Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ В.Г.ШУХОВА

(БГТУ им. В.Г. Шухова)

Кафедра электроэнергетики и автоматики

**КУРСОВОЙ ПРОЕКТ**

По дисциплине “ Микроконтроллеры в электроприводе”

**«Проектирование системы управления регулируемым электроприводом с помощью частотного преобразователя Delta VFD-B»**

Вариант 9

Выполнил:

студент 4 курса

гр. ЭА-191 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.П. Танчук

Проверил:

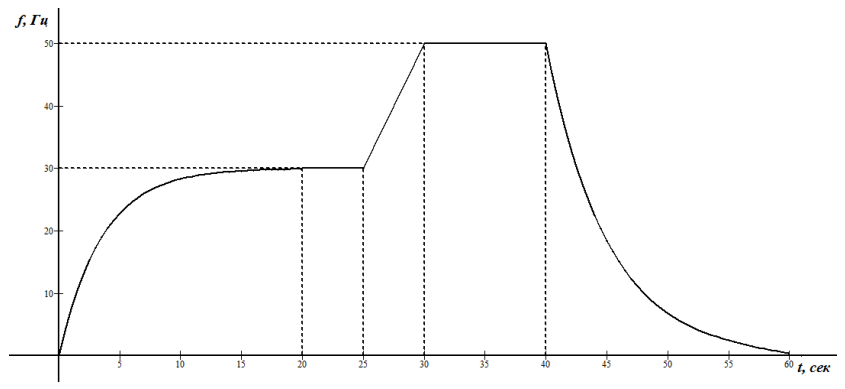
доцент, кандидат техн. наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Солдатенков

Белгород 2023

**Задание** (здесь пока задание неофициальное, потом переделаю, когда Солдатенков даст другое).

Цель задания: приобретение навыков проектирования систем управления частотных электроприводов на базе преобразователей частоты с применением микроконтроллера I7188EX; монтажа, подключения и настройки микроконтроллеров, частотных преобразователей и других интеллектуальных устройств; разработки на языке С++ прикладного программного обеспечения для управления частотно-регулируемым электроприводом; работы с технической документацией, руководствами по подключению, настройке и программированию различных микропроцессорных устройств.

Структура работы. Расчетно-графическое задание предусматривает реализацию специально заданного алгоритма движения (вращения) асинхронного трехфазного электрического двигателя с короткозамкнутым ротором на базе преобразователя частоты с помощью контроллера I7188EX. Основной алгоритм задается в виде тахограммы движения вала двигателя. Номер варианта определяется преподавателем.



В рамках выполнения РГЗ требуется:

₋ Разработать схему подключения преобразователя частоты, управляющего трёхфазным асинхронным двигателем, к контролеру I7188EX. В зависимости от разработанной схемы и используемого оборудования выбрать преобразователи интерфейсов. Дать спецификацию с краткой технической характеристикой преобразователя частоты, контроллера I7188EX и применяемых преобразователей интерфейсов. Привести краткие теоретические сведения о протоколе Modbus и его используемых функциях.

₋ Разработать алгоритм работы программы, который будет реализовывать движение вала в соответствии с заданной тахограммой.

Разработать интерфейс класса преобразователя частоты на языке С++. Данный интерфейс должен содержать необходимые методы, свойства и события для управления преобразователем частоты по протоколу Modbus.

₋ Разработать реализацию необходимых методов, свойств и событий для управления двигателем согласно заданной тахограмме движения вала по протоколу Modbus. При этом должна быть предусмотрена обработка различного типа ошибок: обрыв связи, неверные настройки коммуникационного порта, неверный запрос или ответ, ошибки преобразователя частоты в Modbus-сообщении, несовпадение контрольной суммы при передаче данных.

₋ Разработать пользовательский интерфейс, предусматривающий возможность ввода коммуникационных параметров связи с клавиатуры, а также вывод на экран возникающих ошибок и текущих параметров преобразователя частоты для контроля выполнения разработанного алгоритма в режиме реального времени.

₋ Выполнить реализацию для выполнения разработанных функций и алгоритмов в виде приложения для контроллера I7188EX.

Оформление отчета по РГЗ. Текст отчёта должен быть представлен в машинописном виде на бумаге формата А4. При наборе текста необходимо использовать 12-14 размер шрифта «Timеs New Roman», одинарный или полуторный интервал, выравнивание абзацев по ширине. Все необходимые схемы подключений должны приводиться в отчёте. Отчёт также должен содержать все необходимые сведения об использованном оборудовании и протоколе Modbus, снимки экрана, демонстрирующие результаты работы, листинги исходных файлов программы и блок-схемы алгоритмов работы программы.

В процессе выполнения индивидуального домашнего задания осуществляется контактная работа обучающегося с преподавателем. Консультации проводятся в аудитории и/или посредствам электронной информационно-образовательной среды университета.

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 7](#_Toc127147680)

[ВВЕДЕНИЕ 9](#_Toc127147681)

[1 Исходные данные для проектирования 11](#_Toc127147682)

[2 Управление ПЧ с помощью компьютера на Windows 15](#_Toc127147683)

[2.1 Работа с функциями времени 20](#_Toc127147684)

[2.2 Разработка драйвера последовательного COM порта 20](#_Toc127147685)

[2.3 Разработка драйвера ModbusRTU 20](#_Toc127147686)

[2.4 Разработка драйвера ПЧ 21](#_Toc127147687)

[2.5 Основная программа управления 22](#_Toc127147688)

[3 Управление ПЧ с помощью промышленного контроллера ICP DAS I-7188EX 23](#_Toc127147689)

[3.1 Функции времени на I-7188EX 23](#_Toc127147690)

[3.2 Разработка драйвера UART интерфейса 23](#_Toc127147691)

[3.3 Перенос основной программы 23](#_Toc127147692)

[3.4 Разработка интерфейса взаимодействия с оператором 23](#_Toc127147693)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc127147694)

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

АДКЗ – асинхронный двигатель с короткозамкнутым ротором;

ПЧ – преобразователь частоты;

MSVS – Microsoft Visual Studio 2022;

ВВЕДЕНИЕ

Асинхронный двигатель с короткозамкнутым ротором (АДКЗ) являясь самым распространенным типом электрической машины с развитием математической теории машин переменного тока и усовершенствованием IGBT транзисторов стал управляемым приводом. При частотном регулировании скорости вращения АДКЗ составляем конкуренцию электроприводу постоянного тока по диапазону регулирования, плавности и экономичности. А благодаря простоте конструкции и бесконтактности данного типа двигателя снижается его стоимость и появляется возможность использования во взрывоопасных средах.

Преобразователи частоты предназначены для плавной регулировки скорости и момента вращения вала двигателя путем изменения частоты и напряжения переменного тока при изменении какого-либо технологического параметра.

Использование современных средств управления, включая микропроцессорные, позволяет совершенствовать алгоритмы управления системой «преобразователь-двигатель» и создавать надежные системы регулируемых электроприводов. Высокие динамические и эксплуатационные характеристики электроприводов обеспечиваются за счет разработки и использования новых алгоритмов управления с реализацией на современной микроэлектронной базе – сигнальных процессорах (DSP). Точность цифровой регулировки и наличие промышленных сетевых интерфейсов позволяет легко интегрировать ПЧ в систему автоматизации. Модернизированный таким образом частотно-регулируемый электропривод становится звеном архитектуры АСУ ТП производства.

Целью курсовой работы является практическое изучение современных принципов разработки систем управления регулируемым электроприводом в составе АСУ ТП производства с помощью преобразователей частоты и микропроцессорных систем. В курсовой работе должны быть выполнены следующие задачи:

- реализация алгоритма управления для Windows;

- реализация алгоритма на промышленном контроллере I7188EX;

- реализация пользовательского интерфейса.

# Исходные данные для проектирования

Система управления состоит из асинхронного электродвигателя с короткозамкнутым ротором, преобразователя частоты, промышленного контроллера, преобразователей интерфейсов, персонального компьютера. Приведем краткое описание и технические характеристики каждого из элементов системы.

**Электродвигатель A0-90L-6**

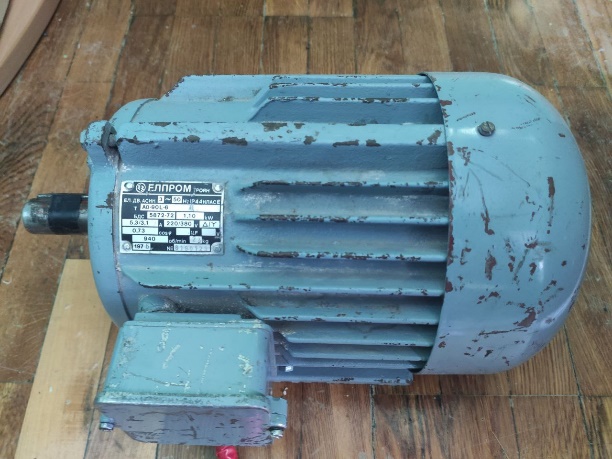


Рисунок . – Внешний вид двигателя A0-90L-6

Тип: АДКЗ.

Название: A0-90L-6.

Производитель: ЕЛПРОМ троян.

Питание 3Ф~50Гц.

Мощность: 1,1 кВт.

Соединение: Δ/Y.

Напряжение: 220/380 В.

Ток: 5,3/3,1 А.

Коэффициент мощности: 0,73.

Частота вращения: 940 об/мин.

Степень защиты: IP44.

Класс изоляции: E.

**Преобразователь частоты Delta Electronics VFD-015B21A**

Преобразователи частоты (ПЧ) компании Delta Electronics, Inc. серии VFD-В предназначены для управления скоростью вращения трехфазных асинхронных электродвигателей с короткозамкнутым ротором мощностью от 0,4 до 75 кВт в составе такого оборудования как, насосы, вентиляторы, миксеры, экструдеры, транспортирующие и подъемные механизмы и другого.

Мощность: 1,5 кВт.

Питание: 1Ф~220 В 50 Гц.

Выходная частота: 0,1–400 Гц.

Потребляемый ток: 9,4 А.

Функциональные возможности: настраиваемая характеристика V/f и векторное управление, основной и дополнительный источники задания частоты, 15 предустановленных скоростей, автоматическое пошаговое управление, ПИД-регулятор, автоматическая компенсация момента и скольжения, связь по MODBUS через последовательный интерфейс RS-485, векторное управление в разомкнутом и замкнутом контуре, авто тестирование двигателя, управления группой электродвигателей, съемный пульт управления, защиты от многих аварийных ситуаций.



Рисунок . – Внешний вид ПЧ

**Промышленный контроллер I-7188EX**

Производитель: ICP DAS.

Модель I-7188EX.

Крепление: монтаж на DIN рейку.

Предустановленная операционная система: MiniOS7.

Питание: напряжений 10 В - 30 В и потребление 2 Вт.

Рабочая температура: -25 ~ 75 °C.

Скорость передачи данных: 300 ~ 115200 кБит/сек.

Процессор: AMD 80188-40 или совместимый с тактовой частотой 40 МГц.

Память: 512 кБ (SRAM) + 512 кБ (Flash) + 31Б (NVRAM) с возможностью расширения до 64 Мб путем установки дополнительных SRAM или Flash плат расширения.

Интерфейсы, разъемы и выходы: винтовые клеммы (RS-485, RS-232), RJ-45 (LAN), 10BASE-T, совместимый с NE2000.

Протоколы: TCP/IP (TCP, UDP, IP, ICMP, ARP), Reverse Address Resolution Protocol.



Рисунок . – Внешний вид контроллера I-7188EX

**Преобразователи интерфейсов**

Конвертер Espada USB/RS-485 на базе микросхемы CH340G и MAX485ESA является преобразователем интерфейсов, предназначенным для перехода с интерфейса USB на порт RS-485. Задача конвертера, подключаемого к USB-порту компьютера или ноутбука, состоит во взаимообратном преобразовании сигналов интерфейсов USB и RS-485. Изделие работает в полудуплексном режиме и поддерживает автоматическое переключение с приема на передачу и обеспечивает подключение контроллеров, измерительных устройств, систем сбора данных к компьютеру с использованием порта USB 2.0.



Рисунок . – Внешний вид преобразователя интерфейсов

**Персональный компьютер**

Любой с ОС Windows 64bit.

# Управление ПЧ с помощью компьютера на Windows

Схема управления ПЧ (рисунок 2.1) работает следующим образом. При включении автоматического выключателя напряжение однофазной сети подается на ПЧ. ПЧ подключен к компьютеру с помощью интерфейса RS-485 через конвертер RS-485/USB. К выходным клеммам ПЧ подсоединен трехфазный АДКЗ. С помощью протокола Modbus RTU компьютер устанавливает параметры ПЧ (частота вращения, время разгона, направление движения), отправляет команды на запуск и остановку двигателя, считывает параметры тока и напряжения во время работы.

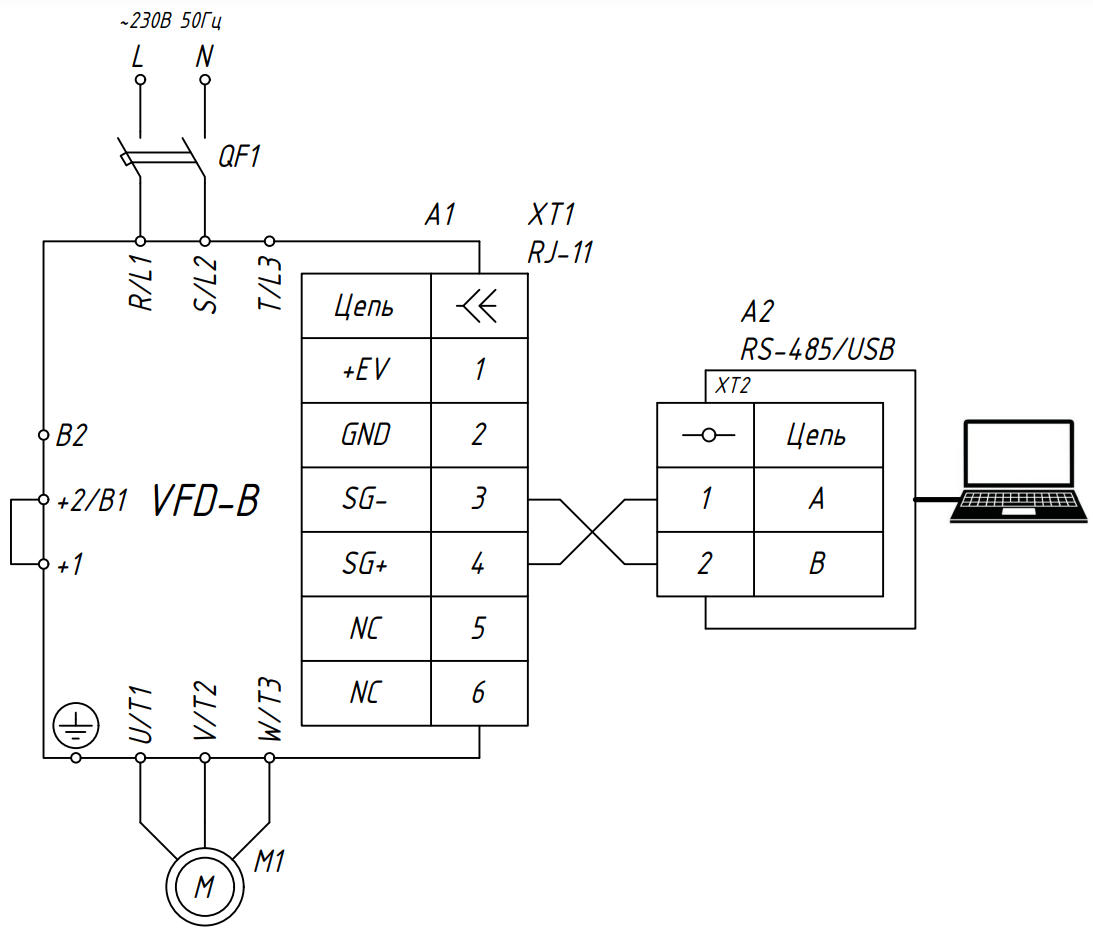


Рисунок . – Принципиальная схема системы управления двигателем с помощью компьютера

Перед созданием программного проекта нужно продумать его архитектуру. Приложение для управления ПЧ с компьютера (рисунок 2.2) работает следующим образом. С помощью графического интерфейса пользователь задает параметры тахограммы вращения двигателя и отправляет команды запуска или остановки. После чего выбранные пользователем параметры и команды записываются в файл конфигурации и запускается программа управления с помощью командной строки. Эта программа считывает требуемые параметры с файла конфигурации, соответственно им задает режим работы двигателя посредством вызова высокоуровневых методов класса VFD. В классе VFD формируются адреса и значения регистров для доступа к требуемым командам и параметрам ПЧ и отправляются в класс ModbusRTU. В классе ModbusRTU формируется строка запроса Modbus и передается на нижний уровень классу COMPort, который с помощью WinAPI отправляет строку запроса в порт.

Полученный от ПЧ ответ расшифровывается в обратном порядке и также записывается в конфигурационный файл, данные из которого считываются программой графического интерфейса и предоставляются пользователю в виде графиков.

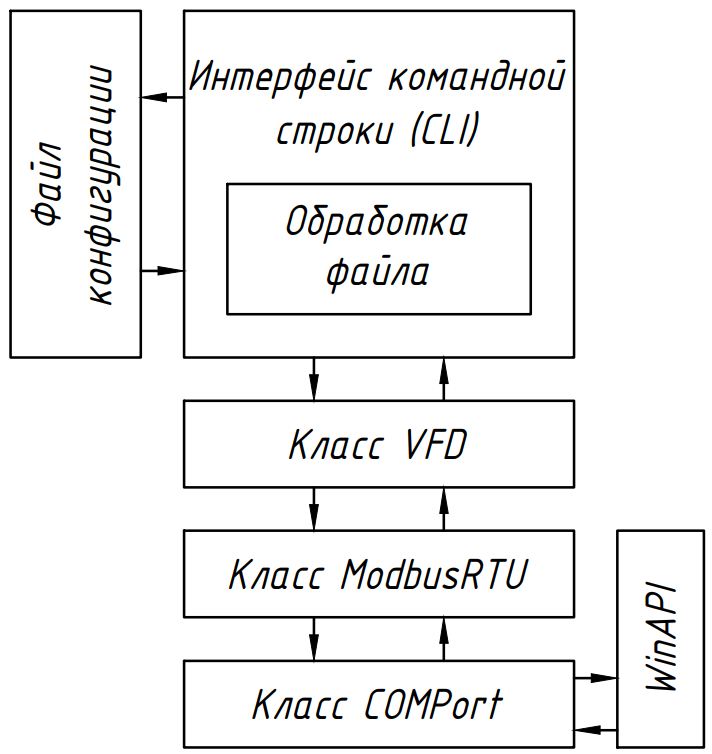


Рисунок . – Архитектура приложения

Для разработки основной программы управления воспользуемся языком **C/C++** и средой разработки Microsoft Visual Studio 2022 (MSVS). Для этого создадим новый проект на основе шаблона *Empty Project*, к которому постепенно будем подключать дополнительные модули.

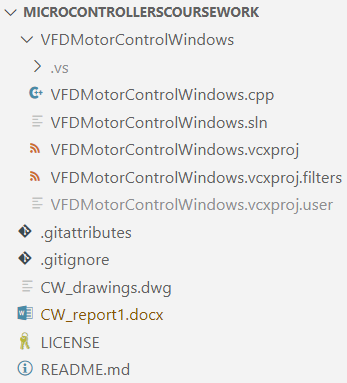
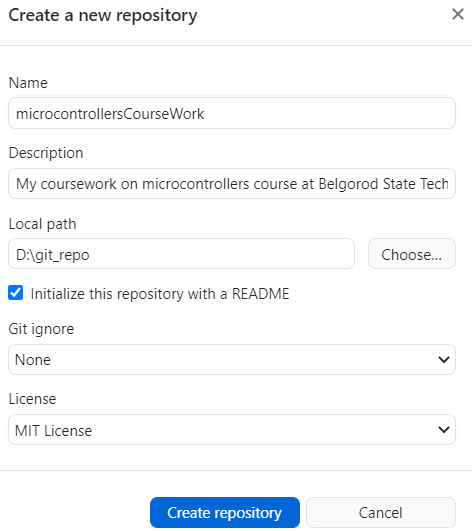
Разработка больших программных проектов требует продуманной организации работы. С этой целью воспользуемся распределенной системой контроля версий Git и Git-хостингом GitHub.

Система контроля версий предназначена для сохранения истории изменений. Как правило, она применяется при разработке программных проектов или набора конфигурационных файлов. История представляет собой снимки проекта, следующие друг за другом в хронологическом порядке. В любой момент можно откатиться к любому состоянию системы в прошлом. Таким образом, можно восстановить поврежденные или случайно удаленные файлы, а также выяснить, кто автор внесенных в код изменений. Также система контроля версий используется при командной работе над проектом [1].

Основная папка проекта Git будет включать в себя как проекты MSVS для Windows и микроконтроллера, так и чертежи, документацию на выбранные элементы системы, описание проекта.

Перед созданием проекта на GitHub зарегистрируемся на сайте <https://github.com/>, установим приложение GitHub Desktop (<https://desktop.github.com/>) и в приложении введем учетные данные от аккаунта GitHub. Git ведет историю изменений всех файлов в выбранной папке. Операционная система Windows и прикладные приложения создают временные файлы при работе, изменения в которых также фиксируются Git. Такое поведение нежелательно, поэтому существует система игнорирования некоторых файлов. В основной папке проекта Git создадим файл *.gitignore*. Для автоматической генерации его содержимого воспользуемся сервисом gitignore.io (<https://www.toptal.com/developers/gitignore/>), на котором зададим ключевые слова: windows, visualstudio, c, c++, microsoftoffice, matlab. В этот же файл добавим шаблоны игнорирования временных файлов AutoCAD (*\*.bak* *\*.dwl* *\*.dwl2*).

Для создания проекта в программе GitHub Desktop выбираем **File→New Repository…**, вводим название и описание проекта, а также путь к папке, в которой будет располагаться репозиторий. В нашем случае уже имеется папка с файлом *.gitignore* и проектом MSVS, поэтому название репозитория должно совпадать с названием этой папки, а в поле расположения репозитория нужно ввести путь к этой папки без указания названия самой папки (рисунок 1.1, *а*).



*a* *б*

Рисунок . – Настройки нового Git репозитория (*а*), и его файловая структура (*б*)

Загружаем созданный репозиторий на GitHub командой **Repository→Push**. Репозиторий будет доступен по адресу <https://github.com/TAN4UKmax/microcontrollersCourseWork>). Структура файлов созданного репозитория показана на рисунке 1.1, *б*. Во время работы над проектом фиксация его состояния осуществляется командой **commit**, а отправка на сервер – **push**.

**Assert**

Проверку работоспособности модулей программы будем выполнять c помощью CppUnitTest. Модульное тестирование (Unit testing) – реализация автоматической проверки корректности отдельных блоков или модулей проекта. Для этого в папке, в которой находится исходный проект, создадим еще один проект с названием VFDMotorControlWindowsTests на основе шаблона Native Unit Test Project. В файл тестирования добавим заголовки классов исходного проекта как показано в листинге ниже.

#include "pch.h"

#include "CppUnitTest.h"

#include "../VFDMotorControlWindows/VFD.h"

#include "../VFDMotorControlWindows/ModbusRTU.h"

#include "../VFDMotorControlWindows/COMPort.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

using namespace std;

namespace VFDMotorControlWindowsTests

{

    TEST\_CLASS(VFDMotorControlWindowsTests)

    {

    public:

        TEST\_METHOD(TestMethod1)

        {

        }

    };

}

Для доступа к определениям классов подключим скомпилированные файлы открывая настройку **Project→Properties→Linker→Input→Additional Dependencies** и с помощью команды Edit… добавляя файлы *COMPort.obj;ModbusRTU.obj;VFD.obj*. Также нужно указать путь к добавленным объектным файлам. В параметре **Project→Properties→Linker→General→Additional Library Directories** добавим строку *..\VFDMotorControlWindows\x64\Debug*. Для запуска тестирования выбираем **Test→Run All Tests**.

Документирование исходного кода будем выполнять в стиле doxygen, так как это является общепринятым в C++. Суть стиля заключается в помещении специально отформатированного комментария в начале каждого файла для краткого описания этого файла (пример для класса COMPort).

/\*\*

 \* @file COMPort.h

 \* @author TAN4UK (tan4ukmak7@gmail.com)

 \* @brief Class for communication with COM port through WinAPI

 \* @version 0.1

 \* @date 2023-02-13

 \*

 \* @copyright Copyright (c) 2023 TAN4UK

 \*

 \*/

Аналогичный комментарий помещается и перед каждым методом.

/\*\*

 \* @brief Writes an array of binary data into COM port

 \*

 \* @param buf - pointer to data buffer

 \* @param length - buffer legth

 \* @return true - if write successful

 \* @return false - if write fails

 \*/

bool Write(char\* buf, unsigned char& length);

Для генерирования документации из подготовленного таким образом исходного кода устанавливаем и запускаем приложение Doxywizard, выбираем расположение проекта и задаем параметры генерации.

/////////////////////////////////////////////////////////////////////////////////////////////////////////////

Тестирование возможно лишнее

## Разработка драйвера последовательного COM порта

Использовать winapi

Вместо СppUnitTest использовать assert

268435456

Функции GetCommProperties и SetupComm

https://en.cppreference.com/w/cpp/error/assert

Поток может использовать функцию PurgeComm для удаления всех символов в выходном или входном буфере устройства

Настройка ресурса

#include <windows.h>

#include <tchar.h>

#include <stdio.h>

void PrintCommState(DCB dcb)

{

// Print some of the DCB structure values

\_tprintf( TEXT("\nBaudRate = %d, ByteSize = %d, Parity = %d, StopBits = %d\n"),

dcb.BaudRate,

dcb.ByteSize,

dcb.Parity,

dcb.StopBits );

}

int \_tmain( int argc, TCHAR \*argv[] )

{

DCB dcb;

HANDLE hCom;

BOOL fSuccess;

TCHAR \*pcCommPort = TEXT("COM1"); // Most systems have a COM1 port

// Open a handle to the specified com port.

hCom = CreateFile( pcCommPort,

GENERIC\_READ | GENERIC\_WRITE,

0, // must be opened with exclusive-access

NULL, // default security attributes

OPEN\_EXISTING, // must use OPEN\_EXISTING

0, // not overlapped I/O

NULL ); // hTemplate must be NULL for comm devices

if (hCom == INVALID\_HANDLE\_VALUE)

{

// Handle the error.

printf ("CreateFile failed with error %d.\n", GetLastError());

return (1);

}

// Initialize the DCB structure.

SecureZeroMemory(&dcb, sizeof(DCB));

dcb.DCBlength = sizeof(DCB);

// Build on the current configuration by first retrieving all current

// settings.

fSuccess = GetCommState(hCom, &dcb);

if (!fSuccess)

{

// Handle the error.

printf ("GetCommState failed with error %d.\n", GetLastError());

return (2);

}

PrintCommState(dcb); // Output to console

// Fill in some DCB values and set the com state:

// 57,600 bps, 8 data bits, no parity, and 1 stop bit.

dcb.BaudRate = CBR\_57600; // baud rate

dcb.ByteSize = 8; // data size, xmit and rcv

dcb.Parity = NOPARITY; // parity bit

dcb.StopBits = ONESTOPBIT; // stop bit

fSuccess = SetCommState(hCom, &dcb);

if (!fSuccess)

{

// Handle the error.

printf ("SetCommState failed with error %d.\n", GetLastError());

return (3);

}

// Get the comm config again.

fSuccess = GetCommState(hCom, &dcb);

if (!fSuccess)

{

// Handle the error.

printf ("GetCommState failed with error %d.\n", GetLastError());

return (2);

}

PrintCommState(dcb); // Output to console

\_tprintf (TEXT("Serial port %s successfully reconfigured.\n"), pcCommPort);

return (0);

Мониторинг событий

#include <windows.h>

#include <tchar.h>

#include <assert.h>

#include <stdio.h>

void \_tmain(

int argc,

TCHAR \*argv[]

)

{

HANDLE hCom;

OVERLAPPED o;

BOOL fSuccess;

DWORD dwEvtMask;

hCom = CreateFile( TEXT("\\\\.\\COM1"),

GENERIC\_READ | GENERIC\_WRITE,

0, // exclusive access

NULL, // default security attributes

OPEN\_EXISTING,

FILE\_FLAG\_OVERLAPPED,

NULL

);

if (hCom == INVALID\_HANDLE\_VALUE)

{

// Handle the error.

printf("CreateFile failed with error %d.\n", GetLastError());

return;

}

// Set the event mask.

fSuccess = SetCommMask(hCom, EV\_CTS | EV\_DSR);

if (!fSuccess)

{

// Handle the error.

printf("SetCommMask failed with error %d.\n", GetLastError());

return;

}

// Create an event object for use by WaitCommEvent.

o.hEvent = CreateEvent(

NULL, // default security attributes

TRUE, // manual-reset event

FALSE, // not signaled

NULL // no name

);

// Initialize the rest of the OVERLAPPED structure to zero.

o.Internal = 0;

o.InternalHigh = 0;

o.Offset = 0;

o.OffsetHigh = 0;

assert(o.hEvent);

if (WaitCommEvent(hCom, &dwEvtMask, &o))

{

if (dwEvtMask & EV\_DSR)

{

// To do.

}

if (dwEvtMask & EV\_CTS)

{

// To do.

}

}

else

{

DWORD dwRet = GetLastError();

if( ERROR\_IO\_PENDING == dwRet)

{

printf("I/O is pending...\n");

// To do.

}

else

printf("Wait failed with error %d.\n", GetLastError());

}

}

## Разработка драйвера ModbusRTU

Сделать на базе libmodbus

## Разработка драйвера ПЧ

Здесь перевожу адреса и коды взаимодействия с ПЧ в понятные методы

## Основная программа управления

Программа должна запускаться из командной строки с 1 обязательным аргументом, который является путем к .csv файлу с координатами графика зависимости скоростей от времени.

# Управление ПЧ с помощью промышленного контроллера ICP DAS I-7188EX

## Функции времени на I-7188EX

## Разработка драйвера UART интерфейса

## Перенос основной программы

## Разработка интерфейса взаимодействия с оператором

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Scott Chacon ProGit. Everything You need to know about Git. [Электронный ресурс] / Scott Chacon, Ben Straub. – Электрон. текстовые данные. – Apress; 2nd ed. edition, 2022. – 538 c. – 978-1484200773. – Режим доступа: <https://git-scm.com/book/ru/v2> – Загл. с экрана.
2. Overview – Doxygen Manual // [Электронный ресурс]. URL: <https://www.doxygen.nl/manual/index.html> (Дата обращения: 22.02.2023).
3. Win32 apps: Ресурсы для обмена данными – Microsoft Learn // [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/windows/win32/devio/communications-resources> (Дата обращения: 23.02.2023).
4. Specification and Implementation Guide for MODBUS over Serial Line – The Modbus organization // [Электронный ресурс]. URL: <https://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf> (Дата обращения: 24.02.2023).
5. MODBUS Application Protocol Specification v1.1b3 – The Modbus organization // [Электронный ресурс]. URL: <https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf> (Дата обращения: 24.02.2023).
6. Руководство по эксплуатации VFD-B – Delta Electronics Inc. // [Электронный ресурс]. URL: <http://www.deltronics.ru/images/manual/VFD-B_manual_rus.pdf> (Дата обращения: 01.03.2023).

<https://filecenter.deltaww.com/Products/download/06/060101/Manual/DELTA_IA-MDS_VFD-B-P_UM_EN_20090406.pdf>

сайт assert <https://en.cppreference.com/w/cpp/error/assert>

Шаньгин Е.С., Управление роботами и робототехническими системами : учеб. пособие для студентов вузов, обучающихся по направлению 652000 "Мехатроника и робототехника" / Е.С. Шаньгин ; Федер. агентство по образованию, Гос. образоват. учреждение высш. проф. образования, Уфим. гос. авиац. техн. ун-т. - Уфа : Уфим. гос. авиац. техн. ун-т, 2005 (Уфа : РИК УГАТУ). - 188 с.

Scott Chacon ProGit. Everything You need to know about Git. [Электронный ресурс] / Scott Chacon, Ben Straub. – Электрон. текстовые данные. – Apress; 2nd ed. edition, 2022. – 538 c. – 978-1484200773. – Режим доступа: <https://git-scm.com/book/ru/v2> – Загл. с экрана.