

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА - Российский технологический университет»**

**РТУ МИРЭА**

Институт искусственного интеллекта

Кафедра проблем управления

**Лабораторная работа №3**

по дисциплине «Программное обеспечение мехатронных и робототехнических систем»

|  |  |
| --- | --- |
| **Выполнили студенты группы**  КРБО-01-22 | Бадалян А.А.  Куприянова А.К.  Цепкин Ю.Ю. |
| **Преподаватель:** | ст. преподаватель Морозов А.А. |
|  |  |
|  |  |
| Работа представлена к защите: | «\_\_\_» \_\_\_\_\_\_\_\_\_ 2025 г. |

Москва 2025

**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 1.1). 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

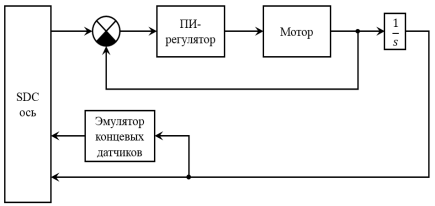


Рис. 1.1. Структура системы управления.

**Порядок выполнения**

1. Создание проекта: при создании проекта необходимо руководствоваться наличием лабораторного стенда, на котором можно проводить тест.

2. Добавление в проект библиотеки и конфигурационные таблицы для создания оси. Для создания SDC-оси необходимо добавить библиотеки Acp10, а также создать в конфигурации NC-мэпинг.

3. Мэпинг новой SDC оси.

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

5. Провести симуляцию, подтверждающую работоспособность программного обеспечения.

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

7. Проведение экспериментов, подтверждающих работоспособность системы управления на реальной модели УРТК.

**ХОД РАБОТЫ**

**1**. **Создание проекта в среде Automation Studio.**

При создании проекта необходимо руководствоваться наличием лабораторного стенда, на котором можно проводить тест. Если тестирование будет проводиться без него, рекомендуется при создании проекта.

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

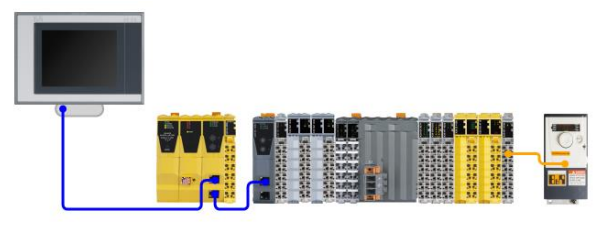


Рисунок 1.2 Конфигурация оборудования

**2. Добавление 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:

-Acp10sdc;

-Acp10man;

-Acp10par;

-Acp10\_MC;

-Acp10sim.

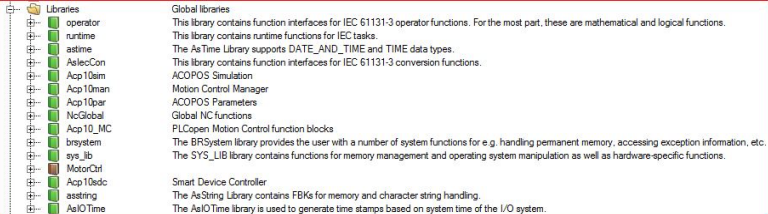


Рисунок 2. Библиотеки проекта

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. В итоге, набор глобальных переменных должен иметь следующий вид.

Таблица 1. Глобальные переменные взаимодействия SDC-оси.

|  |  |  |
| --- | --- | --- |
| Параметр | Тип данных | 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 |

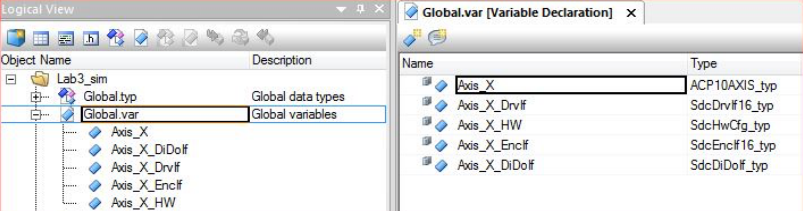


Рисунок 3. Глобальные переменные.

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



Рисунок 4 – Модуль инициализации оси

Далее открываем его и в параметрах инициализации указываем реальные параметры оси УРТК.

Таблица 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 |  |
| trigg\_dir | ncNEGATIVE | Режим подъезда к датчику |
| fix\_dir | ncOFF |  |

Далее добавляем таблицу инициализации оси «ACOPOS Parameter table». В таблице указываем необходимые параметры.

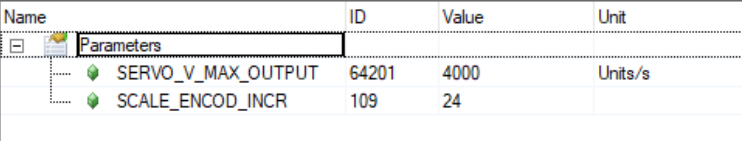


Рисунок 5. Параметры оси

**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».

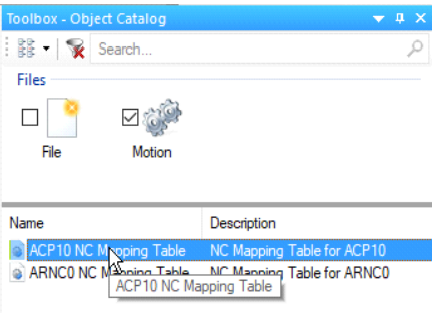


Рисунок 6. Создание новой таблицы мэппинга осей.

В созданную таблицу добавляем SDC ось, открываем новую таблицу, вызываем контекстное меню правым щелчком мыши, создаем новую ось.

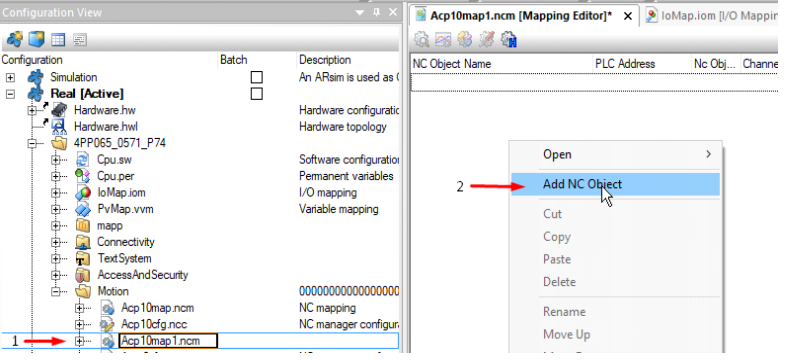


Рисунок 7. Добавление оси

Далее необходимо заполнить параметры новой оси.

Таблица 3. 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 |

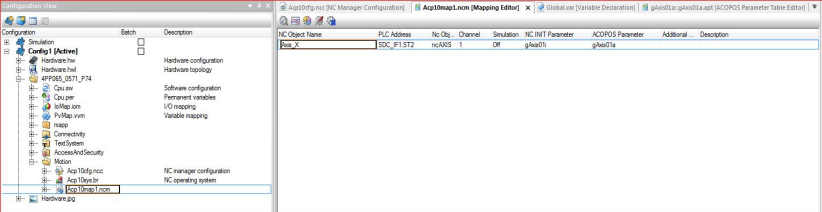


Рисунок 8. Параметры новой оси.

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.

Таблица 4. Параметры функционального блока FB\_Axis.

|  |  |  |  |
| --- | --- | --- | --- |
| Конфигурация | Имя | Тип данных | Описание |
| вход | reset\_error | BOOL | Сброс ошибок мотора |
| вход | endswitch\_a\_rea ched | BOOL | Состояние начального концевого датчика |
| вход | endswitch\_b\_rea ched | 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.

Таблица 5. 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 |

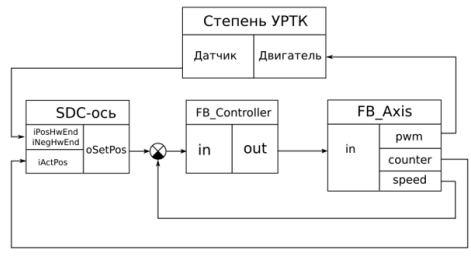


Рисунок 9. Структура программного обеспечения.

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.

**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.

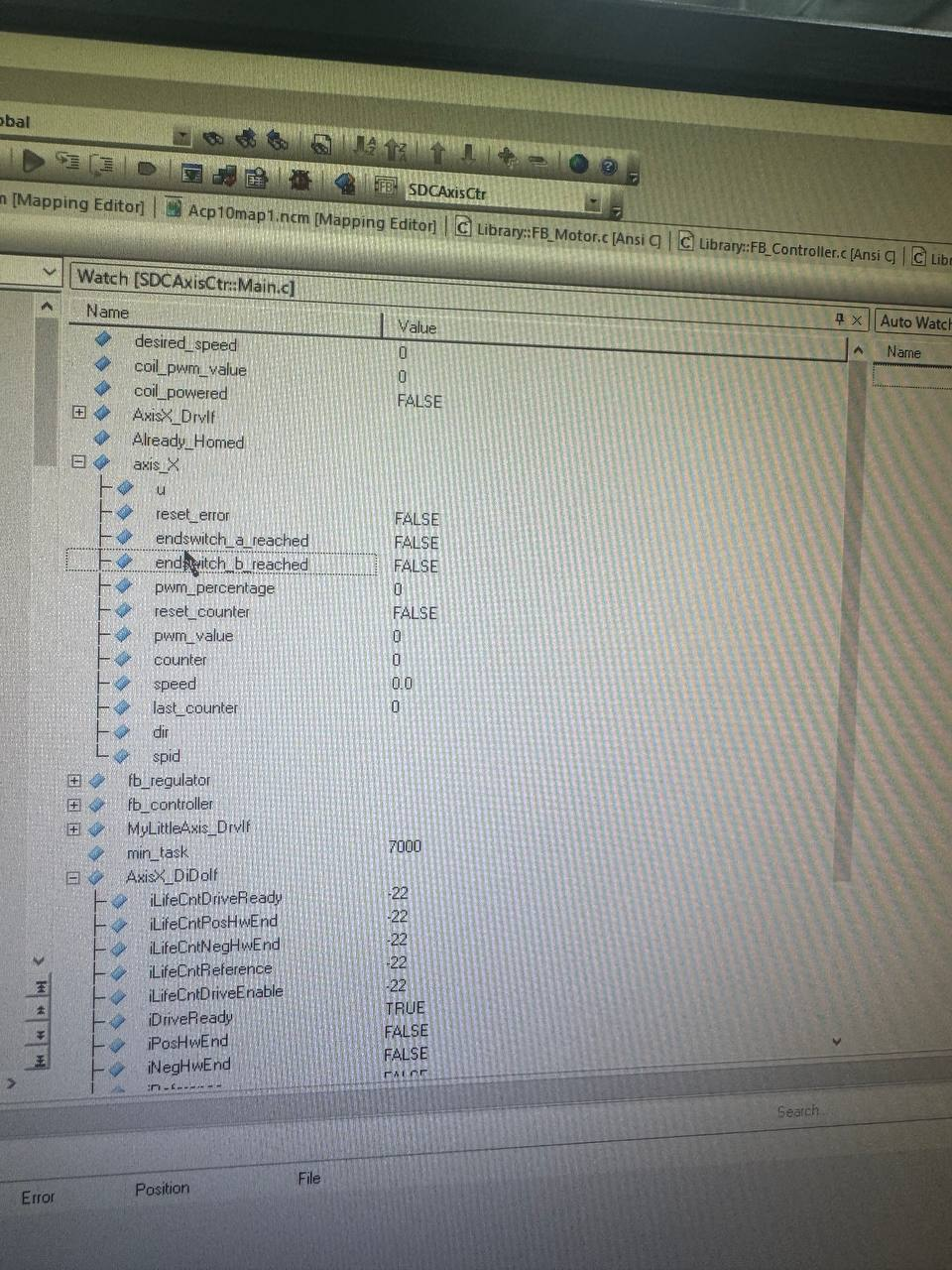


Рисунок 10. Изменение значений в режиме мониторинга.

Таблица 6. 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 «Acp10map.ncm». В открывшемся окне нажимаем на Axis\_X и выбираем «Test» → «Parallel Mode». Далее выполняем инициализацию параметров и включаем контроллер.

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».

****

Рисунок 11. Окно «Watch» до и после реферирования оси.

Снять графики можно средством «NC Trace».

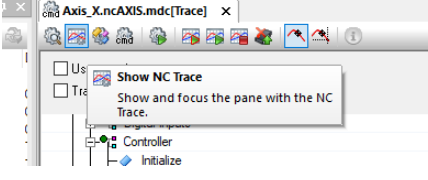


Рисунок 12. NC Trace.

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

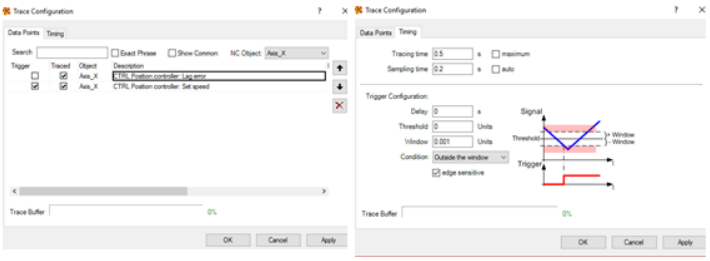


Рисунок 13. Trace Configuration.

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

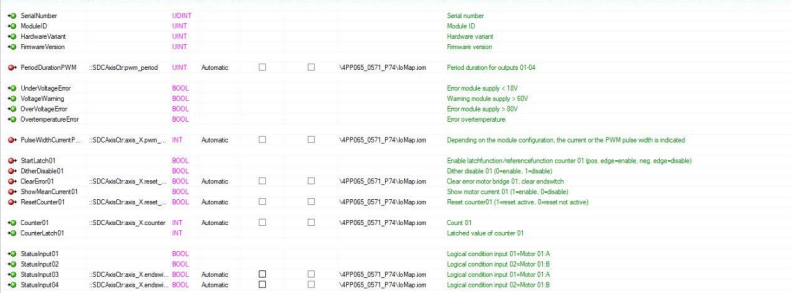


Рисунок 14. Параметры оси

Таблица 7. 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»;

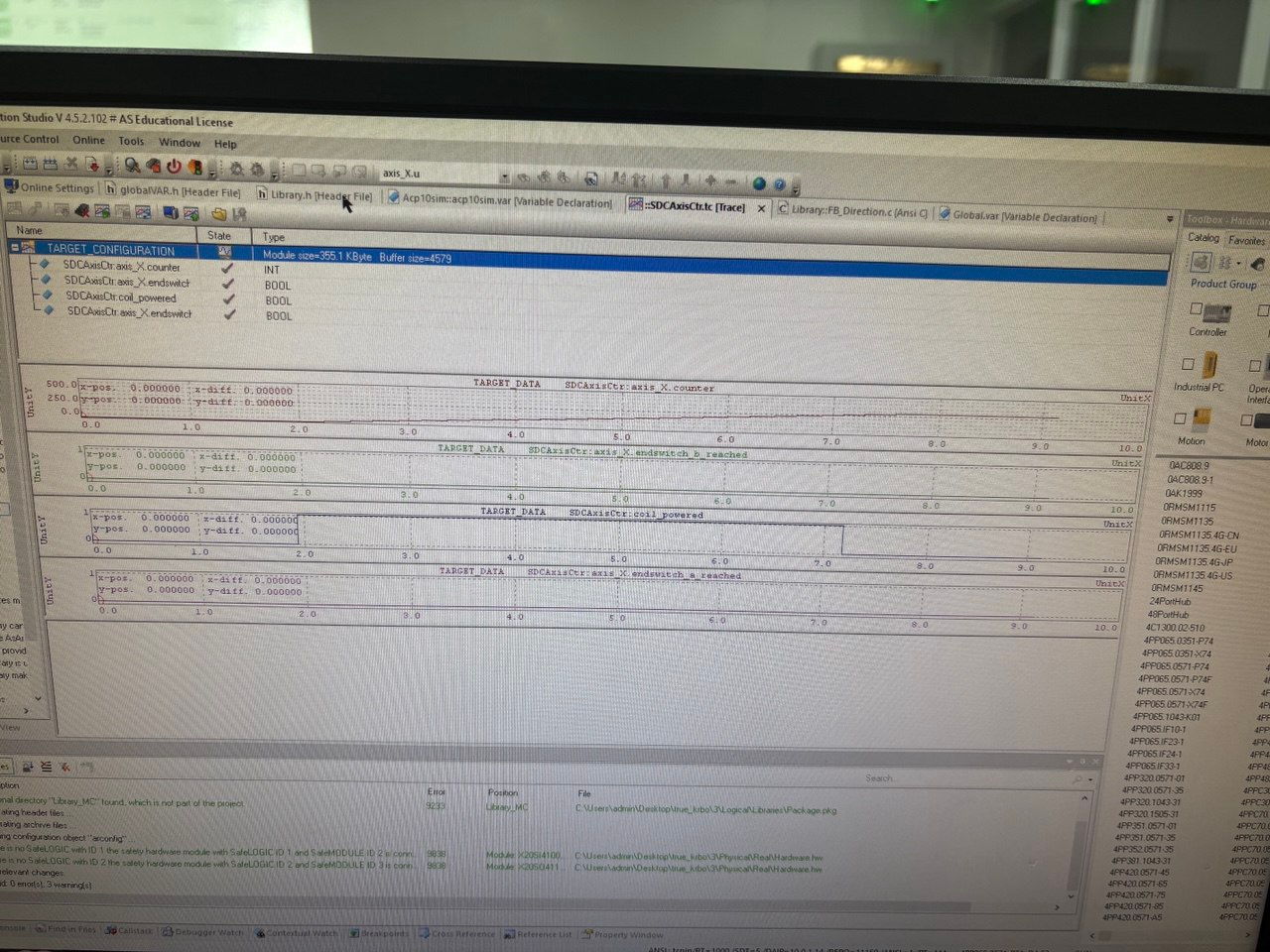


Рисунок 15. Результат экспериментов.

**ВЫВОД**

В ходе лабораторной работы получили навыки создания программного обеспечения систем управления одной степенью подвижности учебного робота.

**Приложение А**

Листинг кода функционального блока FB\_power

#include

#ifdef \_\_cplusplus

extern "C"

{

#endif

#include "Library.h"

#ifdef \_\_cplusplus

};

#endif

/\* TODO: Add your comment here \*/

void FB\_power(struct FB\_power\* inst) // Функциональный блок запуска и референсирования осей

{

switch (inst->State) // Машина состояний для ФБ

{

case Idle: // Состояние покоя

{

if(inst->enable)

{

inst->done = 0;

inst->State = Power; // Переход к состоянию запуска } break; }

case Power: // Состояние запуска

{

inst->Power.Axis = inst->Axis;

inst->Power.Enable = 1;

inst->State = Home; // Переход к состоянию референсирования

break;

}

case Home: // Состояние референсирования осей

{

inst->Home.Axis = inst->Axis;

inst->Home.Execute = 1;

inst->Home.HomingMode = 1;

inst->Home.Position = 0; if (inst->Home.Done)

{

inst->State = Done; // Переход к состоянию завершения

}

break;

}

case Done: // Состояние завершения работы

{

inst->done = 1;

if(inst->done)

{

inst->Home.Done = 0;

inst->enable = 0;

inst->State = Idle; // Переход в состояние покоя после завершения

}

break;

}

}

//Инициализация использованных функциональных блоков MC\_Power(&inst->Power);

MC\_Home(&inst->Home);

**Приложение Б**

Листинг кода функционального блока FB\_move

#include <bur/plctypes.h>

#ifdef \_\_cplusplus

extern "C"

{

#endif

#include "Library.h"

#ifdef \_\_cplusplus

};

#endif

/\* TODO: Add your comment here \*/

void FB\_move(struct FB\_move\* inst) // Функциональный блок движения оси

{

switch (inst->State) // Машина состояний

{

case Idle\_move: // Состояние покоя

{

if(inst->enable)

{

inst->Move.Done = 0;

inst->done = 0;

inst->State = Move; // Переход к состоянию движения

}

break;

}

case Move: // Состояние движение

{

// Передача необходимых параметров в

функциональный блок

inst->Move.Axis = inst->Axis;

inst->Move.Execute = 1;

inst->Move.Distance = inst->length;

inst->Move.Velocity = 3000;

inst->Move.Acceleration = 6000;

inst->Move.Deceleration = 6000;

if(inst->Move.Done)

{

inst->State = Done\_move; // Переход к состоянию

завершения

}

break;

}

case Done\_move: // Состояние завершения

{

inst->done = 1;

if(inst->done)

{

inst->Move.Execute = 0;

inst->length = 0;

inst->enable = 0;

inst->State = Idle\_move;

}

break;

}

}

MC\_MoveAdditive(&inst->Move); // Инициализация Функционального блока движения

}

**Приложение В**

Листинг кода функционального блока FB\_Trajectory

#include <bur/plctypes.h>

#ifdef \_\_cplusplus

extern "C"

{

#endif

#include "Library.h"

#ifdef \_\_cplusplus

};

#endif

/\* TODO: Add your comment here \*/

void FB\_Trajectory(struct FB\_Trajectory\* inst) // Функциональный блок

управления УРТК

{

switch (inst->State) // Машина состояний

{

case Idle\_Trajectory: // Состояние покоя

{

if (inst->enable)

{

inst->done = 0;

inst->State = PowerY; // Переход к запуску оси Y

}

break;

case PowerY : // Запуск оси Y

{

inst->Power\_AxisY.Axis = inst->AxisY; // Объявление

оси Y

inst->Power\_AxisY.enable = 1;

if(inst->Power\_AxisY.done)

{

inst->State = PowerZ; // Переход к запуску оси Z

}

break;

}

case PowerZ :

{

inst->Power\_AxisZ.Axis = inst->AxisZ; // Объявление

оси Z

inst->Power\_AxisZ.enable = 1;

if(inst->Power\_AxisZ.done)

{

inst->State = side1; // Переход к состоянию

движения по траектории 1-й стороны

}

break;

}

case side1 : // Состояние движения по траектории 1-й стороны

{

inst->Move\_Axis.Axis = inst->AxisY;

inst->Move\_Axis.length = inst->length\*100;

inst->Move\_Axis.enable = 1;

if(inst->Move\_Axis.done)

{

inst->State = side2; // Переход к состоянию

движения по траектории 2-й стороны

}

break;

}

case side2 : // Состояние движения по траектории 2-й стороны

{

inst->Move\_Axis.Axis = inst->AxisZ;

inst->Move\_Axis.length = inst->length\*100;

inst->Move\_Axis.enable = 1;

if(inst->Move\_Axis.done)

{

inst->State = side3; // Переход к состоянию

движения по траектории 3-й стороны

}

break;

}

case side3 : // Состояние движения по траектории 3-й стороны

{

inst->Move\_Axis.Axis = inst->AxisY;

inst->Move\_Axis.length = -inst->length\*100;

inst->Move\_Axis.enable = 1;

if(inst->Move\_Axis.done)

{

inst->State = side4; // Переход к состоянию

движения по траектории 4-й стороны

}

break;

}

case side4 : // Состояние движения по траектории 4-й стороны

{

inst->Move\_Axis.Axis = inst->AxisZ;

inst->Move\_Axis.length = -inst->length\*100;

inst->Move\_Axis.enable = 1;

if(inst->Move\_Axis.done)

{

inst->State = Stop\_Trajectory;

}

break;

}

case Stop\_Trajectory : // Состояние остановки и выключения УРТК

{

inst->Power\_AxisY.Power.Enable = 0;

inst->Power\_AxisZ.Power.Enable = 0;

if (!inst->Power\_AxisY.Power.Status && !inst-

>Power\_AxisZ.Power.Status)

{

inst->State = Done\_Trajectory;

}

}

case Done\_Trajectory : // Состояние завершения работы ФБ

{

inst->done = 1;

if(inst->done)

{

inst->Move\_Axis.done = 0;

inst->Move\_Axis.done = 0;

inst->enable = 0;

inst->State = Idle\_Trajectory;

}

break;

}

}

// Инициализация использованных функциональных блоков

FB\_move(&inst->Move\_Axis);

FB\_power(&inst->Power\_AxisY);

FB\_power(&inst->Power\_AxisZ);

**Приложение Г**

Листинг кода основной программы User\_Interface

#include <bur/plctypes.h>

#ifdef \_DEFAULT\_INCLUDES

#include <AsDefault.h>

#endif

void \_INIT ProgramInit(void)

{

// Объявление осей Y и Z по адресу

Trajectory.AxisY = &AxisY;

Trajectory.AxisZ = &AxisZ;

// Мгновенное присваивание значения 0 при инициализации переменной

Trajectory.length = 0;

Move\_axis.length = 0;

}

void \_CYCLIC ProgramCyclic(void)

{

FB\_Trajectory(&Trajectory); // Инициализация функционального блока

}

void \_EXIT ProgramExit(void)

{

}