МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА
Институт кибернетики
Кафедра проблем управления

Лабораторная работа №3

по дисциплине «Программное обеспечение мехатронных и робототехнических систем»

Тема: Программное обеспечение системы управления одной степенью робота УРТК

Работу выполнил
студент КРБО-01-17:
Денисов Д.С.

Преподаватель:
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. 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.

Ход работы:

Был создан новый проект в среде Automation Studio, конфигурация оборудования представлена на рисунках 1 и 2.

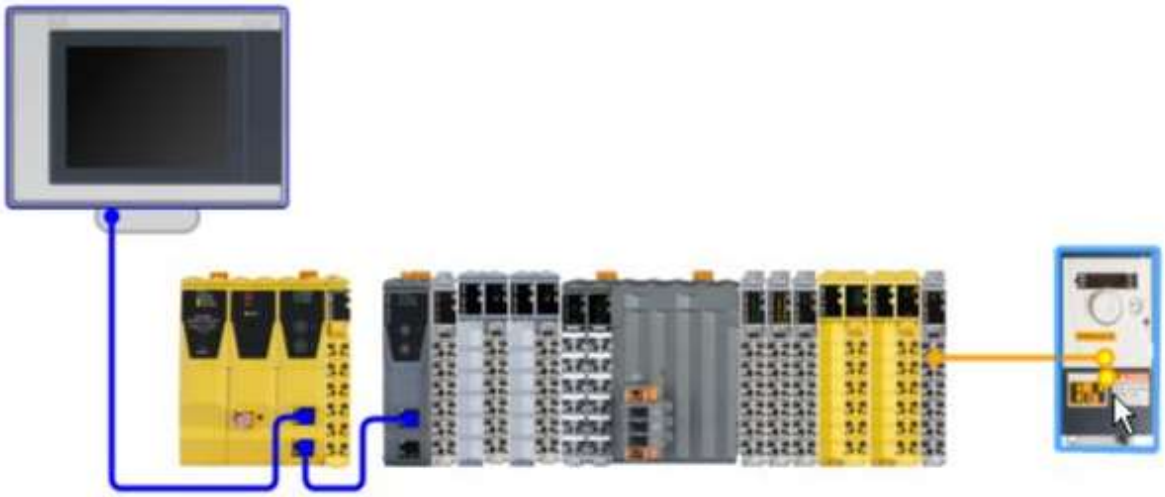

Рисунок 1. Конфигурация оборудования

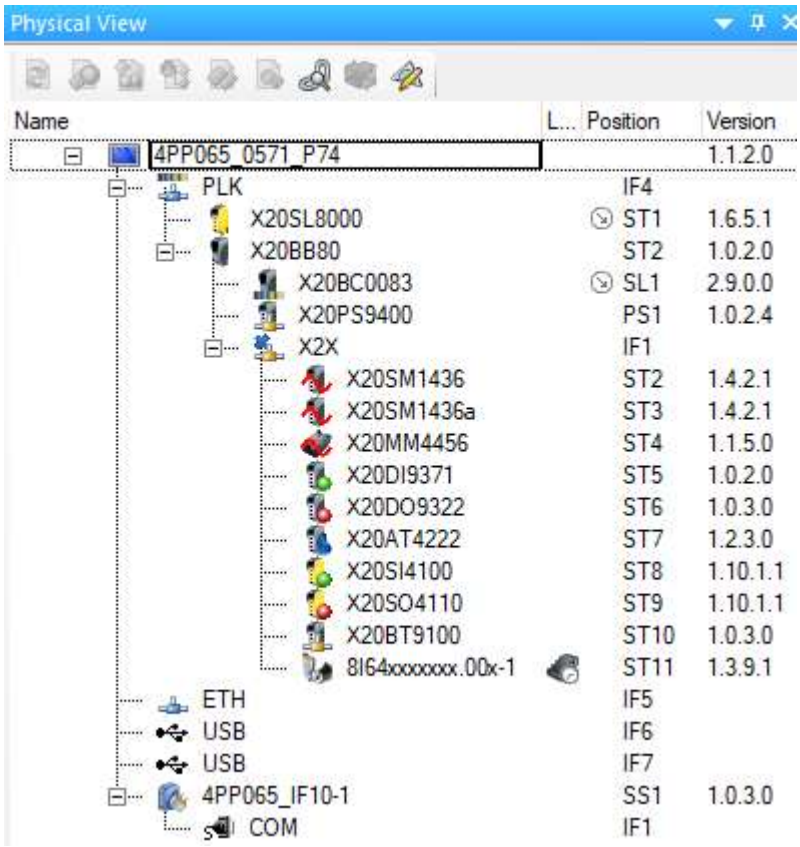

Рисунок 2. Список используемых компонентов

Затем была создана SDC-ось управления одной степенью подвижности УРТК, структурная схема которой изображена на рисунке 3.

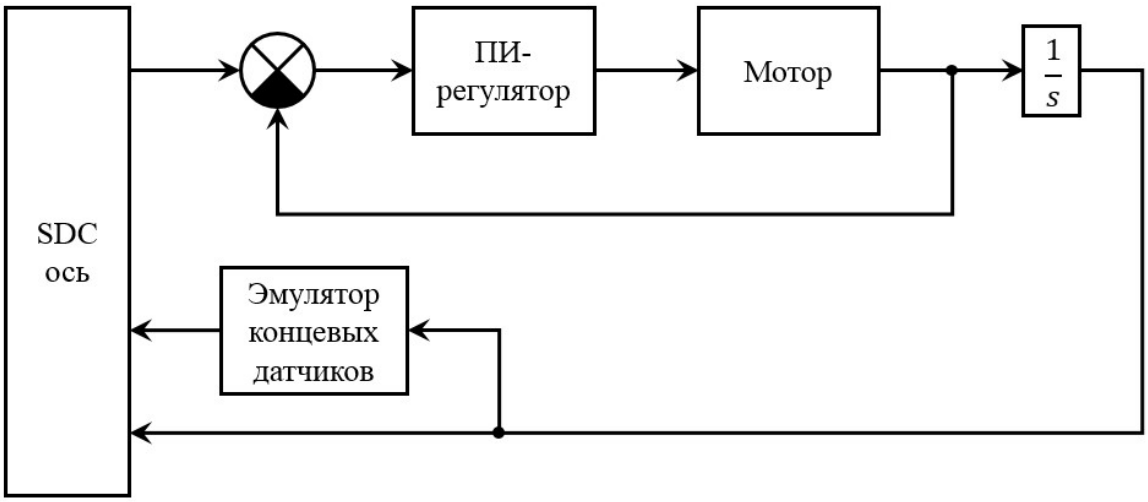

Рисунок 3. Структурная схема управления одной степенью подвижности

Для создания SDC-оси была добавлена библиотека Acp10. После этого в проекте, помимо стандартного набора, появился следующий набор библиотек (рисунок 4):

-Acp10sdc;

-Acp10man;

-Acp10par;

-Acp10_MC;

-Acp10sim.

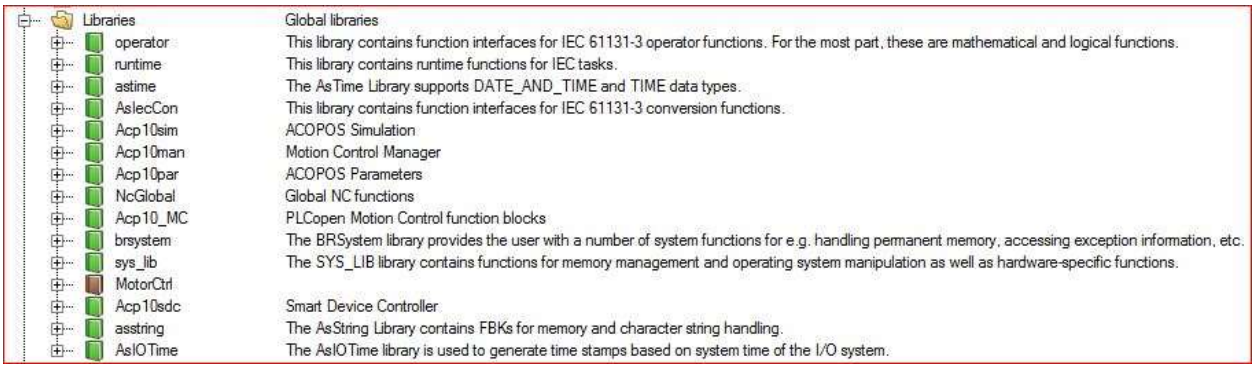

Рисунок 4. Библиотеки

Затем была добавлена библиотека B&R AsIOTime.

Созданы переменные и структуры, необходимые для работы SDC (таблица 1).

Таблица 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

Набор глобальных переменных имеет следующий вид (рисунок 5):

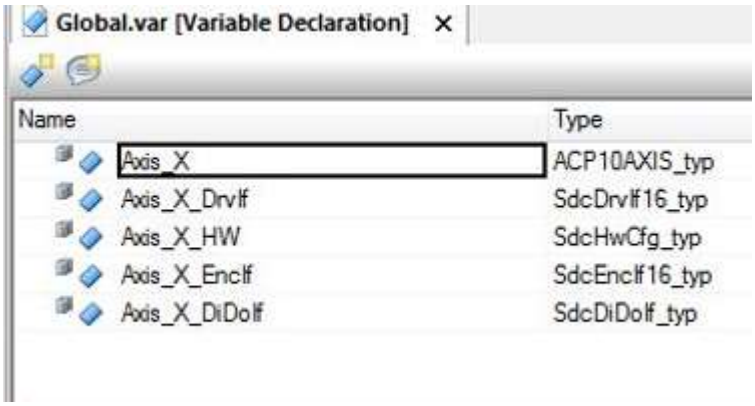

Рисунок 5. Глобальные переменные

Для создания SDC-оси, средством Toolbox был добавлен модуль инициализации оси «ACP10 Axis».

Далее в параметрах инициализации оси «ACP10 Axis» были указаны реальные параметры оси УРТК (таблица 2).

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	Режим подъезда к датчику
trigg_dir	ncNEGATIVE	
fix_dir	ncOFF	

Далее средством Toolbox была добавлена таблица инициализации оси «ACOPOS Parameter table». В таблице были указаны необходимые параметры (таблица 3).

Таблица 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

В каталоге «4PP065_0571_P74\Motion» средствами Toolbox – Object Catalog, выбрав фильтр «Motion», была добавлен в папку Motion файл «ACP10 NC Mapping Table».

В созданную таблицу была добавлена SDC ось.

Далее были заполнены параметры новой оси (таблица 4).

Таблица 4 - Параметры мэпинга создаваемой SDC-оси

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

Скопировав в данный проект библиотеку «MotorControl» из лабораторной работы №1, в нее был добавлен функциональный блок «FB_Axis» (таблица 5).

Таблица 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
внутреннее состояние	desired_speed	REAL	Требуемая скорость
внутреннее состояние	regulator	FB_Regulator	Регулятор скорости
внутреннее состояние	counter_buffer	INT	Буфер для счетчика импульсов
внутреннее состояние	sdc_enc	SdcEncIf16_typ	Переменная для контроля датчиков двигателя
внутреннее состояние	sdc_drv	SdcDrvIf16_typ	Переменная для контроля интерфейса привода
внутреннее состояние	sdc_dido	SdcDiDoIf16_typ	Переменная для контроля цифровых входов/выходов интерфейса привода

Блок реализует обработку инкрементного датчика. Фиксируется изменение переменной «counter» для оценки текущей скорости двигателя. Так как частота вычислительных циклов велика, то изменение сначала заносится в специальный буфер. Это необходимо, чтобы увеличить количество уровней дискретизации на допустимом диапазоне скорости. Затем блок передает состояние концевых датчиков в порт ввода/вывода SDC-оси. Алгоритм блока заканчивается с установкой времени импульса ШИМ-сигнала. Для этого значение переменной oSetPos приводится к Unit/sec. Его разность с текущей вычисленной скоростью подается на регулятор, коэффициенты которого были

настроены для корректной работы конечной системы с обратной связью. Листинг функционального блока FB_Axis приведен в Приложении А.

Для создания программы обработки одной оси была создана ANSI C Program с названием «SDCAxisCtrlX» и в нее были добавлены необходимые переменные (таблица 6).

Таблица 6 – Переменные программы SDCAxisCtrlX

Имя	**Тип данных**	**Описание**
axis_X	FB_Axis	Переменная оси
coil_powered	BOOL	Включение обмотки возбуждения
coil_pwm_value	INT	Время импульса ШИМ
pwm_period	UINT	Период ШИМ
motor_X	FB_Motor	Симулируемый электродвигатель

Код программы вызывает функциональный блок переменной оси axis_X.

Так как тестирование лабораторной работы происходит в симуляции, то реальный двигатель был заменен моделью из первой лабораторной работы. Концевые датчики в работе активируются по значению переменной counter.

Затем была выполнена симуляция средством «Test». Для этого во вкладке «Configuration View» в папке «Motion» был открыт файл «Acp10map.ncm». После этого была выполнена инициализация параметров и включен контроллер.

Далее была произведена операция «Homing», после которой ось отреферировалась. Данная операция была произведена по рисунку 6.

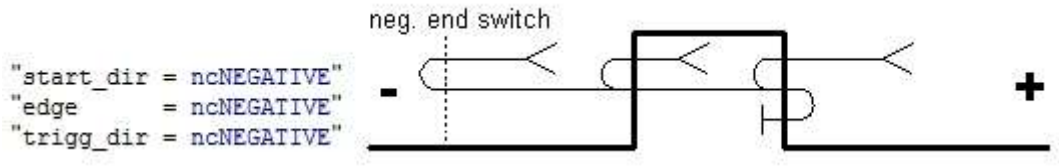

Рисунок 6. Процедура хоуминга

Средством «Trace» записать графики перемещения оси (рисунок 7). На графиках изображены: состояние концевого датчика, текущее положение, желаемая скорость и оценка реальной текущей скорости. По графикам видно, что реальная скорость несколько отстает от уставки контроллера. Также видны погрешности, возникающие из-за дискретной природы инкрементного датчика.

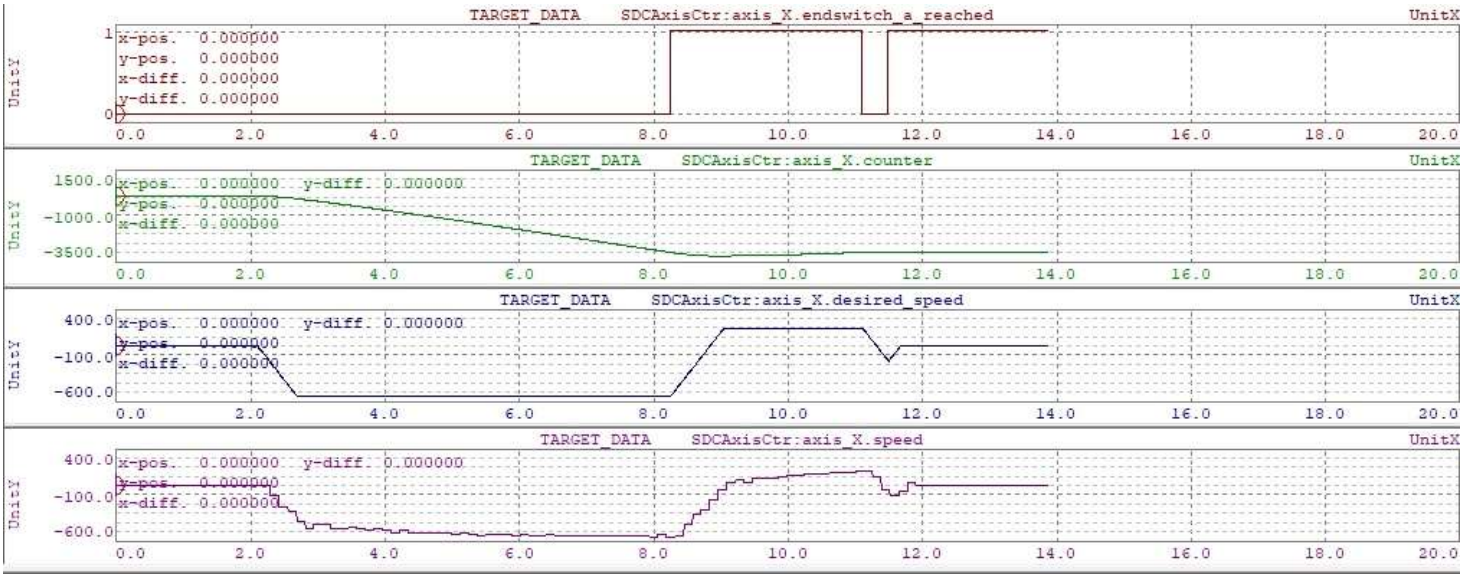

Рисунок 7. Графики перемещения оси во время процедуры «homing»

Вывод: получены навыки создания программного обеспечения систем управления одной степенью подвижности учебного робота. В ходе выполнения работы был проведен эксперимент, показывающий работоспособность системы управления одной степенью подвижности учебного робота.

Приложение А

Листинг функционального блока FB-Axis

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

void FB_Axis(struct FB_Axis* inst)
{
	handleEncoder(inst);
	handleEndSwitches(inst);
	setPwm(inst);
}

void handleEncoder(struct FB_Axis* inst){
	inst->counter_buffer += inst->counter - inst->last_counter;
	inst->last_counter = inst->counter;
	inst->sdc_enc->iActPos = inst->counter;
	if((inst->sdc_enc->iLifeCnt) % COUNTER_BUFFER_CYCLES == 0){
```

```
            inst->speed = (inst->counter_buffer) / (0.002 *
COUNTER_BUFFER_CYCLES); // cycle = 2 ms
            inst->counter_buffer = 0;
        }
}

void handleEndSwitches(struct FB_Axis* inst){
        inst->sdc_dido->iNegHwEnd = inst->endswitch_a_reached;
        inst->sdc_dido->iPosHwEnd = inst->endswitch_b_reached;
}

void setPwm(struct FB_Axis* inst){
        inst->desired_speed = inst->sdc_drv->oSetPos * UNITS_PER_ROTATION
/ 32767;
        inst->regulator.e = inst->desired_speed - inst->speed;
        FB_Regulator(&(inst->regulator));
        inst->pwm_value = (inst->regulator.u) / (inst->regulator.max_abs_value) *
(inst->pwm_period);
}
```

Приложение Б

Листинг основной программы

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

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

void _INIT ProgramInit(void)
{
	// DiDo initialization
	Axis_X_HW.EncIf1_Typ = ncSDC_ENC16;
	Axis_X_HW.DiDoIf_Typ = ncSDC_DIDO;
	Axis_X_HW.DrvIf_Typ = ncSDC_DRVSERVO16;
	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");

	// SDC-axis initialization
	Axis_X_ModuleOk = 1;
	Axis_X_EncIf.iEncOK = 1;
	Axis_X_DrvIf.iDrvOK = 1;
	Axis_X_DrvIf.iStatusEnable = 1;
	Axis_X_DiDoIf.iDriveReady = 1;

	// Axis_X initialization
	axis_X.counter = axis_X.last_counter = 0;
	axis_X.pwm_period = pwm_period = 0.05;
```

```
	axis_X.sdc_enc = &Axis_X_EncIf;
	axis_X.sdc_drv = &Axis_X_DrvIf;
	axis_X.sdc_dido = &Axis_X_DiDoIf;
	axis_X.regulator.dt = 0.002;
	axis_X.regulator.integrator.dt = 0.002;
	axis_X.regulator.k_i = 0.0072;
	axis_X.regulator.k_p = 0.0072;
	axis_X.regulator.max_abs_value = 24.0;

	// Simulated motor initialization
	motor_X.dt = 0.01;
	motor_X.ke = 1;
	motor_X.Tm = 0.1;
	motor_X.u = 0;
	motor_X.phi = 0;
	motor_X.integrator1.dt = 0.002 / motor_X.Tm;
	motor_X.integrator2.dt = 0.002;
}

void _CYCLIC ProgramCyclic(void)
{
	incrementLifeCounters();
	FB_Axis(&axis_X);
	simulateSensors(&axis_X, &motor_X);
}

void _EXIT ProgramExit(void)
{
```

```
}

void incrementLifeCounters() {
	Axis_X_EncIf.iLifeCnt++;
	Axis_X_EncIf.iActTime = (INT)AsIOTimeCyclicStart();
	Axis_X_DrvIf.iLifeCnt++;
	Axis_X_DrvIf.oSetPos = Axis_X.monitor.v;
	Axis_X_DiDoIf.iLifeCntDriveEnable++;
	Axis_X_DiDoIf.iLifeCntDriveReady++;
	Axis_X_DiDoIf.iLifeCntNegHwEnd++;
	Axis_X_DiDoIf.iLifeCntPosHwEnd++;
	Axis_X_DiDoIf.iLifeCntReference++;
}

void simulateSensors(struct FB_Axis* axis, struct FB_Motor* motor){
	if(axis->sdc_drv->oSetPos != 0){
		motor->u = axis->pwm_value / axis->pwm_period *
MAX_UNITS_PER_SECOND;
		FB_Motor(motor);
		axis->counter = motor->phi;

		axis->endswitch_a_reached = axis->counter <= -3000 ? 1 : 0;
		axis->endswitch_b_reached = axis->counter >= 3000 ? 1 : 0;
	}
}
```