МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибернетики
Кафедра проблем управления

Лабораторная работа №3

Тема: «Программное обеспечение системы управления одной степенью робота УРТК»

_____

Выполнил:
Студент 4-го курса
группы КРБО-01-17
Эсаулов И. Д.

Преподаватель:

_____

Морозов А.А

г. Москва,
2020

Цель работы

Получение навыков создания программного обеспечения систем управления одной степенью подвижности учебного робота.

Задание на выполнение лабораторной работы

Создать функциональный блок, основанный на библиотеке Acp10sdc, который будет осуществлять управление одной осью УРТК. Включая обработку концевых датчиков, реферирование к датчику, расположенному со стороны мотора (после реферирования ось переходит в состояние «отключено»). Управление должно быть реализовано из тестового режима. Схема системы управления одной степенью подвижности представлена на рисунке 1:

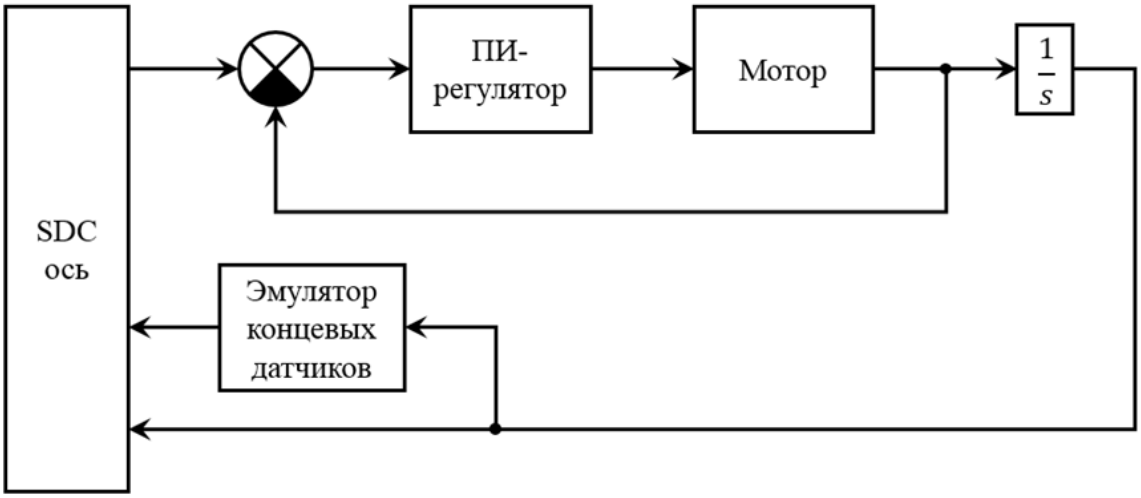

Рис. 1 – Итоговая структурная схема системы управления

Ход работы

В B&R из модулей и интерфейса собирается лабораторный проект. В него входит следующая цепочка элементов:

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

Для дальнейшей работы с модулями добавляются библиотеки Acp10 (для создания SDC-оси) и ASIOTime (для работы с переменными и структурами оси). Из-за различия в версиях возможны некоторые отличия в перечне библиотек, необходимых для реализации работы, однако к ним изменения не применяются (рис. 2).

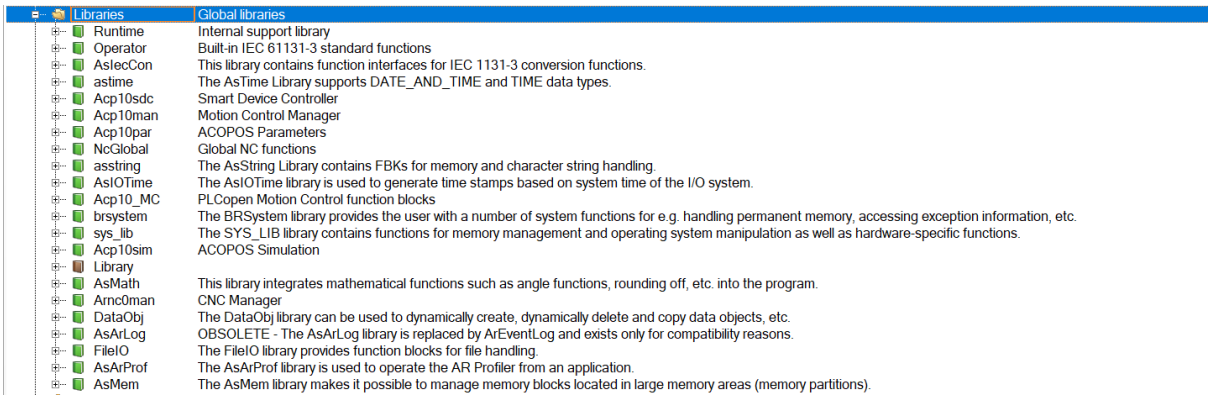

Рис. 2 – перечень подключенных библиотек

Добавляем к переменным в Global.Var, созданным после добавления библиотек, приставку Axis_X (рис. 3). Информация о переменных содержится в таблице 1.

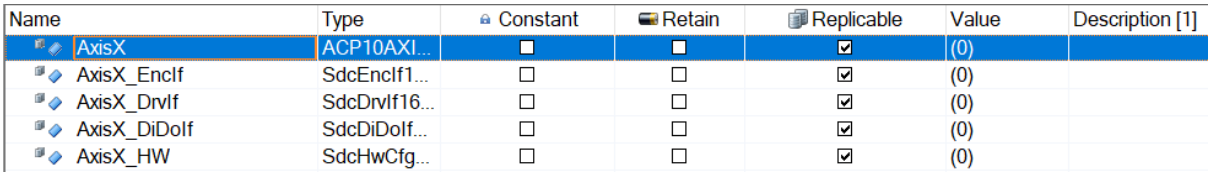

Рис. 3 – Глобальные переменные

Таблица 1. Переменные и их назначение

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

Модуль инициализации оси ACP10 Axis, добавленный автоматически, наполняется переменными, которые содержат реальные параметры оси УРТК (рис. 4). С назначением переменных можно ознакомиться в табл. 2.

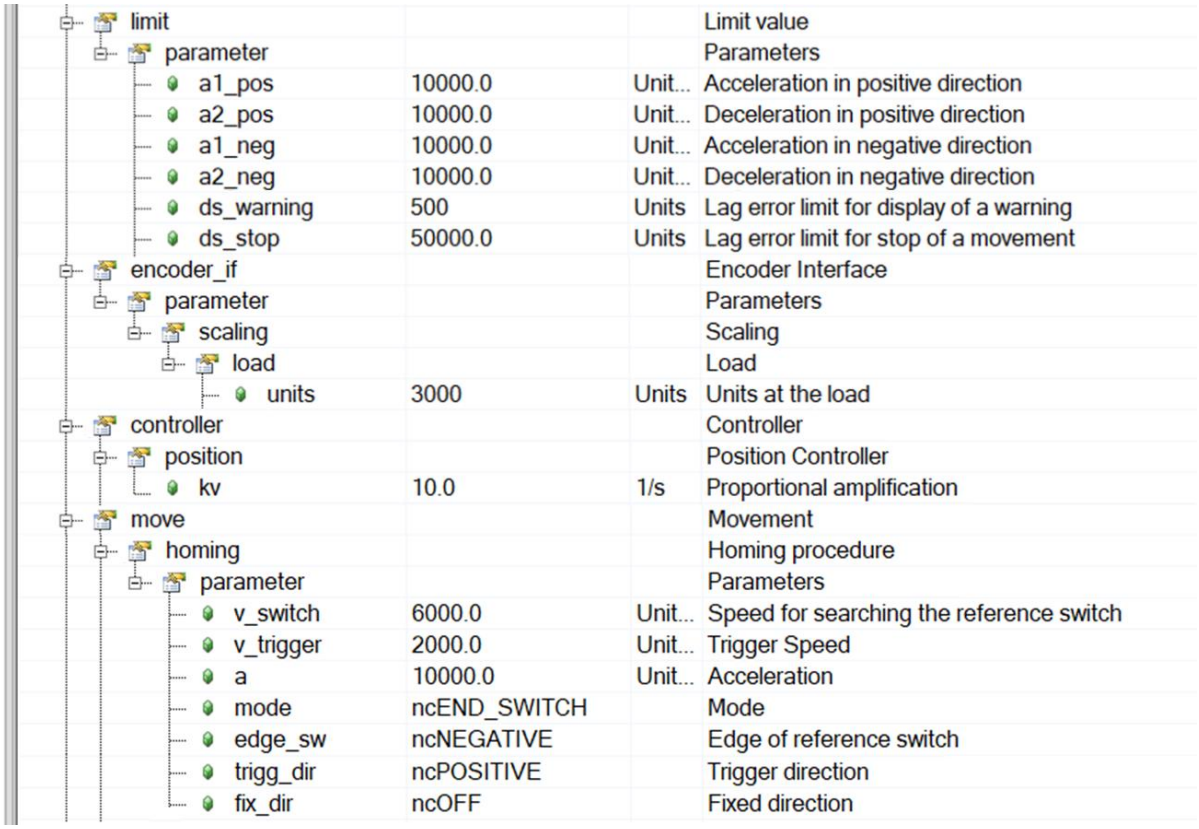

Рис. 4 – Переменные, инициализированные для управления осью

Таблица 2. Значения параметров оси УРТК

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	

Для инициализации оси понадобится таблица «ACOPOS Parameter table» (рис. 5) с параметрами, указанными в методичке.

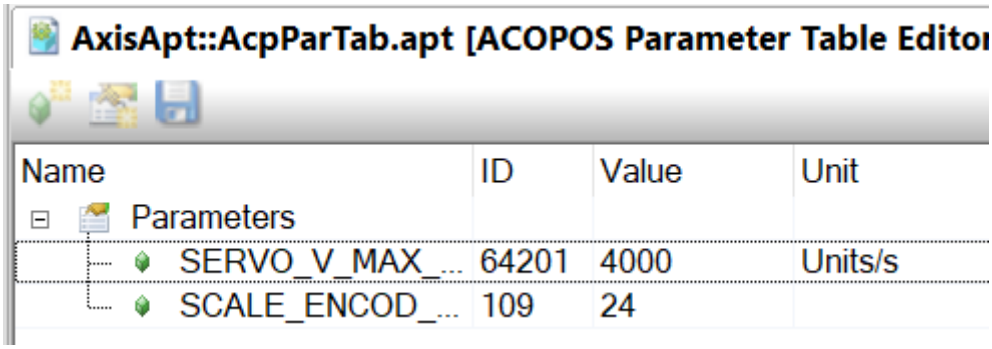

Рис. 5 – Параметры Скорости вращения и Частоты импульсов

Во вкладке “Configuration View” расположена конфигурация “Real”, которая открывает каталог. В каталоге “4PP065_0571_P74\Motion” средствами Toolbox – Object Catalog, выбрав фильтр “Motion”, добавляется в папку Motion файл “ACP10 NC Mapping Table”. В открывшемся окне создается новая AxisX ось (рис. 6).

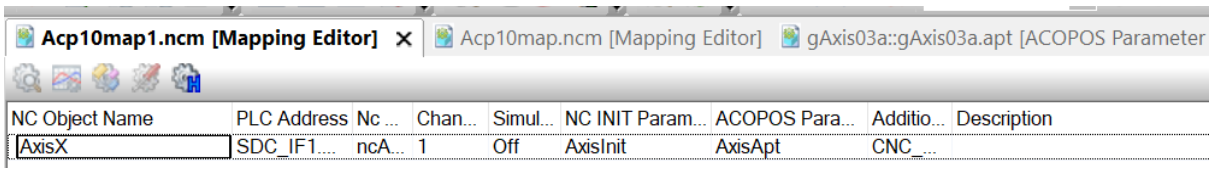

Рис. 6 – Параметры оси

Управление осью разрабатывается на основе результатов, полученных из 1-й лабораторной работы. В текущий проект перемещается библиотека "MotorControl", которая модифицируется путем добавления в ее содержимое функциональный блок "FB_Axis". Блок определяет входное воздействие на ось двигателя с помощью ШИМ сигнала. Параметры функционального блока находятся в таблице 3.

Таблица 3 – Параметры функционального блока

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

После инициализации всех вышеперечисленных параметров создается программа управления осью разработанной в 1-й лабораторной работе моделью управления двигателем. Схема логики ПО на рисунке 7.

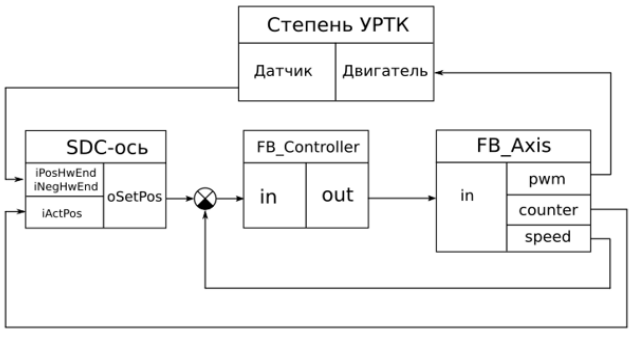

Рис. 7 – Структура Программного Обеспечения

Все значения в вычислениях приводятся к одним единицам измерения. Устанавливаются начальные параметры SDC-оси (листинг 1) и указывается число iActTime как время начала цикла., чтобы ось не падала в ошибку.

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;
```

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();
```

Для проверки работоспособности реализованного проекта необходимо запустить его в симуляции. 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.

Симуляция выполняется через функцию “test”. Для этого во вкладке «Configuration View» в папке «Motion» открываем файл “Acp10map.ncm”. В открывшемся окне нажимаем на Axis_X правой кнопкой мыши и выбираем “Test” (рис. 8).

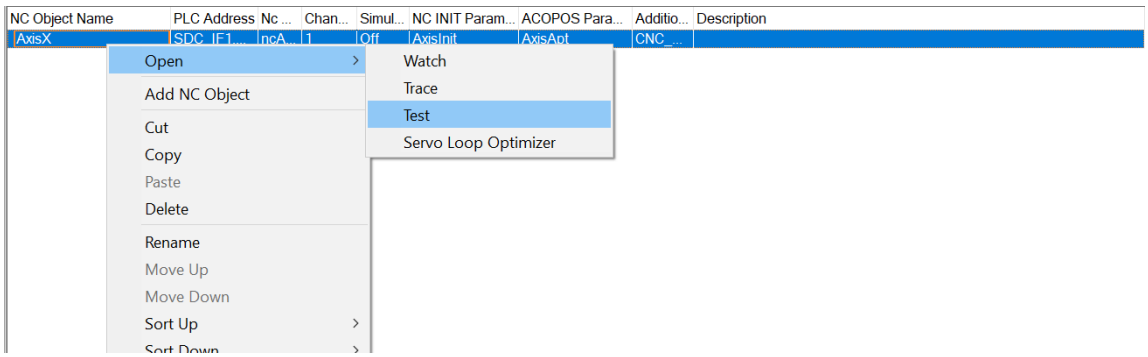

Рис. 8 – Переход в режим симуляции

Во всплывающем окне выбирается режим “Parallel Mode”. Зеленый индикатор свидетельствует о том, что проект запущен корректно (рис. 9).

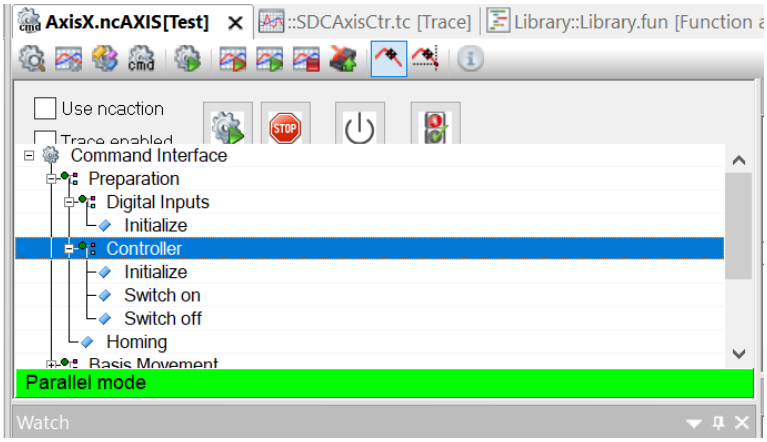

Рис. 9 – Окно AxisX.ncAXIS

Результаты работы можно отследить в окне Trace. Настраивается оно через вкладку Configuration (рис. 10).

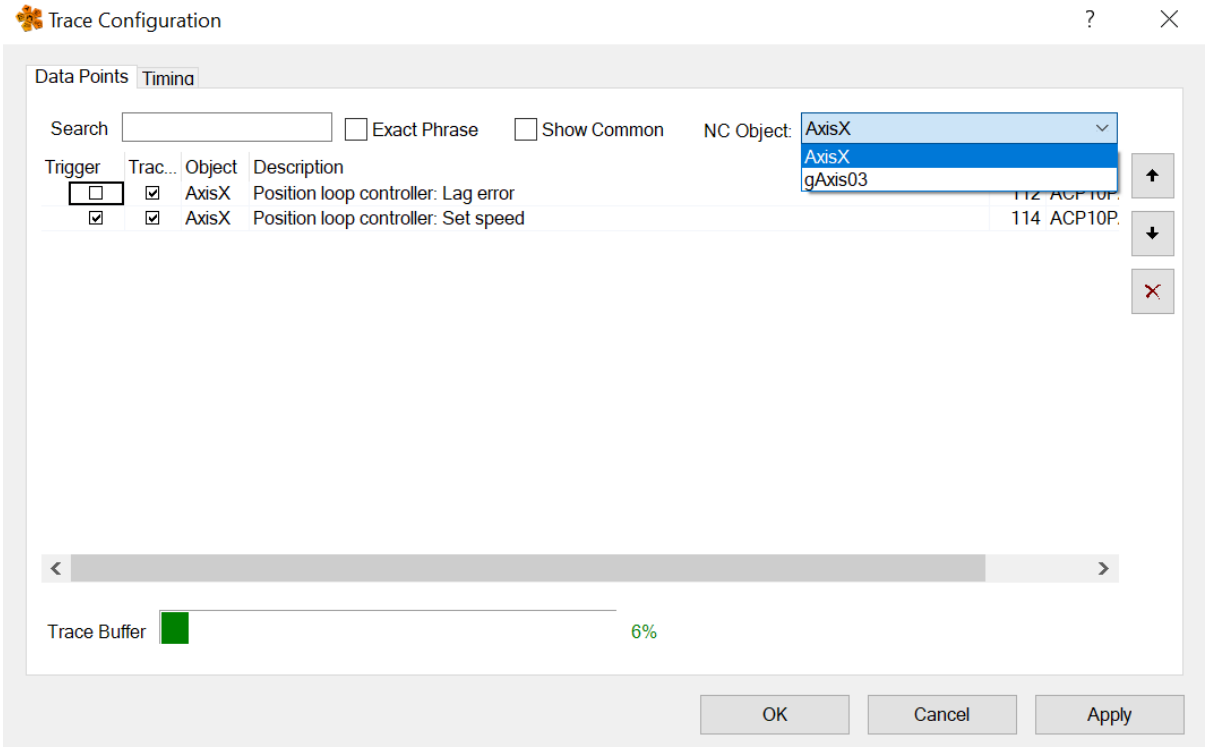

Рис. 10 – Вкладка Configuration

В окне Trace выводятся следующие графики зависимостей: Lag Error, Set Speed, Status negative end switch, Status negative end switch, Set position (рис. 11).

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

Вывод

В ходе выполнения лабораторной удалось реализовать рабочий проект системы управления одной степенью подвижности учебного робота. Были получены навыки создания функциональных блоков и написания для них программ в среде разработки ПО B&R. С одними только модулями программы и кодом, связующим и управляющим собранной моделью, получилось проверить работоспособность проекта в симуляции.

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++;
                    }
```

FB_Control:

```
#include <bur/plctypes.h>
#ifdef __cplusplus
	extern "C"
	{
#endif
	#include "Library.h"
#ifdef __cplusplus
	};
#endif

void FB_Controller(struct FB_Controller* inst)
{
	REAL a = inst->in * inst->k_p;
	REAL b = inst->in * inst->k_i;
	inst->integrator.dt = inst->dt;
	inst->integrator.in = b;
	FB_Integrator(&inst->integrator);
	a = a > inst->max_abs_value ? inst->max_abs_value : a;
	a = a < -inst->max_abs_value ? -inst->max_abs_value : a;
	inst->sum = a + inst->integrator.out;
	inst->out = inst->sum;
	inst->out = inst->out > inst->max_abs_value ? inst->max_abs_value : inst->out;
	inst->out = inst->out < -inst->max_abs_value ? -inst->max_abs_value : inst->out;
	inst->last_sum = (inst->out - inst->sum);
	inst->integrator.state = inst->integrator.state + inst->last_sum;
}
```

FB_Integrator:

```
#include <bur/plctypes.h>
#ifdef __cplusplus
	extern "C"
	{
#endif
	#include "Library.h"
#ifdef __cplusplus
	};
#endif
/* TODO: Add your comment here */
void FB_Integrator(struct FB_Integrator* inst)
{
	inst->out = inst->dt * inst->in + inst->state;
	inst->state = inst->out;
}
```