МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2023 г.

# РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СТЕНДА ДЛЯ ИССЛЕДОВАНИЯ ДИНАМИЧЕСКИХ ПРОЦЕССОВ ПНЕВМОКЛАПАНОВ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2023.129 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ П. О. Шабуров
«___»_____2023 г.

Автор работы,
студент группы КЭ-405
_____ Д. В. Жуков
«___»_____ 2023 г.

Нормоконтролёр,
ст. препод. каф. ЭВМ
_____С. В. Сяськов
«___»_____ 2023 г.

Челябинск-2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой,
_____ Д.В. Топольский
«___»_____ 2023 г.

# ЗАДАНИЕ

**на выпускную квалификационную работу бакалавра**

студенту группы КЭ-405
Жукову Даниилу Вячеславовичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

**1. Тема работы:** «Разработка программного обеспечения стенда для исследования динамических процессов пневмоклапанов» утверждена приказом по университету от «25» апреля 2023 г. №753-13/12

**2. Срок сдачи студентом законченной работы:** 22.05.2023 г.

**3. Исходные данные к работе.**

Стенд предназначен для формирования пульсирующего давления произвольной формы с целью исследования динамических характеристик датчиков давления.

**Функциональное значение:**

Функциональным назначением программно-аппаратного комплекса является обеспечение функционирования стенда исследования динамических характеристик датчиков давления.

**Требования к аппаратной части:**

Аппаратная часть комплекса должна обеспечивать возможность выполнения следующих функций:

– Функции выбора режима работы стенда;

– Функции приема аналоговых сигналов генератора сигналов произвольной формы;

– Функции приема аналоговых сигналов контрольного датчика давления (КДД) с выходным сигналом 4-20 мА;

– Функции связи с персональным компьютером по каналу USB;

– Функции управления работой клапанов в соответствии с заданным алгоритмом.

**Требования к временным характеристикам:**

Время выполнения одного цикла расчетов в соответствии с алгоритмом приложения А должно быть не более 0,3 мс.

**Требования к исходным кодам и языкам программирования:**

Исходные коды ПО должны быть реализованы на языке Си.

**4. Перечень подлежащих разработке вопросов:**

1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы;

2. Формирование требований;

3. Проектирование архитектуры программного обеспечения;

4. Разработка и тестирование программного обеспечения.

**5. Дата выдачи задания:** «1» декабря 2022 г.

Руководитель работы_____/ П. О. Шабуров /

Студент _____/ Д. В. Жуков /

КАЛЕНДАРНЫЙ ПЛАН

| Этап | Срок сдачи | Подпись руководителя |
|---|---|---|
| Анализ предметной области. Обзор аналогов | 30.03.2023 | |
| Определение требований | 06.04.2023 | |
| Проектирование архитектуры | 15.04.2023 | |
| Разработка и тестирование | 20.04.2023 | |
| Компоновка текста работы и сдача на нормоконтроль | 15.05.2023 | |

Руководитель работы _____ /П. О. Шабуров/

Студент _____ /Д. В. Жуков /

# АННОТАЦИЯ

Жуков Д. В. Разработка программного обеспечения стенда для исследования динамических процессов пневмоклапанов. – Челябинск: ФГАУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН, 2023, 34 с., 7 ил., список исп. источников – 3 наим.

Тема выпускной квалификационной работы: «Разработка программного обеспечения стенда для исследования динамических процессов пневмоклапанов».

В данной работе был проведён аналитический обзор предметной области и проведён обзор аналогов данной технологии. Были установлены требования к создаваемому проекту и спроектирована архитектура программного обеспечения. Также были разработан алгоритм управления пневмоклапанами и программа высокого уровня, которая обеспечивает связь пользователя со стендом. Весь программный комплекс был протестирован. В результате работы было разработано программное обеспечение для стенда, исследующего динамические процессы пневмоклапанов.

ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

Поставленная цель данной выпускной квалификационной работы является разработка программного обеспечения стенда для исследования динамических процессов пневмоклапанов. Задачи, которые были поставлены:

– Анализ предметной области и обзор аналогов;

– Сформировать требования для разработки ПО;

– Спроектировать архитектуру ПО;

– Разработать алгоритм управления пневмоклапанами и программу высокого уровня;

– Провести тестирование программного обеспечения.

Пневмоклапан – это небольшое устройство, которое предназначается для изменения направления движения потока сжатого воздуха в воздушных линиях пневматического привода.

Пневматические клапаны нашли очень широкое применение в таких сферах как приготовление пищи, окрас текстиля, синтез химических веществ, покраска автомобилей, вакуумирование, индукционный нагрев, производство фармацевтических препаратов, разлив напитков, производство электроэнергии, изготовление полупроводниковых плат, литье под давлением, производство сжатого воздуха, термическая формовка, лазерная резка и т. д.

Стенд пульсирующего давления предназначен для исследования динамических характеристик датчиков давления. Изобретение относится к устройствам, предназначенных для поверки и калибровки датчиков давления. Подобные устройства имеют коллектор для подключения образцового и проверяемого датчика давления, устройство для формирования давления и вычислительно-управляющее устройство.

Микроконтроллер – микросхема, предназначенная для управления электронными устройствами. Это сочетание на одном кристалле функции процессора и периферийных устройств. У них имеется своё оперативно запоминающее устройство (ОЗУ), а в некоторых внедрена собственное постоянное запоминающее устройство (ПЗУ).

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ОБЗОР АНАЛОГОВ

Целью выпускной квалификационной работы является разработка программного обеспечения для управления клапанами устройства формирования пульсирующего давления.

В рамках данной работы, используется микроконтроллер NUCLEO-F303RE прописанный в техническом задании от заказчика.

Плата Nucleo состоит из двух частей, как показано на рисунке 1. Часть с разъемом mini-USB представляет собой встроенный отладчик ST-LINK 2.1, который используется для загрузки микропрограммы в целевой микроконтроллер и выполнения пошаговой отладки. Интерфейс ST-LINK также предоставляет виртуальный COM-порт, который можно использовать для обмена данными и сообщениями с хост ПК. Одна из главных особенностей плат Nucleo заключается в том, что интерфейс STLINK можно легко отделить от остальной части платы. Таким образом, ее можно использовать в качестве автономного программатора ST-LINK [1].

Программное обеспечение – это совокупность программных и документальных средств для создания и эксплуатации систем обработки данных средствами вычислительной техники.

Ниже приведен обзор уже существующих аналогов, которые работают по тому же принципу, что описан выше.

Авторы работ [2] разработали инструмент для проверки и настройки серво- и пропорциональных клапанов. Технология ориентирована на клиентов, занимающихся тестированием клапанов.

В изобретении используется электронная плата технологии SMD, а обработка цифровой информации возложена на программируемый чип ALTERA. Алгоритм, вшитый в универсальную плату имеет функционал для работы стенда в разных режимах. Данная плата, представленная на рисунке 1, позволяет подключить в неё датчики давления, мерительный поршень, клапаны и компьютер.

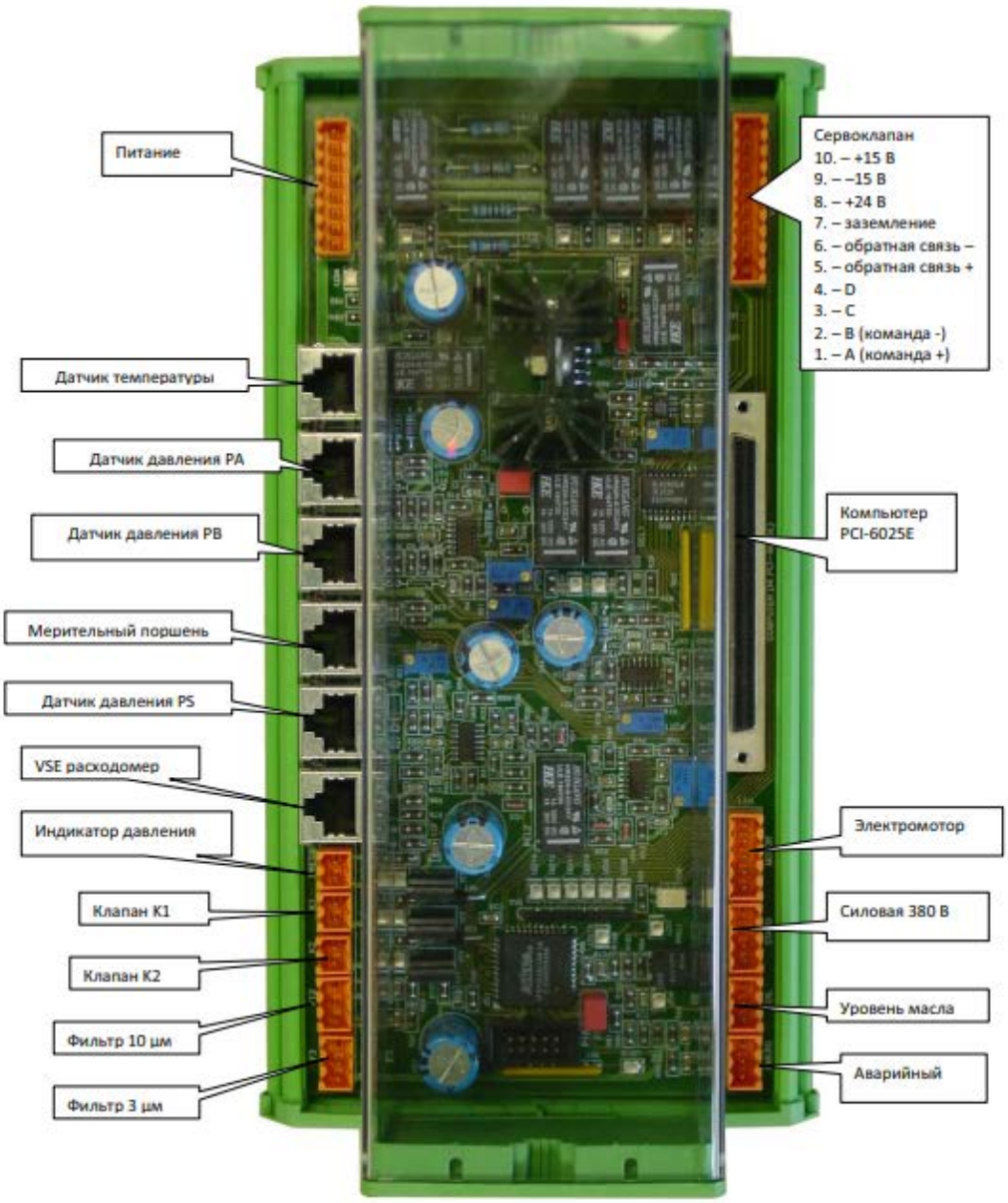


Рисунок 1 – Отладочная плата

Отдельно от стенда, авторы проекта создали специальное программное обеспечение со своим интерфейсом, предназначенное для настройки и выбора режима работы инструмента. Но данная технология запатентована за рубежом и доступность к данному продукту остаётся под вопросом.

Автор работы [3] использовал отладочную плату Arduino Mega 2560 фирмы Atmel для реализации своего проекта. Особенностью данной работы, является то, что микроконтроллер считывает данные с измерительных датчиков и оправляет их на персональный компьютер через последовательный порт, затем сравнивает полученные данные с входным сигналом и подает команду на открытие или закрытие клапанов. Так же рассчитывается время, на которое открываются

клапана, и задержка между открытием клапанов. Тем самым производиться регулировка уровня давления в ресивере. Так же функцией микроконтроллера, выступающего в качестве блока управления клапанами, является расчет погрешности сформированного давления и отправка этих данных на персональный компьютер через последовательный порт. Вся эта система работает при условии, что уже существует программа высокого уровня, с которой должен взаимодействовать пользователь.

Таким образом, основываясь на вышеперечисленных аналогах, можно сделать вывод, что хоть уже и существуют подобные технологии, но потребность к разработке программно-аппаратной части с алгоритмом управления и доступной программой высокого уровня к подобным стендам остаётся актуальной.

# 2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

В первой главе был проведён обзор уже существующих технологий управления клапанами на стендах исследования динамических характеристик. В данной главе необходимо определить функциональные и нефункциональные требования к данному проекту.

## 2.1 Функциональные требования

Так как работа предполагает разработку программно-аппаратной части, то для более удобного представления функционала разделим требования на аппаратные и требования к ПО.

Требования к аппаратной части:

– Функции выбора режима работы стенда;

– Функции приема аналоговых сигналов генератора сигналов произвольной формы;

– Функции приема аналоговых сигналов контрольного датчика давления (КДД) с выходным сигналом 4-20 мА;

– Функции связи с персональным компьютером по каналу USB;

– Функции управления работой клапанов в соответствии с заданным алгоритмом.

Требования к ПО:

– Функции расчета расходных характеристик клапанов и управления клапанами в соответствии с алгоритмом приложения А.

– Функции тестирования клапанов в соответствии с циклограммой представленной на рисунке 2.

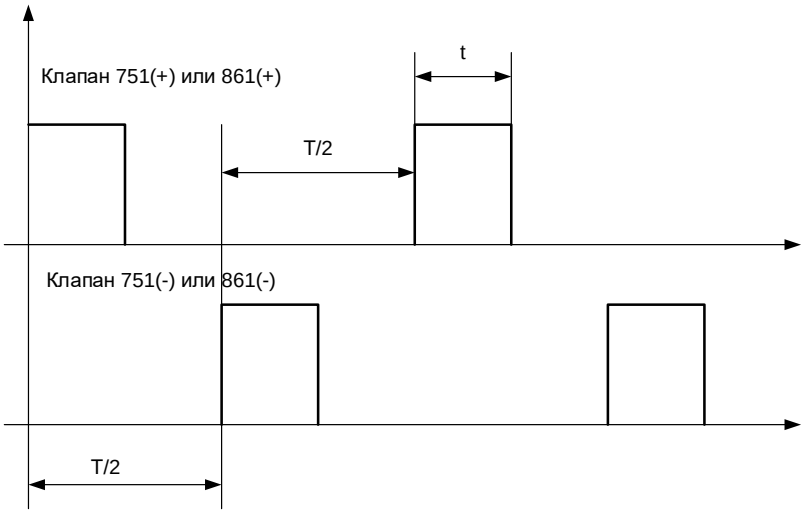


Рисунок 2 – Циклограмма работы режима тестирования

– Функции управления режимами работы стенда представленные в таблице 1.1.

Таблица 1.1 – Режимы работы

| Тумблер SA2 блока управления | Тумблер SA3 блока управления | Режим работы клапанов |
|---|---|---|
| 0 | 0 | Все клапаны закрыты |
| 1 | 0 | Открыты каналы клапана 861 на увеличение давления в соответствии с табл. 3.2 |
| 0 | 1 | Открыты каналы клапана 861 на сброс давления в соответствии с табл. 3.2 |
| 1 | 1 | Управление клапанами по алгоритму (приложение А) или по циклограмме рис.1. |

– Функции записи в ПЗУ блока управления режимов работы стенда и констант для расчётов в соответствии с таблицей 1.2.

Таблица 1.2 – Константы для расчёта

| Наименование | Обозначение, размерность | Значение по умолчанию | Допустимый диапазон |
|---|---|---|---|
| Режим формирования давления по алгоритму приложения А, код режима PC1=1, PC0=0. | | | |
| Нагрузка токовой петли контрольного датчика | $R_н$, кОм | 0,15 | 0,05 – 0,15 |
| Давление настройки контрольного датчика избыточное | $P_{мах}$, бар | 10,00 | 0,10 – 10,00 |
| Давление входное верхнее абсолютное | $P_в$, бар | 7,00 | 0,10 – 7,00 |
| Давление входное нижнее абсолютное | $P_н$, бар | 1,00 | 0,00 – 5,00 |
| Минимальное рассогласование | Δ, бар | 0,01 | 0,001 – 0,050 |
| Объем рабочей камеры | $V_{раб}$, л | 0,015 | 0,010 – 0,100 |
| Канал увеличения давления: | | | |
| пропускная способность | $C_+$, нл/(бар*с) | 0,800 | 0,100 – 3,000 |
| критический коэффициент | $b_+$ | 0,200 | 0,000 – 1,000 |
| Канал сброса давления | | | |
| пропускная способность | $C_-$, нл/(бар*с) | 0,800 | 0,100 – 3,000 |
| критический коэффициент | $b_-$ | 0,200 | 0,000 – 1,000 |
| Период дискретизации | Δt, с | 0,0015 | 0,0010 – 0,0050 |
| Задержка срабатывания клапана (открытие) | $t_{зад}$, с | 0,0005 | 0,0001 – 0,0030 |
| Режим тестирования клапанов, код режима PC1=0, PC0=1. | | | |
| Расход клапана 751(+) | 751(+) | 0 | 0 – 15* |
| Расход клапана 751(-) | 751(-) | 0 | 0 – 15* |
| Расход клапана 861(+) | 861(+) | 1 | 0 - 63 |
| Расход клапана 861(-) | 861(-) | 1 | 0 - 63 |
| Период переключения | T, мс | 300 | 3 - 1000 |
| Длительность открытого состояния | t, мс | 120 | 1 - 400 |

## 2.2 Нефункциональные требования

К нефункциональным требованиям данной работы относятся:

– Время выполнения одного цикла расчетов в соответствии с алгоритмом приложения А должно быть не более 0,3 мс.

– Надежное (устойчивое) функционирование ПО должно быть обеспечено выполнением при эксплуатации стенда требований руководства по эксплуатации

– Время восстановления функционирования ПО, вызванного прерыванием питания стенда, должно быть не более 5 с.

– Любые некорректные действия оператора не должны приводить к фатальным (невосстанавливаемым) отказам ПО.

– Восстановление функционирования ПО после некорректных действий оператора, должно выполняться отключением питания стенда на время не менее 0,5 с.

# 3 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ

Во второй главе были определены функциональные и нефункциональные требования к данному проекту. В данной главе необходимо спроектировать архитектуру программного обеспечения и структуру алгоритмов работы программ.

## 3.1 Архитектура программного обеспечения

Для полного понимания работы всего комплекса была разработана диаграмма классов UML, представленная на рисунке 3.


Рисунок 3 – Диаграмма классов UML

На данной диаграмме видны основные классы:

– Пользователь – имеет определённые знания для работы со стендом, может «общаться» со стендом с помощью программы высокого уровня, в которую он может установить режим работы стенда и ввести определённые значения для расчёта.

– Программа Windows – программа высокого уровня, которая обеспечивает связь с микроконтроллером, с помощью неё отправляются все введённые данные в микроконтроллер.

– Микроконтроллер – на нём уже происходит сам процесс управления пневмоклапанами. Им учитывается полученная информация от программы

высокого уровня, вычисляются нужные для управления значения и посылаются сигналы на пневмоклапан.

– Пневмоклапан – часть стенда, на которую посылаются сигналы для сброса или подачи давления.

### 3.2 Алгоритм работы программы

Алгоритм управления пневмоклапанами должен быть на языке Си. В качестве среды разработки была выбрана программа CubeMX.

Алгоритм сначала инициализирует все порты, подключенные к микроконтроллеру, затем считывает данные из флеш-памяти, далее производит опрос двух тумблеров, и в зависимости от положения этих тумблеров выбирается один из трёх видов работы стенда:

– Отключение всей системы,

– Уменьшение/ увеличение давления

– Режим формирования давления, (два режима: расчет расходных характеристик или тестирование клапанов).

Основной алгоритм программы представлен на рисунке 4.

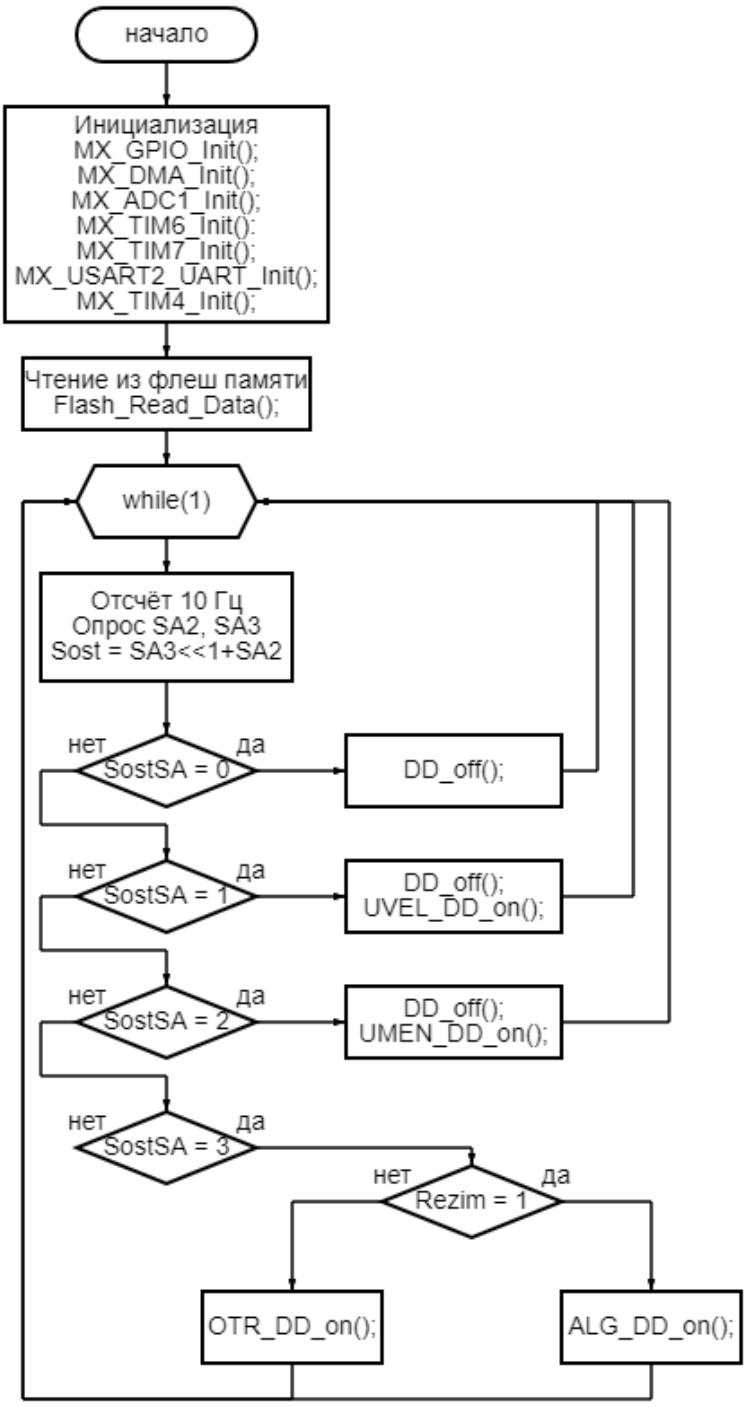


Рисунок 4 – Блок схема главного алгоритма управления пневмоклапанами

Алгоритм расчета расходных характеристик клапанов представлен на рисунке 5.

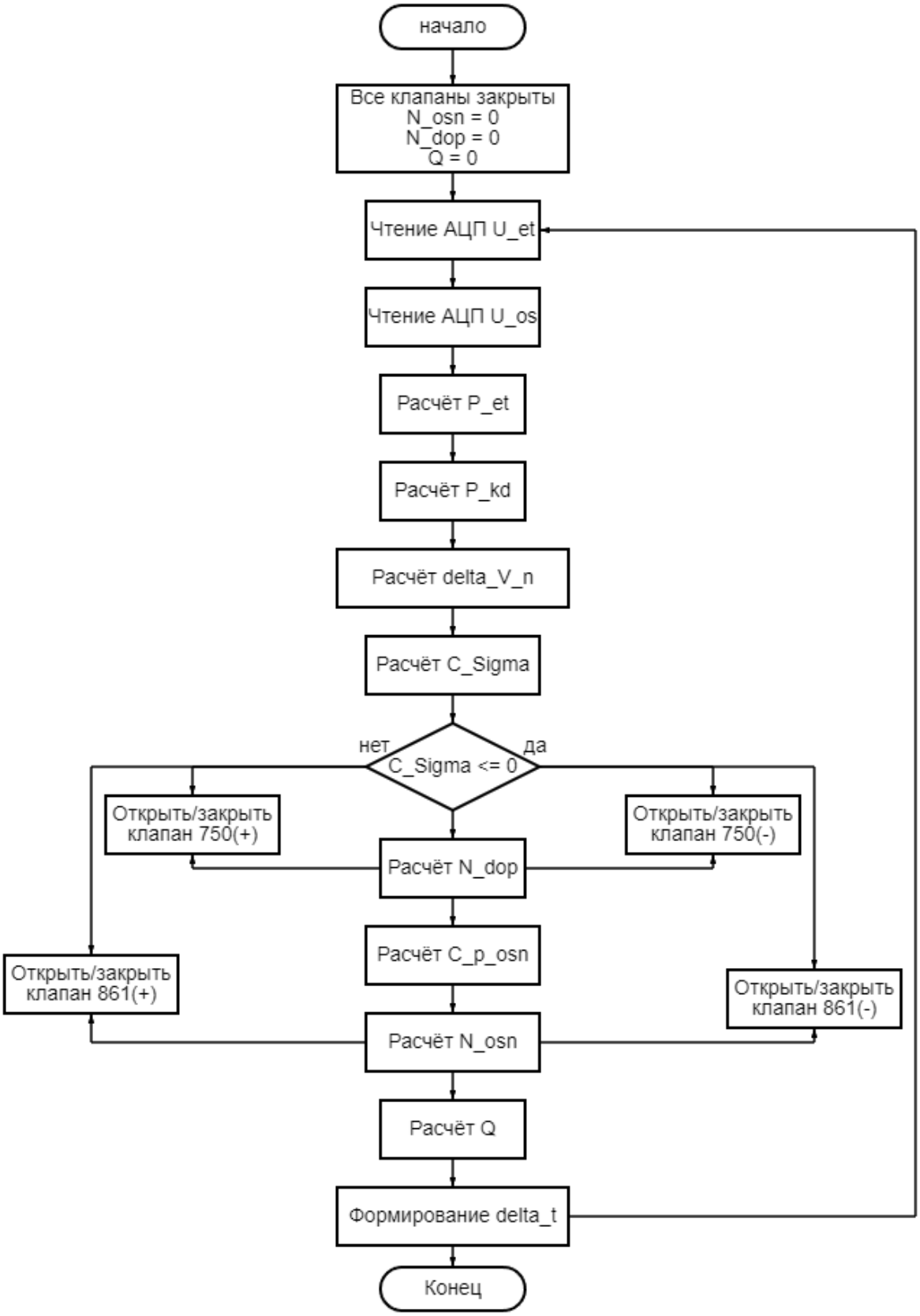


Рисунок 5 – Блок схема алгоритма расчет расходных характеристик

Для взаимодействия между пользователем и микроконтроллером требуется написать программу высокого уровня для операционной системы Windows.

Программа предназначена для управления клапанами подачи и сброса давления в стенде исследования динамических характеристик датчиков давления.

Программа осуществляет выполнение следующих функций:

– Запись в ПЗУ блока управления режимов работы стенда и констант для расчетов;

– Управление режимами работы стенда;

– Тестирование клапанов;

– Расчет расходных характеристик клапанов и управление клапанами.

# 4 РАЗРАБОТКА И ТЕСТИРОВАНИЕ

В третьей главе была спроектирована архитектура программного обеспечения проекта. В данной главе необходимо разработать спроектированный алгоритм программы и разработать программу высокого уровня.

## 4.1 Разработка алгоритма управления

В основном алгоритме программы происходит чтение констант из флеш-памяти. Далее следует бесконечный цикл, в котором проводится опрос двух тумблеров на частоте 10 Гц. В зависимости от состояния этих тумблеров выполняются методы в соответствии с блок схемой на рисунке 2.

Для правильной и лаконичной работы всей системы для каждого действия и каждого режима в алгоритме должен быть разработан свой метод прерывания.

В алгоритме реализованы следующие прерывания:

– Для опроса двух тумблеров был разработан алгоритм прерывания по системному таймеру SysTick, представленный на листинге 4.1.

Листинг 4.1 – Опрос двух тумблеров.

```
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  sostSA = LL_GPIO_IsInputPinSet(GPIOB, LL_GPIO_PIN_0)
           + LL_GPIO_IsInputPinSet(GPIOC, LL_GPIO_PIN_12) * 2;

  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */

  /* USER CODE END SysTick_IRQn 1 */
}
```

– Для выполнения режима расчета расходных характеристик, был разработан алгоритм прерывания по DMA1 канала 1. Алгоритм данного прерывания представлен в приложении А.

– Для выполнения режима тестирования клапанов был разработан алгоритм прерывания по таймеру 7.

– Для обмена информацией микроконтроллера с компьютером был разработан алгоритм прерывания по таймеру 4. При срабатывании данного прерывания определяется команда соответствующая чтению или записи. При чтении происходит чтение констант из флеш-памяти. При записи константы записываются в регистры флеш-памяти.

Для работы с флеш-памятью были разработаны методы по чтению и записи данных, представленные на листинге 4.2.

Листинг 4.2 – Реализация методов чтения/записи данных во флеш память.

```
//Запись во флеш-память
void WriteToFlash(uint32_t Value)
{
     uint8_t i;
     uint32_t pageAdr;

        //NVIC_VectTab_FLASH |
     pageAdr =  FIRMWARE_PAGE_OFFSET;

     FLASH_Unlock();
     FLASH_ErasePage(pageAdr);

     FLASH_ProgramWord((uint32_t)(pageAdr), (uint32_t)Value);
     FLASH_Lock();
}

//Чтение из флеш памяти
uint32_t Flash_Read_Data (volatile uint32_t StartSectorAddress)
{

  return *(__IO uint32_t *)StartSectorAddress;

}
```

В методе записи данных во флеш-память сначала идет разблокировка памяти, затем стирание всей страницы регистров, далее происходит запись слова размерностью 4 байта в регистр флеш-памяти по заданному адресу.

В методе чтения из флеш-памяти по заданному адресу памяти возвращается сохраненное значение.

– Для отправки данных на компьютер используется прерывание по USART2. Алгоритм данного прерывания представлен в приложении Б.

Для управления пневмоклапанами были разработаны методы для подачи/сброса давления.

Метод, отключающий все системы подачи или сброса давления представлен на листинге 4.3.

Листинг 4.3 – Реализация метода отключения всех систем

```
void DD_off(void) {
  LL_GPIO_SetPinPull(GPIOB,     LL_GPIO_PIN_1     |     LL_GPIO_PIN_6,
LL_GPIO_PULL_DOWN);
  LL_GPIO_SetPinPull(GPIOB, LL_GPIO_PIN_8, LL_GPIO_PULL_DOWN);
  LL_GPIO_SetPinPull(GPIOB, LL_GPIO_PIN_9, LL_GPIO_PULL_DOWN);

  LL_GPIO_SetPinPull(GPIOB,    LL_GPIO_PIN_10    |    LL_GPIO_PIN_15,
LL_GPIO_PULL_DOWN);

  LL_GPIO_SetPinPull(GPIOA,     LL_GPIO_PIN_6     |     LL_GPIO_PIN_10,
LL_GPIO_PULL_DOWN);
  LL_GPIO_SetPinPull(GPIOA,    LL_GPIO_PIN_13    |    LL_GPIO_PIN_15,
LL_GPIO_PULL_DOWN);

  LL_GPIO_SetPinPull(GPIOC,     LL_GPIO_PIN_4     |     LL_GPIO_PIN_9,
LL_GPIO_PULL_DOWN);

}
```

Метод, включающий клапан 861 на увеличение давления представлен на листинге 4.4.

Листинг 4.4 – Метод увеличение давления, используя клапан 861.

```
void UVEL_DD_on(void) {
  LL_GPIO_SetPinPull(GPIOB,     LL_GPIO_PIN_1     |     LL_GPIO_PIN_6,
LL_GPIO_PULL_UP);
  LL_GPIO_SetPinPull(GPIOB, LL_GPIO_PIN_8, LL_GPIO_PULL_UP);
  LL_GPIO_SetPinPull(GPIOB, LL_GPIO_PIN_9, LL_GPIO_PULL_UP);

  LL_GPIO_SetPinPull(GPIOB,    LL_GPIO_PIN_10    |    LL_GPIO_PIN_15,
LL_GPIO_PULL_DOWN);
}
```

Метод, включающий клапан 861 на сброс давления представлен на листинге 4.5.

Листинг 4.5 – Метод сброса давления, используя клапан 861.

```
void UMEN_DD_on(void) {
  LL_GPIO_SetPinPull(GPIOA,     LL_GPIO_PIN_6     |     LL_GPIO_PIN_10,
LL_GPIO_PULL_DOWN);
  LL_GPIO_SetPinPull(GPIOA,    LL_GPIO_PIN_13    |    LL_GPIO_PIN_15,
LL_GPIO_PULL_DOWN);

  LL_GPIO_SetPinPull(GPIOC,     LL_GPIO_PIN_4     |     LL_GPIO_PIN_9,
LL_GPIO_PULL_DOWN);}
```

4.2 Разработка программы высокого уровня

Так как программу высокого уровня требуется разработать для платформы Windows, то удобнее использовать среду разработки Windows Forms используя .NET Framework.

Интерфейс программы представлен на рисунке 6.

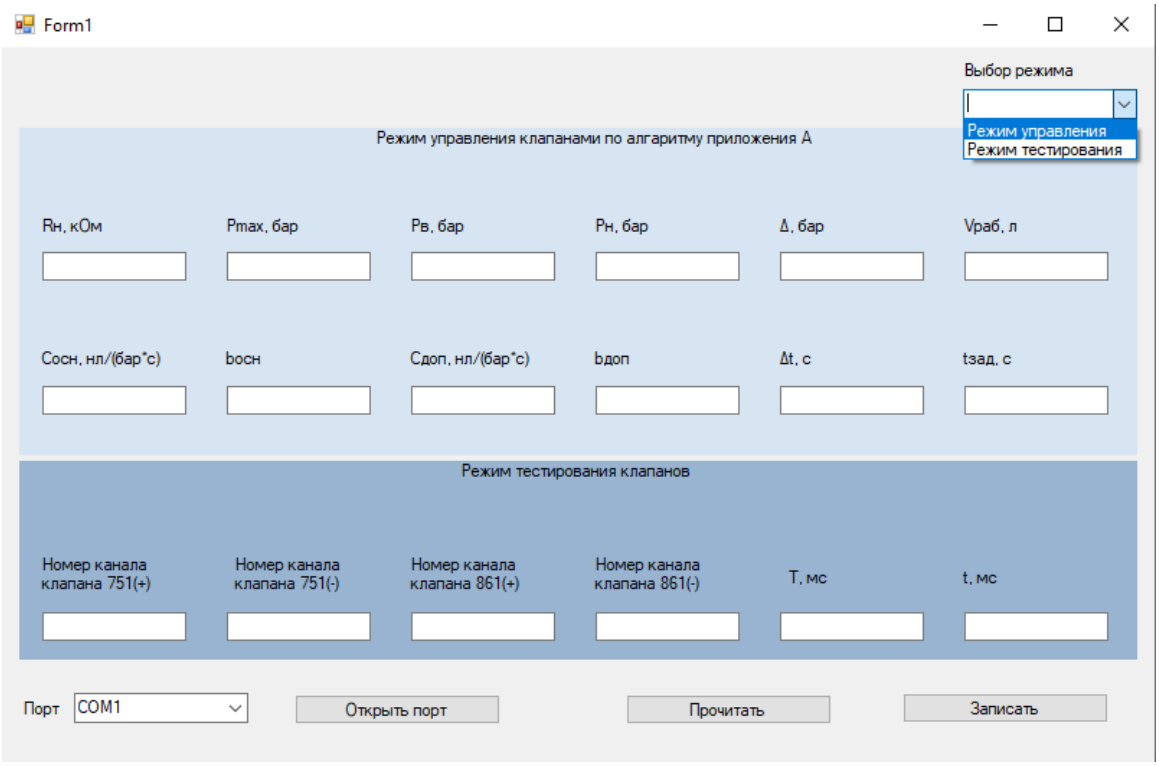


Рисунок 6 – Интерфейс программы

По рисунку видно, что для пользователя есть возможность: задавать определённые физические величины для расчёта, задавать режим работы стенда, выбрать порт, подсоединённый к микроконтроллеру.

Для данной программы были написаны следующие методы:

– Всплывающее окно предлагает список доступных COM портов используя библиотеку SerialPort. Алгоритм выбора доступного COM порта представлен на листинге 4.6.

Листинг 4.6 – Реализация выбора доступного COM порта

```
private void Form1_Load(object sender, EventArgs e)
{
    string[] ports = SerialPort.GetPortNames();
    comboBox2.Items.AddRange(ports);
}
```

– Кнопка «Открыть порт» связывается с микроконтроллером по выбранному порту для возможности записи и чтения. Алгоритм открыти порта представлен на листинге 4.7.

Листинг 4.7 – Реализация метода подключения к порту

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        serialPort1.PortName = comboBox2.Text;
        serialPort1.Open();

        label24.Text = "Порт открыт";

    }
    catch
    {
        MessageBox.Show("Connection error");
    }

}
```

– Кнопка «Прочитать» считывает данные с микроконтроллера. Алгоритм чтения данных с микроконтроллера представлен на листинге 4.8.

Листинг 4.8 – Чтение данных с микроконтроллера.

```
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        byte[] data = new byte[1024];
        int bytesRead = serialPort1.Read(data, 0, data.Length);
                string  output  =  Encoding.ASCII.GetString(data,  0,
bytesRead);

         //= serialPort1.ReadLine();
        label23.Text = "Чтение прошло успешно";

    } catch
    {
        MessageBox.Show("Error");
    }
}
```

Проверим возможность чтения данных из микроконтроллера. Для того, чтобы прочесть данные, необходимо сперва открыть порт, по которому соединён микроконтроллер. Затем, нажать кнопку «Прочитать». Текстовые поля программы заполняются сохранёнными данными на микроконтроллере. В результате

появляется строка сверху окна программы о успешном чтении данных. Результат работы чтения данных показан на рисунке 7.

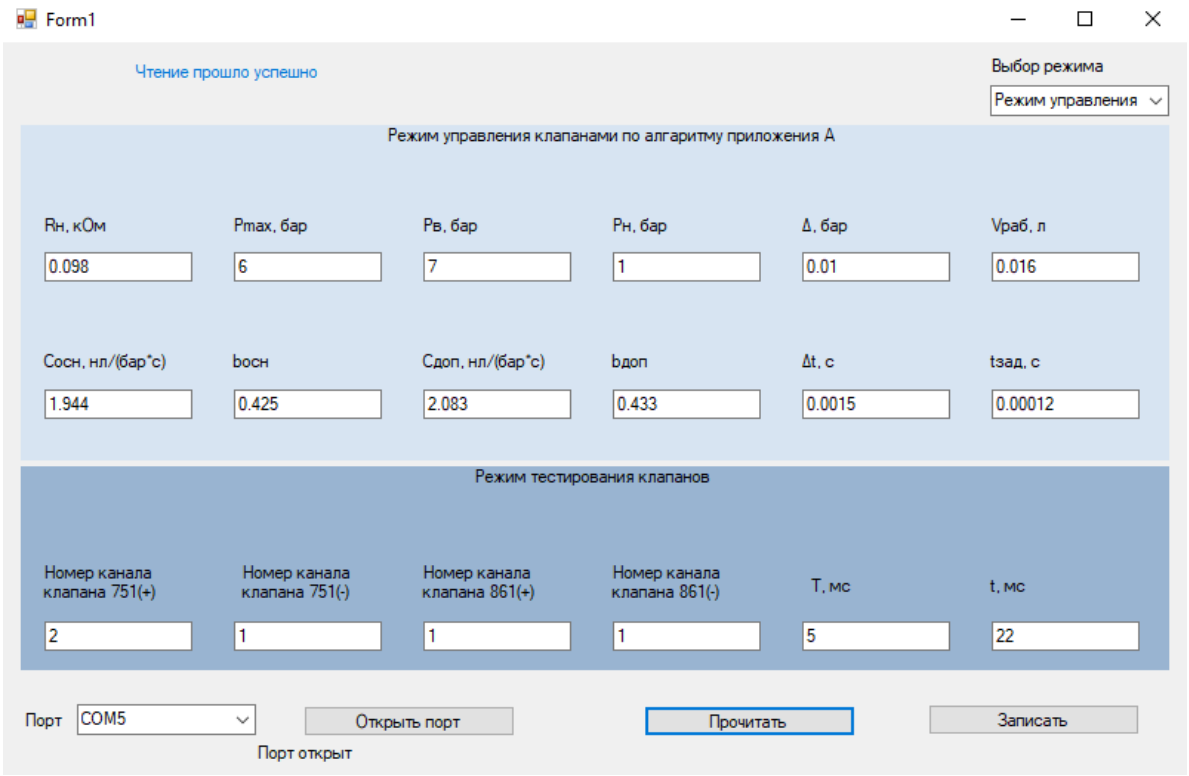


Рисунок 7 – Чтение данных

– Кнопка «Записать» записывает в микроконтроллер данные, которые были введены пользователем в полях текста соответствующих переменных в зависимости от выбранного режима. Алгоритм записи представлен на листинге 4.8.

Листинг 4.8 – Запист данных в микроконтроллер.

```
private void button3_Click(object sender, EventArgs e)
{
    int Rezim;
    try
    {
        if (comboBox1.Items[0].ToString() == "Режим управления")
          {
              Rezim = 0;
              if (serialPort1.IsOpen)
              {
                  serialPort1.Write(Rezim.ToString());
                  serialPort1.Write(textBox1.Text);
                  serialPort1.Write(textBox2.Text);
                  serialPort1.Write(textBox3.Text);
                  serialPort1.Write(textBox4.Text);
                  serialPort1.Write(textBox5.Text);
                  serialPort1.Write(textBox6.Text);
```

```
                serialPort1.Write(textBox12.Text);
                serialPort1.Write(textBox11.Text);
                serialPort1.Write(textBox10.Text);
                serialPort1.Write(textBox9.Text);
                serialPort1.Write(textBox8.Text);
                serialPort1.Write(textBox7.Text);
            }
    }   else   if   (comboBox1.Items[0].ToString()   ==   "Режим
тестирования")
            {
                Rezim = 1;

                if (serialPort1.IsOpen)
                {
                serialPort1.Write(Rezim.ToString());
                serialPort1.Write(textBox18.Text);
                serialPort1.Write(textBox17.Text);
                serialPort1.Write(textBox16.Text);
                serialPort1.Write(textBox15.Text);
                serialPort1.Write(textBox14.Text);
                serialPort1.Write(textBox13.Text);
                }
            }

        } catch
        {
            MessageBox.Show("Error");
        }
}
```

# ЗАКЛЮЧЕНИЕ

Поставленная цель данной выпускной квалификационной работы является разработка программного обеспечения стенда для исследования динамических процессов пневмоклапанов.

Решённые задачи, которые были поставлены при выполнении работы:

– Проведён анализ предметной области и обзор аналогов;

– Сформированы требования для разработки ПО;

– Спроектирована архитектура ПО;

– Разработаны алгоритм управления пневмоклапанами и программа высокого уровня;

– Проведено тестирования программного обеспечения.

Цель данной работы достигнута. Программное обеспечение позволяет исследовать динамические процессы пневмоклапанов.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Кармин Новиелло Отладочная плата Nucleo : Пер. с англ. Освоение STM32 / Кармин Новиелло. – Leanpub, 2018. – 825 с.

2. DIETZ automation GmbH Автоматический испытательный стенд для проверки, технического обслуживания и регулировки сервоклапанов / [Электронный ресурс] // 2004. – Режим доступа: https://dietzautomation.com/wp-content/uploads/2023/02/ValveExpert-2.x-Manual-ru.pdf

3. Трусова, Е. А. Разработка программного обеспечение для динамического измерительного стенда: пояснительная записка ВКР / Е. А. Трусова. – Челябинск, 2017. – 74 с.

# ПРИЛОЖЕНИЯ

## ПРИЛОЖЕНИЕ А
Алгоритм работы режима расчёта расходных характеристик

```
void DMA1_Channel1_IRQHandler(void)
{
  /* USER CODE BEGIN DMA1_Channel7_IRQn 0 */
  //DD_off();
  N_osn = 0;
  N_dop = 0;
  Q = 0;
  // sChitivanie ACP U_et, U_os

  //Raschet P_et, P_kd
  P_et = (((U_et - 4.0f * P_n)*P_max)/16*P_n) + 1.013f;
  P_kd = (((U_kd - 4.0f * P_n)*P_max)/16*P_n) + 1.013f;

  //Raschet delta_V_n
  if (P_et - P_kd > 0) {
    delta_V_n = 0.987f * (P_et - P_kd) * V_rab;
  } else if (P_et - P_kd <= 0) {
    delta_V_n = 0;
  }

  //Raschet C_sigma
    if ((delta_V_n - t_zad * Q > 0) && (P_kd / P_v > b_plus)) {

    C_sigma = (delta_V_n - t_zad * Q)/(detla_t * P_v * sqrt(1.0f -
powf((P_kd/P_v - b_plus)/(1-b_plus), 2.0f)));

  } else if ((delta_V_n - t_zad * Q > 0) && (P_kd / P_v <= b_plus))
{

    C_sigma = (delta_V_n - t_zad * Q)/(detla_t * P_v);

  } else if ((delta_V_n - t_zad * Q > 0) && (P_kd / P_v == 1.01f)) {

    C_sigma = (delta_V_n - t_zad * Q)/(detla_t * P_v * 0,001);

  } else if ((delta_V_n - t_zad * Q > 0)&&(C_sigma >= (C_plus*(C_osn
+ C_dop))/(C_plus + C_osn + C_dop))) {

    C_sigma = C_sigma >= (C_plus*(C_osn + C_dop))/(C_plus + C_osn +
C_dop);

  } else if ((delta_V_n - t_zad * Q <= 0) && (P_n / P_kd > b_minus))
{

    C_sigma = (delta_V_n - t_zad * Q)/(detla_t * P_kd * sqrt(1.0f -
powf((P_n/P_kd - b_minus)/(1 - b_minus), 2.0f)));
```

```
  } else if ((delta_V_n - t_zad * Q <= 0) && (P_n / P_kd <=
b_minus)) {

    C_sigma = (delta_V_n - t_zad * Q)/(detla_t * P_kd);

  } else if ((delta_V_n - t_zad * Q <= 0) && (P_n / P_kd == 1.01f))
{

    C_sigma = (delta_V_n - t_zad * Q)/(detla_t * P_kd * 0,001);

  } else if ((delta_V_n - t_zad * Q <= 0)&&(C_sigma <=
(C_minus*(C_osn + C_dop))/(C_minus + C_osn + C_dop))) {

    C_sigma = C_sigma >= (C_minus*(C_osn + C_dop))/(C_minus + C_osn
+ C_dop);

  }

  if (C_sigma > 0) {
    //open klapan 750(+) and 861(+)
  } else if (C_sigma <= 0) {
    //open klapan 750(-) and 861(-)
  }

  //Raschet N_dop
    if (delta_V_n - t_zad * Q > 0) {

    N_dop = trunc((8.0f * (C_osn * C_plus - fabs(C_sigma) * (C_osn +
C_plus)))/(C_dop * fabs(C_sigma) - C_plus));

  } else if ((8.0f * (C_osn * C_plus - fabs(C_sigma) * (C_osn +
C_plus)))/(C_dop * fabs(C_sigma) - C_plus) > 7.0f) {

    N_dop = 8.0f;

  } else if ((8 * (C_osn * C_plus - fabs(C_sigma) * (C_osn +
C_plus)))/(C_dop * fabs(C_sigma) - C_plus) <= 0) {

    N_dop = 0;

  } else if (delta_V_n - t_zad * Q <= 0) {

    N_dop = trunc((8.0f * (C_osn * C_minus - fabs(C_sigma) * (C_osn
+ C_minus)))/(C_dop * fabs(C_sigma) - C_minus));

  } else if ((8.0f * (C_osn * C_minus - fabs(C_sigma) * (C_osn +
C_minus)))/(C_dop * fabs(C_sigma) - C_minus) > 7.0f) {

    N_dop = 8.0f;
```

```
} else if ((8.0f * (C_osn * C_minus - fabs(C_sigma) * (C_osn +
C_minus)))/(C_dop * fabs(C_sigma) - C_minus) <= 0) {

    N_dop = 0;

  }
  //close klapan 750 +-

  //Raschet C_p_osn

  //Rashet N_osn
    if (delta_V_n - t_zad * Q > 0) {

    N_osn = trunc(((((C_plus * C_dop * N_dop) / 8.0f) -
fabs(C_sigma) * (((C_dop * N_dop) / 8.0f) + C_plus)) * 63.0f) /
((fabs(C_sigma) - C_plus) * C_osn));

  } else if ((((((C_plus * C_dop * N_dop) / 8.0f) - fabs(C_sigma) *
(((C_dop * N_dop) / 8.0f) + C_plus)) * 63.0f) / ((fabs(C_sigma) -
C_plus) * C_osn)) >= 63.0f) {

    N_osn = 63.0f;

  } else if ((((((C_plus * C_dop * N_dop) / 8.0f) - fabs(C_sigma) *
(((C_dop * N_dop) / 8.0f) + C_plus)) * 63.0f) / ((fabs(C_sigma) -
C_plus) * C_osn)) <= 0) {

    N_osn = 0;

  } else if (delta_V_n - t_zad * Q <= 0) {

    N_osn = trunc(((((C_minus * C_dop * N_dop) / 8.0f) -
fabs(C_sigma) * (((C_dop * N_dop) / 8.0f) + C_minus)) * 63.0f) /
((fabs(C_sigma) - C_minus) * C_osn));

  } else if ((((((C_minus * C_dop * N_dop) / 8.0f) - fabs(C_sigma) *
(((C_dop * N_dop) / 8.0f) + C_minus)) * 63.0f) / ((fabs(C_sigma) -
C_minus) * C_osn)) >= 63.0f) {

    N_osn = 63.0f;

  } else if ((((((C_minus * C_dop * N_dop) / 8.0f) - fabs(C_sigma) *
(((C_dop * N_dop) / 8.0f) + C_minus)) * 63.0f) / ((fabs(C_sigma) -
C_minus) * C_osn)) <= 0) {

    N_osn = 0;

  }
  //close klapan 861 +-
```

```
//Rashet Q
    if (delta_V_n - t_zad * Q > 0 && P_kd/P_v > b_plus) {

    Q = ((((C_osn * N_osn) / 63.0f) + ((C_dop * N_dop) / 8.0f)) *
C_plus * P_v * sqrt(1.0f - powf((P_kd/P_v - b_plus)/(1-b_plus),
2.0f)))/(((C_osn * N_osn) / 63.0f) + ((C_dop * N_dop) / 8.0f) +
C_plus);

  } else if (delta_V_n - t_zad * Q > 0 && P_kd/P_v <= b_plus) {

    Q = ((((C_osn * N_osn) / 63.0f) + ((C_dop * N_dop) / 8.0f)) *
C_plus * P_v) / (((C_osn * N_osn) / 63.0f) + ((C_dop * N_dop) /
8.0f) + C_plus);

  } else if (delta_V_n - t_zad * Q <= 0 && P_n/P_kd > b_minus) {

    Q = ((((C_osn * N_osn) / 63.0f) + ((C_dop * N_dop) / 8.0f)) *
C_minus * P_v * sqrt(1.0f - powf((P_n/P_kd - b_minus)/(1 - b_minus),
2.0f)))/(((C_osn * N_osn) / 63.0f) + ((C_dop * N_dop) / 8.0f) +
C_minus);

  } else if (delta_V_n - t_zad * Q <= 0 && P_n/P_kd <= b_minus) {

    Q = ((((C_osn * N_osn) / 63.0f) + ((C_dop * N_dop) / 8.0f)) *
C_minus * P_v) / (((C_osn * N_osn) / 63.0f) + ((C_dop * N_dop) /
8.0f) + C_minus);

  }

  //formerovanie delta_t

  /* USER CODE END DMA1_Channel7_IRQn 0 */

  /* USER CODE BEGIN DMA1_Channel7_IRQn 1 */

  /* USER CODE END DMA1_Channel7_IRQn 1 */
}
```

ПРИЛОЖЕНИЕ Б
Алгоритм прерывания по USART2

```
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  LL_USART_InitTypeDef USART_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_USART2);

  LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
  /**USART2 GPIO Configuration
  PA2   ------> USART2_TX
  PA3   ------> USART2_RX
  */
  GPIO_InitStruct.Pin = LL_GPIO_PIN_2|LL_GPIO_PIN_3;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_HIGH;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USART2 DMA Init */

  /* USART2_TX Init */
  LL_DMA_SetDataTransferDirection(DMA1,             LL_DMA_CHANNEL_7,
LL_DMA_DIRECTION_MEMORY_TO_PERIPH);

  LL_DMA_SetChannelPriorityLevel(DMA1,              LL_DMA_CHANNEL_7,
LL_DMA_PRIORITY_LOW);

  LL_DMA_SetMode(DMA1, LL_DMA_CHANNEL_7, LL_DMA_MODE_NORMAL);

  LL_DMA_SetPeriphIncMode(DMA1,                     LL_DMA_CHANNEL_7,
LL_DMA_PERIPH_NOINCREMENT);

  LL_DMA_SetMemoryIncMode(DMA1,                     LL_DMA_CHANNEL_7,
LL_DMA_MEMORY_INCREMENT);

  LL_DMA_SetPeriphSize(DMA1,                        LL_DMA_CHANNEL_7,
LL_DMA_PDATAALIGN_BYTE);

  LL_DMA_SetMemorySize(DMA1,                        LL_DMA_CHANNEL_7,
LL_DMA_MDATAALIGN_BYTE);
```

```
  /* USART2 interrupt Init */
  NVIC_SetPriority(USART2_IRQn,
NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));
  NVIC_EnableIRQ(USART2_IRQn);

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  USART_InitStruct.BaudRate = 38400;
  USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
  USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
  USART_InitStruct.Parity = LL_USART_PARITY_NONE;
  USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
  USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
  USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
  LL_USART_Init(USART2, &USART_InitStruct);
  LL_USART_DisableIT_CTS(USART2);
  LL_USART_ConfigAsyncMode(USART2);
  LL_USART_Enable(USART2);
  /* USER CODE BEGIN USART2_Init 2 */
  sprintf(str1,"String %04drn",i);
  USART_TX((uint8_t*)str1,13);
  /* USER CODE END USART2_Init 2 */

}
```