

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА - Российский технологический университет»**

**РТУ МИРЭА**

Институт искусственного интеллекта

Кафедра проблем управления

**Лабораторная работа №2**

по дисциплине «Программное обеспечение мехатронных и робототехнических систем»

|  |  |
| --- | --- |
| **Выполнил студент группы**  КРБО-01-20 | Голубятников Н.С.  Назаренко А.В. |
| **Преподаватель:** | Морозов А.А. |
|  |  |
|  |  |
| Работа представлена к защите: | «\_\_\_» \_\_\_\_\_\_\_\_\_ 2023 г. |

# Отладка программного обеспечения роботехнических систем с использованием виртуального моделирования

**Программное обеспечение системы управления  
мехатронного модуля управления защитной дверью  
Цель работы**: получение навыков построения программного обеспечения  
промышленных систем управления на базе функциональных блоков и конечных  
автоматов.  
**Задание**: разработать программное обеспечение системы автоматического  
управления приводом защитной двери. Схема механизма представлена на  
рисунке 1. 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, небольшую 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.

|  |
| --- |
|  |
| Рисунок 1. Устройство защитной двери |

# Ход работы

В среде моделирования AutomationStudio была программно воссоздана система управления дверью, представленная на рисунке 1.

В базовую конфигурацию оборудования были добавлены следующие модули (рисунок 2, рисунок 3):

* X20BC0083
* X20PS9400
* X20SM1436
* X20SM1436a
* X20MM4456
* X20DI9731
* X20DO9322
* X20AT4222
* X20SI4100
* X20SO4110
* X20BT9100

|  |
| --- |
|  |
| Рисунок 2. Конфигурация оборудования |

|  |
| --- |
|  |
| Рисунок 3. Конфигурация оборудования |

Для моделирования работы асинхронного двигателя в конфигурации частотного преобразователя были внесены изменения в соответствии с таблицей 1 (рисунок 4).

|  |  |  |
| --- | --- | --- |
| Таблица 1. Параметры асинхронного двигателя | | |
| Наименование параметра | Название в конфигурационной таблице [eng.] | Заданное значение |
| Номинальное питающее напряжение двигателя [B] | 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 А] | 10 |

|  |
| --- |
| C:\Users\One\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{051E6767-97AB-4431-9731-65A5C10F15D8}\{3466656B-2135-4601-AB8D-621F8CB846E2}\ResourceMap\{955C57BA-39E3-465D-A23D-5C98D3FAF82B} |
| Рисунок 4. Вкладка конфигурации частотного преобразователя |

Далее в проекте были созданы необходимые объекты ANSI C Program и ANSI C Library « DriveLib». В библиотеке «DriveLib» создать функциональные блоки: «DoorStateMachine», «DriveStateMachine» и «LedStateMachine» (рисунок 5).

|  |
| --- |
|  |
| Рисунок 5. Структура проекта |

«DriveStateMashine» – функциональный блок управления

частотным преобразователем. Параметры функционального блока двигателя представлены в таблице 2. Код функционального блока представлен в приложении А.

Таблица 2. Параметры функционального блока двигателя.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

«DoorStateMashine» - функциональный блок, в котором была реализована  
основная логика программы – задание направления вращения и скорости  
движения воротами в зависимости от их положения и требуемого направления  
вращения. Для упрощения описания состояний ворот во вкладке «DrivLib -> Types.typ» был создан новый тип данных: перечисление «States». Сами состояния, которые необходимо добавить приведены в таблице 3, а их связь представлена в функциональной схеме (рис. 6). Параметры функционального блока управления воротами представлены в таблице 4. Исходный код функционального блока представлен в приложении Б.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 3. Состояния ворот.   |  |  | | --- | --- | | Состояние | Описание | | ST\_INIT | Инициализация параметров и ожидание включения частотного преобразователя | | ST\_UNKNOWN | Ворота в неизвестном положении | | ST\_OPENED | Ворота открыты | | ST\_CLOSED | Ворота закрыты | | ST\_ACC\_OP | Ускорение ворот в сторону открытия | | ST\_ACC\_CL | Ускорение ворот в сторону закрытия | | ST\_OP | Движение к открытию | | ST\_OP | Движение к закрытию | | ST\_SLOW\_OP | Замедление ворот в сторону открытия | | ST\_SLOW\_CL | Замедление ворот в сторону закрытия | |

Таблица 4. Структура функционального блока управления воротами.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

|  |
| --- |
|  |
| Рисунок 6. Функциональная схема переходов состояний ворот |

«LedStateMashine» – машина состояний обработки светодиодных индикаторов. Параметры функционального блока управления воротами представлены в таблице 5. Исходный код функционального блока представлен в приложении В.

Таблица 5. Структура функционального блока управления воротами.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | BOOL | Таймер |

На рисунке 7 представлены параметры всех функциональных блоков.

|  |
| --- |
|  |
| Рисунок 7. Параметры функциональных блоков |

Далее была проведена операция мэпинга индикаторов, датчиков положения, частотного преобразователя. Соответствие входов частотного преобразователя (модуля 8164xxxxxx.00x-1) и переменных приведены в  
таблице 6. В меню Physical View был выбран частотный преобразователь и в контекстном меню выбран пункт «Configuration» (рис. 8) и во вкладке «I/O Mapping» изменены вкладки ETAD, ETI, CMOD, CMI из положения unpacked в packed; далее в меню «I/O Mapping» частотного преобразователя были изменены параметры аналогично рисунку 9; произвести мэпинг переменных согласно таблице 6:

|  |
| --- |
|  |
| Рисунок 8. Изменение вида представления регистров. |
|  |
| Рисунок 9. Мэпинг переменных для частотного преобразователя. |

Таблица 6. Мэпинг переменных для частотного преобразователя.

|  |  |  |
| --- | --- | --- |
| Наименование | Переменная | Описание |
| ETAD\_Input | drive\_sm.state | Текущее состояние частотного преобразователя |
| CMDD\_Output | drive\_sm.command | Команды на управление |
| LFRD\_Output | drive\_sm.speed | Заданное значение скорости |

Для мэпинга индикаторов для модуля X20DO9322 (DO - Digital Output),  
соответствия каналов соответственно будут (таблица 7):

Таблица 8. Мэппинг переменных для индикаторов

|  |  |
| --- | --- |
| Имя канала | Используемая переменная |
| DigitalOutput02 | led\_sm.led1 |
| DigitalOutput03 | led\_sm.led2 |
| DigitalOutput04 | led\_sm.led3 |
| DigitalOutput05 | led\_sm.led4 |

Для мэпинга датчиков для модуля X20DI9371 (DI - Digital Input),  
соответствия каналов соответственно будут (таблица 8):

Таблица 9. Мэпинг переменных для датчиков положения

|  |  |
| --- | --- |
| Имя канала | Используемая переменная |
| DigitalInput01 | door\_sm.direction |
| DigitalInput02 | door\_sm.sw1 |
| DigitalInput03 | door\_sm.sw2 |
| DigitalInput04 | door\_sm.sw3 |
| DigitalInput05 | door\_sm.sw4 |

Была написана программа управления частотным преобразователем. Блок-схема управления ЧП представлена на рисунке 10. Листинг исходного кода представлен в приложении Г.

|  |
| --- |
|  |
| Рисунок 10. Блок-схема управления частотным преобразователем |

Кольцевые датчики срабатывают одноразовым нажатием. В полностью открытом состоянии двери светодиоды не горят. В полностью закрытой двери все светодиоды зажжены. Светодиод пройденного кольцевого датчика зажжен или потух в зависимости от направления движения двери. Движение двери в сторону одного из кольцевых датчиков обозначено морганием (изменением состояния) соответствующего светодиода.

Далее была проверена работа программы. На рисунках представлены состояния переменных при моделировании Trace. Рисунок 11 соответствует изменению состояния датчиков, световых индикаторов и скорости при переходе двери из полностью закрытого состояния в полностью открытое.

|  |
| --- |
|  |
| Рисунок 11. Trace состояния переменных при открытии двери |

Рисунок 12 соответствует изменению состояния датчиков, световых индикаторов и скорости из полностью открытого состояния в полностью закрытое.

|  |
| --- |
|  |
| Рисунок 12. Trace состояния переменных при закрытии двери |

Рисунок 13 соответствует изменению состояния датчиков, световых индикаторов и скорости при переходе двери из полностью открытого состояния в закрытое до 3-его кольцевого датчика, а затем изменение направления движения до полного открытия двери.

|  |
| --- |
|  |
| Рисунок 13. Trace состояния переменных при закрытии-открытии двери |

# Вывод

В ходе данной лабораторной работы были получены навыки моделирования объекта управления в промышленных системах автоматического управления. С помощью инструмента типа данных перечисление и условного оператора switch-case были налажены переходы состояния положения двери.

# Приложение А

Листинг исходного код функционального блока DriveStateMachine.c.

#include <bur/plctypes.h>  
#ifdef \_\_cplusplus  
 extern "C"  
 {  
#endif  
 #include "DrivLib.h"  
#ifdef \_\_cplusplus  
 };  
#endif  
/\* TODO: Add your comment here \*/  
void DriveStateMachine(struct DriveStateMachine\* inst)  
{  
 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 Б

Листинг исходного код функционального блока DoorStateMachine.c.

#include <bur/plctypes.h>  
#ifdef \_\_cplusplus  
 extern "C"  
 {  
#endif  
 #include "DrivLib.h"  
#ifdef \_\_cplusplus  
 };  
#endif  
/\* TODO: Add your comment here \*/  
   
   
 // direction = 0 -> door is opening  
 // direction = 1 -> door is closing  
   
 // sensors are placed as follows:  
 // CLOSED s3 -- s2 -- s1 -- s0 OPENED  
 // sw1 = s0 -> sw1 == opened  
 // sw2 = s1  
 // sw3 = s2  
 // sw4 = s3 -> sw4 == closed  
  
void DoorStateMachine(struct DoorStateMachine\* inst)  
{  
 switch(inst->state)**{** // INIT and UNKNOWN  
 case ST\_INIT:  
 inst->state = ST\_UNKNOWN;  
 // dodelat'  
 case ST\_UNKNOWN:  
 inst->speed = 200;  
 if(inst->sw4)  
 inst->state = ST\_OP;  
 else if(inst->sw3)  
 inst->state = ST\_ACC\_OP;  
 else if(inst->sw2)  
 inst->state = ST\_SLOW\_OP;  
 else if(inst->sw1)  
 inst->state = ST\_OPENED;  
 break;  
   
 // fully closed/opened states  
 case ST\_CLOSED:  
 inst->speed = 0;  
 if(inst->direction == 0) // if in closed state the dir changed to opening  
 inst->state = ST\_OP; // normal opening  
 break;  
 case ST\_OPENED:  
 inst->speed = 0;  
 if(inst->direction == 1) // if in open state the dir is to close  
 inst->state = ST\_CL; // normal closing  
 break;  
   
 // dir = 1  
 // closing  
 case ST\_CL:  
 inst->speed = -200;  
 if(inst->direction == 0)  
 inst->state = ST\_SLOW\_OP;  
 else if(inst->sw2)  
 inst->state = ST\_ACC\_CL;  
 break;  
 case ST\_ACC\_CL:  
 inst->speed = -700;  
 if(inst->direction == 0)   
 inst->state = ST\_ACC\_OP;  
 else if(inst->sw3)  
 inst->state = ST\_SLOW\_CL;  
 break;  
 case ST\_SLOW\_CL:  
 inst->speed = -200;  
 if(inst->direction == 0)   
 inst->state = ST\_OP;  
 else if(inst->sw4)  
 inst->state = ST\_CLOSED;  
 break;  
   
 // dir == 0  
 // opening  
 case ST\_OP:  
 inst->speed = 200;  
 if(inst->direction == 1)  
 inst->state = ST\_SLOW\_CL;  
 else if(inst->sw3)  
 inst->state = ST\_ACC\_OP;  
 break;  
 case ST\_ACC\_OP:  
 inst->speed = 700;  
 if(inst->direction == 1)  
 inst->state = ST\_ACC\_CL;  
 else if(inst->sw2)  
 inst->state = ST\_SLOW\_OP;  
 break;  
 case ST\_SLOW\_OP:  
 inst->speed = 200;  
 if(inst->direction == 1)  
 inst->state = ST\_CL;  
 else if(inst->sw1)  
 inst->state = ST\_OPENED;  
 break;  
 **}**   
}

# Приложение В

Листинг исходного код функционального блока LedStateMachine.c.

#include <bur/plctypes.h>  
#ifdef \_\_cplusplus  
 extern "C"  
 {  
#endif  
 #include "DrivLib.h"  
#ifdef \_\_cplusplus  
 };  
#endif  
/\* TODO: Add your comment here \*/  
  
 // direction = 0 -> door is opening  
 // direction = 1 -> door is closing  
  
 // sensors are placed as follows:  
 // OPENED s0 -- s1 -- s2 -- s3 CLOSED  
 // sw1 = s0 -> sw1 == OPENED  
 // sw2 = s1  
 // sw3 = s2  
 // sw4 = s3 -> sw4 == CLOSED  
  
 // lights defenitions:  
 // \* - light is blinking (the door is going to that sensor)  
 // 0 - lights off  
 // 1 - lights on  
void LedStateMachine(struct LedStateMachine\* inst)  
{  
 inst->timer +=1;  
 switch(inst->state)**{** // INIT and UNKNOWN  
 case ST\_INIT:  
 inst->led1 = 0;  
 inst->led2 = 0;  
 inst->led3 = 0;  
 inst->led4 = 0;  
 break;  
 case ST\_UNKNOWN: // state UNKNOWN -> \*\*\*\*  
 if(inst->timer % 25 == 0){  
 inst->led1 = !inst->led1;  
 inst->led2 = inst->led1;  
 inst->led3 = inst->led1;  
 inst->led4 = inst->led1;  
 }  
 break;  
   
 // from closed to open  
 case ST\_CLOSED: // full closed  
 // 1111  
 inst->led1 = 1;  
 inst->led2 = 1;  
 inst->led3 = 1;  
 inst->led4 = 1;  
 break;  
 case ST\_OP: // from full closed stating to open  
 // 11\*0  
 inst->led1 = 1;  
 inst->led2 = 1;  
 if(inst->timer % 15 == 0) inst->led3 = !inst->led3;  
 inst->led4 = 0;  
 break;  
 case ST\_ACC\_OP: // middle  
 // 1\*00  
 inst->led1 = 1;  
 if(inst->timer % 15 == 0) inst->led2 = !inst->led2;  
 inst->led3 = 0;  
 inst->led4 = 0;  
 break;  
 case ST\_SLOW\_OP: // almost opened  
 // \*000  
 if(inst->timer % 15 == 0) inst->led1 = !inst->led1;  
 inst->led2 = 0;  
 inst->led3 = 0;  
 inst->led4 = 0;  
 break;  
 case ST\_OPENED: // full opened   
 // 0000  
 inst->led1 = 0;  
 inst->led2 = 0;  
 inst->led3 = 0;  
 inst->led4 = 0;  
 break;  
 // from opened to close  
 case ST\_CL: // 1\*00  
 inst->led1 = 1;  
 if(inst->timer % 15 == 0) inst->led2 = !inst->led2;  
 inst->led3 = 0;  
 inst->led4 = 0;  
 break;  
 case ST\_ACC\_CL: // 11\*0  
 inst->led1 = 1;  
 inst->led2 = 1;  
 if(inst->timer % 15 == 0) inst->led3 = !inst->led3;  
 inst->led4 = 0;  
 break;  
 case ST\_SLOW\_CL: // 111\*  
 inst->led1 = 1;  
 inst->led2 = 1;  
 inst->led3 = 1;  
 if(inst->timer % 15 == 0) inst->led4 = !inst->led4;  
 break;  
 **}**   
}

# Приложение Г

Листинг исходного код блока программы Program.

#include "DrivLib.h"  
#include <bur/plctypes.h>  
  
#ifdef \_DEFAULT\_INCLUDES  
 #include <AsDefault.h>  
#endif  
  
  
void \_INIT ProgramInit(void)  
{  
 drive\_sm.enable = 1;  
 door\_sm.state = ST\_INIT;  
}  
  
void \_CYCLIC ProgramCyclic(void)  
{  
 led\_sm.state = door\_sm.state;  
 DriveStateMachine(&drive\_sm);  
 speed = door\_sm.speed;  
 DoorStateMachine(&door\_sm);  
 LedStateMachine(&led\_sm);  
}  
  
void \_EXIT ProgramExit**(**void**)**{  
 drive\_sm.enable = 0;  
 door\_sm.state = ST\_INIT;  
}