

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего
образования

«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибернетики
Кафедра проблем управления

Лабораторная работа №2

Тема: «Программное обеспечение системы управления мехатронного
модуля управления защитной дверью»



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

Преподаватель:
Морозов А.А

г. Москва,
2020

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

Задание на выполнение лабораторной работы: разработать программное обеспечение системы автоматического управления приводом защитной двери. Дверь оснащена асинхронным двигателем и четырьмя датчиками положения ($S0 - S3$), 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 $s1s2$, меньшую скорость на отрезках $s0s1$ и $s2s3$ (для обеспечения безопасности движения). Индикаторы должны показывать местоположение двери.

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

Ход Работы

Откроем исходную конфигурацию и создадим модуль частотного преобразователя, в проекте будут использоваться его настройки по умолчанию.

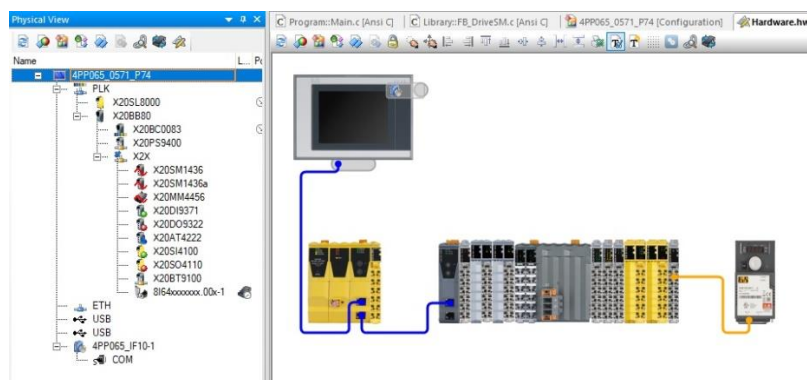


Рисунок 1. Конфигурация оборудования

Далее перейдем к конфигурации асинхронного двигателя, используя документацию к нему.

Таблица 1 – 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

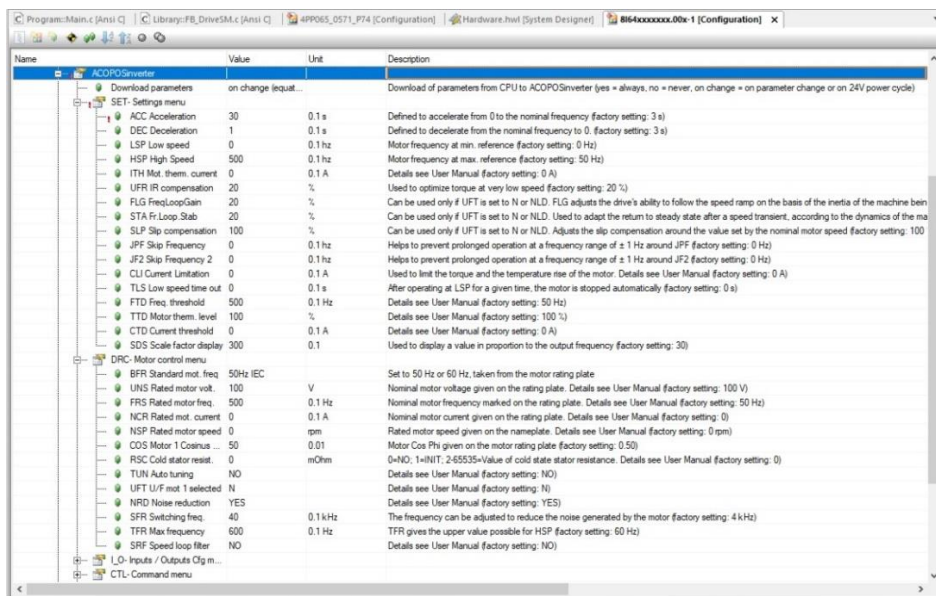


Рисунок 2. Новые параметры конфигурации

Теперь необходимо создать объекты для управления. Это

- функциональный блок «DriveStateMashine», который реализуется управление частотным преобразователем.
- функциональный блок «DoorStateMashine», в котором будет реализована основная логика программы – задание направления вращения и скорости движения воротами в зависимости от их положения и требуемого направления вращения.
- функциональный блок «LedStateMashine» - машина состояний обработки светодиодных индикаторов.

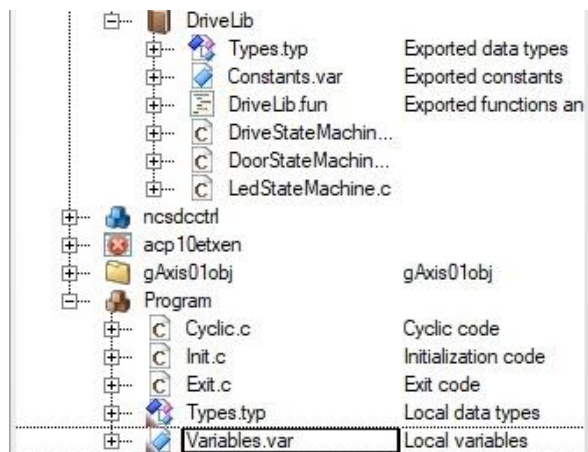


Рисунок 3. Внешний вид функциональных блоков

Для упрощения описания состояний ворот необходимо создать новый тип данных «DoorStates». Необходимые состояния приведены в таблице 2.

Параметры функциональных блоков предоставлены в таблицах

Таблица 2 - Состояния ворот для типа данных «DoorStates»..

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

Таблица 3 - Параметры функционального блока «DriveStateMashine»

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

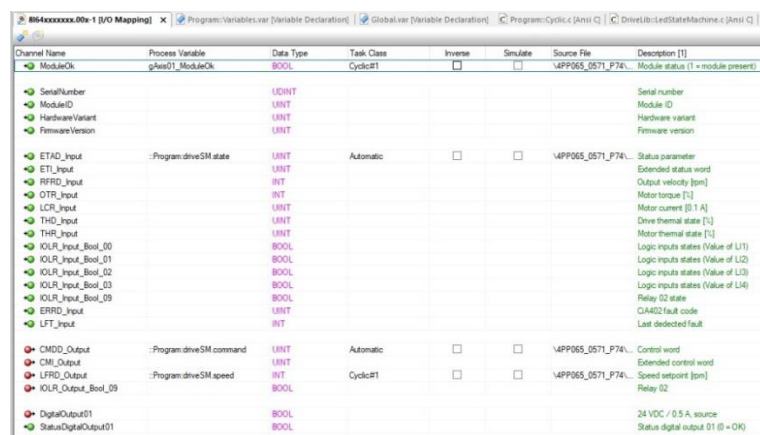
Таблица 4 – Структура функционального блока «DoorStateMashine»

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

Таблица 5 – Структура функционального блока «LedStateMashine»

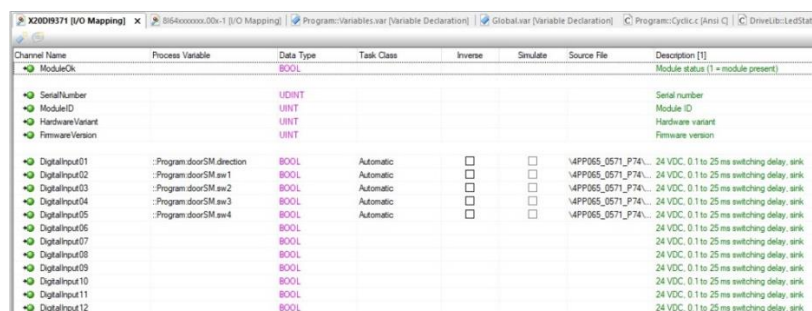
Конфигурация	Имя	Тип данных	Описание
ВХОД	State	UINT	Состояние частотного преобразователя
ВЫХОД	led1	BOOL	Сигнал работы функционального блока
ВЫХОД	led2	BOOL	Сигнал работы функционального блока
ВЫХОД	led3	BOOL	Сигнал работы функционального блока
ВЫХОД	led4	BOOL	Сигнал работы функционального блока
ВЫХОД	timer	INT	Заданная скорость

Теперь необходимо объединить программный код с реальной установкой для этого перейдем во вкладку “I/O Mapping” и настоим взаимодействие блоков



Channel Name	Process Variable	Data Type	Task Class	Inverse	Simulate	Source File	Description [1]
ModuleOk	gAes01_ModuleOk	BOOL	CycleR1			\AP065_0571_P74\...	Module status (1 = module present)
SerialNumber		UINT					Serial number
ModuleID		UINT					Module ID
HardwareVariant		UINT					Hardware variant
FirmwareVersion		UINT					Firmware version
ETAD_Input	:Program driveSM state	UINT	Automatic			\AP065_0571_P74\...	Status parameter
ETI_Input		UINT					Extended status word
RFRD_Input		INT					Output velocity [rpm]
OTR_Input		INT					Motor torque [%]
LCR_Input		UINT					Motor current [0.1 A]
THD_Input		UINT					Drive thermal state [%]
THR_Input		UINT					Motor thermal state [%]
KOLR_Input_Boo0_01		BOOL					Logic inputs states (Value of L1)
KOLR_Input_Boo0_02		BOOL					Logic inputs states (Value of L2)
KOLR_Input_Boo0_03		BOOL					Logic inputs states (Value of L3)
KOLR_Input_Boo0_09		BOOL					Logic inputs states (Value of L4)
ERRO_Input		UINT					Relay 02 state
LFT_Input		INT					CIA02 fault code
CMDO_Output	:Program driveSM command	UINT	Automatic			\AP065_0571_P74\...	Control word
CMI_Output		UINT					Extended control word
LFRD_Output	:Program driveSM speed	INT	CycleR1			\AP065_0571_P74\...	Speed setpoint [rpm]
KOLR_Output_Boo0_09		BOOL					Relay 02
DigitalOutput01		BOOL					24 VDC / 0.5 A source
StatusDigitalOutput01		BOOL					Status digital output 01 (0 = OK)

Рисунок 4. Внешний вид частотного преобразователя



Channel Name	Process Variable	Data Type	Task Class	Inverse	Simulate	Source File	Description [1]
ModuleOk		BOOL					Module status (1 = module present)
SerialNumber		UINT					Serial number
ModuleID		UINT					Module ID
HardwareVariant		UINT					Hardware variant
FirmwareVersion		UINT					Firmware version
DigitalInput01	:Program doorSM direction	BOOL	Automatic			\AP065_0571_P74\...	24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput02	:Program doorSM sw1	BOOL	Automatic			\AP065_0571_P74\...	24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput03	:Program doorSM sw2	BOOL	Automatic			\AP065_0571_P74\...	24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput04	:Program doorSM sw3	BOOL	Automatic			\AP065_0571_P74\...	24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput05	:Program doorSM sw4	BOOL	Automatic			\AP065_0571_P74\...	24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput06		BOOL					24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput07		BOOL					24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput08		BOOL					24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput09		BOOL					24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput10		BOOL					24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput11		BOOL					24 VDC. 0.1 to 25 ms switching delay, sink
DigitalInput12		BOOL					24 VDC. 0.1 to 25 ms switching delay, sink

Рисунок 5. Внешний вид блока X20DI9371

Channel Name	Process Variable	Data Type	Task Class	Inverse	Simulate	Source File	Description [1]
ModuleOk		BOOL					Module status (1 = module present)
SerialNumber		UDINT					Serial number
ModuleID		UINT					Module ID
HardwareVariant		UINT					Hardware variant
FirmwareVersion		UINT					Firmware version
DigitalOutput01		BOOL					24 VDC / 0.5 A, source
DigitalOutput02	::ProgramIedSM.Ied1	BOOL	Automatic	<input type="checkbox"/>	<input type="checkbox"/>	\4PP065_0571_P74\...	24 VDC / 0.5 A, source
DigitalOutput03	::ProgramIedSM.Ied2	BOOL	Automatic	<input type="checkbox"/>	<input type="checkbox"/>	\4PP065_0571_P74\...	24 VDC / 0.5 A, source
DigitalOutput04	::ProgramIedSM.Ied3	BOOL	Automatic	<input type="checkbox"/>	<input type="checkbox"/>	\4PP065_0571_P74\...	24 VDC / 0.5 A, source
DigitalOutput05	::ProgramIedSM.Ied4	BOOL	Automatic	<input type="checkbox"/>	<input type="checkbox"/>	\4PP065_0571_P74\...	24 VDC / 0.5 A, source
DigitalOutput06		BOOL					24 VDC / 0.5 A, source
DigitalOutput07		BOOL					24 VDC / 0.5 A, source
DigitalOutput08		BOOL					24 VDC / 0.5 A, source
DigitalOutput09		BOOL					24 VDC / 0.5 A, source
DigitalOutput10		BOOL					24 VDC / 0.5 A, source
DigitalOutput11		BOOL					24 VDC / 0.5 A, source
DigitalOutput12		BOOL					24 VDC / 0.5 A, source
StatusDigitalOutput01		BOOL					Status digital output 01 (0 = OK)
StatusDigitalOutput02		BOOL					Status digital output 02 (0 = OK)
StatusDigitalOutput03		BOOL					Status digital output 03 (0 = OK)
StatusDigitalOutput04		BOOL					Status digital output 04 (0 = OK)
StatusDigitalOutput05		BOOL					Status digital output 05 (0 = OK)
StatusDigitalOutput06		BOOL					Status digital output 06 (0 = OK)
StatusDigitalOutput07		BOOL					Status digital output 07 (0 = OK)
StatusDigitalOutput08		BOOL					Status digital output 08 (0 = OK)
StatusDigitalOutput09		BOOL					Status digital output 09 (0 = OK)
StatusDigitalOutput10		BOOL					Status digital output 10 (0 = OK)
StatusDigitalOutput11		BOOL					Status digital output 11 (0 = OK)
StatusDigitalOutput12		BOOL					Status digital output 12 (0 = OK)

Рисунок 6. Внешний вид блока X20DO9322отвечающего за кнопки

Логика программы состояний ворот и возможных переходах между ними была реализована на основе блок-схемы, предоставленной на рисунке 7. На рисунке 8 предоставлена блок-схема управления частотным преобразователем.

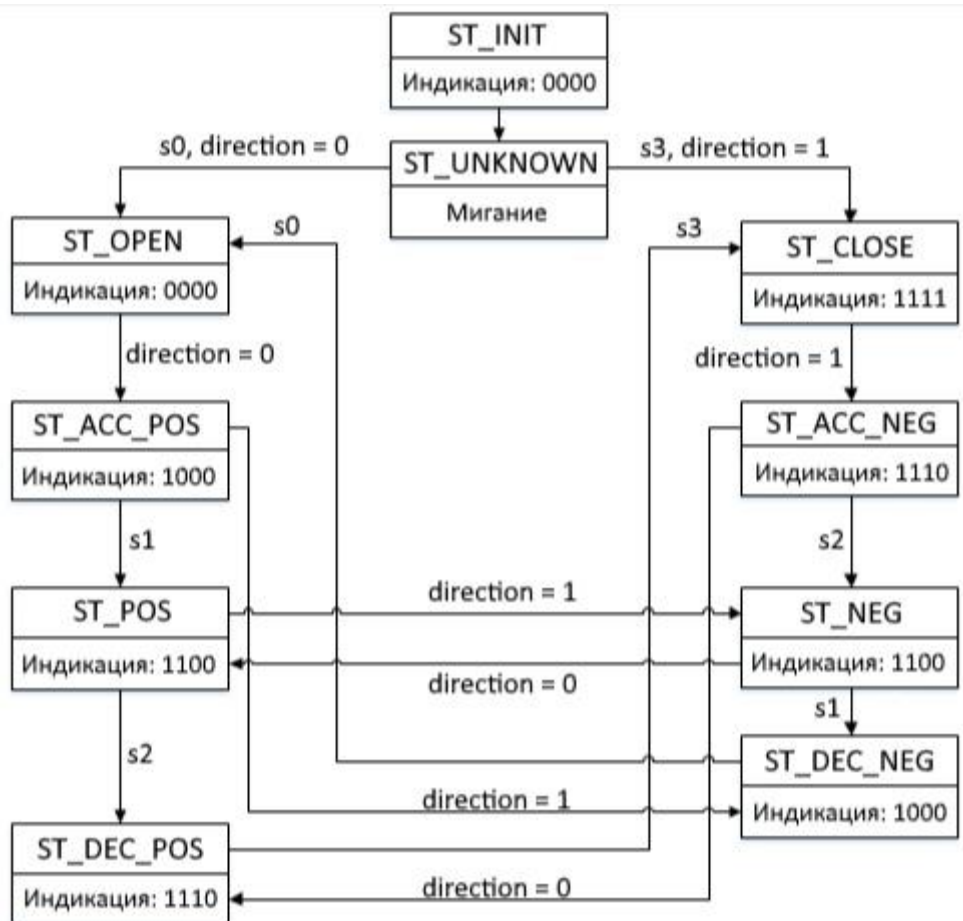


Рисунок 7. Блок-схема состояний ворот и возможных переходах между ними

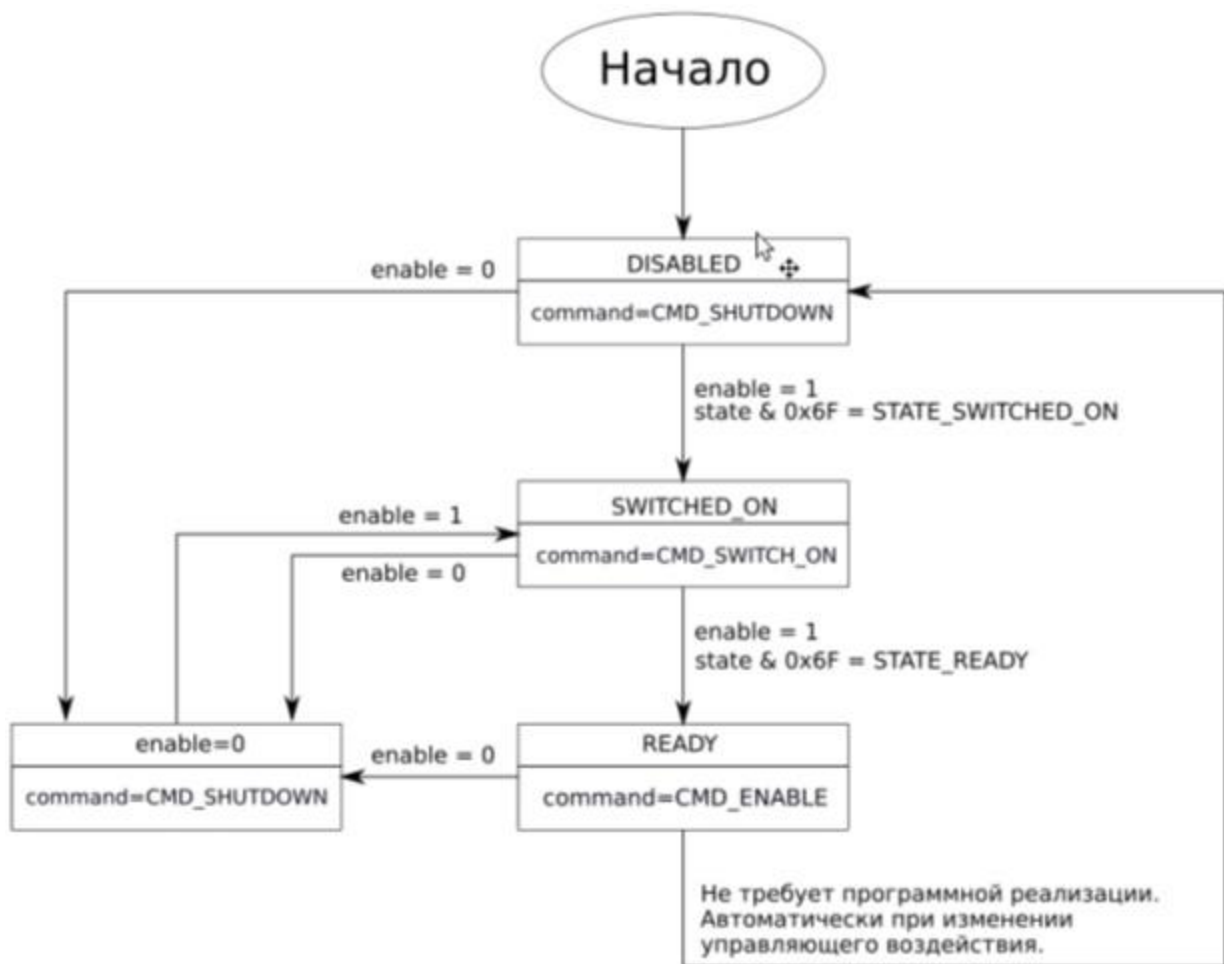


Рисунок 8. Блок-схема управления частотным преобразователем

Как результат получим работающую программу управления асинхронным двигателем. На графике 9 стоит обратить внимание на то как изменение скорости влияет на мерцание светодиодов на панели.

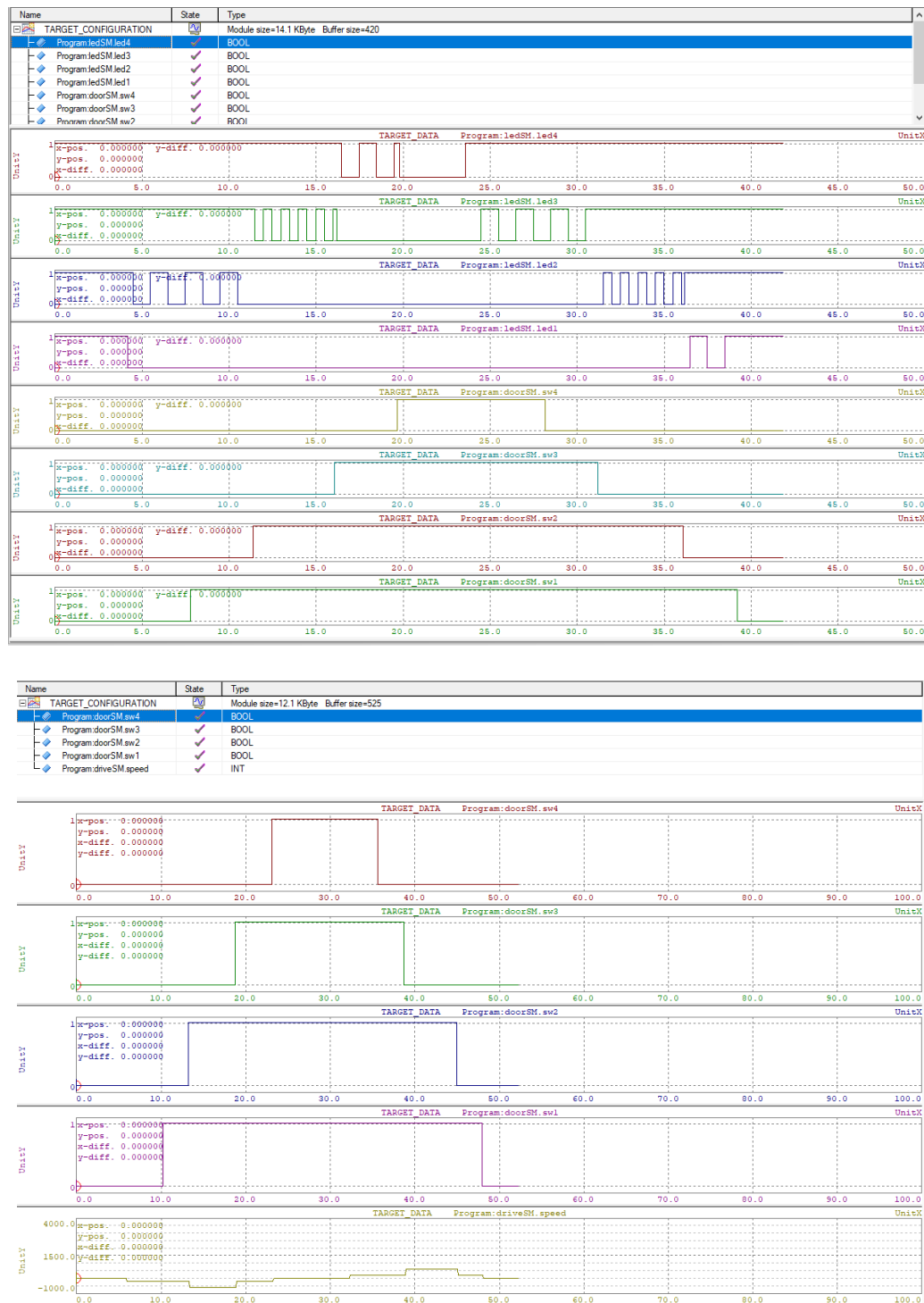


Рисунок 9. Показания по скорости двигателя и состояние датчиков sw

Вывод

В результате выполнения лабораторной работы были получены навыки моделирования систем автоматического управления в среде разработки Automation Studio B&R. На основе смоделированной модели системы управления защитной дверью были получены графики скорости двигателя, состояний светодиодов и датчиков. Был получен навык в моделировании систем управления и работы с лабораторным стендом.

Приложение А

Листинг программного кода main (cyclic, init, exit)

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

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

void _INIT ProgramInit(void)
{
driveSM.enable=1;
}

void _CYCLIC ProgramCyclic(void)
{
DoorStateMachine(&doorSM);
driveSM.speed = doorSM.speed;
DriveStateMachine(&driveSM);
ledSM.state = doorSM.state;
LedStateMachine(&ledSM);
}

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

Приложение Б

```
#include <bur/plctypes.h>
#ifdef __cplusplus
extern "C"
{
#endif
#include "DriveLib.h"
#ifdef __cplusplus
};
#endif
/* TODO: Add your comment here */
void DriveStateMachine(struct DriveStateMachine* inst)
{
/*TODO: Add your code here*/
UINT mask = inst->state & 0x6f;
if(!inst->enable)
{
    inst->command = CMD_SHUTDOWN;
}
else
{
    switch (mask)
    {
        case STATE_DISABLED:
            inst->command = CMD_SHUTDOWN;
            break;

        case STATE_READY:
            inst->command = CMD_ENABLED;
            break;

        case STATE_SWITCHED_ON:
            inst->command = CMD_SWITCHED_ON;
            break;
    }
}
}
```

Приложение В

```
#include <bur/plctypes.h>
#ifdef __cplusplus
extern "C"
{
#endif
#include "DriveLib.h"
#ifdef __cplusplus
};
#endif
/* TODO: Add your comment here */
void DoorStateMachine(struct DoorStateMachine* inst)
{
/*TODO: Add your code here*/
switch(inst->state)
{
    case ST_INIT:
    {
        if(inst->sw1==0 && inst->sw2==0 && inst->sw3==0 && inst->sw4==0)
            inst->state=ST_UNKNOWN;
        break;
    }
    case ST_UNKNOWN:
    {
        if(inst->direction==0 && inst->sw1!=inst->sw1)
            inst->state=ST_CLOSE;
        if(inst->direction==1 && inst->sw1!=inst->sw1)
            inst->state=ST_NEG;
        if(inst->direction==1 && inst->sw2!=inst->sw2)
            inst->state=ST_ACC_NEG;
        if(inst->direction==1 && inst->sw3!=inst->sw3)
            inst->state=ST_DEC_NEG;
        if(inst->direction==1 && inst->sw4!=inst->sw4)
            inst->state=ST_OPEN;
        if(inst->direction==0 && inst->sw4!=inst->sw4)
            inst->state=ST_POS;
        if(inst->direction==0 && inst->sw3!=inst->sw3)
            inst->state=ST_ACC_POS;
        if(inst->direction==0 && inst->sw2!=inst->sw2)
            inst->state=ST_DEC_POS;
    }
}
```

```

        break;
    }
case ST_CLOSE:
{
    inst->speed = 0;
    if(inst->direction==1)
        inst->state=ST_NEG;
    break;
}
case ST_NEG:
{
    inst->speed = -200;
    if(inst->direction==1 && inst->sw2!=inst-
>sw2)
        inst->state=ST_ACC_NEG;
    if(inst->direction==0)
        inst->state=ST_DEC_POS;
    break;
}
case ST_ACC_NEG:
{
    inst->speed = -600;
    if(inst->direction==1 && inst->sw3!=inst-
>sw3)
        inst->state=ST_DEC_NEG;
    if(inst->direction==0)
        inst->state=ST_ACC_POS;
    break;
}
case ST_DEC_NEG:
{
    inst->speed = -200;
    if(inst->direction==1 && inst->sw4!=inst-
>sw4)
        inst->state=ST_OPEN;
    if(inst->direction==0)
        inst->state=ST_POS;
    break;
}
case ST_OPEN:
{
    inst->speed = 0;
    if(inst->direction==0)
        inst->state=ST_POS;
    break;
}
case ST_POS:
{
    inst->speed = 200;

```

```

        if(inst->direction==0 && inst->sw3!=inst-
>sw3)
            inst->state=ST_ACC_POS;
            if(inst->direction==1)
                inst->state=ST_DEC_NEG;
            break;
        }
    case ST_ACC_POS:
        {
            inst->speed = 600;
            if(inst->direction==0 && inst->sw2!=inst-
>sw2)
                inst->state=ST_DEC_POS;
                if(inst->direction==1)
                    inst->state=ST_ACC_NEG;
                break;
            }
        case ST_DEC_POS:
            {
                inst->speed = 200;
                if(inst->direction==0 && inst->sw1!=inst-
>sw1)
                    inst->state=ST_CLOSE;
                    if(inst->direction==1)
                        inst->state=ST_NEG;

                break;
            }
        }
    inst->sw1=inst->sw1;
    inst->sw2=inst->sw2;
    inst->sw3=inst->sw3;
    inst->sw4=inst->sw4;
}

```


Приложение Г

```
#include <bur/plctypes.h>
#ifdef __cplusplus
    extern "C"
    {
#endif
    #include "DriveLib.h"
#ifdef __cplusplus
    };
#endif
/* TODO: Add your comment here */
void LedStateMachine(struct LedStateMachine* inst)
{
    switch(inst->state)
    {
        case ST_INIT:
        {
            break;
        }

        case ST_UNKNOWN:
        {
            if(inst->timer%5==4)
            {
                inst->led1=!inst->led1;
                inst->led2=!inst->led2;
                inst->led3=!inst->led3;
                inst->led4=!inst->led4;
            }
            break;
        }

        case ST_CLOSE:
        {
            inst->led1=1;
            inst->led2=1;
            inst->led3=1;
            inst->led4=1;
            break;
        }

        case ST_NEG:
        {
            inst->led1=0;
            if(inst->timer%10==9)
                inst->led2=!inst->led2;
            inst->led3=1;
            inst->led4=1;
            break;
        }
    }
}
```

```

    }
case ST_ACC_NEG:
{
    inst->led1=0;
    inst->led2=0;
    if(inst->timer%5==4)
        inst->led3=!inst->led3;
    inst->led4=1;
    break;
}
case ST_DEC_NEG:
{
    inst->led1=0;
    inst->led2=0;
    inst->led3=0;
    if(inst->timer%10==9)
        inst->led4=!inst->led4;
    break;
}
case ST_OPEN:
{
    inst->led1=0;
    inst->led2=0;
    inst->led3=0;
    inst->led4=0;
    break;
}
case ST_POS:
{
    inst->led1=0;
    inst->led2=0;
    if(inst->timer%10==9)
        inst->led3=!inst->led3;
    inst->led4=1;
    break;
}
case ST_ACC_POS:
{
    inst->led1=0;
    if(inst->timer%5==4)
        inst->led2=!inst->led2;
    inst->led3=1;
    inst->led4=1;
    break;
}
case ST_DEC_POS:
{
    if(inst->timer%10==9)
        inst->led1=!inst->led1;
    inst->led2=1;
    inst->led3=1;

```

```
inst->led4=1;
break;
    }
}
inst->timer++;
}
```