МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий
Кафедра вычислительной техники

ЛАБОРАТОРНАЯ РАБОТА № 3

по дисциплине «Программное обеспечение мехатронных и робототехнических систем»

«Программное обеспечение системы управления одной степенью робота УРТК»

Студент группы КРБО-03-17. Зеленский Д.П.

Руководитель: Морозов А.А._____

Москва

2020

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. Рис. 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.

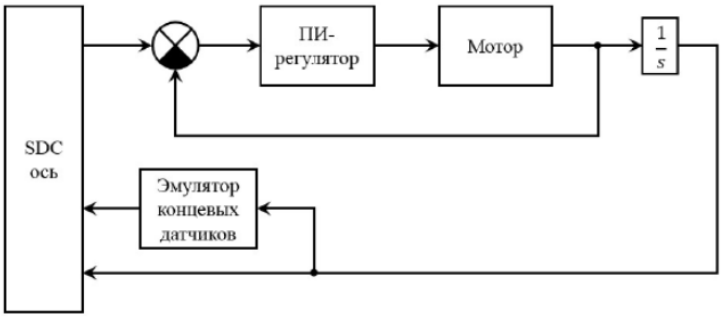

Рис.1 Структура системы управления одной степенью подвижности робота

Основная часть

- Была создана симуляцию проекта на ПК, используя модули и интерфейсы, предложенные в методических указаниях.

  - 5LS182.6-1;
  - X20BC0083;
  - X20MM4456
  - X20DI9371;
  - X20D09322;
  - 80VD100PS.C02X-01.

- Были добавлены в проект необходимые библиотеки (Рис.2) и также добавлены конфигурационные таблицы для создания оси. Для создания SDC-оси добавлены библиотеки Acp10, а также создан в конфигурации

NC-мэпинг.

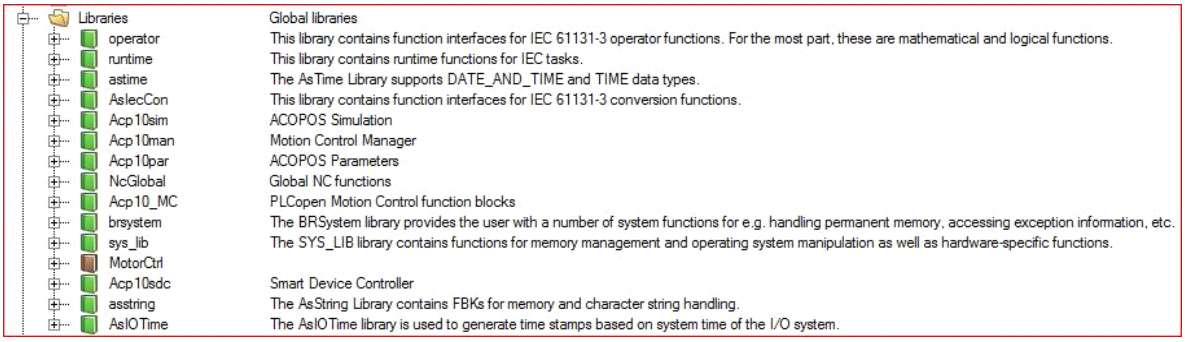

Рис. 2 Библиотеки

В Global variables автоматически были созданы переменные и структуры, необходимые для работы SDC. Для создаваемой в рамках данной лабораторной работы оси так же необходимы аналогичные структуры. Для этого они были скопированы и переименованы только название оси, заменив его на Axis_X.

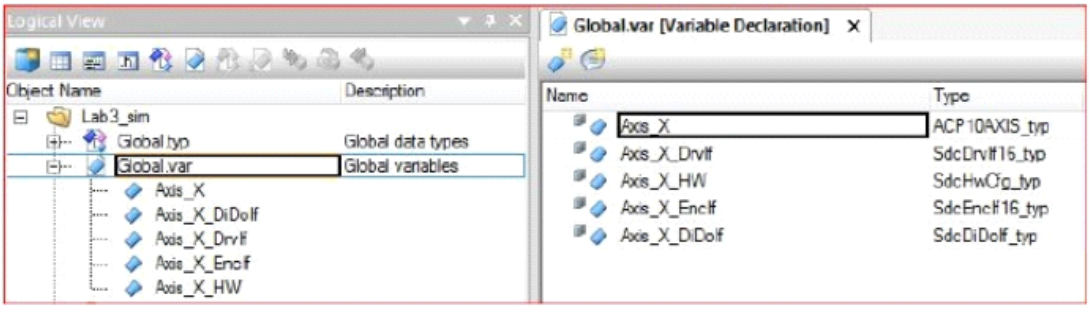

Рис. 3 Глобальные переменные

Добавлен в папку проекта, средством Toolbox, модуль инициализации оси «ACP10 Axis».

Затем он был открыт и в параметрах инициализации были указаны реальные параметры оси УРТК (Таблица 1 и Рис. 4)

Таблица 1. 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	

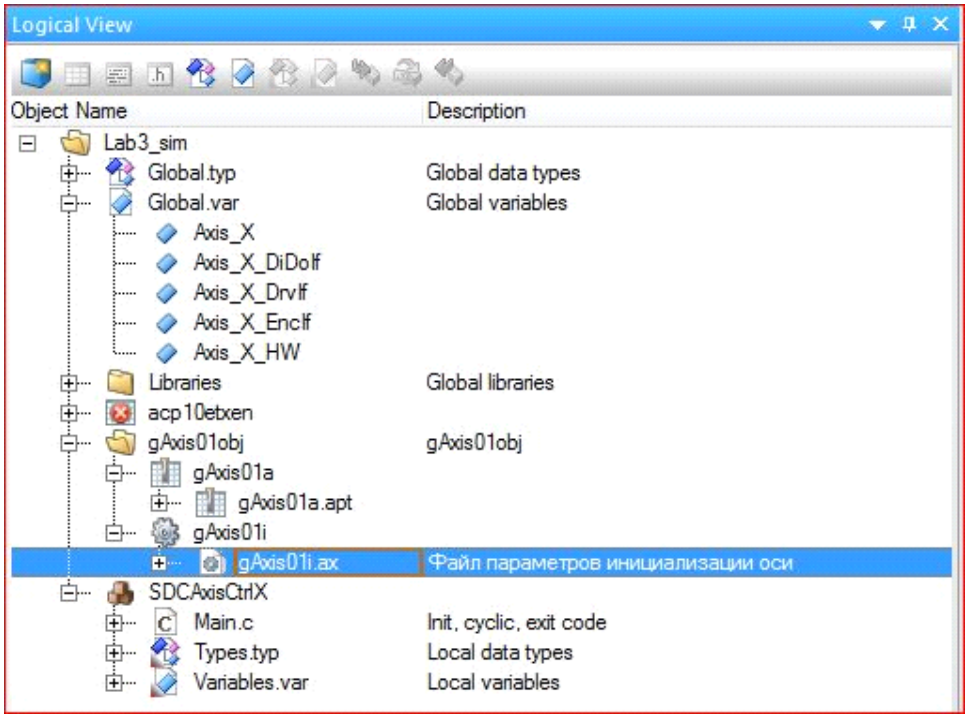

Рис 4. Файл параметров инициализации

Далее была добавлена таблица инициализации оси «ACOPOS Parameter table».

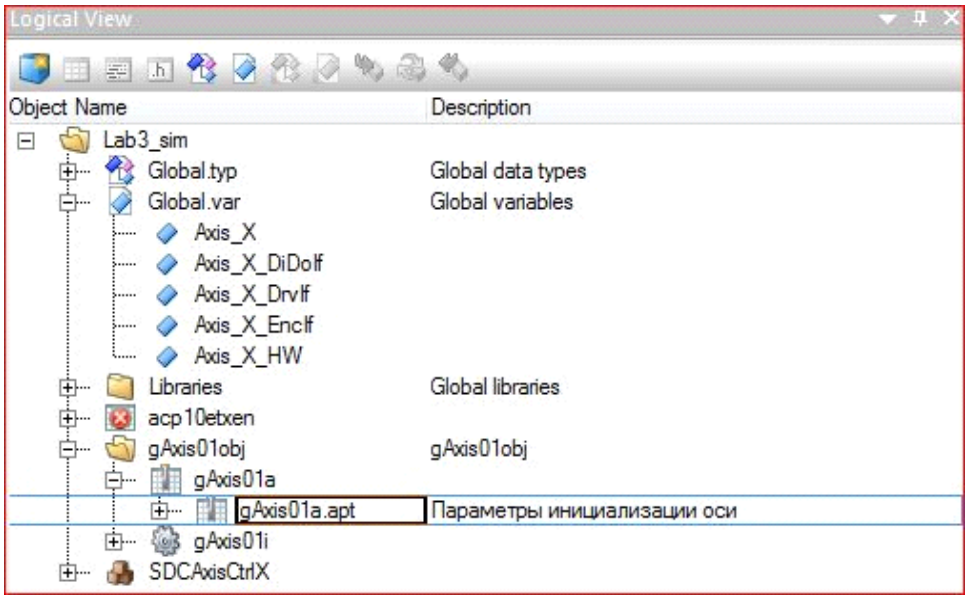

Рис. 5 Параметры инициализации оси

- Привязка новой SDC оси.

Во вкладке «Configuration View» была открыта конфигурацию «Real» и в каталоге «4PP065_0571_P74\Motion» средствами Toolbox – Object Catalog, был выбран фильтр «Motion», добавленный в папку Motion файл «ACP10 NC Mapping Table» (Рис. 6).

В созданную таблицу добавили SDC ось.

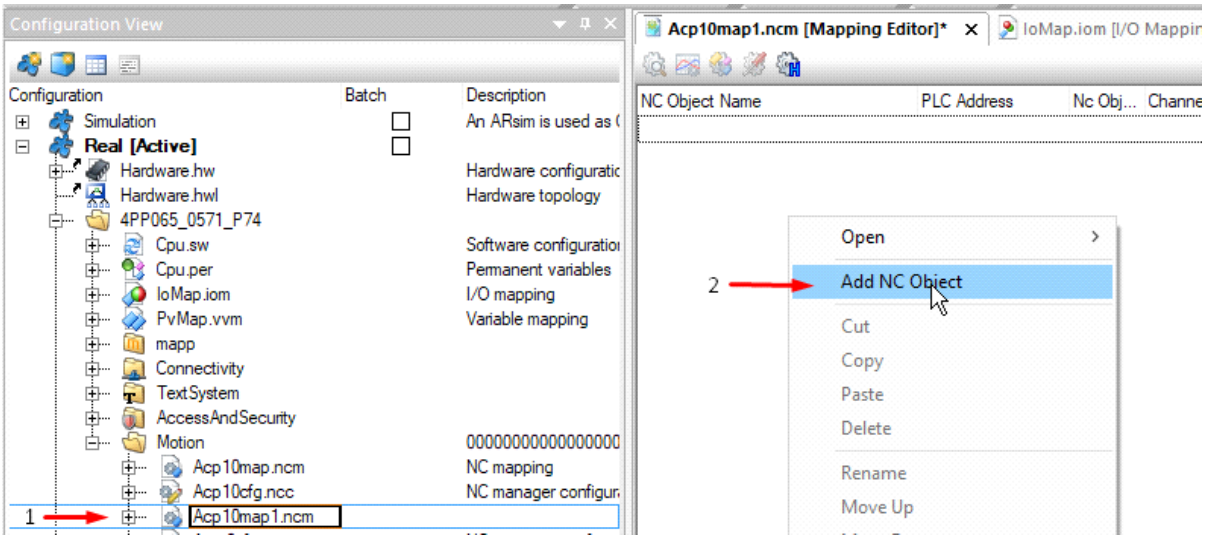

Рис. 6 Добавление оси

Были заполнены параметры новой оси.

Таблица 2. Параметры мэпинга создаваемой SDC-оси

Имя NC объекта	PLC адрес	Тип NC	Таблица ини-	Таблица параметров

(оси)		объекта	циализации	ACOPOS
Axis_X	SDC_IF1.ST2	ncAXIS	gAxis01i	gAxis01a

Скопировав в данный проект библиотеку «MotorControl» из лабораторной работы №1, добавили в нее функциональный блок «FB_Axis». Данный функциональный блок отвечает за определение входного воздействия на ось двигателя путем задания ШИМ сигнала.

- Создание программы обработки одной оси с применением библиотеки управления двигателем.

Для реализации этого был создан ANSI C Program с названием «SDCAxisCtrlX» и добавлены в нее переменные из методических указаний.

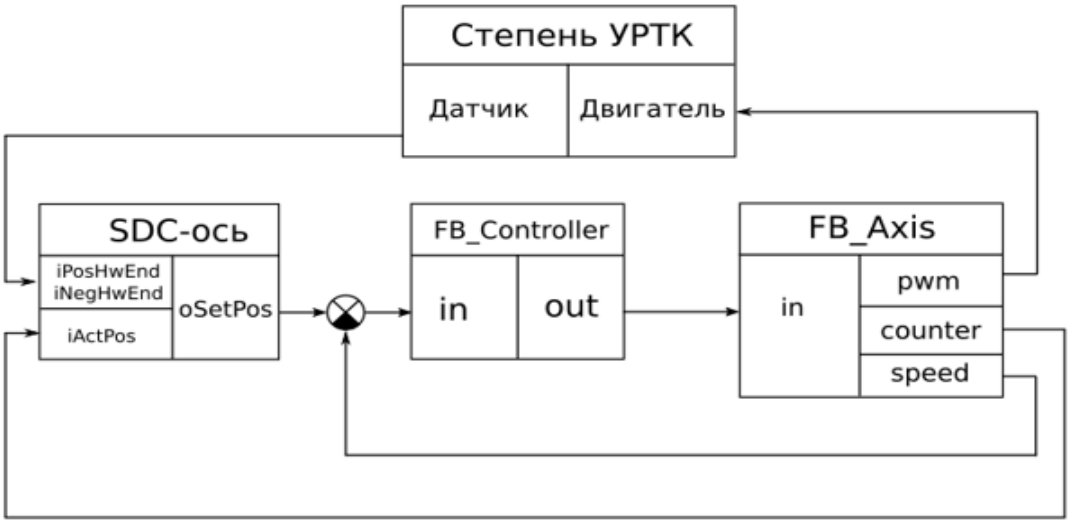

Рис. 7 Структура программного ПО

Дополнительно, в части инициализации основной программы, были:

- Установлены параметры и входы готовности SDC-оси (листинг 1);
- Установлены константы для FB_Control

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 было также:

- Инкрементировать LiveCounter-ы SDC оси (см. листинг 2), чтобы ось не падала в ошибку, было указано 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();
```

- Проведена симуляция, подтверждающая работоспособность данного программного обеспечения.

Прежде чем тестировать разработанную программу на УРТК, была выполнена симуляцию, которая позволила выявить ошибки и отладить программное обеспечение.

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 endswitch_a_reached или endswitch_b_reached посредством опции monitor (см. Рис. 8.)

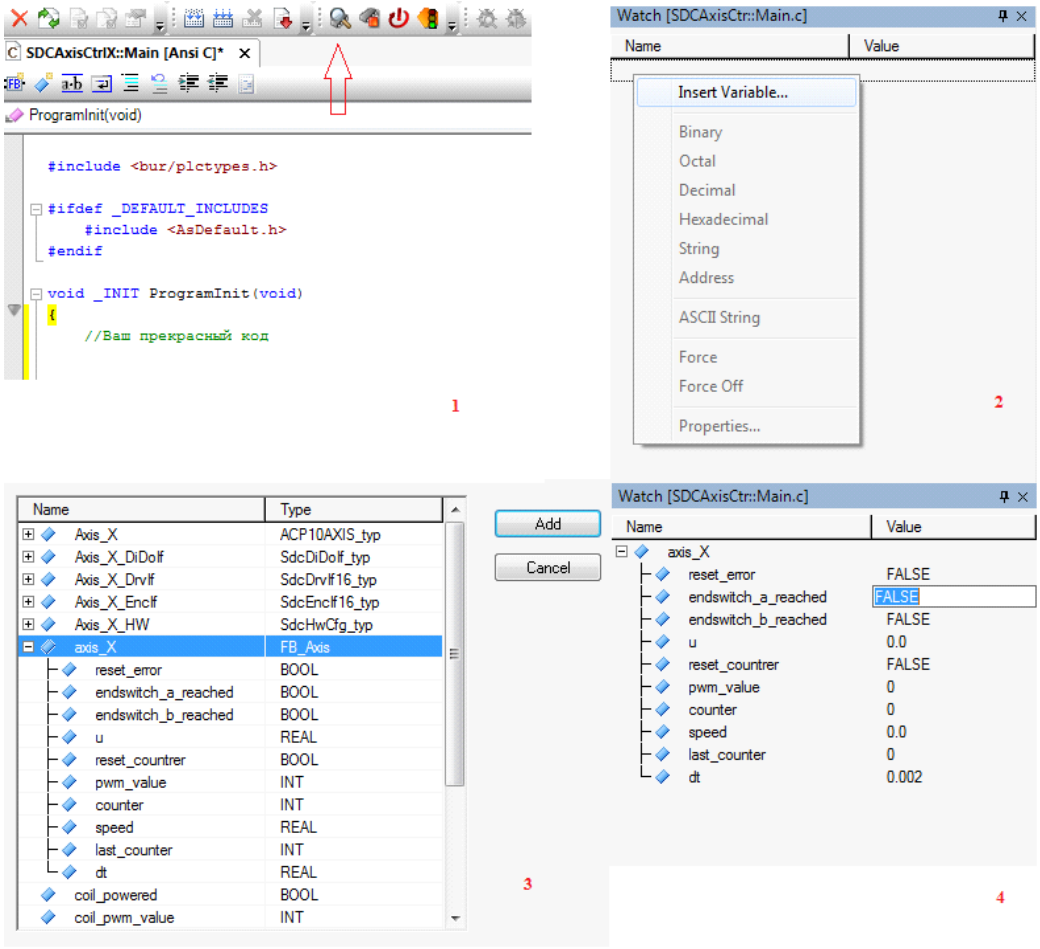

Рис. 8 Изменение значений в режиме мониторинга

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 было выполнить мэпинг переменных.

Была выполнена симуляцию средством «Test». Для этого во вкладке «Configuration View» в папке «Motion» был открыт файл «Acp10map.ncm». В открывшемся окне нажали на Axis_X правой кнопкой мыши и выбирали

«Test» → «Parallel Mode». Далее была выполнена инициализация параметров и включен контроллер (Рис. 9)

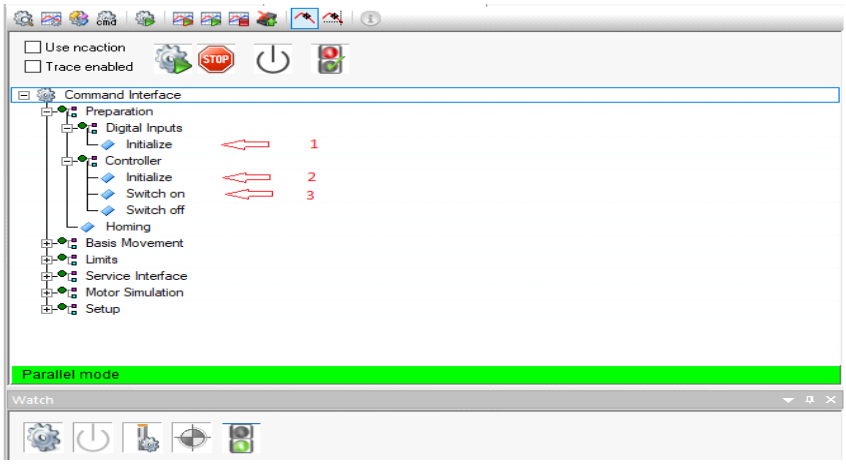

Рис. 9 Инициализация и включение контроллера

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». 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).

Нажав правой кнопкой мыши в открывшемся окне «Trace» и открыв «Configuration» были выбраны значения для записи и во вкладке «Timing» увеличено время снятия трейса (Рис. 10).

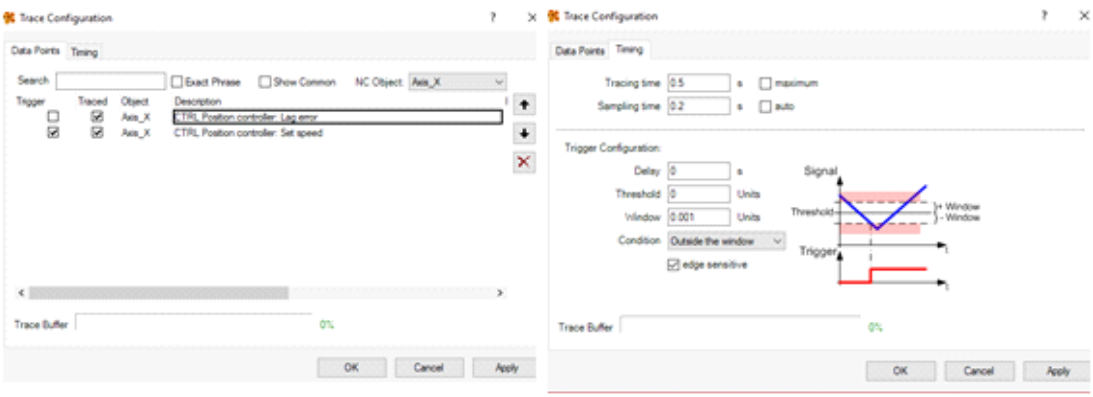

Рис. 10 Trace Configuration

- Средством «Trace» записаны графики перемещения оси по ParID:

  - 111
  - 112
  - 114
  - 113
  - 462

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

Для работы с реальным оборудованием была создана ещё одна конфигурация (Configuration View → Add Configuration…). Изменен код, аккуратно убрав элементы имитации.

Мэпинг переменных SDC-оси указан в таблице в методичке.

7. Проведены 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

8. Результаты экспериментальных исследований

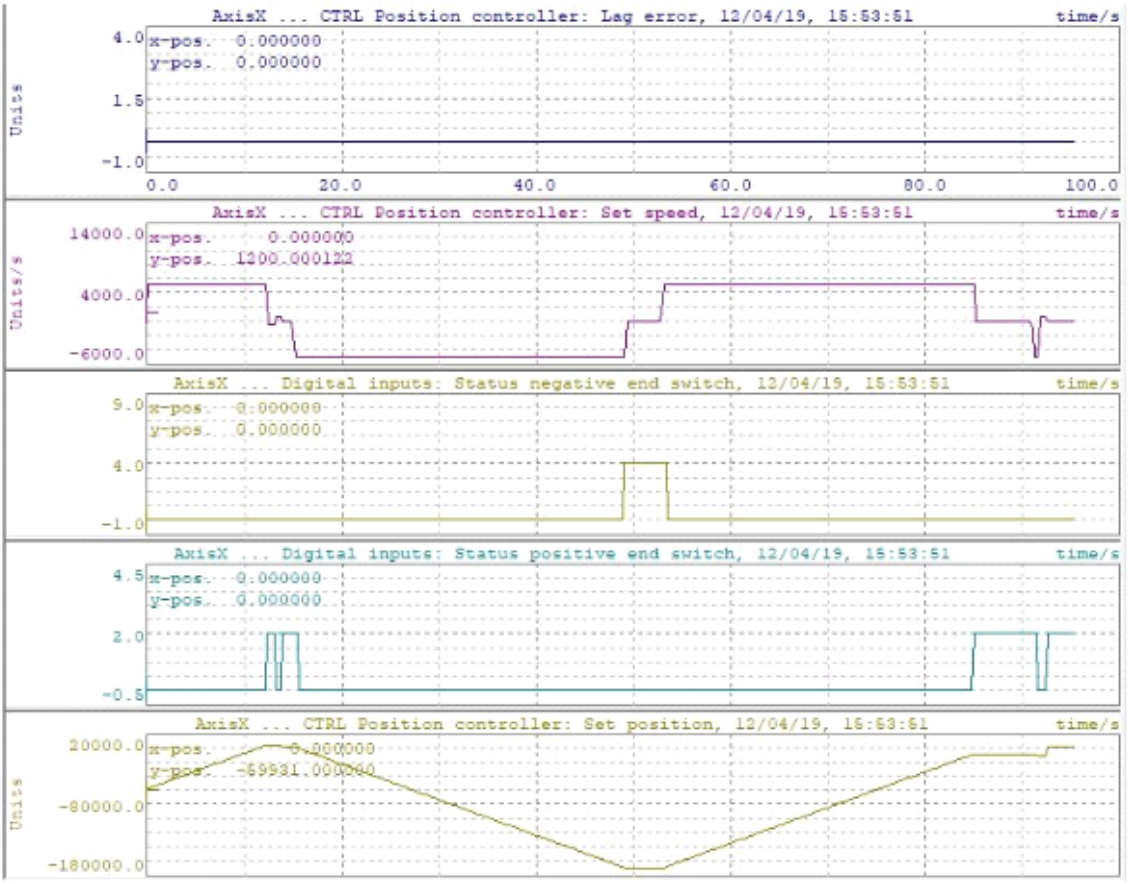

Рис. 11 Графики зависимости Lag Error, Set Speed, Status negative end switch, Status negative end switch, Set position

Вывод: были получены навыков создания программного обеспечения систем управления одной степенью подвижности учебного робота.

Приложение 1.

Листинг кода

Main:

```
#include <bur/plctypes.h>

#ifdef _DEFAULT_INCLUDES
        #include <AsDefault.h>
#endif

void increase_counters(void);
```

```
void Homing(void);

void _INIT ProgramInit(void)
{

        AxisX_HW.EncIf1_Typ = ncSDC_ENC16;
        AxisX_HW.DiDoIf_Typ = ncSDC_DIDO;
        AxisX_HW.DrvIf_Typ = ncSDC_DRVSERVO16;


        strcpy(AxisX_HW.EncIf1_Name, "AxisX_EncIf");
        strcpy(AxisX_HW.DrvIf_Name, "AxisX_DrvIf");
        strcpy(AxisX_HW.DiDoIf_Name, "AxisX_DiDoIf");


        AxisX_EncIf.iEncOK = 1;
        AxisX_DrvIf.iDrvOK = 1;
        AxisX_DrvIf.iStatusEnable = 1;
        AxisX_DiDoIf.iDriveReady = 1;

        fb_regulator.dt = 0.002;
        fb_regulator.k_i = 0.16;
        fb_regulator.k_p = 0.0064;
        fb_regulator.max_abc_value = 24.0;

        pwm_period = 200;
}

void _CYCLIC ProgramCyclic(void)
{
        increase_counters();
        AxisX_EncIf.iActTime = (INT)AsIOTimeCyclicStart();
        AxisX_DiDoIf.iPosHwEnd = axis_X.endswitch_a_reached;
        AxisX_DiDoIf.iNegHwEnd = axis_X.endswitch_b_reached;

        if(coil_powered == 1)
        {
                /*if(!Already_Homed)
                {
                        Homing();
                }
```

```
                else
                {*/
                        increase_counters();
                        axis_X.reset_counter = 0;
                        FB_Axis(&axis_X);
                        coil_pwm_value = 32767;
                AxisX_DiDoIf.iPosHwEnd = axis_X.endswitch_a_reached;

                AxisX_DiDoIf.iNegHwEnd = axis_X.endswitch_b_reached;
                        FB_Axis(&axis_X);

                        fb_regulator.e = (AxisX_DrvIf.oSetPos*100)  - axis_X.speed;//oSetPos;
                        FB_Regulator(&fb_regulator);

                        axis_X.u = fb_regulator.u ;
                        FB_Axis(&axis_X);//}
        }
        else
        {
                coil_pwm_value = 0;
                fb_regulator.e = 0 ;
                FB_Regulator(&fb_regulator);
                axis_X.u = 0;
                FB_Axis(&axis_X);

        }
        // COUNTER1 = axis_X.counter

        //COUNTER2 += axis_X.counter - COUNTER1;
}

void increase_counters(void)
{
        AxisX_EncIf.iLifeCnt++;
        AxisX_DiDoIf.iLifeCntDriveEnable++;
        AxisX_DiDoIf.iLifeCntDriveReady++;
        AxisX_DiDoIf.iLifeCntNegHwEnd++;
        AxisX_DiDoIf.iLifeCntPosHwEnd++;
        AxisX_DiDoIf.iLifeCntReference++;
        AxisX_DrvIf.iLifeCnt++;
}
```

```
void Homing(void)
{
	increase_counters();
	coil_pwm_value = 32767;
	iPosHwEnd = axis_X.endswitch_a_reached;
	iNegHwEnd = axis_X.endswitch_b_reached;
	FB_Axis(&axis_X);

	fb_regulator.e = (AxisX_DrvIf.oSetPos * 10) - axis_X.speed;//oSetPos;
	FB_Regulator(&fb_regulator);

	axis_X.u = fb_regulator.u ;
	FB_Axis(&axis_X);
	if (iPosHwEnd)
	{
		coil_pwm_value = 0;
		fb_regulator.e = 0;
		FB_Regulator(&fb_regulator);
		axis_X.u = 0;
		FB_Axis(&axis_X);
		axis_X.reset_counter = 1;
		FB_Axis(&axis_X);
		if(k == 200)
		{
			Already_Homed = 1;
			FB_Axis(&axis_X);
		}
		k++;

	}

}

void _EXIT ProgramExit(void)
{
	// Insert code here

}
```

fb_AXIS:

```
#include <bur/plctypes.h>
#ifdef __cplusplus
        extern "C"
        {
#endif
        #include "Library.h"
#ifdef __cplusplus
        };
#endif
/* TODO: Add your comment here */
void FB_Axis(struct FB_Axis* inst)
{
        /*if (inst->endswitch_a_reached == 1 )
        {
                inst->dir = 1;
        }
        if (inst->endswitch_b_reached == 1 )
        {
                inst->dir = 0;
        }


        if(inst->dir == 1)
        {
                //inst->reset_counter = 0;
                inst->u = -inst->u;
                inst->pwm_value = (inst->u/24.0) * 32767;
        }
        if(inst->dir == 0)
        {*/
                //inst->reset_counter = 0;
                inst->pwm_value = (inst->u/24.0) * 32767;
//      }
        if (inst->i == 1000)
        {
                inst->speed = (inst->counter - inst->last_counter)/2;

                inst->last_counter = inst->counter;
```

	inst->i = 0;
	}
	inst->spid = inst->speed;
	inst->i++;
	}