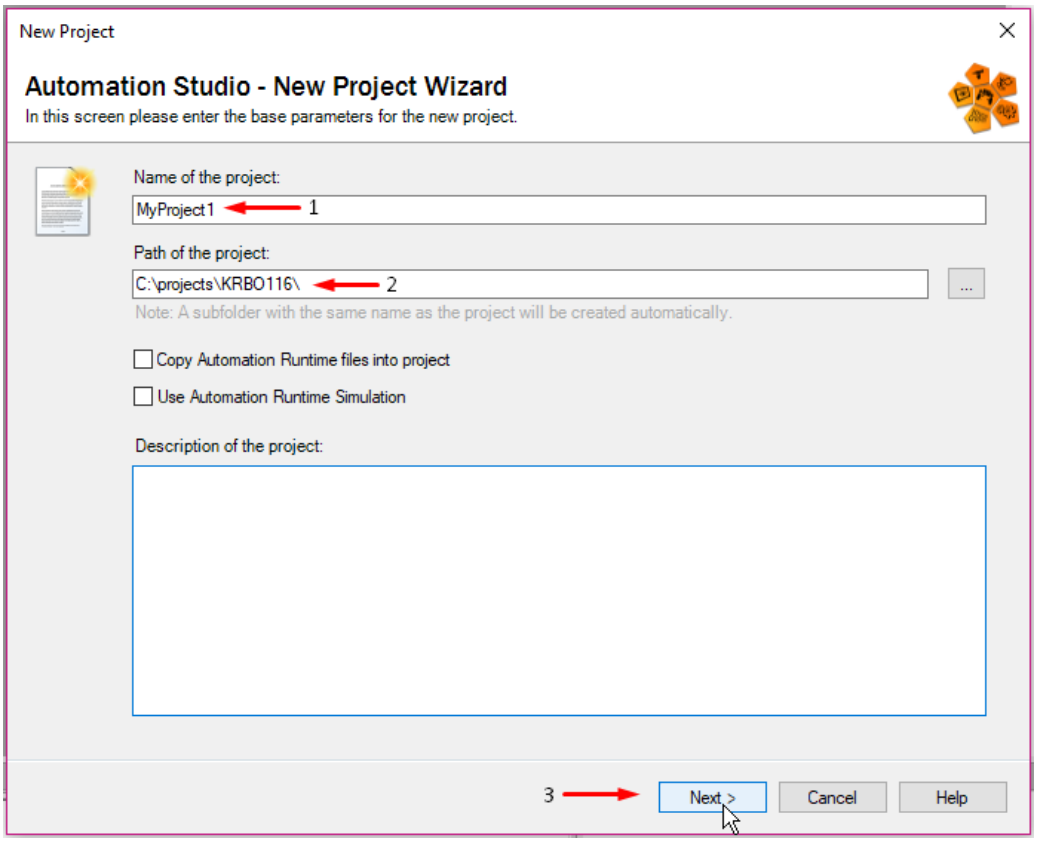

Рисунок 4.1.3. Создание нового проекта с конфигурацией оборудования.

2. Конфигурация оборудования

При создании проекта с конфигурацией оборудования, указываем имя конфигурации ([1] на рис. 4. 2 .4), указываем ручное создание конфигурации ([2] на рис. 4. 2 .4), и нажимаем далее конфигурации ([3] на рис. 4. 2 .4).

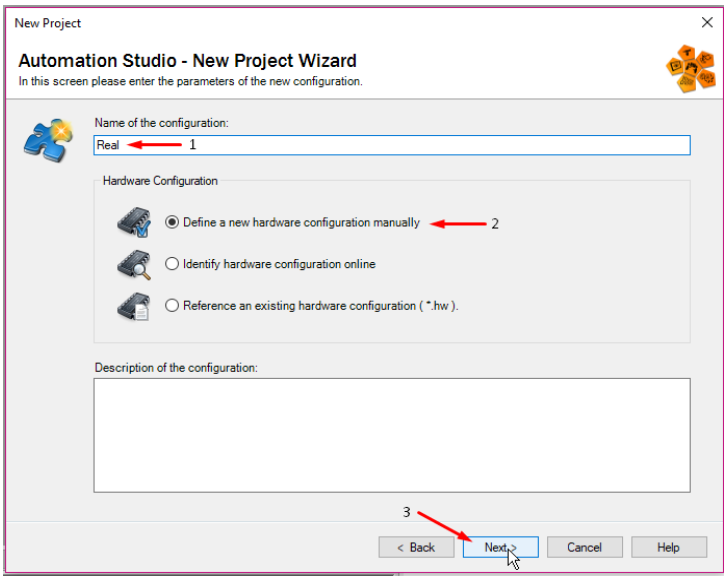

Рисунок 4.2.4. Создание новой конфигурации.

Выбираем в меню контроллер стенда: 4PP065_0571_P74:

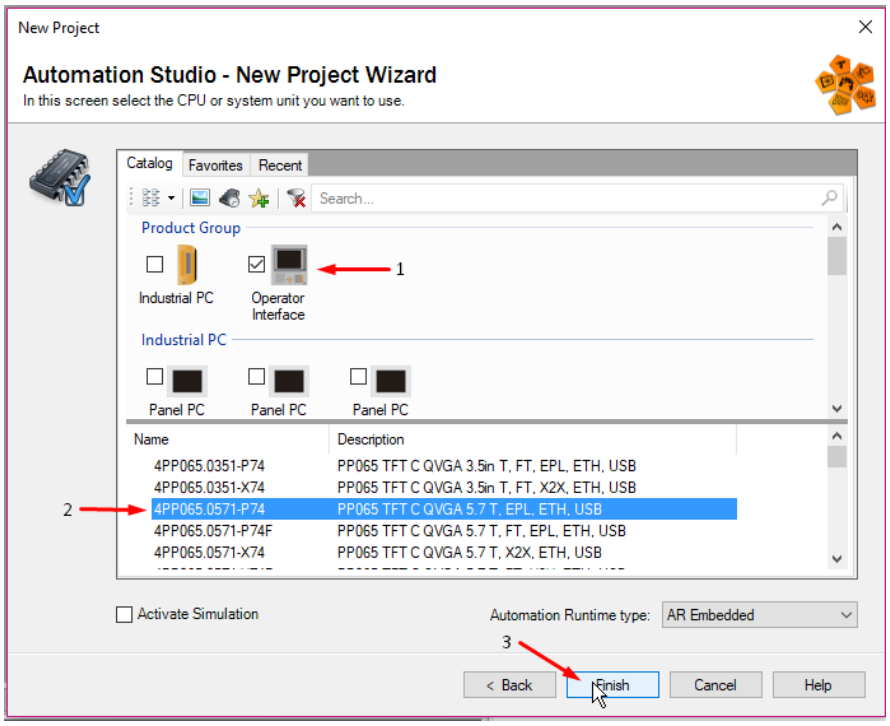

Рисунок 4.2.5. Выбор контроллера при конфигурации.

В созданном проекте, в визуальной среде настройки конфигурации «System Desiner» составляем следующую конфигурацию.

Таблица 3.13 Модули учебного стенда.

Модуль	Назначение	Порты коммутации модулей
4PP065_0571_P74	ПЛК	IF4 - PLC1
X20SL8000	Safety модуль	
X20BC0083	Контроллер шины X2X	PLC2 – PLC1
X20SM1436	Блок управления шаговым двигателем	X2X
X20SM1436	Блок управления шаговым двигателем	

X20MM4456	Блок управления двигателям постоянного тока	
X20DI9371	Блок цифровых входов	
X20DO9322	Блок цифровых выходов	
X20AT4222	Блок обработки датчиков температуры	
X20SL4410	Safety модуль	
X20SO4410	Safety модуль	
X20BT9100	Модуль подключения внешних устройств по X2X	X2X1 - X2X1
8I64xxxxxxx.00x-1	Частотный преобразователь	

В итоге должна получиться следующая конфигурация, аналогичная рисунку 2.7.

3. Создание программ, библиотек и конфигурация функциональных блоков.

3.1. Создание программы

3.1.1. Во вкладке “Logical View” следуем, нажав на названии проекта, в меню “Logical View” щелкнуть по кнопке “New Object”:

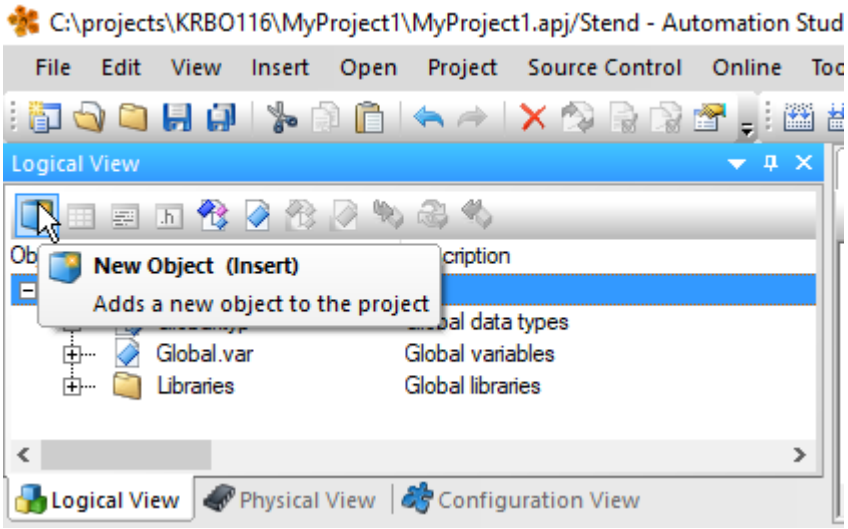

Рисунок 4.4.6. Создание новой программы.

3.1.2. В открывшемся окне “Toolbox” выделить из папки “Program” ([1] на рис. 4. 4 .7) и дважды щелкнуть на меню ANSI C Program ([2] на рис. 4. 4 .7)

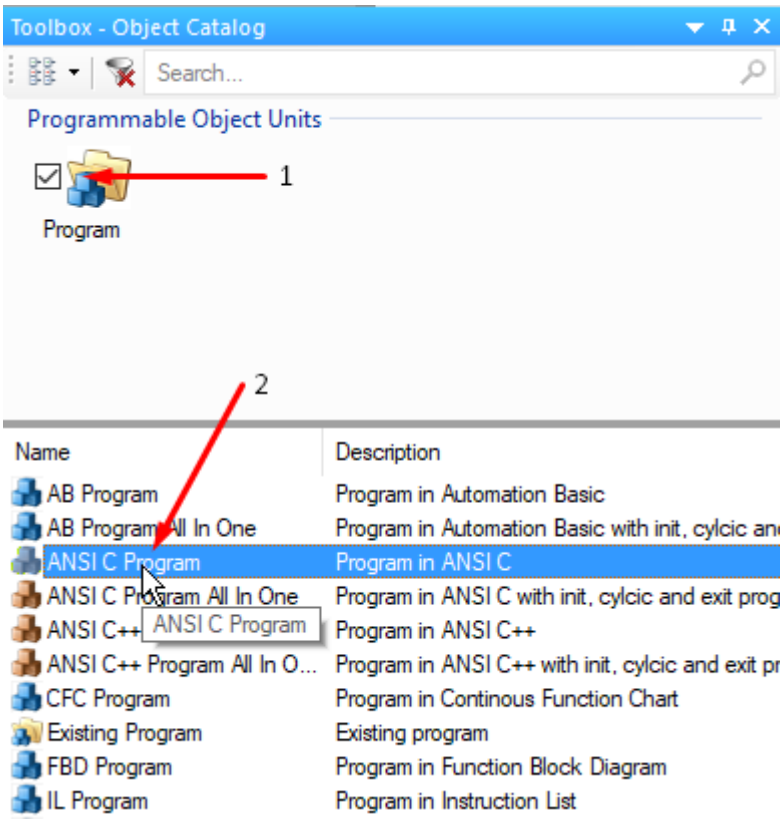

Рисунок 4.4.7. Создание новой программы.

В результате появится программа, название которой можно изменить опцией “rename”.

Программа содержит в себе следующие объекты:

-Cyclic.c – основной цикл программы;

-Init.c Подпрограмма инициализации;

-Exite.c;

-Types.typ;

-Variables.var.

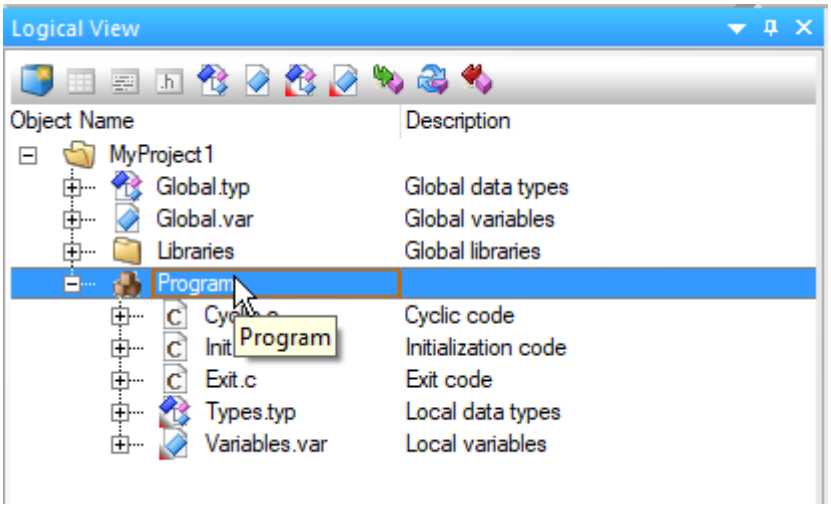

Рисунок 4.4.8. Результат создания новой программы.

3.2. Создание библиотеки

3.2.1. В проекте выбрать папку с библиотеками

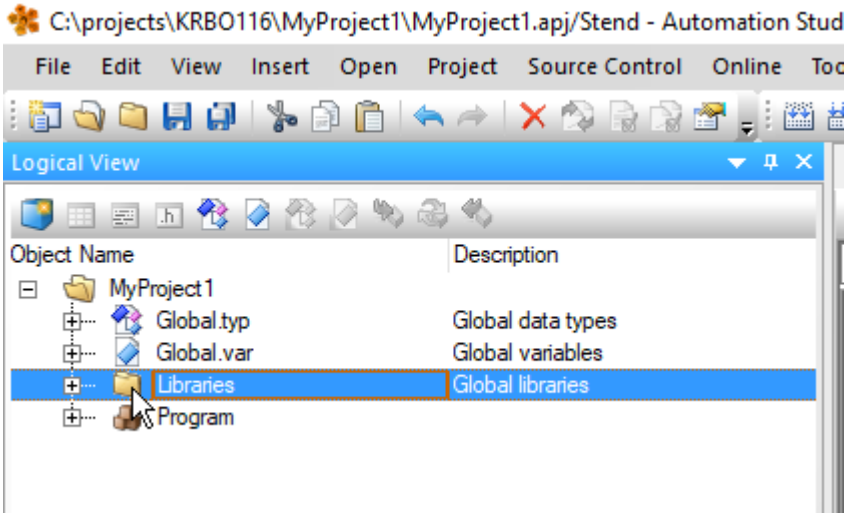

Рисунок 4.4.9. Выбор папки с библиотеками.

1.1.1. В меню ToolBox выбрать папку “Library” ([1] на рис. 4. 4 .10) и создать библиотеку двойным кликом “ANSI C Library” ([2] на рис. 4. 4 .10).

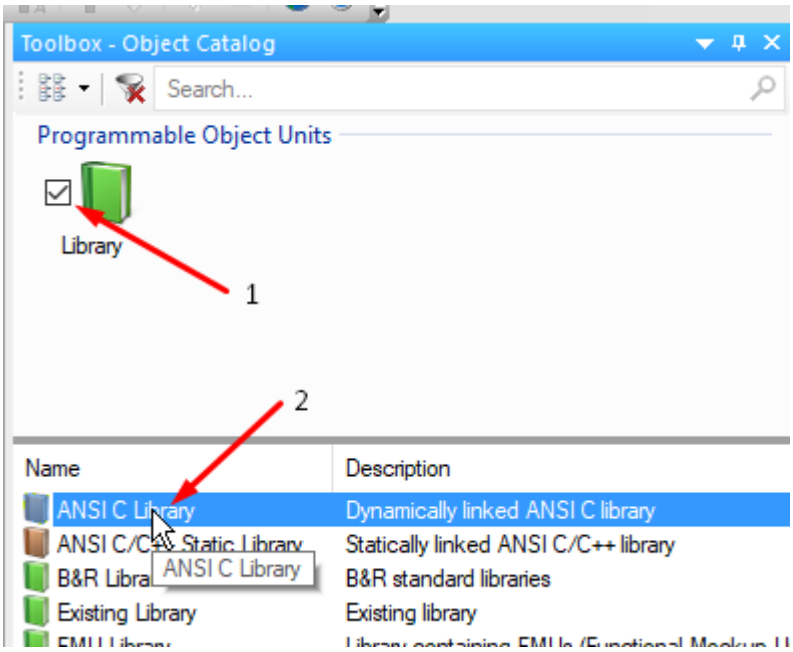

Рисунок 4.4.10. Создание новой библиотеки.

В результате создаётся пользовательская библиотека, название которой можно изменить опцией “rename”.

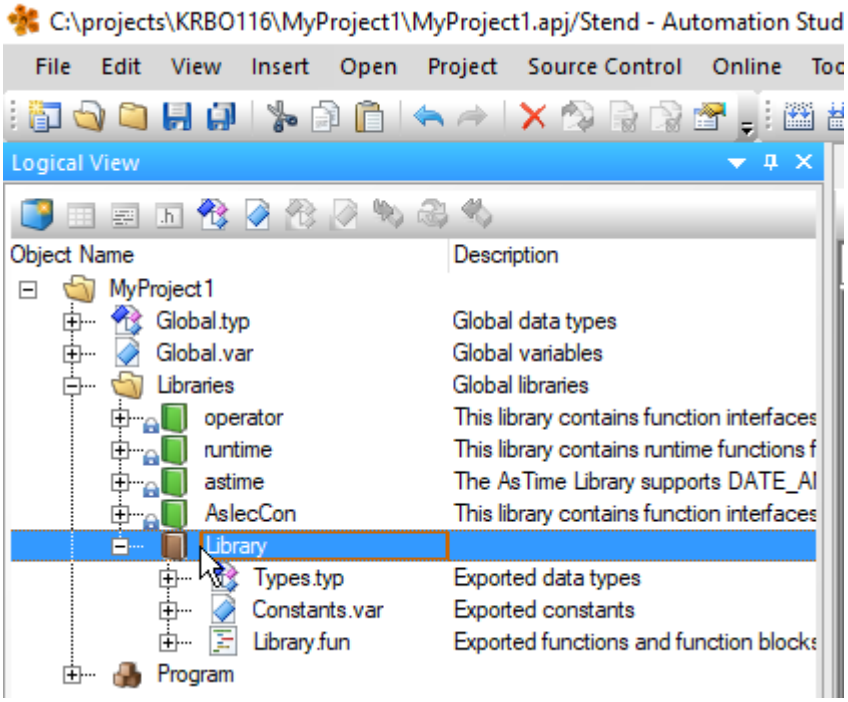

Рисунок 4.4.11. Результат создания новой библиотеки.

1.1. Создание функционального блока в библиотеке

1.1.1. Выбрав требуемую библиотеку ([1] на рис. 4. 4 .12), в панели Toolbox нажимаем на создание функции/функционального блока ([2] на рис. 4. 4 .12).

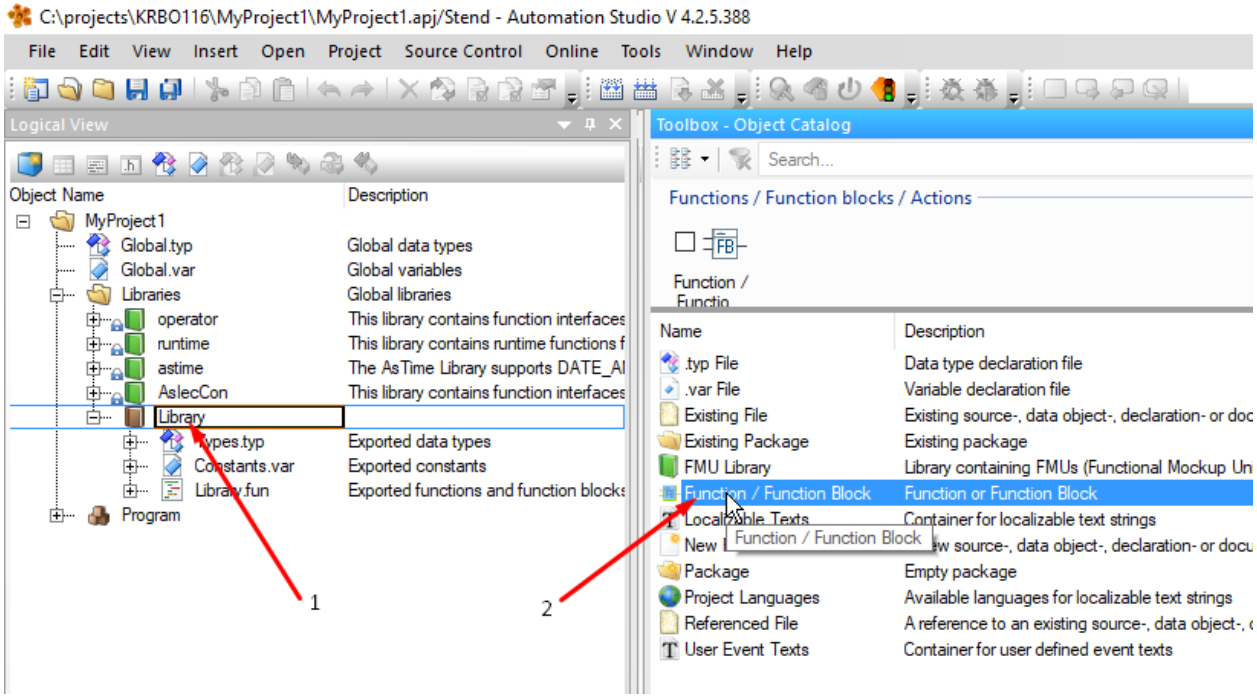

Рисунок 4.4.12. Результат создания новой библиотеки.

1.1.1. В открывшемся окне задаем имя новому функциональному блоку ([1] на рис. 4. 4 .13), выбираем, что хотим создать именно функциональный блок, а не функцию ([2] на рис. 4. 4 .13), выбираем язык ANSI C ([3] на рис. 4. 4 .13), нажимаем «далее» ([4] на рис. 4. 4 .13).

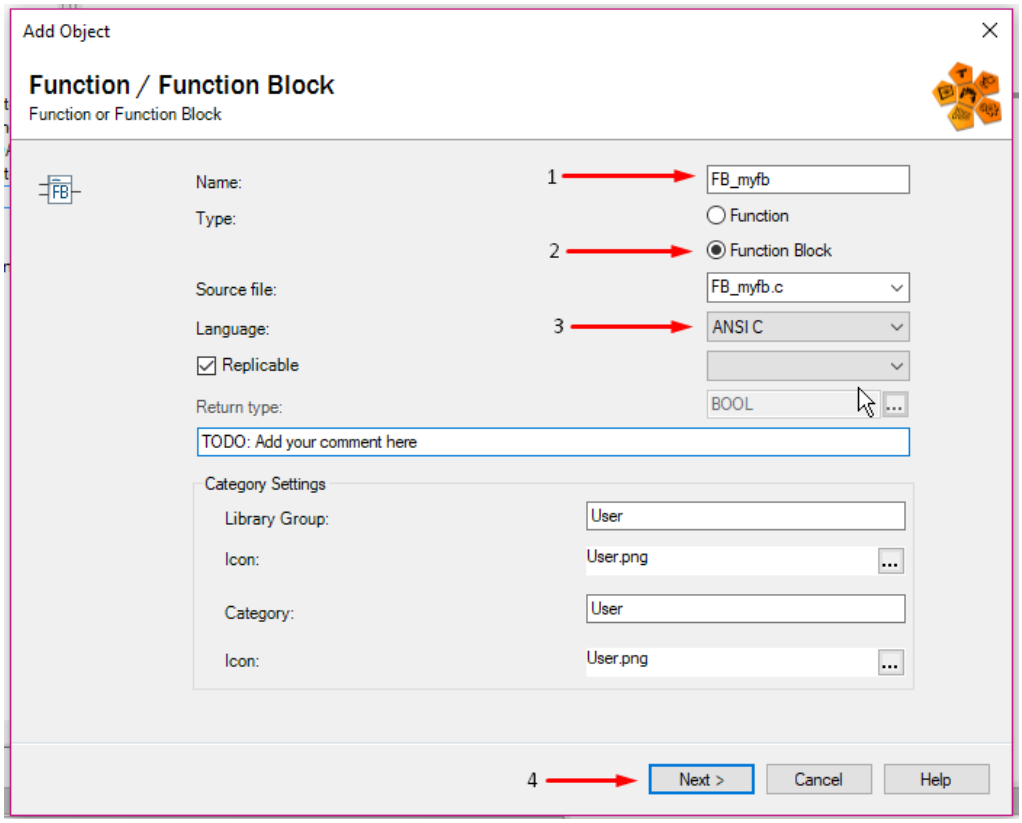

Рисунок 4.4.13. Результат создания новой библиотеки.

1.1.2. Открывается окно создания структуры функционального блока, в котором, нажимая кнопку «Add» (рис. 4. 4 .14). Выбрать назначение переменной (вход, выход или внутреннее состояние ФБ) можно в колонке Scope (рис. 4. 4 .15.а). Тип данных указывается в графе Type, простым написанием типа, либо открытием окна расширенного выбора кнопкой (рис. 4.4.15.б), которое позволяет:

- выбрать категорию типа данных ([1] на рис. 4. 4 .16);

- рассмотреть весь набор и выбрать из него требуемый тип данных ([2] на рис. 4. 4 .16);

- воспользоваться поиском ([3] на рис. 4. 4 .16).

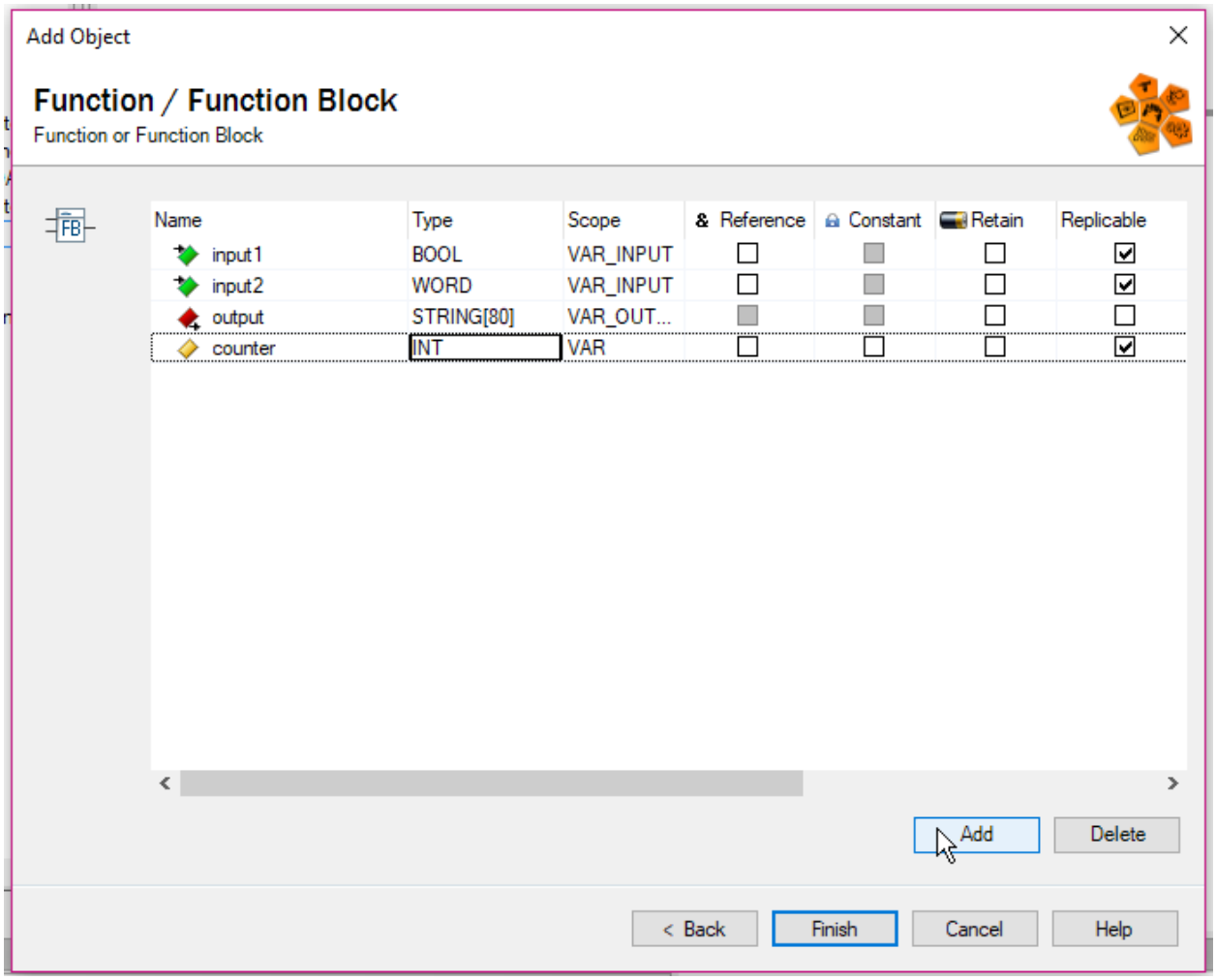

Рисунок 4.4.14. Добавление в структуру ФБ переменных.

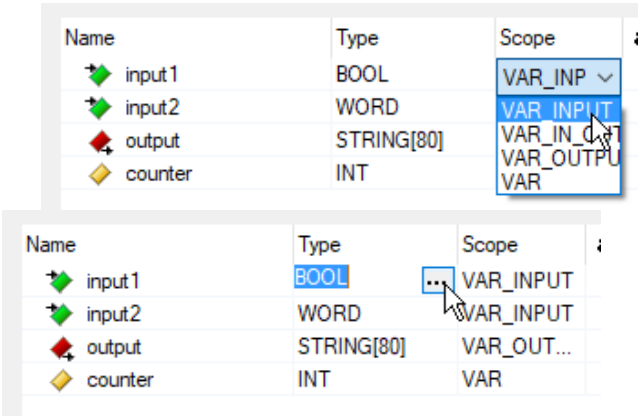

Рисунок 4.4.15 Выбор назначения переменной и выбор типа данных переменной

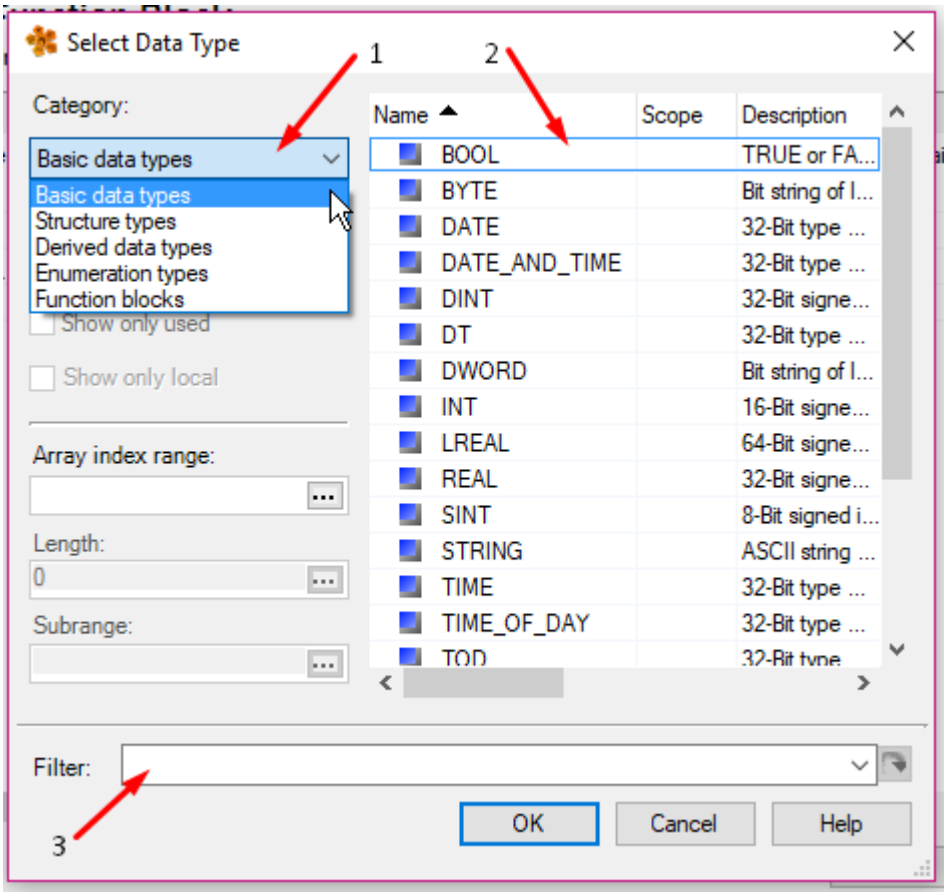

Рисунок 4.4.16. Расширенный выбор типа данных переменной.

4. Прошивка физического стенда: Offline Installer и Online Install

1.2. Прошивка путем Offline Install

1.2.1. Необходимо открыть Tools-Offline Install

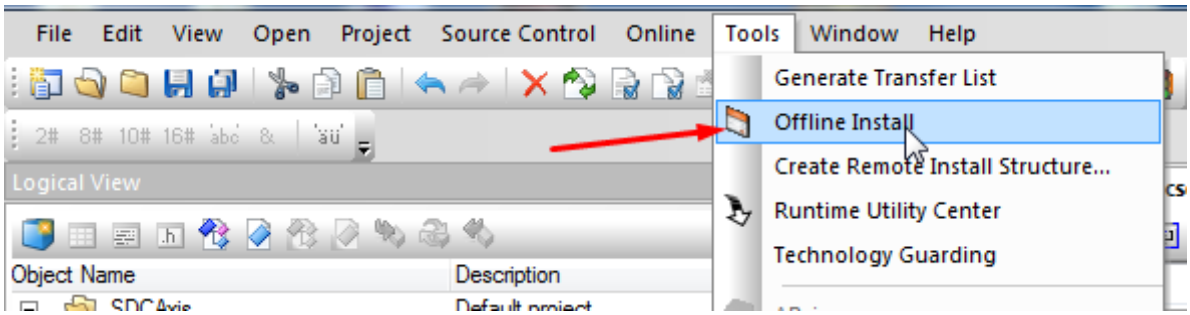

Рисунок 4.5.17 Открытие окна Offline Install.

1.2.2. Выбрав карту памяти ([1] на рис. 4. 5 .18), нажать на кнопку загрузки ([2] на рис. 4. 5 .18). Согласиться с форматированием карты (рис. 4. 5 .19) памяти и выйти из окна загрузки программы (рис. 4. 5 .20).

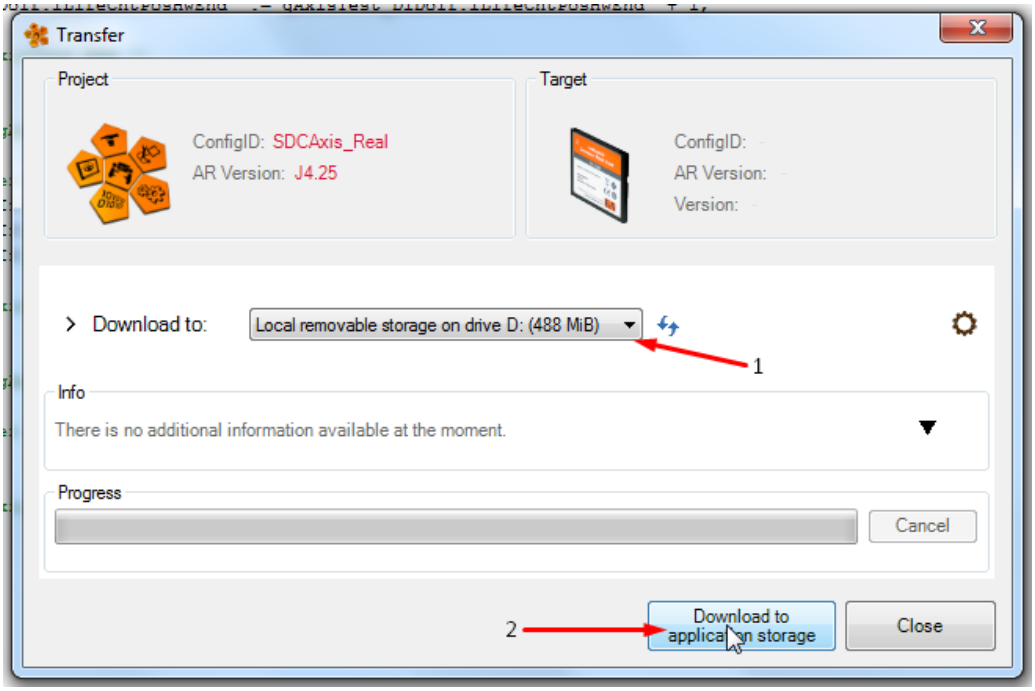

Рисунок 4.5.18 Загрузка программы на карту памяти.

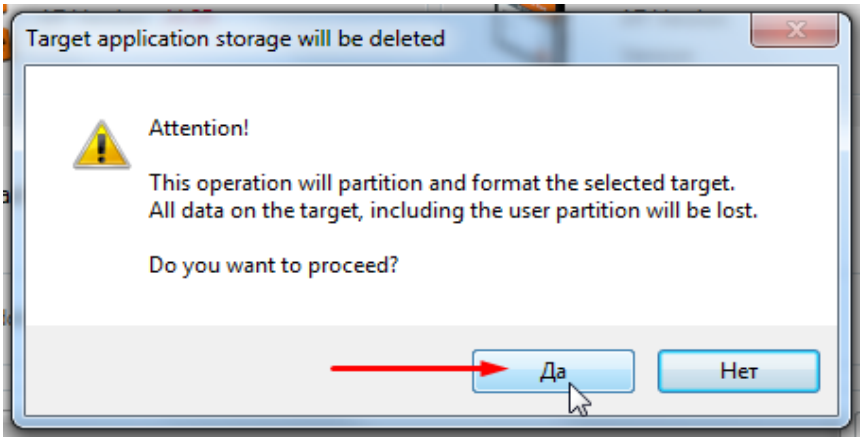

Рисунок 4.5.19. Форматирование карты памяти.

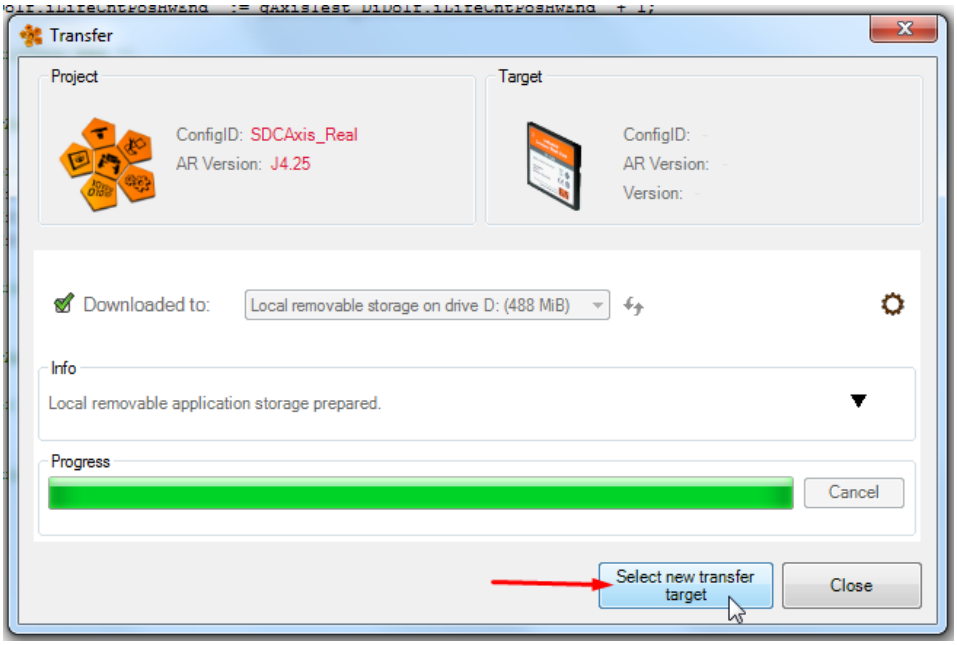

Рисунок 4.5.20. Расширенный выбор типа данных переменной.

1.3. Online Install.

1.3.1. Для установки программ по сети, изначально необходимо подключить ПЛК, открыв в меню Online-Settings…

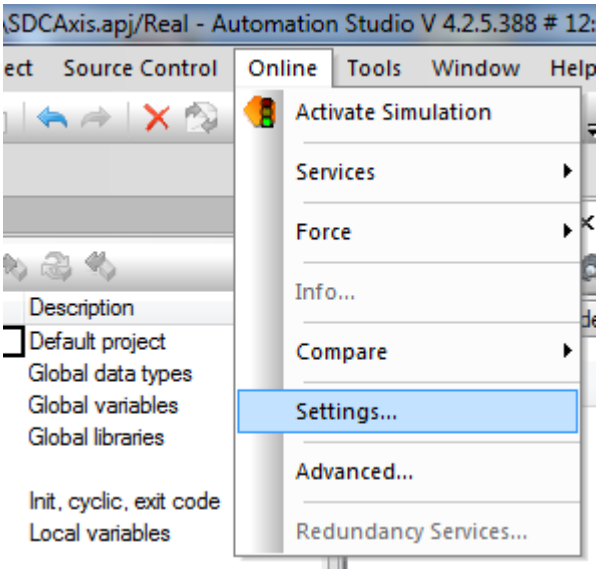

Рисунок 4.5.21. Открытие окна конфигурации сетевого подключения.

1.3.2. Далее открываем вкладку Browse (рис. 4. 5 .22).

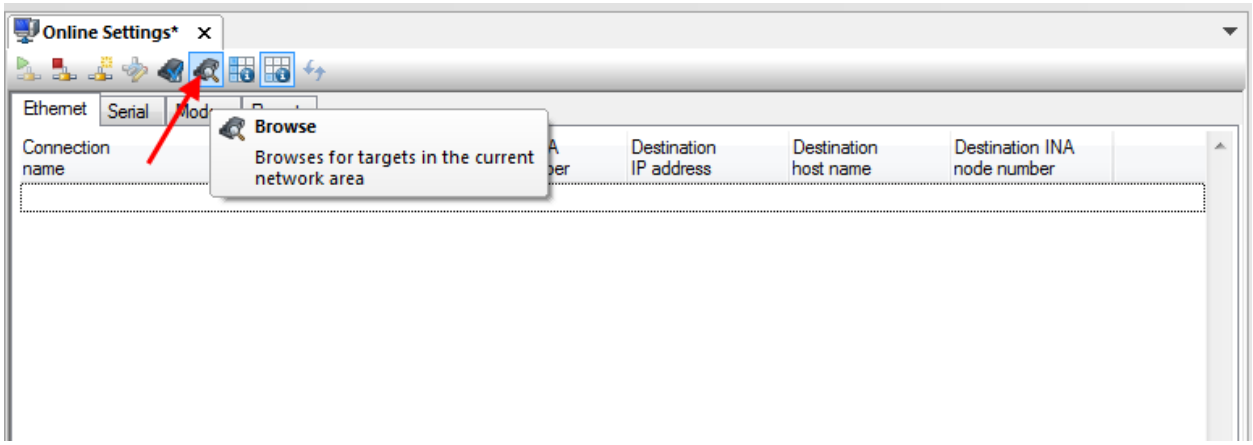

Рисунок 4.5.22. Конфигурация сетевого подключения.

1.1.1. Если IP и маска подсети горят красным цветом (рис. 4. 5 .23)., то они выставлены не правильно и следует их переконфигурировать, вызвав контекстное меню и открыв опцию Set IP Parameters (рис. 4. 5 .24).

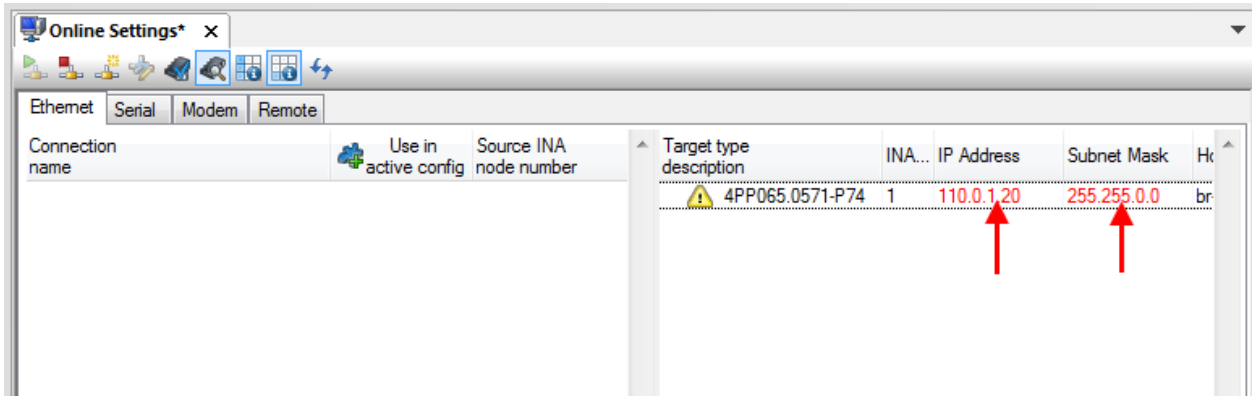

Рисунок 4.5.23. Конфигурация сетевого подключения.

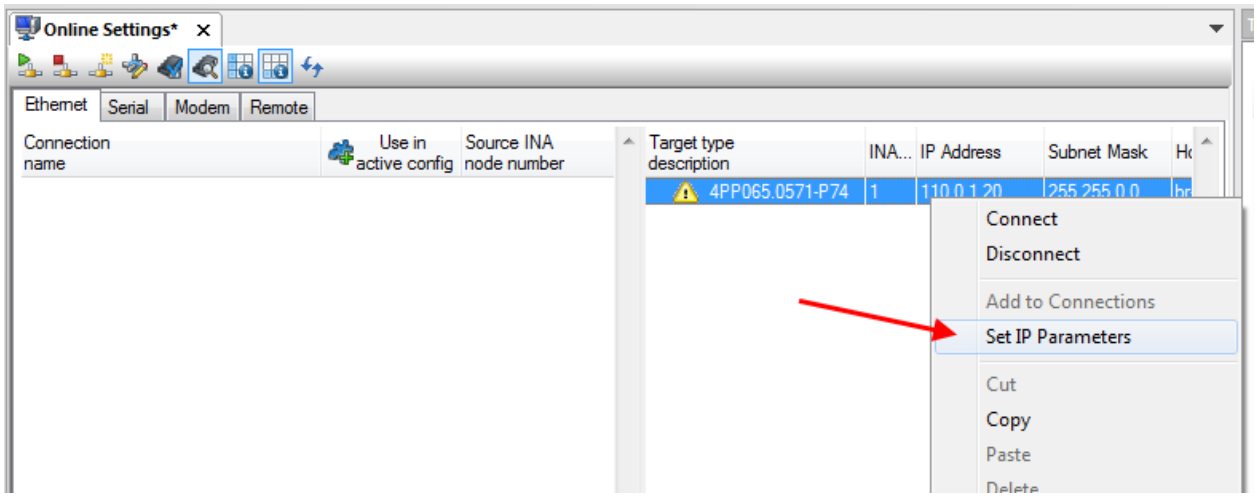

Рисунок 4.5.24. Конфигурация сетевого подключения.

1.1.1. Установить следующие IP и маску, аналогично рис. 4. 5 .25.

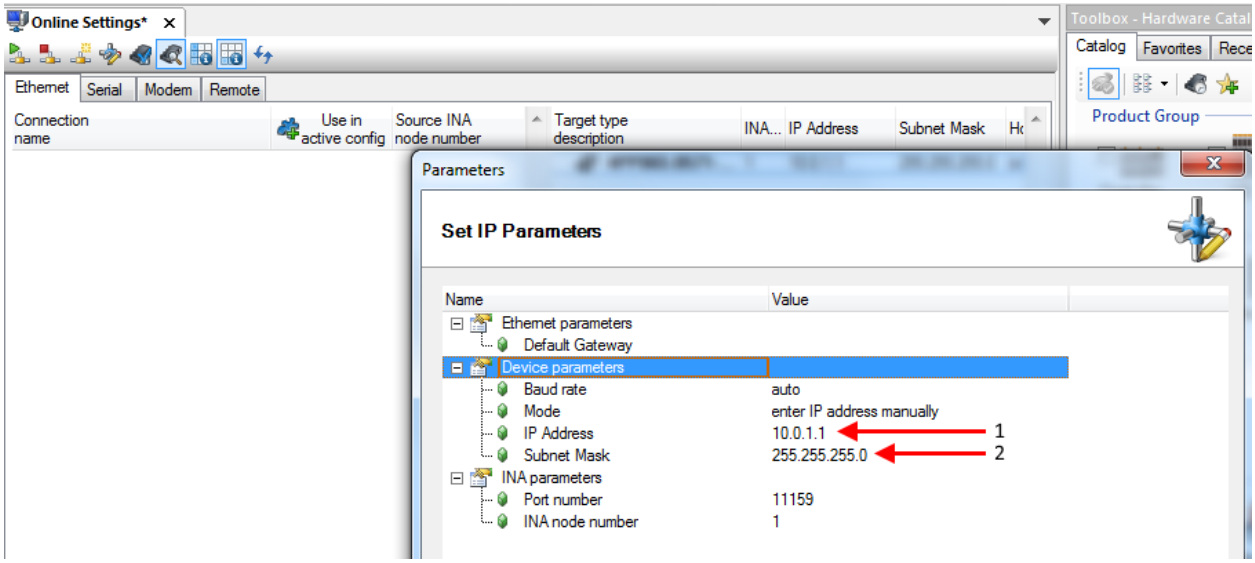

Рисунок 4.5.25. Установка IP и MAC адресов.

1.1.1. Добавить в подключения сконфигурированный ПЛК (рис. 4. 5 .26).

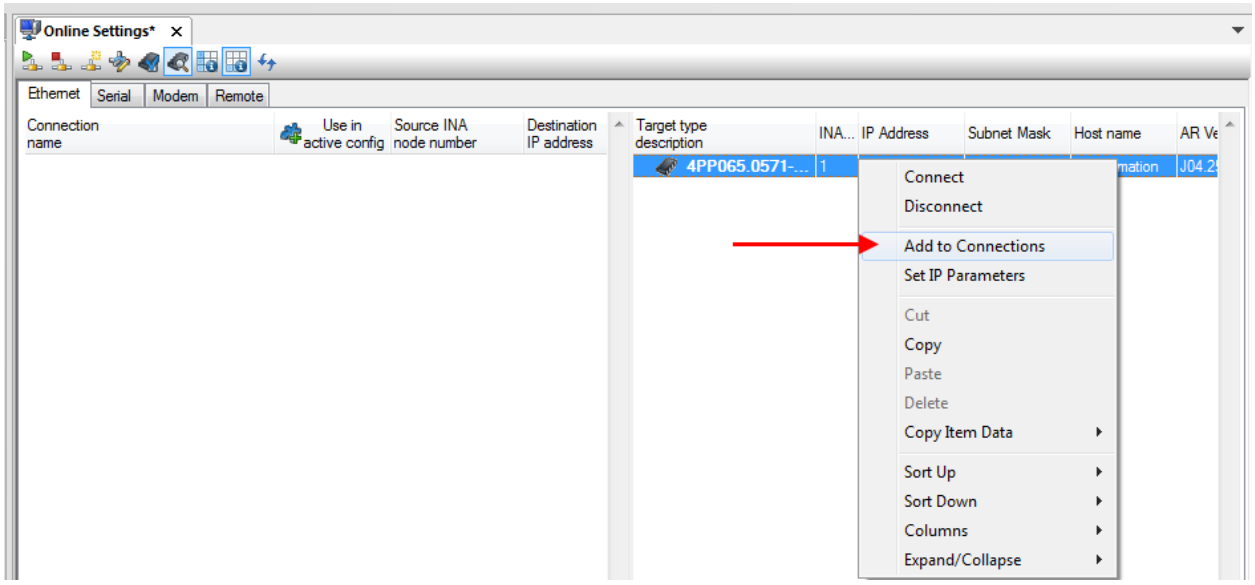

Рисунок 4.5.26. Конфигурация сетевого подключения.

1.1.1. Установив флаг «Use in active config» ([1] на рис. 4. 5 .27), вызвав контекстное меню, необходимо нажать на кнопку «Connect» ([2] на рис. 4. 5 .27).

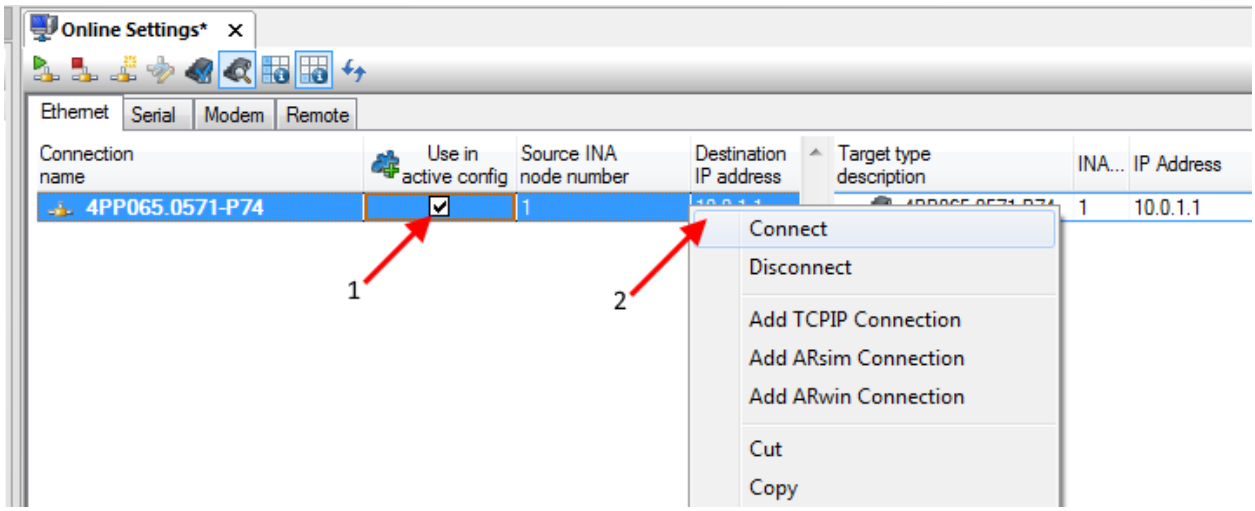

Рисунок 4.5.27. Конфигурация сетевого подключения.

1.1.1. Если все предыдущие шаги были выполнены верно, в нижнем правом углу должна появиться зеленым цветом надпись «RUN» ([1] на рис. 4. 5 .28), Теперь, для сохранения настроек, и в дальнейшем, проливать проект нужно будет лишь нажимая кнопку «Transfer» » ([2] на рис. 4. 5 .28). Так же можно перезагрузить ПЛК нажав кнопку [3] на рис. 4. 5 .28.

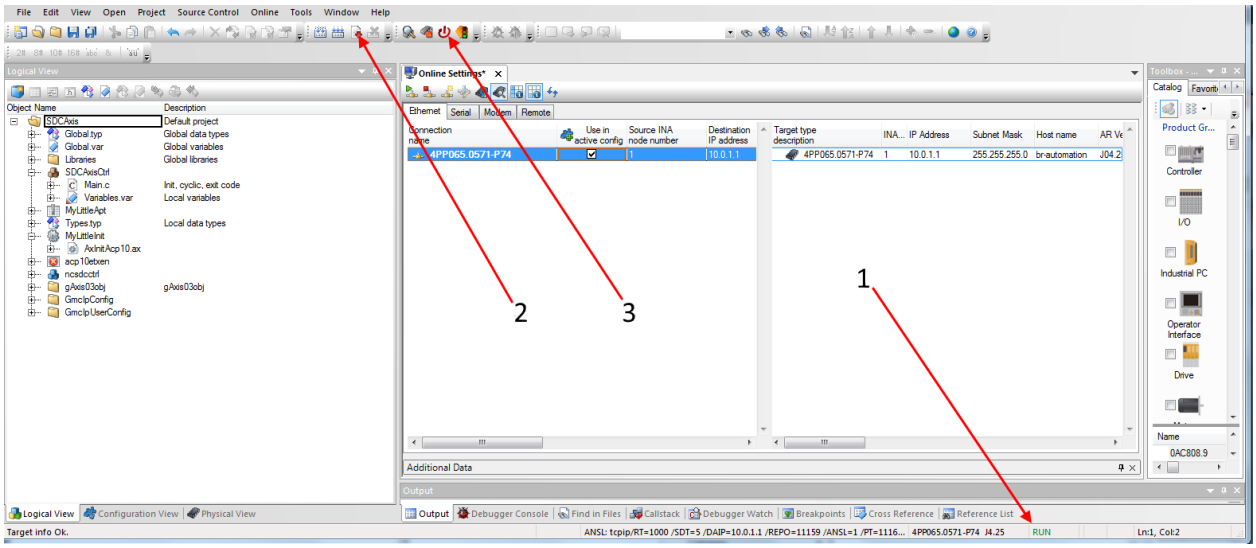

Рисунок 4.5.28. Окно AS при подключенном ПЛК.

5. Методы решения возможных проблем при выполнении работы

1.4. Ошибка версии Safety/Runtime

Для смены версии Safety/Runtime под необходимые при компиляции проекта, следует проделать следующий набор действий (рис. 4. 6 .29. и 4. 6 .30.).

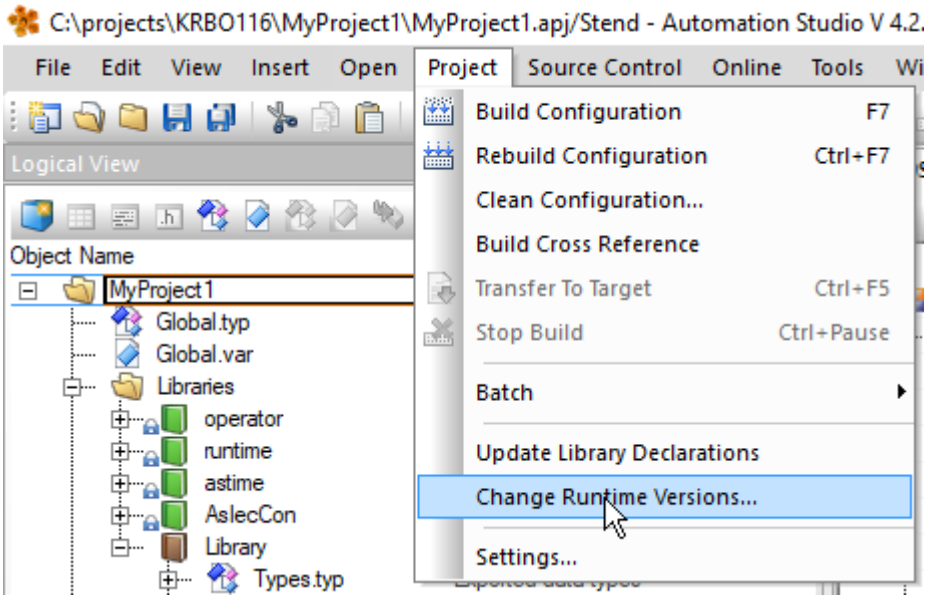

Рисунок 4.6.29. Создание нового проекта.

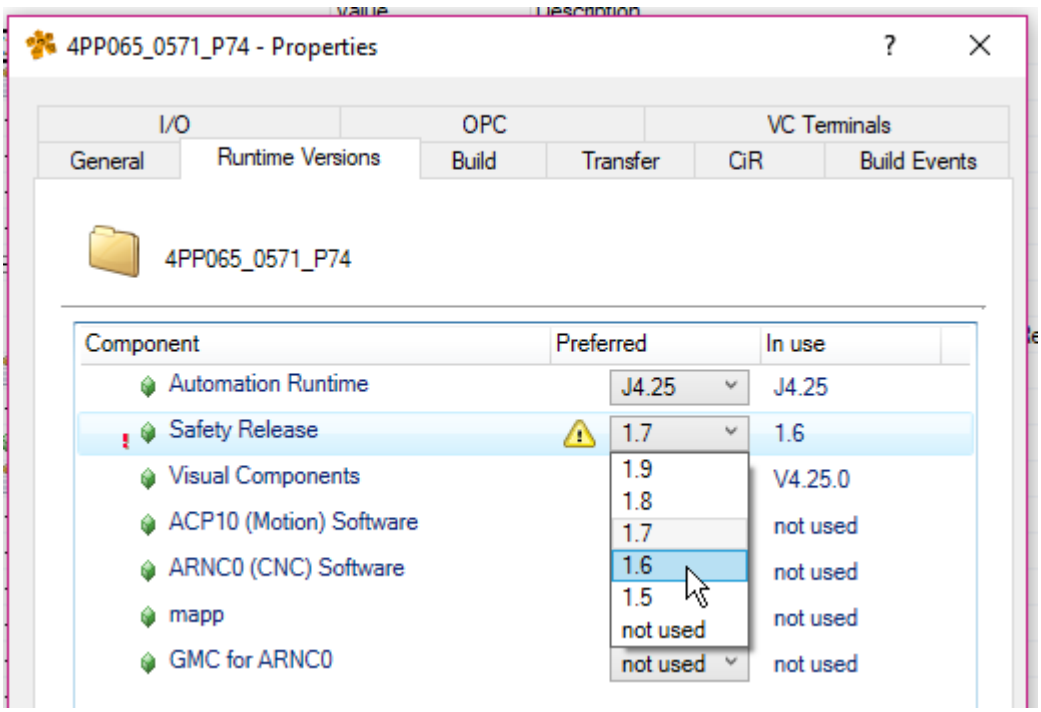

Рисунок 4.6.30. Создание нового проекта.

Список рекомендованной литературы

1. Лукинов, А.П. Проектирование мехатронных и робототехнических устройств + CD. [Электронный ресурс] : Учебные пособия — Электрон. дан. — СПб. : Лань, 2012. — 608 с. — Режим доступа: http://e.lanbook.com/book/2765 — Загл. с экрана.

2. Польский, В.А. Изучение способов управления электроприводом переменного тока на базе программируемых логических контроллеров : метод. указания по курсу «Электроприводы роботов». [Электронный ресурс] : Учебно-методические пособия / В.А. Польский, А.В. Ванин. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана, 2010. — 35 с. — Режим доступа: http://e.lanbook.com/book/52353 — Загл. с экрана.

3. Булгаков, А.Г. Промышленные роботы. Кинематика, динамика, контроль и управление. [Электронный ресурс] : Монографии / А.Г. Булгаков, В.А. Воробьев. — Электрон. дан. — М. : СОЛОН-Пресс, 2008. — 488 с. — Режим доступа: http://e.lanbook.com/book/13760 — Загл. с экрана.

4. Медведев, М.Ю. Программирование промышленных контроллеров. [Электронный ресурс] : Учебные пособия / М.Ю. Медведев, В.Х. Пшихопов. — Электрон. дан. — СПб. : Лань, 2011. — 288 с. — Режим доступа: http://e.lanbook.com/book/686 — Загл. с экрана.

5. Предко, М. Устройства управления роботами: схемотехника и программирование / М. Предко .— М. : ДМК-Пресс, 2010 .— (В помощь радиолюбителю) .— ISBN 5-94074-226-2 .— ISBN 978-5-94074-226-2 (http://rucont.ru/efd/203185)

6. Федоров, Ю.Н. Справочник инженера по АСУТП: проектирование и разработка. Электронная версия. [Электронный ресурс] : Справочники — Электрон. дан. — Вологда : "Инфра-Инженерия", 2015. — 928 с. — Режим доступа: http://e.lanbook.com/book/65111 — Загл. с экрана.