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»

РАБОТА ПРОВЕРЕНА
Консультант
_____ А. Р. Ишбулатов
«___»_____ 2023 г.

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

# Разработка пользовательского интерфейса программы диагностики и настройки электронных блоков управления самоходных машин

## ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
## К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
## ЮУРГУ–090301.2023.130 ПЗ ВКР

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____М. А. Алтухова
«___»_____2023 г.

Автор работы,
студент группы КЭ-405
_____И. С. Захаров
«___»_____2023 г.

Нормоконтролёр,
ст. преп. 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 ЭВМ
_____ Д. В. Топольский
«___»_____2023 г.

# ЗАДАНИЕ

**на выпускную квалификационную работу бакалавра**

студенту группы КЭ-405
Захарову Ивану Сергеевичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

**1. Тема работы:** «Разработка пользовательского интерфейса программы диагностики и настройки электронных блоков управления самоходных машин» утверждена приказом по университету от 25.04.2023 г. №753-13/12.

**2. Срок сдачи студентом законченной работы:** 01 июня 2023 г.

**3. Исходные данные к работе.**

Требуется разработка графического пользовательского интерфейса для программы диагностики электронных блоков управления самоходных машин, предназначенной заменить использующуюся на предприятии на данный момент программу диагностики BODAS Service. Интерфейс должен быть максимально приближен по функционалу и внешнему виду к интерфейсу оригинальной программы.

С помощью предоставленных предприятием функций и алгоритмов обмена информацией с контроллером, интерфейс должен представлять полученную информацию в удобном графическом виде, иметь возможность

запрашивать значения параметров у пользователя и передавать их на сохранение в память контроллера.

Интерфейс должен поддерживать операционные системы Windows 7 и Windows 10.

**4. Перечень подлежащих разработке вопросов:**

1) анализ предметной области;
2) формирование основных требований;
3) проектирование интерфейса;
4) реализация интерфейса;
5) тестирование интерфейса.

**5. Дата выдачи задания:** 2 декабря 2022 г.

Руководитель работы_______/М. А. Алтухова/

Студент_____/И. С. Захаров/

КАЛЕНДАРНЫЙ ПЛАН

| Этап | Срок сдачи | Подпись руководителя |
|---|---|---|
| Анализ предметной области | 01.03.2023 | |
| Формирование основных требований | 07.03.2023 | |
| Проектирование интерфейса | 15.03.2023 | |
| Реализация интерфейса | 01.05.2023 | |
| Тестирование интерфейса | 15.05.2023 | |
| Компоновка текста работы и сдача на нормоконтроль | 24.05.2023 | |
| Подготовка презентации и доклада | 30.05.2023 | |

Руководитель работы_______/М. А. Алтухова/

Студент_____/И. С. Захаров/

# АННОТАЦИЯ

И. С. Захаров. Разработка пользовательского интерфейса программы диагностики и настройки электронных блоков управления самоходных машин. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2023, 77 с., 26 ил., библиогр. список – 22 наим.

В рамках выпускной квалификационной работы был проведён аналитический обзор интерфейса подлежащей замене аналогичной программы BODAS Service и интерфейсов других аналогичных систем. В результате проведенного обзора были выявлены недостатки, которые необходимо устранить в выпускной квалификационной работе, и достоинства этих систем, которые необходимо включить в выпускную квалификационную работу.

Рассмотрены основные технологии, применяющиеся в разработке программ для персональных компьютеров, и выбраны наиболее подходящие для данного проекта.

После аналитического обзора аналогов и основных технических решений были произведены: проектирование, разработка и тестирование пользовательского интерфейса программы диагностики и настройки электронных блоков управления для персональных компьютеров. Результатом работы является графический пользовательский интерфейс программы для персональных компьютеров под управлением операционных систем Windows 7 и Windows 10, реализующий заявленный функционал.

ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

Одним из важнейших элементов практически всех современных автомобилей и самоходной техники, например, бульдозеров, кабелеукладчиков, погрузчиков, трубоукладчиков и т. д. является электронный блок управления (далее – ЭБУ) различными электрическими системами. Управление происходит в реальном времени. Данные о текущих условиях поступают на ЭБУ от датчиков, они подвергаются обработке в соответствии с алгоритмами и используются для управления исполнительными механизмами и устройствами [1].

На самоходной технике наиболее часто используются ЭБУ комбинированного типа, совмещающие в себе блоки управления двигателем, трансмиссией и навесным оборудованием, например, ножом кабелеукладчика, отвалом бульдозера и т. д.

Для корректной работы ЭБУ требуется загружаемое в него программное обеспечение (далее – ПО), называемое прошивкой, регламентирующее алгоритм работы блока, и настройка различных параметров. ЭБУ может передавать ошибки в электронных системах, информацию о портах ЭБУ, показатели различных датчиков в реальном времени и прочую информацию, необходимую для отладки и диагностики электронных систем машины [1].

Для взаимодействия с ЭБУ и обеспечения работоспособности перечисленных выше функций существуют системы диагностики ЭБУ для различных платформ, представляющие из себя программное обеспечение с графическим пользовательским интерфейсом.

Графический пользовательский интерфейс (Graphical user interface, GUI) – система средств для взаимодействия пользователя с электронными устройствами, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т. п.). Графический

пользовательский интерфейс является неотъемлимой частью всех программ диагностики и настройки ЭБУ для персональных компьютеров.

Предприятию требуется графический пользовательский интерфейс для собственной программы диагностики ЭБУ для возможности его настройки и изменения под нужды предприятия.

Актуальность проекта обусловлена заказом предприятия.

Целью выпускной квалификационной работы является разработка графического пользовательского интерфейса для программы диагностики электронных блоков управления самоходных машин, предназначенной заменить использующуюся на предприятии на данный момент программу диагностики BODAS Service, для операционных систем Windows 7 и Windows 10.

Для достижения поставленной цели, необходимо решить следующие задачи:

1) провести анализ интерфейса использующейся на предприятии программы BODAS Service и интерфейсов других аналогичных программ, выявить их достоинства и недостатки, сделать вывод;

2) провести анализ существующих технологий разработки программ для персональных компьютеров и выбрать из них оптимальные для выполнения выпускной квалификационной работы, обосновать выбор;

3) провести анализ существующих шаблонов проектирования и выбрать из них оптимальный для выполнения выпускной квалификационной работы, обосновать выбор;

4) определить функциональные и нефункциональные требования к разрабатываемому интерфейсу;

5) разработать структурно-функциональную схему проекта;

6) выполнить программную реализацию проекта;

7) провести тестирование разработанного интерфейса на корректность выполнения задач, выполнить отладку.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Обзор аналогов

На данный момент предприятие использует программу BODAS Service для диагностики и настройки электронных блоков управления Bosch Rexroth BODAS, устанавливающихся на выпускаемую предприятием самоходную технику. В связи с невозможностью будущего использования предприятием лицензионной версии данной программы, предприятию требуется замена этой конкретной программы.

Программный инструмент **BODAS Service** для ПК обеспечивает удобный и простой в использовании способ выполнения сервисных функций для контроллеров BODAS от Bosch Rexroth. Параметры могут быть отображены и отредактированы, могут быть отображены переменные процесса, их значения могут быть записаны и выведены на график. Могут быть выведены и сброшены сообщения об ошибках контроллера, а также может быть настроена диагностическая конфигурация. Кроме того, доступны чтение и запись функций в EEPROM (англ. Electrically Erasable Programmable Read-Only Memory – электрически стираемое перепрограммируемое постоянное запоминающее устройство) контроллера [2].

Пользовательский интерфейс программы представлен на рисунке 1.1.

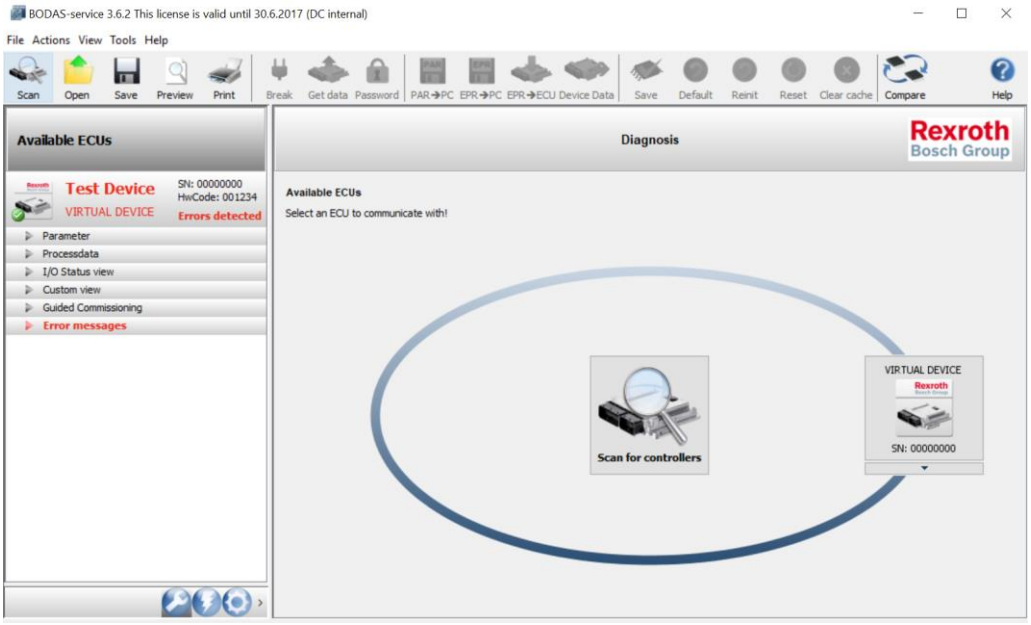


Рисунок 1.1 – Графический пользовательский интерфейс программы BODAS Service версии 3.6.2

Достоинства программы BODAS Service:

1) возможность вывода всех выводимых и возможность изменения всех изменяемых параметров контроллера семейства Bosch Rexroth BODAS;

2) интуитивный и удобный интерфейс, функции и вкладки группированы;

3) предусмотрена возможность сохранять пользовательские настройки и диагностические проекты, т.е. создавать шаблон или «слепок» проекта с уже заданными параметрами для определенного контроллера;

4) предусмотрена возможность строить графики;

5) можно сохранить текущее состояние контроллера с его характеристиками, значениями всех параметров и ошибками в удобном читаемом виде;

6) можно изменять способ вывода параметров и их значений, тему и цвета интерфейса;

7) кнопки, функции которых невыполнимы в данный момент или выполнение функций которых не приведет к ожидаемому результату, являются неактивными и недоступными для нажатия;

8) панель задач в нижней части пользовательского интерфейса программы, содержащая информацию о текущем подключении;

9) невысокие системные требования относительно широких возможностей функционала;

10) подробные документация и техническая спецификация к программе.

Недостатки программы BODAS Service:

1) работа преимущественно с контроллерами Bosch Rexroth BODAS, о чем заявляют разработчики в технической спецификации программы;

2) интерфейс программы реализован на устаревшей платформе Windows Forms (WF), значительно уступающей по удобству и функционалу платформе Windows Presentation Foundation (WPF);

3) стиль всех шрифтов не определен, поэтому используется исходный шрифт платформы Windows Forms «Segoe UI», который в разных блоках, таких как: текстовые, кнопки и т. д., выглядит по-разному, в зависимости от параметров блока – текст может быть размыт или отображаться некорректно в связи с его неправильным положением;

4) платная лицензия;

5) нет исходного кода программы в открытом доступе, следовательно, невозможность вносить изменения в алгоритмы работы или интерфейс, не предусмотренные в настройках.

В связи с рекомендацией производителя использующихся на предприятии контроллеров Bosch Rexroth BODAS, для диагностики и настройки данных контроллеров используется программа BODAS Service, в то время как в остальных аналогичных программах предусмотрена лишь диагностика и настройка автомобильных ЭБУ, значительно уступающих Bosch Rexroth BODAS по функциональности. Следовательно, обзоры следующих аналогов будут исключать рассмотрение их функциональной составляющей.

**PCMSCAN** – это полнофункциональный универсальный сканер OBD-II и диагностический инструмент, который поддерживает широкий спектр аппаратных интерфейсов OBD-II. Он позволяет просматривать, составлять графики, читать и выводить диагностические данные в режиме реального времени через порт диагностических данных OBD-II автомобиля. Он также позволяет просматривать диагностические коды неисправностей (DTC), т.е. коды ошибок автомобиля, данные стоп-кадра (текущее состояние) и другую информацию об автомобиле.

Пользовательский интерфейс программы представлен на рисунке 1.2.


Рисунок 1.2 – Графический пользовательский интерфейс программы PCMSCAN версии 2.4.12

Нефункциональные достоинства программы PCMSCAN:

1) создание графиков с одновременным отображением значений до 8 параметров;

2) создание полного отчета и запись лога о полученных данных с возможностью экспорта в файл;

3) визуализация в окне программы различных датчиков;

4) все функции, выводимая информация о текущем состоянии контроллера и автомобиля группированы и находятся в определенных вкладках;

5) ранее открытые вкладки не закрываются при выходе из них автоматически, а сохраняются в оперативной памяти, что дает возможность вернуться к ним без повторной загрузки в любой момент;

6) все функции и разделы имеют оригинальные тематические картинки, дающие представление о назначении функции или раздела без необходимости читать описание;

7) предусмотрена возможность сохранять пользовательские настройки и диагностические проекты, т.е. создавать шаблон или «слепок» проекта с уже заданными параметрами для определенного контроллера;

8) кнопки, функции которых невыполнимы в данный момент или выполнение функций которых не приведет к ожидаемому результату, являются неактивными и недоступными для нажатия;

9) панель задач в нижней части пользовательского интерфейса программы, содержащий описание функции кнопки, на которую наведен курсор мыши.

Нефункциональные недостатки программы PCMSCAN:

1) большая часть второстепенных окон имеют неактивную кнопку закрытия в правом верхнем углу, вместо которой присутствует отдельная кнопка Close, что представляется неудобным и непривычным обычному пользователю;

2) графики стилизованы под осциллограф, что бывает не всегда удобным.

Программа **OBD Scan Tech** служит для диагностики автомобилей по протоколам ISO 9141 и KWP 2000 через разъем OBD с помощью K-Line адаптера и позволяет получить данные по двигателю, коробке передач и некоторым узлам автомобиля, работающим по K-Line. Стоить отметить, что программа не позволяет настраивать и изменять никакие параметры, кроме информации об автомобиле, заполнение которой избавит пользователя от поиска описания ошибок, некоторых параметров и других данных,

отличающихся в зависимости от производителя. Другими словами, программа служит только для чтения информации с контроллера.

Пользовательский интерфейс программы представлен на рисунке 1.3.

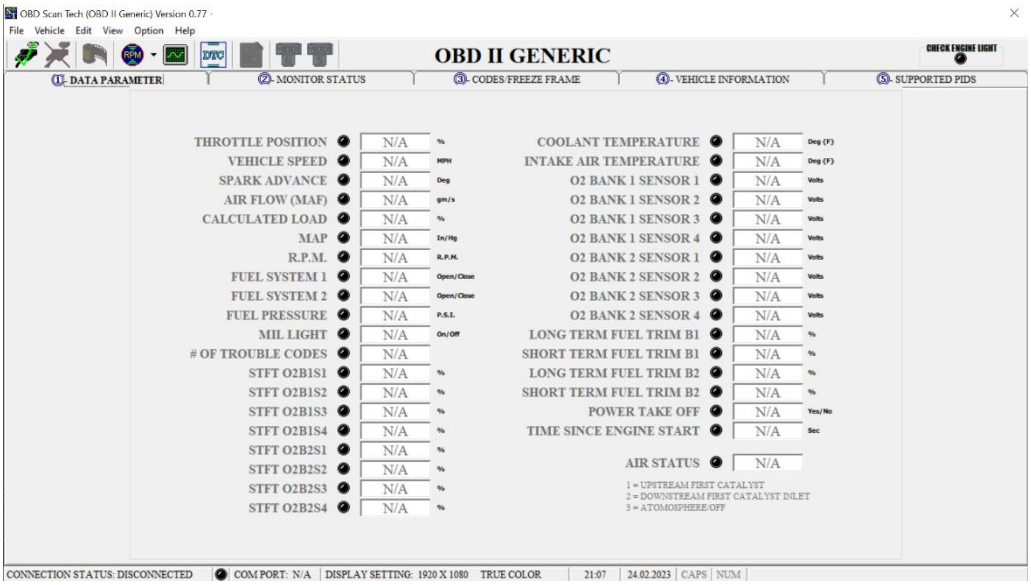


Рисунок 1.3 – Графический пользовательский интерфейс программы OBD Scan Tech версии 0.77

Нефункциональные достоинства программы OBD Scan Tech:

1) функции и вкладки группированы аналогично программе BODAS Service;

2) кнопки, функции которых невыполнимы в данный момент или выполнение функций которых не приведет к ожидаемому результату, являются неактивными и недоступными для нажатия;

3) несмотря на довольно скудный функционал, программа имеет подробную документацию для каждой настройки, функции и каждого вывода;

4) позволяет записывать и сохранять логи, сохранять текущее состояние контроллера в текстовом виде.

Нефункциональные недостатки программы OBD Scan Tech:

1) параметры и ошибки никак не группированы, искать необходимый параметр или ошибку неудобно;

2) интерфейс программы крайне непривлекательный;

3) из-за неправильной разметки некоторые вкладки заграждают другие;

4) способ вывода нельзя настроить, доступен только выбор единиц измерения;

5) программа не позволяет строить графики, т.е. представлять изменение какого-либо параметра во времени для удобного отслеживания возможной неисправности;

6) внешний вид программы недоступен для изменения в настройках.

Вывод: ввиду будущего отказа предприятия от использующейся на данный момент программы диагностики BODAS Service, предприятию необходим графический пользовательский интерфейс программы диагностики, способной заменить BODAS Service, лишенный критических недостатков данной системы и перенявший некоторые нефункциональные достоинства остальных рассмотренных аналогов.

## 1.2 Анализ основных технологических решений

### 1.2.1 Выбор платформы для разработки

На сегодняшний день существует множество платформ для разработки графических интерфейсов программ для персональных компьютеров: Windows Forms, Windows Presentation Foundation, JavaFX, NW.JS, Qt, GTK, Tkinter и другие. Существуют также «обёртки» многих платформ, поддерживающих одни языки программирования, для их применения в разработке на других языках. Так, например, функционал фреймворка Qt имплементирован библиотекой QtJambi для языка Java, библиотекой PyQt для языка Python, библиотекой PHP-Qt для языка PHP; функционал фреймворка GTK имплементирован библиотекой PyGTK для языка Python, библиотекой PHP-GTK для языка PHP и так далее, в то время, как, например, Windows Presentation Foundation поддерживает разработку на любом .NET – совместимом языке программирования, в которые входят: C#, F#, VB.NET, C++, Ruby, Python, Delphi, Lua и многие другие.

При выборе фреймворка для проекта, следует выделять те, для которых предусмотрена поддержка языка C#, и отдавать им предпочтение, так как

разрабатываемый интерфейс будет использовать предоставленные предприятием функции и алгоритмы обмена информацией с контроллерами, написанные на языке C#.

Рассмотрим наиболее популярные из перечисленных фреймворки, выделим их достоинства и недостатки.

**Windows Forms** является предшественником фреймворка Windows Presentation Foundation (WPF). Подход к разработке программ на Windows Forms основывается на графическом интерфейсе GDI (Graphics Device Interface, Graphical Device Interface) – интерфейсе операционной системы Windows для представления графических объектов и передачи их на устройства отображения, такие как мониторы и принтеры.

GDI отвечает за отрисовку линий и кривых, отображение шрифтов и обработку палитры. Задача отрисовки окон, меню и т.п. закреплена за пользовательской подсистемой, располагающейся в user32.dll и основывающейся на GDI.

Достоинства Windows Forms:

1) низкий порог вхождения, простота разработки;

2) подробная документация, большое количество учебных материалов, эскизов, шаблонов и готовых элементов в открытом доступе;

3) поддержка и обновление фреймворка компанией-разработчиком, не смотря на сравнительно большой возраст технологии.

Недостатки Windows Forms:

1) описание любых элементов интерфейса осуществляется исключительно кодом на одном из поддерживаемых языков программирования;

2) сложность разделения визуальной и логической частей интерфейса, связанная с отсутствием вспомогательных инструментов описания элементов интерфейса;

3) измерения производятся в пикселях, т.е. самым простым, примитивным методом, следовательно, зависимость от разрешения экрана;

4) отсутствие возможности аппаратного ускорения в связи с использованием базового графического интерфейса системы Windows – GDI.

Графическая составляющая фреймворка **Qt**, в отличии от Windows Forms, основывается на более продвинутых графических интерфейсах OpenGL и Vulkan, активно применяющихся в разработке требовательных к аппаратной части игр. Отличительная особенность – использование метаобъектного компилятора – предварительной системы обработки исходного кода. Метаобъектная система – часть ядра фреймворка Qt для поддержки в C++ таких возможностей, как сигналы и слоты для коммуникации между объектами в режиме реального времени и динамических свойств системы.

Достоинства Qt:

1) подробная документация, сопровождающаяся большим количеством примеров, исходный код которых содержит подробные комментарии и описание;

2) возможность использования языка разметки XML для создания графической составляющей интерфейса, что упрощает процесс разработки;

3) фреймворк реализован на языке C++, следовательно, при использовании его на языке C++, код будет выполняться быстрее, чем на других фреймворках в связке с другими языками, то есть фреймворк обеспечит лучшее быстродействие;

4) возможность аппаратного ускорения благодаря графическим интерфейсам OpenGL и Vulkan, лежащим в основе графической составляющей Qt.

Недостатки Qt:

1) высокий порог вхождения, сложность освоения фреймворка относительно аналогов

2) отсутствие сборщика мусора, следовательно, вероятность «засорения» памяти оперативно запоминающего устройства неиспользующимися переменными и объектами;

3) из-за обратной совместимости со старыми версиями фреймворка разработчики поддерживают в том числе неоптимальные решения;

4) визуальная непривлекательность существующих элементов графического интерфейса, сложность изменения стилей элементов и создания новых элементов.

Фреймворк **GTK** поддерживает множество графических интерфейсов, таких как OpenGL, EGL, GL ES, GLX, DRI и другие. GTK написан на языке Си, но тем не менее, является объектно-ориентированным. Одной из причин для выбора Си в качестве языка было желание облегчение разработки интерфейсов на других языках программирования.

Достоинства GTK:

1) высокое быстродействие интерфейсов, реализованных с помощью GTK;

2) возможность использования языка разметки XML для создания графической составляющей интерфейса, что упрощает процесс разработки;

3) простота использования, удобный и интуитивно понятный API;

4) возможность применения тем, определяющих внешний вид основных элементов управления, из большого их количества в открытом доступе;

5) поддержка большинства популярных современных языков программирования;

6) возможность аппаратного ускорения.

Недостатки GTK:

1) прекращена поддержка предпочтительного языка C#;

2) необходимость создания элементов управления, похожих на элементы интерфейса программы BODAS Service, реализованной с помощью фреймворка Windows Forms, «с нуля»;

3) меньшая популярность и распространенность, чем у перечисленных выше Windows Forms и Qt, а значит меньшее количество ресурсов и примеров.

**Windows Presentation Foundation (WPF)** [3] - аналог Windows Forms, система для построения клиентских приложений для операционной системы Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык разметки XAML. В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Windows Forms, где используется GDI/GDI+. Весь код транслируется в код для трансляции в DirectX с помощью библиотеки milcore.dll.

Достоинства Windows Presentation Foundation:

1) поддержка языка разметки XAML, упрощающего разработку визуальной составляющей интерфейса и позволяющего разделить визуальную часть интерфейса от логической;

2) независимость от разрешения экрана. Измерения производятся не в пикселях, как в Windows Forms, а в точках на дюйм (принцип векторной графики), что позволяет окну разворачиваться на любом мониторе;

3) возможность выводить свойства элементов в отдельные стили, удобное CSS-подобное описание стилей элементов;

4) возможность аппаратного ускорения, благодаря DirectX;

5) является самым популярным фреймворком для разработки программ для настольных компьютеров под управлением операционной системы Windows, следовательно, имеет огромную базу учебных материалов и примеров. Отдел компании Microsoft регулярно дополняет теорией и примерами и редактирует документацию к фреймворку.

Недостатки Windows Presentation Foundation:

1) сравнительно большой занимаемый программой, использующей фреймворк WPF, объем дискового пространства;

2) невысокая скорость запуска программ.

В рамках выпускной квалификационной работы очень важны такие критерии, как скорость и удобство разработки, отдельно отмеченные у фреймворка Windows Presentation Foundation. Данный фреймворк позволяет разделить визуальную и логическую части проекта, легко изменять шаблоны стилей элементов и создавать на их основе новые.

Одним из основных нефункциональных требований к разрабатываемому интерфейсу является визуальное сходство с интерфейсом используемой на предприятии на данный момент программы диагностики ЭБУ BODAS Service, интерфейс которой был реализован с помощью платформы Windows Forms (WF).

Сравнение рассмотренных фреймворков по выделенным критериям представлено в таблице 1.1.

Таблица 1.1 – сравнение фреймворков по выделенным критериям

| Фреймворк / Критерий | Windows Forms | Qt | GTK | WPF |
|---|---|---|---|---|
| Совместимость с языком C# | + | - | + (поддержка прекращена) | + |
| Аппаратное ускорение | - | + | + | + |
| Объем документации и количество учебного материала | **** | **** | *** | ***** |
| Поддержка языка разметки | - | + (XML) | + (XML) | + (XAML) |

* – оценка по 5-балльной шкале

Исходя из сравнительной таблицы и родства фреймворков Windows Forms и WPF, мной была выбрана платформа Windows Presentation Foundation (WPF), как более совершенный наследник платформы Windows Forms и наиболее подходящая для реализации выпускной квалификационной работы.

### 1.2.2 Выбор языка программирования

Как было сказано выше, фреймворк Windows Presentation Foundation поддерживает разработку на любом .NET – совместимом языке программирования, в которые входят: C#, F#, VB.NET, C++, Ruby, Python, Delphi, Lua и многие другие. Так как разрабатываемый интерфейс будет использовать предоставленные предприятием функции и алгоритмы обмена информацией с контроллерами, написанные на языке C#, и бОльшая часть обучающих материалов и официальной документации к фреймворку WPF написана на языке C#, был сделан выбор в пользу языка C#.

### 1.2.3 Выбор шаблона проектирования

Хотя каждый из шаблонов имеет довольно много отличий, их цели похожи: отделить представление (View) от кода логики (Presenter, Controller, ViewModel и т. д.) и кода обработки данных (Model). Это позволяет каждому из них работать самостоятельно. Например, можно изменить внешний вид и стиль приложения, не затрагивая логику и данные.

Существует 3 основных шаблона проектирования программ с графическим пользовательским интерфейсом: Model-View-Controller (MVC), Model-View-Presenter (MVP) и Model-View-ViewModel (MVVM), а также смешанные шаблоны, например, MVPVM, MVCVM и другие [4].

Модель – это набор объектов или логика приложения. Модель независима от контроллеров / презентеров / моделей представления и представлений [5].

Представление отображает данные модели в пользовательском интерфейсе и не может напрямую влиять на модель.

В шаблоне **MVC** связующим звеном между моделью и представлением выступает контроллер [5]. Контроллер определяет, какое представление должно быть отображено в данный момент, реагируя на события представления. Возможно несколько представлений только для одного контроллера. Схема шаблона MVC представлена на рисунке 1.4.

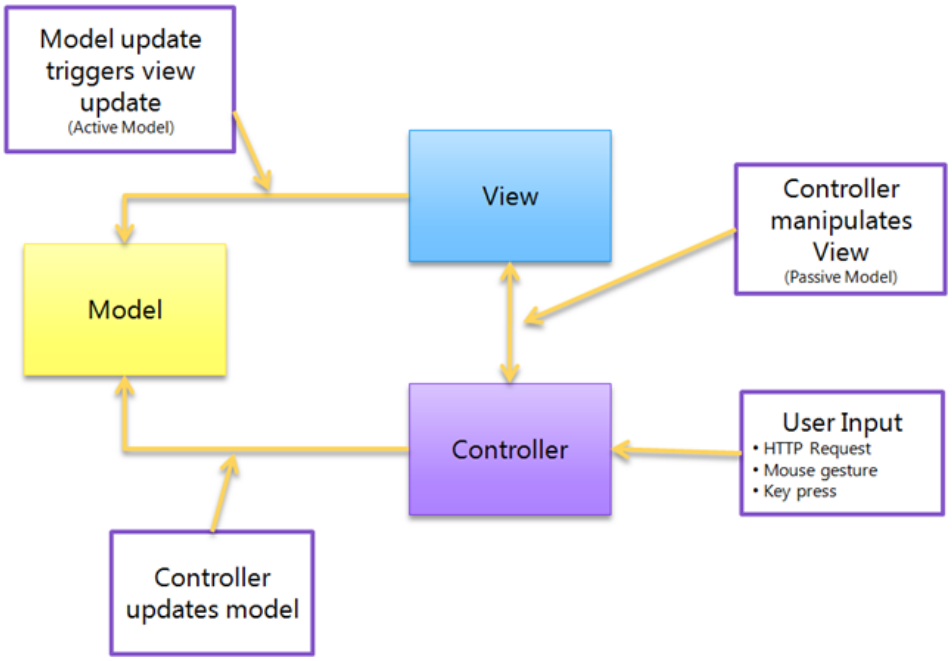


Рисунок 1.4 – Схема шаблона MVC

В шаблоне **MVP** модель и представление связывает презентер [5]. Когда представление уведомляет презентер, что пользователь что-то сделал, презентер принимает решение об обновлении модели и синхронизирует все изменения между моделью и представлением. Презентер взаимодействует с представлением через интерфейс представления. На рисунке 1.5 представлена схема шаблона MVP.

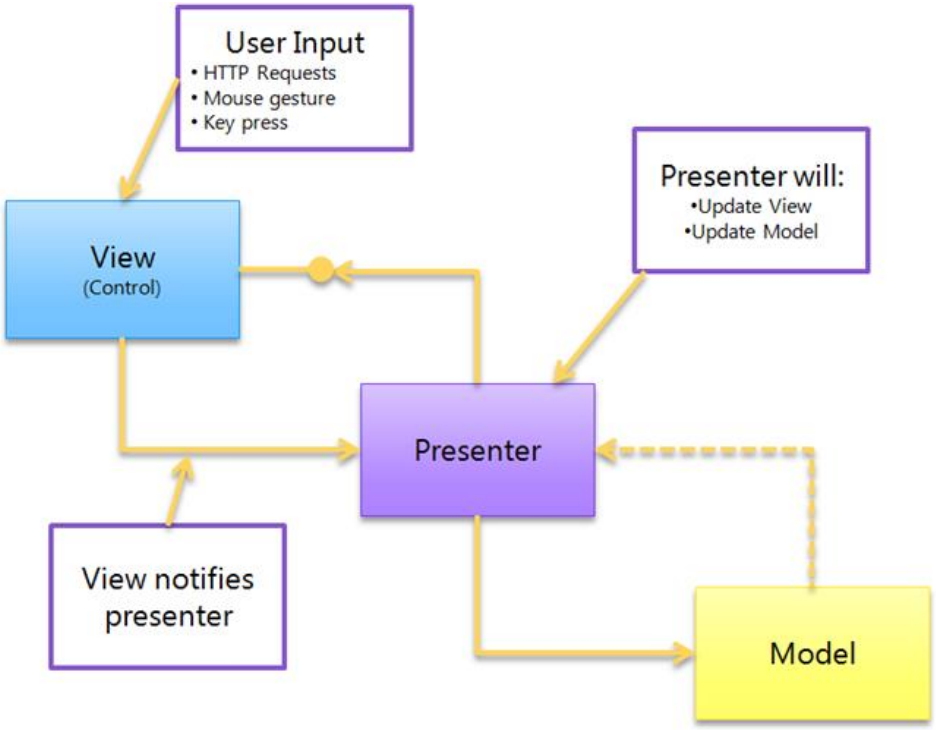


Рисунок 1.5 – Схема шаблона MVP

В шаблоне **MVVM** связующим звеном между моделью и представлением выступает модель представления [5]. Модель представления – это абстракция представления. Обычно это означает, что свойства представления совпадают со свойствами модели представления. Модель представления не имеет ссылки на интерфейс представления (IView), изменение ее состояния автоматически изменяет представление и наоборот, поскольку используется механизм связывания данных (Bindings). Обычно одна модель представления связана с одним представлением. Схема шаблона MVVM представлена на рисунке 1.6.

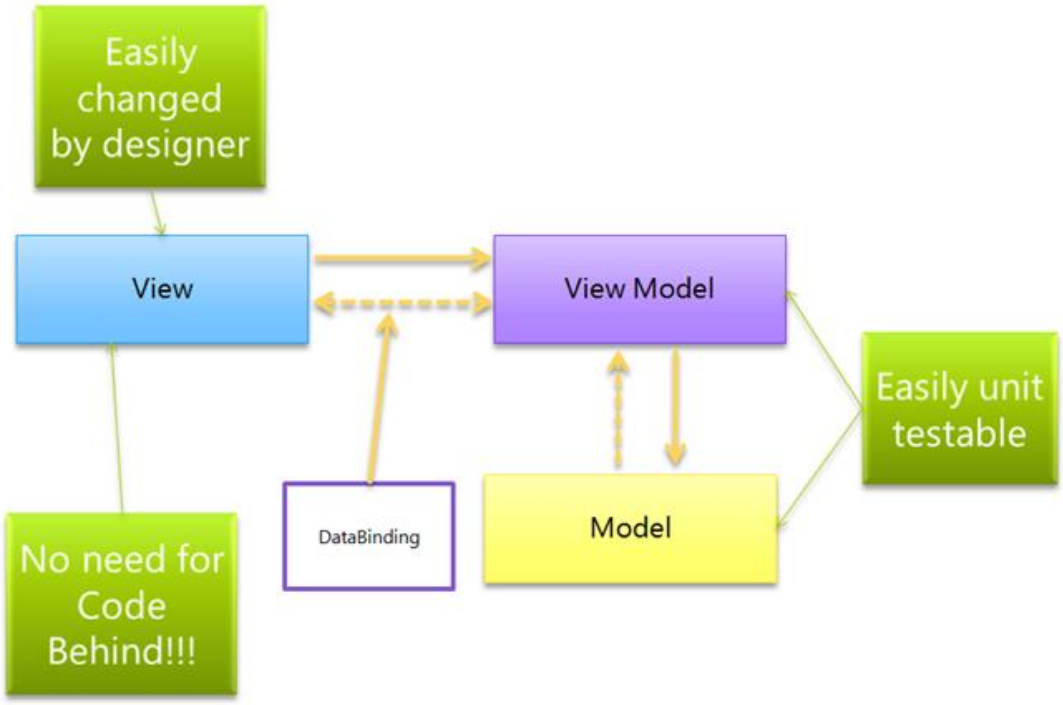


Рисунок 1.6 – Схема шаблона MVVM

Так как технология WPF позволяет связывать данные представления и модели представления, то есть исключает необходимость реализовывать специальные интерфейсы представлений IView, что значительно упрощает и ускоряет разработку, а шаблон MVVM основывается именно на этом механизме связывания данных, для проекта самым оптимальным будет использование шаблона MVVM.

### 1.2.4 Выбор среды разработки

Исходя из выбранных раннее фреймворка и языка программирования, необходимо, чтобы среда разработки (Integrated development environment – IDE) поддерживала технологию WPF и язык C#. К таким относятся: Visual Studio, Visual Studio Code, JetBrains Rider. Рассмотрим достоинства и недостатки каждой из перечисленных среды разработки и составим сравнительную таблицу.

Интегрированная среда разработки **Visual Studio** является творческой стартовой площадкой, которую можно использовать для редактирования, отладки и сборки кода, а также для публикации приложения. В дополнение к стандартному редактору и отладчику, предоставляемых большинством интегрированных сред разработки, Visual Studio включает компиляторы, средства завершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки программного обеспечения.

Достоинства Visual Studio:

1) рефакторинг кода дает возможность интеллектуального переименования переменных, извлечения одной или нескольких строк кода в новый метод и изменения порядка параметров методов;

2) IntelliSense позволяет отображать сведения о коде непосредственно в редакторе, которая умеет перехватывать ошибки, предлагать правильные варианты написания кода и в некоторых случаях автоматически создавать небольшие отрывки кода. По сути, это встроенная в редактор базовая документация, которая избавляет от необходимости искать информацию в других источниках;

3) единый компонент поиска находит нужные функции среды разработки или элементы кода, чем упрощает работу, исключая необходимость поиска вручную;

4) CodeLens помогает находить ссылки на код, изменения кода, связанные с кодом ошибки, рабочие элементы, проверки кода и модульные тесты не выходя из редактора;

5) горячая перезагрузка позволяет редактировать файлы кода приложения и сразу же применять изменения кода к работающему приложению;

6) встроенный менеджер пакетов NuGet предоставляет доступ к огромной базе, состоящей из более чем 100 000 уникальных пакетов в виде устанавливаемых в проект библиотек DLL для проектов .NET;

7) XAML Designer позволяет создавать и редактировать интерфейс программы без написания кода;

8) интуитивный стиль кодирования. По умолчанию Visual Studio форматирует код по мере его ввода, автоматически вставляя необходимые отступы, открывая и закрывая скобки, применяя цветовое кодирование, выделяя ошибки и предупреждения волнистыми линиями и многое другое. Параметры автоматического форматирования можно настраивать;

9) многофункциональный отладчик, позволяющий расставлять точки останова, просматривать текущее состояние переменных в памяти и деревьев вызова функций и многое другое;

10) бесплатная версия Visual Studio Community Edition включает все необходимые для реализации интерфейса функции и инструменты.

Недостатки Visual Studio:

1) большой занимаемый средой разработки объем дискового пространства;

2) сравнительно невысокая скорость работы, оправдываемая мощностью функционала систем среды.

**Visual Studio Code**, фактически, является редактором кода с большой базой расширений, максимально приближающей его к полноценной среде разработки с отличием, заключающимся лишь в отсутствии интегрированного компилятора.

Достоинства Visual Studio Code:

1) присутствует рефакторинг кода;

2) IntelliSense, предлагающий правильные варианты написания кода и в некоторых случаях автоматически создающий небольшие отрывки кода. Однако, менее функционален и интуитивен, чем в Visual Studio, работает не со всеми функциями или иногда работает неправильно;

3) наличие отладчика, но менее функционального, чем в Visual Studio;

4) очень высокая скорость работы;

5) доступность менеджера пакетов NuGet через расширения, а, значит, и большой базы устанавливаемых с его помощью библиотек;

6) доступность автоматического форматирования через расширения, не уступающего форматированию в Visual Studio;

7) небольшой занимаемый объем дискового пространства;

8) поставляется бесплатно.

Недостатки Visual Studio Code:

1) многие из расширений, необходимых для выполнения задания ВКР, неофициальные и поддерживаются сторонними разработчиками;

2) расширение PowerShell Pro Tools, позволяющее создавать и редактировать элементы интерфейса без написания кода на языке программирования или языке разметки, менее функционально, чем встроенный в Visual Studio XAML Designer;

3) возможности и корректность работы IntelliSense сильно уступают встроенному IntelliSense в Visual Studio.

**JetBrains Rider –** это интегрированная среда для .NET-разработки, сочетает возможности ReSharper в части анализа .NET-кода с функциональностью IntelliJ-платформы. ReSharper (R#), в свою очередь, – расширение, разработанное компанией JetBrains, для повышения продуктивности работы в Microsoft Visual Studio. R# проводит статический анализ кода (поиск ошибок в коде до компиляции) в масштабе всего решения, предусматривает дополнительные средства автозаполнения, навигации, поиска, подсветки синтаксиса, форматирования, оптимизации и генерации кода и многое другое.

Достоинства JetBrains Rider:

1) статический анализ кода с подсветкой ошибок и неоптимальных по тем или иным причинам фрагментов;

2) Quick-Fixes – возможность быстрого исправления ошибок и замечаний, удаление избыточных элементов кода;

3) Context Actions — быстрые преобразования кода по типичным сценариям;

4) мгновенный поиск функций и настроек среды разработки или элементов кода;

5) улучшенный вариант IntelliSense, учитывающий контекст, т.е. предлагающий варианты написания кода, на основе существующего в проекте;

6) многофункциональный отладчик, почти не уступающий по функционалу отладчику в Visual Studio;

7) Value Analysis — анализ control flow и data flow внутри функций, выявляющий избыточные проверки, присваивания и логические операторы, ветки кода, недостижимые ни при каких входных данных, показывающий предупреждения о возможных исключениях при вызове методов и операторов;

8) внушительный объем правил форматирования, инспекций кода и рефакторингов, в сравнении со всеми аналогами.

Недостатки JetBrains Rider:

1) высокая цена – US $169.00 в год за среду разработки и плагин ReSharper;

2) средство предварительного просмотра XAML позволяет лишь просматривать интерфейс без возможности создания и редактирования его элементов управления без написания кода;

3) сложность в освоении, если ранее не было опыта работы с продуктами компании JetBrains.

Сравнение рассмотренных сред разработки по выделенным критериям представлено в таблице 1.2.

Таблица 1.2 – сравнение IDE по выделенным критериям

| IDE / Критерий | MS Visual Studio | MS Visual Studio Code | JetBrains Rider |
|---|---|---|---|
| Возможность создания и редактирования элементов интерфейса без написания кода | + | + | - |
| Удобство написания кода (автоматическое форматирование, рефакторинг и т. д.) | **** | *** | ***** |
| Скорость работы среды и поиска | *** | ***** | ***** |
| Бесплатность | + | + | - |

* – оценка по 5-балльной шкале

Таким образом, среда JetBrains Rider больше ориентирована на простоту, правильность и скорость написания кода, в то время как Visual Studio – на функциональные возможности. Visual Studio Code, хоть и включает весь необходимый функционал для реализации проекта благодаря расширениям, но заметно уступает аналогам по возможностям этого функционала. Было бы целесообразным выбрать среду Rider от компании JetBrains, т.к. она позволит разработать интерфейс в кратчайшие сроки, но очень высокая цена за годовую лицензию заставила сделать выбор в пользу бесплатной версии Visual Studio Community среды Visual Studio.

**Вывод.**

Для разработки графического пользовательского интерфейса программы диагностики ЭБУ самоходных машин были выбраны следующие технологии: WPF – для создания архитектуры интерфейса, XAML – для создания представления (дизайна) интерфейса, C# – для написания логики взаимодействия представлений с моделями представлений и предоставленных предприятием функций взаимодействия с контроллером, среда разработки Visual Studio.

# 2 ФОРМИРОВАНИЕ ОСНОВНЫХ ТРЕБОВАНИЙ

## 2.1 Функциональные требования

В результате аналитического обзора предметной области были выявлены следующие общие функциональные требования к интерфейсу:

– главное меню программы, включающее кнопки с выпадающими списками: «File» – сохранение, загрузка, создание диагностического проекта и т.д.; «Actions» – кнопки панели инструментов Toolbar в виде компактного списка без картинок и описания; «View» – возможность включать и отключать отображение панелей Toolbar и Taskbar; «Tools» – кнопки: «Options» – окно настроек, «Interface» – выбор интерфейса, с помощью которого осуществляется подключение к контроллеру, «Language» – выбор языка контроллера; «Help» – кнопки, открывающие окна: документации, информации о лицензии, информации о программе, правах и разработчике;

– панель инструментов Toolbar, включающая следующий набор кнопок: «Scan» – обнаружить подключенный контроллера, перевод необходимых кнопок в активное состояние, отображение вкладок для работы, получение информации с контроллера; «Break» – разорвать соединение с контроллером; «Get data» - получить информацию с контроллера; «Save» – сохранить и записать в контроллер значения параметров, измененные пользователем; «Default» – выставить значения всех параметров в значения по умолчанию;
«Reinit» – выставить значения все параметров в значение при подключении; «Reset» – перезагрузить контроллер и подключиться снова; «Help» – открыть окно документации;

– должны быть реализованы вкладки, доступные из списка вкладок в левой части интерфейса после подключения к контроллеру: «Device info» – вывод основной информации о контроллере, его программном обеспечении и т.п.; «Parameter» – вывод всех параметров контроллера; «Processdata» – вывод данных процесса; «I/O status view» – текущие значения на входах и

выходах (портах) контроллера; «Error messages» – вывод всех ошибок контроллера;

– справа от списка вкладок поле с представлением активной вкладки;

– панель Taskbar в нижней части интерфейса, содержащей информацию о текущем подключении или его отсутствии.

Функциональные требования к отдельным вкладкам интерфейса представлены в таблице 2.1.

Таблица 2.1 – функциональные требования к вкладкам интерфейса

| Название вкладки | Функциональные требования |
|---|---|
| Parameter | – Раскрывающиеся и закрывающиеся списки групп параметров (parameter Submenu), объединенные в меню параметров (parameterMenu);<br>– в открытой группе представлены параметры этой группы в виде списка, содержащего: индекс параметра, название, текущее значение в поле для его изменения, кнопку для сброса значения параметра в значение по умолчанию;<br>– возможность выбора значения параметра из списка значений (для типа list) или диапазона значений (для типа number), кнопкой On/off (для типа switch);<br>– для параметров с возможностью выбора значения из диапазона значений (типы number и hex number) предусмотреть окно для ручного ввода значения;<br>– при наведении курсора или выборе клавиатурой появляется небольшое всплывающее окно (popup), содержащее информацию о всех полях параметра, таких как: тип, текущее значение, минимальное и максимальное значения, значение по умолчанию и т. д.;<br>– интервал обновления значений по умолчанию равен 3000 мс;<br>– поле для вывода текущего интервала обновления в мс. |
| Processdata | – Вывод параметров и их значений аналогично вкладке Parameter;<br>– интервал обновления значений параметров по умолчанию равен 250 мс;<br>– поле для вывода текущего интервала обновления в мс;<br>– при наведении курсора или выборе клавиатурой появляется небольшое всплывающее окно (popup), содержащее информацию о всех полях параметра. |
| I/O Status | – Вывод всей полученной информации о текущем состоянии портов контроллера;<br>– интервал обновления значений по умолчанию равен 500 мс;<br>– поле для вывода текущего интервала обновления в мс. |
| Custom view | – Выбор вкладок для их отображения в одном представлении и одновременной работы с данными каждой выбранной вкладки. |

Продолжение таблицы 2.1

| | |
|---|---|
| Error messages | – Разделение представления вкладки на 2 равные части по вертикали: несохраненные ошибки (Active errors), сохраненные ошибки (Saved errors);<br>– списки ошибок в виде таблицы со столбцами: индекс, код ошибки, появления, сообщение ошибки;<br>– получение подробного описания ошибки при наведении курсора, если такое имеется;<br>– интервал обновления значений по умолчанию равен 3000 мс;<br>– поле для вывода текущего интервала обновления в мс;<br>– возможность добавления активной ошибки в сохраненные. |

Сформированные функциональные требования можно графически описать диаграммой вариантов использования, представленную на рисунке 2.1. Разделение пользователей на категории не предусматривается, так как использование программы предполагается исключительно мастером.


Рисунок 2.1 – Диаграмма вариантов использования

## 2.2 Нефункциональные требования

В результате аналитического обзора предметной области были выявлены следующие нефункциональные требования:

– совместимость со всеми выпусками операционных систем Windows 7 и Windows 10;

– внешняя и структурная схожесть с программой BODAS Service;

– шрифт всех элементов – «Segoe UI» 14 кегля, черного цвета;

– возможность изменять ширину поля со списком вкладок и поля с представлением активной вкладки перетаскиванием границы (элемента GridSplitter) между ними;

– автоматическое появление элементов ScrollBar для горизонтальной и/или вертикальной прокрутки поля или представления, если требуется;

– кнопки панели инструментов Toolbar имеют цветные стилизованные картинки, дающие представление пользователю о назначении кнопки;

– адаптация интерфейса под любой доступный размер окна.

# 3 ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА

## 3.1 Архитектура предлагаемого решения

При разработке приложения используется архитектура MVVM, позволяющая, как было сказано раннее, связывать данные представления и модели представления без использования вспомогательных интерфейсов представления IView или контроллеров Controller. Представлением называется часть графического интерфейса, которая посредством взаимодействия пользователя с интерфейсом может заменяться другой частью графического интерфейса, т.е. другим представлением [6].

Для удобства проект структурно разделен на несколько частей:

– Icons – каталог, который содержит все использующиеся в пользовательском интерфейсе изображения и картинки;

– MVVM – каталог, который содержит:

– Models – каталог с моделями данных, представляющими из себя классы, используемые для вывода информации моделями представлений в представления;

– ViewModels – каталог с моделями представлений;

– Views – каталог с представлениями и связанными с ними файлами отделенного кода, называемыми Code-Behind, необходимыми для инициализации представления, но, как правило, не использующиеся для какой-либо логики или связывания в паттерне MVVM, т.к. всем управляют модели представлений;

– файлы BaseViewModel.cs, ObservableObject.cs и RelayCommand.cs с необходимыми для реализации паттерна MVVM классами BaseViewModel, ObservableObject и RelayCommand соответстенно;

– Styles – каталог, который содержит файлы расширения .xaml с реализованными стилями границ, управляющих элементов и т.д.;

– файл App.xaml, выполняющий функцию включения в ресурсы проекта всех стилей, привязки моделей представлений к представлениям и задания представления, которое будет показано при запуске программы;

– AssemblyInfo.cs – файл, предоставляющий возможность указать внешние ресурсы для проекта.

Интерфейс имеет 4 окна: главное окно MainWindow, окно настроек OptionsWindow, окно выбора языка интерфейса и языка контроллера LanguageSelectionWindow и окно выбора интерфейса связи с контроллером InterfaceConfigurationWindow. Главное окно содержит меню Menu, панель инструментов Toolbar, панель доступных вкладок Tabs, ContentControl с представлением активной вкладки и панель задач Taskbar.

Для реализации заявленного функционала потребуется 28 представлений, имеющих собственный графический интерфейс.

В качестве механизма переключения представлений было решено использовать привязку Binding [7] к необходимой модели представления, связанную с соответствующим представлением [8], для вкладок главного окна и окна настроек, и использование элемента TabControl с жесткой привязкой его элементов TabItem к нужным представлениям для дочерних представлений вкладок Parameter, Processdata, I/O Status view и Custom view. Следовательно, каждому представлению будет соответствовать одна связанная с ним модель представления, кроме тех, что будут использоваться в элементе TabControl.

Для описания диаграмм и схем используется язык графического описания для объектного моделирования UML.

Схема архитектуры представлений и связанных с ними моделей представлений, иллюстрирующая включение представления в то или иное окно или другое представление, представлена на рисунке 3.1.

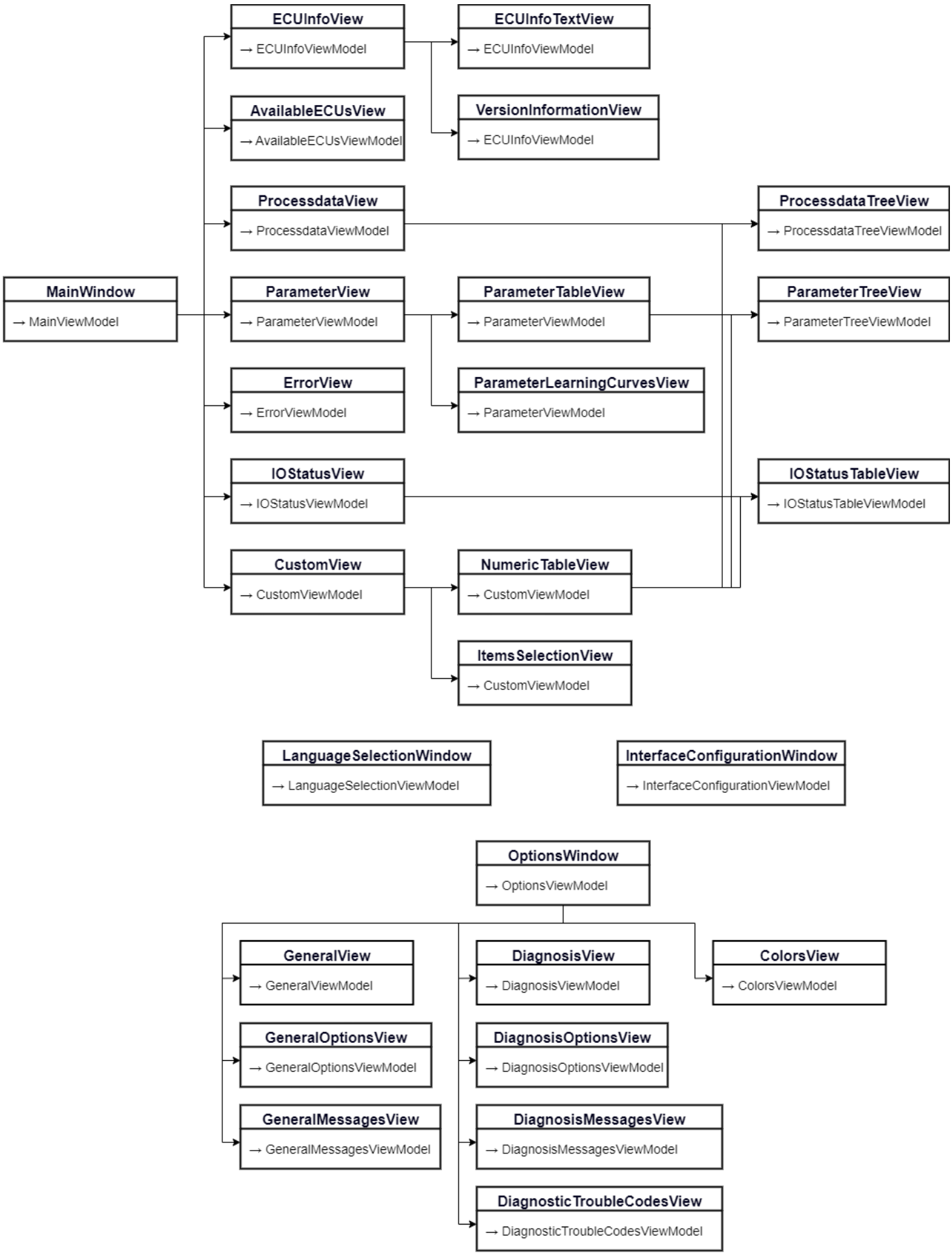


Рисунок 3.1 – Схема представлений

Назначение всех представлений и окон приведены в таблице 3.1.

Таблица 3.1 – Представления и окна разрабатываемого интерфейса

| Название окна / представления | Назначение |
|---|---|
| MainWindow | Главное окно программы. Возможность подключиться, отключиться и переподключиться к контроллеру, сохранить изменения параметров в память контроллера, выставить все значения в значение по умолчанию, отобразить активное представление, открыть любое другое окно и т.д. |
| DefaultView | Представление, позволяющее инициировать поиск и выбрать контроллер для работы. Является отображаемым представлением по умолчанию. |
| ECUInfoView | Представление с основной информацией о подключенном контроллере. |
| ECUInfoTextView | Представление, являющееся частью ECUInfoView. Позволяет изменить отправляемое контроллером его краткое описание. |
| VersionInformationView | Представление, являющееся частью ECUInfoView. Отображает всю доступную информацию о контроллере и его прошивке. |
| ParameterView | Представление для работы с изменяемыми параметрами контроллера. |
| ParameterTableView | Представление, являющееся частью ParameterView. Включает в себя ParameterTreeView. |
| ParameterTreeView | Представление дерева параметров контроллера. |
| ParameterLearningCurvesView | Представление, являющееся частью ParameterView. Отображает значения изменяемых параметров в виде кривых. |
| ProcessdataView | Представление для просмотра данных процесса и их полей. Включает в себя ProcessdataTreeView. |
| ProcessdataTreeView | Предсталение дерева данных процесса. |
| IOStatusView | Представление с информацией о портах контроллера. Включает в себя IOStatusTableView. |
| IOStatusTableView | Представление таблицы портов контроллера. |
| CustomView | Представление, позволяющее группировать несколько вкладок в одну и работать с данными каждой выбранной вкладки. |
| ErrorView | Представление с активными и сохраненными ошибками. |
| LanguageSelectionWindow | Окно выбора языка пользовательского интерфейса и языка контроллера. |
| InterfaceConfigurationWindow | Окно выбора интерфейса подключения. |
| OptionsWindow | Окно настроек. |
| GeneralView | Представление с описанием общей группы настроек GeneralOptionsView и GeneralMessagesView. |
| GeneralOptionsView | Представление с возможностью сбросить настройки, настройкой пути к данным программы, проверки обновлений и др. |

Продолжение таблицы 3.1

| | |
|---|---|
| GeneralMessagesView | Представление, позволяющее настроить некоторые неспецифические ошибки и сообщения с предупреждением. |
| DiagnosisView | Представление с описанием группы диагностических настроек DiagnosisOptionsView, DiagnosisMessagesView и DiagnosticTroubleCodesView. |
| DiagnosisOptionsView | Представление с основными настройками модуля диагностики. |
| DiagnosisMessagesView | Представление с настройками отображаемых ошибок и сообщений с предупреждением модуля диагностики. |
| DiagnosticTroubleCodesView | Представление с настройкой отображения ошибок модуля диагностики, непосредственно касающихся его параметров. |
| FlashToolView | Представление с описанием группы настроек модуля записи FlashToolOptionsView и FlashToolMessagesView. |
| FlashToolOptionsView | Представление с основными настройками модуля записи. |
| FlashToolMessagesView | Представление с настройкой отображения ошибок и сообщений с предупреждением модуля записи. |
| ConfiguratorView | Представление с описанием группы настроек конфигуратора ConfiguratorOptionsView и ConfiguratorMessagesView. |
| ConfiguratorOptionsView | Представление с основными настройками конфигуратора. |
| ConfiguratorMessagesView | Представление с настройкой отображения ошибок и сообщений с предупреждением конфигуратора. |
| ColorsView | Представление с возможностью выбора визуальной темы интерфейса, изменения цвета заднего фона, границы и текста полей и элементов управления. |

## 3.2 Описание данных

Входными данными для графического представления являются данные, полученные путем вызова соответствующих предоставленных предприятием функций. Входные данные для вкладок Parameter и Processdata (параметры) представляют из себя объекты структуры Data, представленной в таблице 3.2, данные для вкладки Error – объекты структуры Error, представленной в таблице 3.3.

Таблица 3.2 – Структура Data

| Поле | Тип данных | Описание |
|---|---|---|
| dataIdentifier | short | Идентификатор |
| isAccessible | bool | Доступность |
| name | string | Название |
| valueType | byte | Тип значения |

Продолжение таблицы 3.2

| | | |
|---|---|---|
| minValue | short | Минимальное допустимое значение |
| maxValue | short | Максимальное допустимое значение |
| step | ushort | Шаг (для параметров типа number) |
| unitCode | byte | Код единиц измерения |
| multiplier | ushort | - |
| divisor | ushort | - |
| digits | ushort | Длина значения / количество знаков после запятой |
| accessLevel | byte | Уровень доступа |
| defaultValue | short | Значение по умолчанию |
| eepromPage | byte | Страница в EEPROM |
| eepromAddress | byte | Адрес в EEPROM |
| value | short | Текущее значение |

Таблица 3.3 – Структура Error

| Поле | Тип данных | Описание |
|---|---|---|
| errorCode | ushort | Код ошибки |
| occurence | ushort | Количество появлений |
| parameter | ushort | Параметр |
| timestamp | uint | Время появления |
| description | string | Описание ошибки |

Для удобства вывода параметров вкладок Parameter и Processdata в интерфейс будет создан абстрактный класс Parameter и производные классы: NumberParameter, ListParameter, SwitchParameter, HexNumberParameter, BinaryNumberParameter и BargraphParameter, класс Error для вкладки Error messages, класс IOPort для вкладки I/O Status view. Парсинг полученных параметров в виде объектов структуры Data или структуры Error в объекты упомянутых выше классов будет происходить на этапе вызова функции чтения данных с контроллера (при подключении и переподключении к контроллеру, при нажатии кнопки Get Data и автоматически с заданным интервалом обновления для параметров вкладки Processdata, портов вкладки I/O Status view и ошибок вкладки Error messages).

# 4 РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА

При написании кода проекта использовались источники [9-22].

## 4.1 Реализация стилей и шаблонов элементов управления

Предоставляемые фреймворком WPF элементы управления [9] зачастую имеют неподходящие для реализации интерфейса шаблоны и стили. Для реализации всех окон и представлений было написано в общей сложности 11 шаблонов контейнеров StackPanel / DockPanel, 12 шаблонов текстовых элементов TextBlock, 15 шаблонов кнопок Button / RadioButton / ToggleButton, 9 стилей границ Border для элементов управления, 2 шаблона дерева TreeView и 2 шаблона таблицы DataGrid. Для каждого шаблона и стиля настроены привязки, триггеры данных и триггеры событий.

Исходные коды главных элементов управления представлены в приложении А: исходный код шаблона дерева параметров ParameterTreeView представлен в листинге А.1, шаблона дерева параметров ProcessdataTreeView – в листинге А.2, шаблон таблицы IOStatusTableView – в листинге А.3, шаблон таблицы ErrorTableView – в листинге А.4.

## 4.2 Реализация окон и представлений

Пользовательский интерфейс представляет собой программу для персонального компьютера с 4 окнами: основное окно, окно выбора языка контроллера, окно выбора интерфейса подключения и окно настроек. Главное окно при запуске программы представлено на рисунке 4.1.

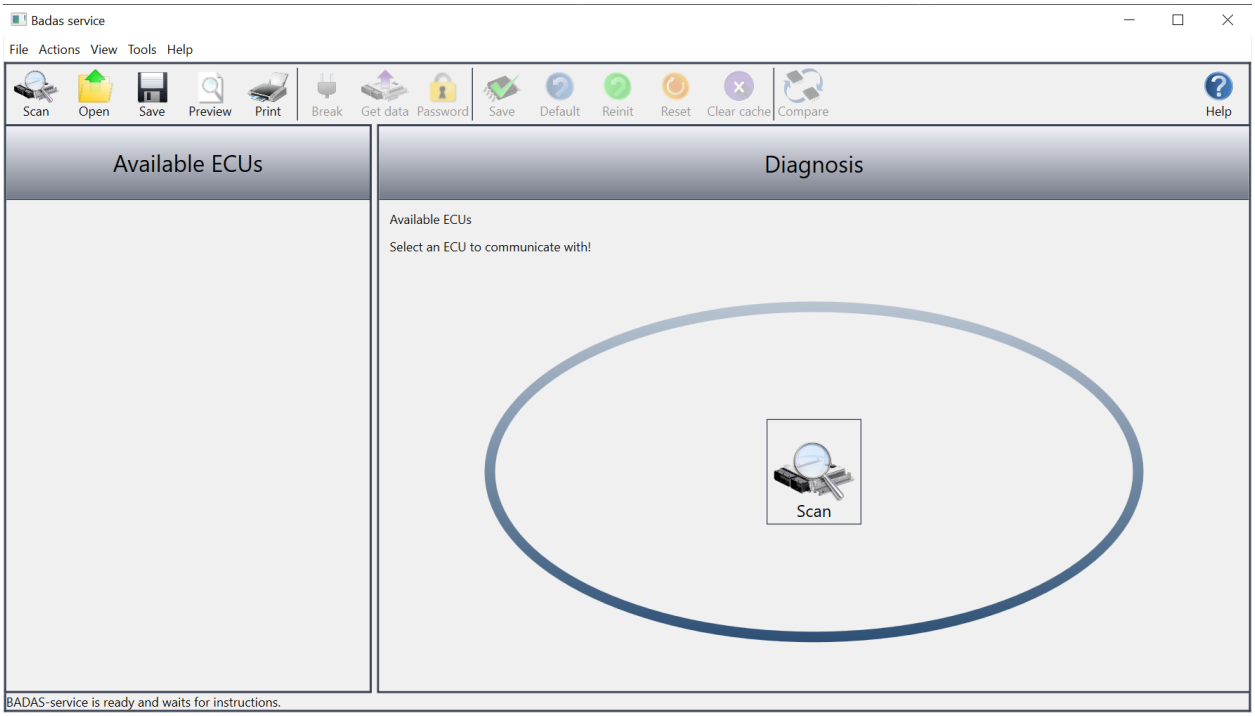


Рисунок 4.1 – Главное окно при запуске

На главном окне расположены: главное меню, предоставляющее пользователю доступ ко всему функционалу интерфейса, панель инструментов с основными необходимыми для работы с контроллером кнопками, панель выбора отображаемой вкладки, поле с представлением активной (выбранной) вкладки. При запуске программы открывается главное окно, панель выбора вкладок скрыта от пользователя (пользователь не запросил подключение к контроллеру), активной и единственной доступной для отображения вкладкой является вкладка подключения и выбора контроллера.

После нажатия пользователем одной из кнопок для выполнения команды Scan, становится доступной работа с подключенным контроллером, если инициализация прошла успешно, или с виртуальным (симулированным) контроллером в ином случае. Главное окно с «обнаруженным» виртуальным контроллером представлено на рисунке 4.2.

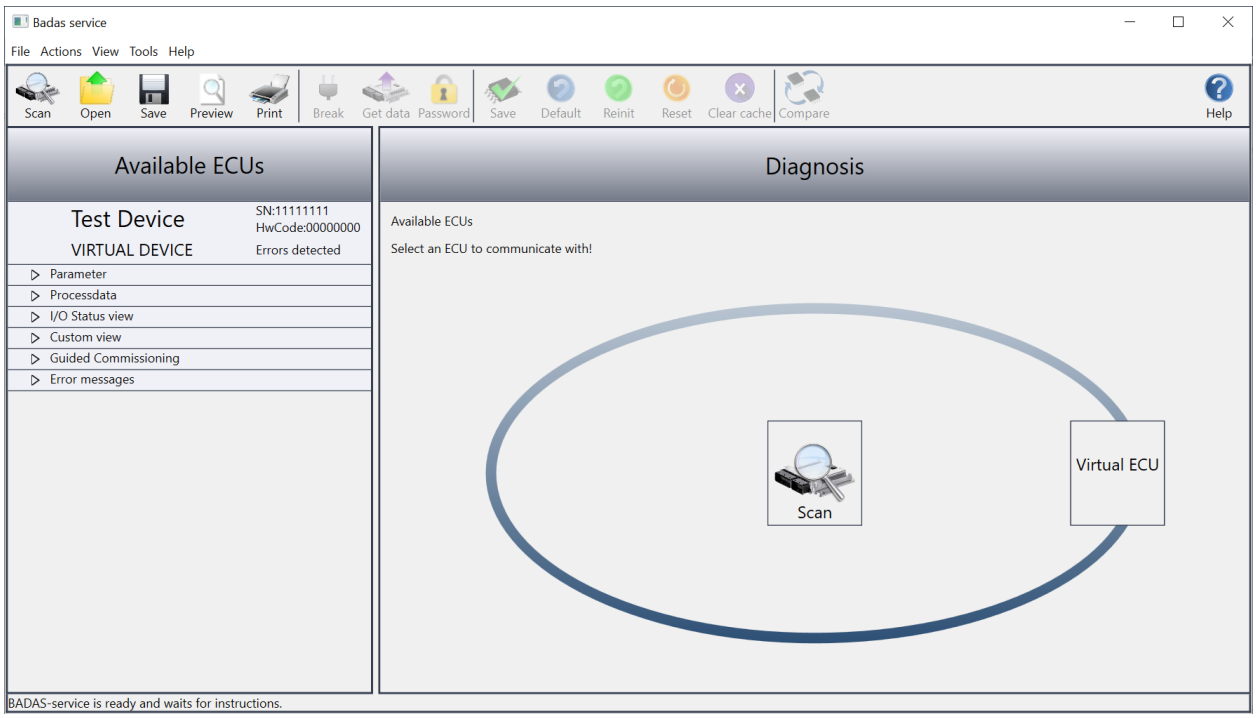


Рисунок 4.2 – Главное окно с «обнаруженным» виртуальным контроллером

После успешной инициализации или эмуляции, вызывается конструктор модели представления каждой вкладки, который вызывает все методы для чтения данных из контроллера (или создания в случае симуляции) и их парсинга в соответствующие классы для вывода в интерфейс.

Открытая вкладка ECUInfo с основной информацией о подключенном контроллере представлена на рисунке 4.3.

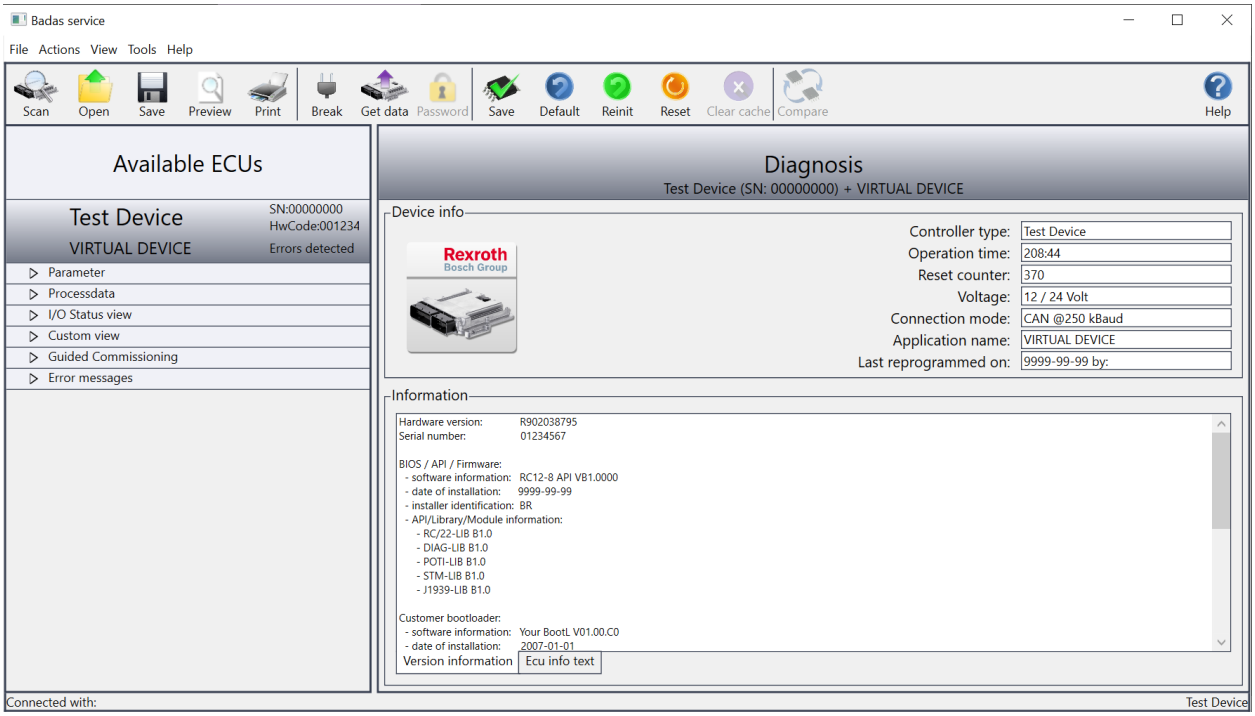


Рисунок 4.3 – Вкладка ECUInfo

На представленном выше рисунке видно, что вкладка содержит поля:

– Controller type (тип контроллера);

– Operation time (время в работе);

– Reset counter (счетчик количества полных перезагрузок);

– Voltage (поддерживаемое напряжение питания);

– Connection mode (тип подключения);

– Application name (название прошивки);

– Last reprogrammed on (дата последней прошивки).

Поле Information содержит 2 подвкладки:

– Version information (вся доступная информация о программном обеспечении контроллера);

– ECU info text (краткое описание контроллера, задается пользователем). Представление подвкладки подержит поле для ввода текста (описания контроллера) и кнопку сохранения описания в EEPROM контроллера, представлено на рисунке 4.4.

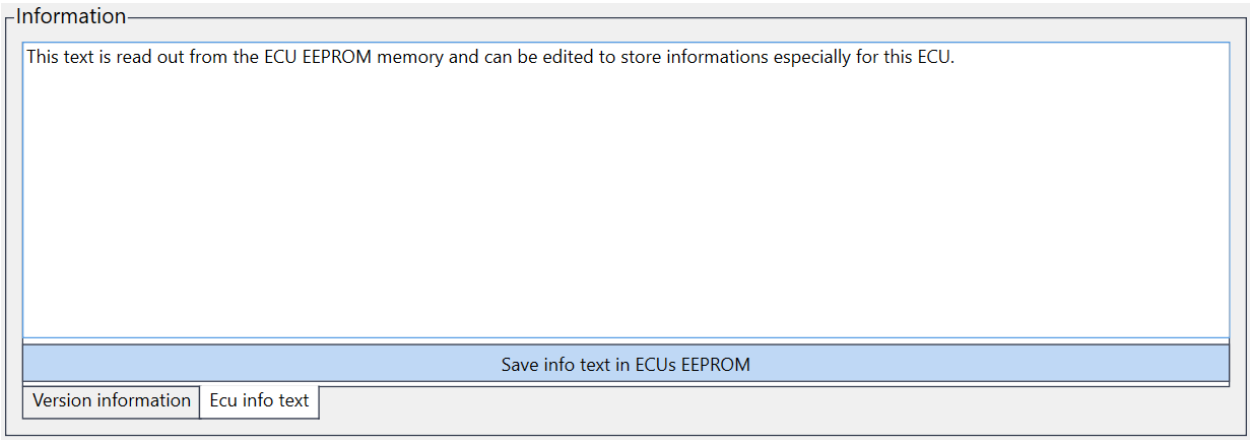


Рисунок 4.4 – Подвкладка ECU info text

Открытая вкладка Parameter для работы с доступными изменяемыми параметрами контроллера показана на рисунке 4.5.

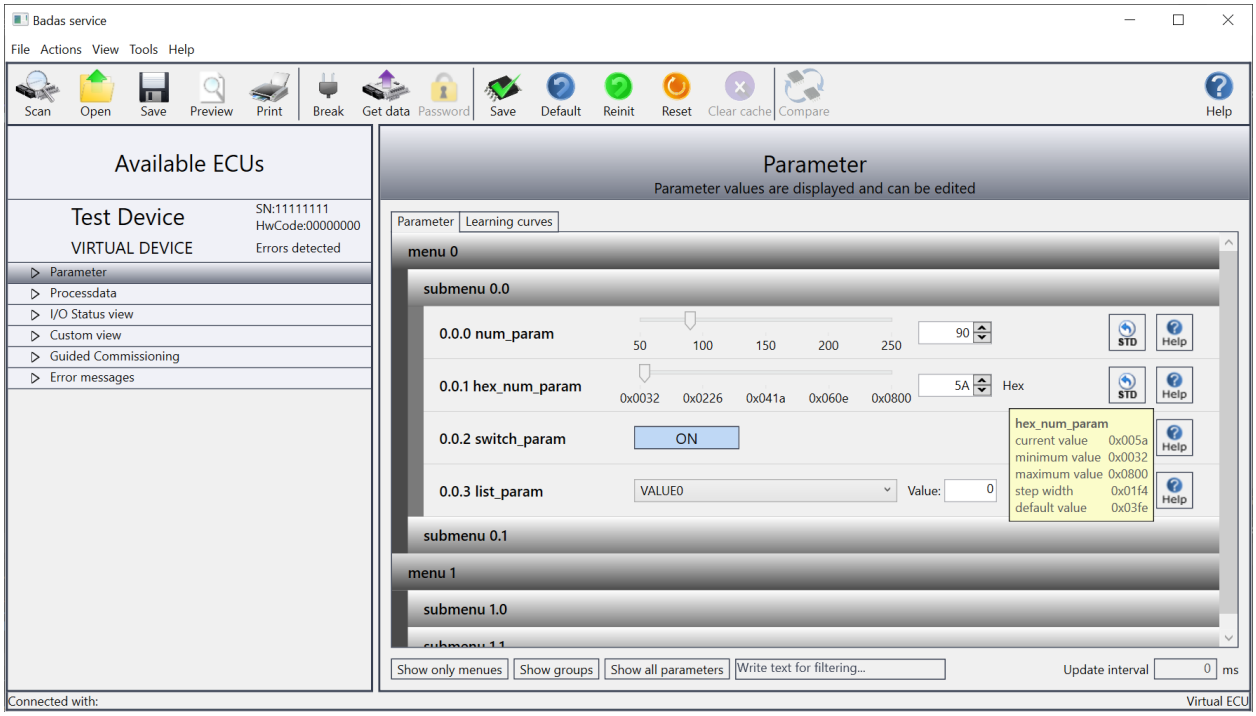


Рисунок 4.5 – Вкладка Parameter

Как видно из рисунка, вкладка Parameter содержит 2 подвкладки: Parameter, в которой представлено дерево параметров, и Learning curves, в которой по команде пользователя запрашиваются и строятся кривые обучения (на данный момент функция недоступна).

На подвкладке Parameter представлено дерево параметров, представляющее из себя выпадающие списки меню, подменю и соответствующих им параметров. Поле каждого параметра содержит: номер,

название, поле с визуализацией значения (слайдер и тестовое поле для параметров числового типа (Number), кнопка включения / выключения (SwitchButton) для параметров булевого типа (Switch), выпадающий список значений и текстовое поле с выбранным значением для параметров типа List), поле с обозначением единицы измерения и кнопки STD (set to default – выставить значение параметра в значение по умолчанию) и Help (описание полей параметров, для всех типов одинаковое). При наведении курсора на параметр появляется всплывающая подсказка с информацией о всех используемых данным типом полях.

Открытая вкладка Processdata для просмотра данных процесса (параметры, доступные только для чтения) показана на рисунке 4.6.

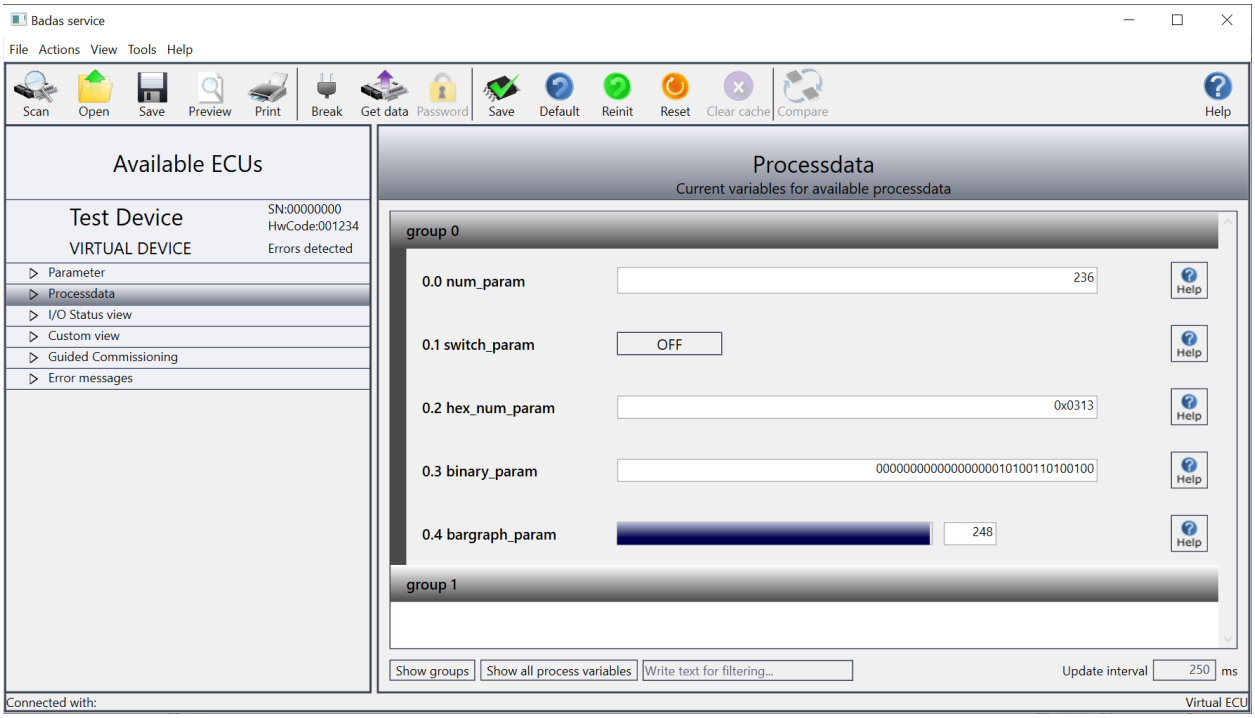


Рисунок 4.6 – Вкладка Processdata

Вкладка содержит дерево параметров, подобно вкладке Parameter, за исключением полей и кнопок, необходимых для изменения значений параметров. Все поля параметров данной вкладки недоступны для изменения. Интервал обновления значений параметров по умолчанию равен 250 мс.

Открытая вкладка I/O Status view для просмотра данных о входах и выходах контроллера показана на рисунке 4.7.

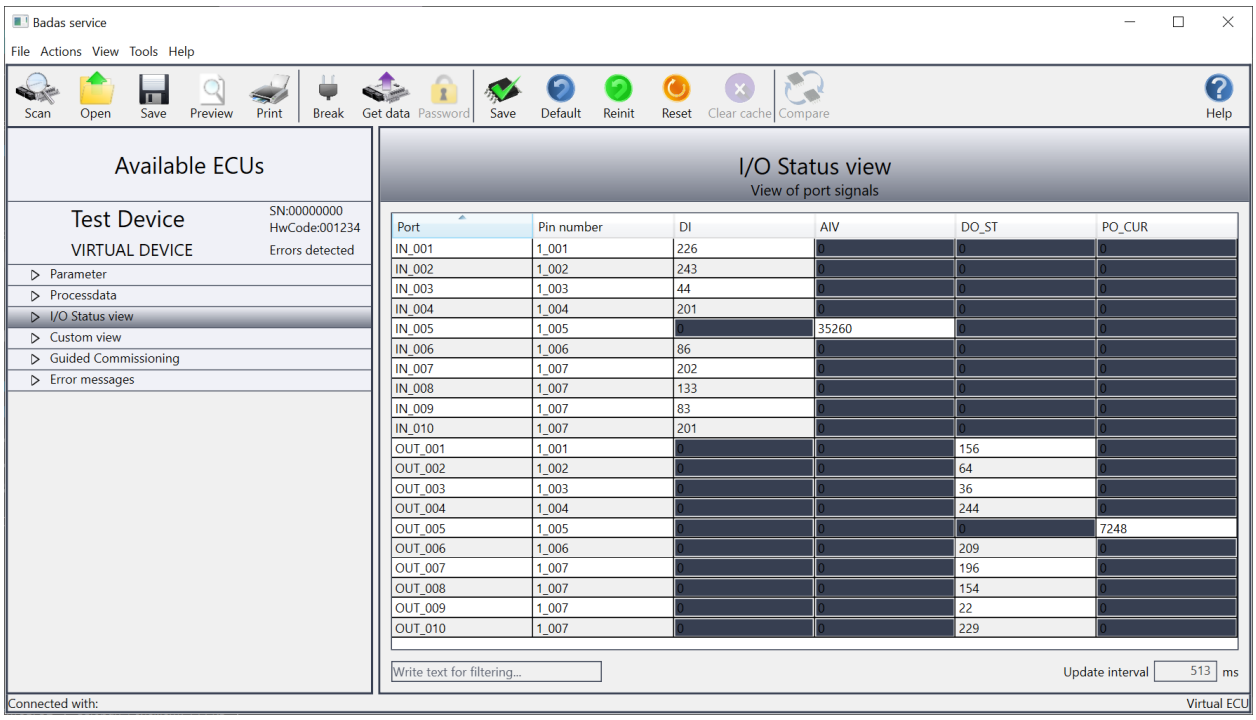


Рисунок 4.7 – Вкладка I/O Status view

Вкладка содержит таблицу, в которой строки – это отдельные порты, а столбцы – их параметры (DI – digital input (цифровой вход), AI – analog input (аналоговый вход) DO – digital output (аналоговый выход) и др.) Интервал обновления данных таблицы по умолчанию равен 500 мс.

Открытая вкладка Custom view для работы одновременно с несколькими выбранными вкладками показана на рисунке 4.8.

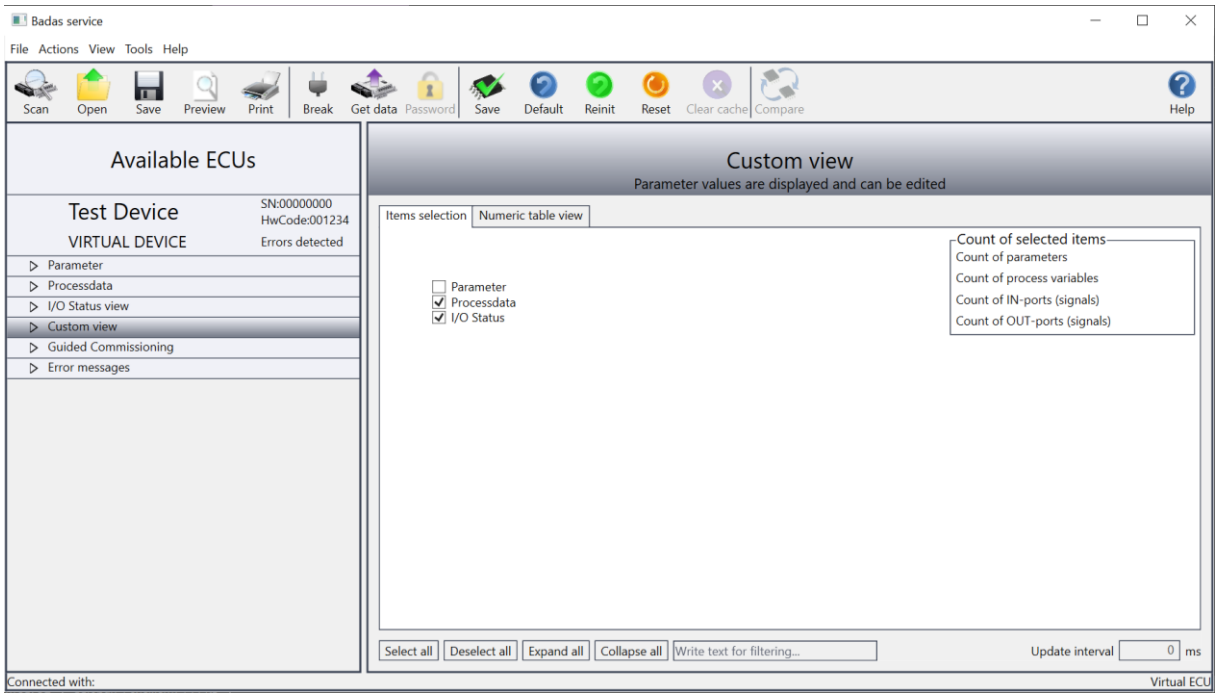


Рисунок 4.8 – Вкладка Custom view

На подвкладке Items selection доступен выбор вкладок для отображения на подвкладке Numeric table view.

При выборе, допустим, Processdata и I/O Status, на подвкладке Numeric table view станут доступны данные вкладок Processdata и I/O Status (рисунок 4.9), которые будут обновляться с наибольшим интервалом из выбранных вкладок (таким образом, при интервале обновления дерева параметров Processdata, равным 250 мс и интервале обновления таблицы I/O Status, равным 500 мс, будет выбран интервал в 500 мс).

Подвкладка Numeric table view вкладки Custom view использует те же модели представления, что и вкладки Parameter, Processdata и I/O Status view, следовательно, любые изменения данных отразятся и в остальных представлениях.

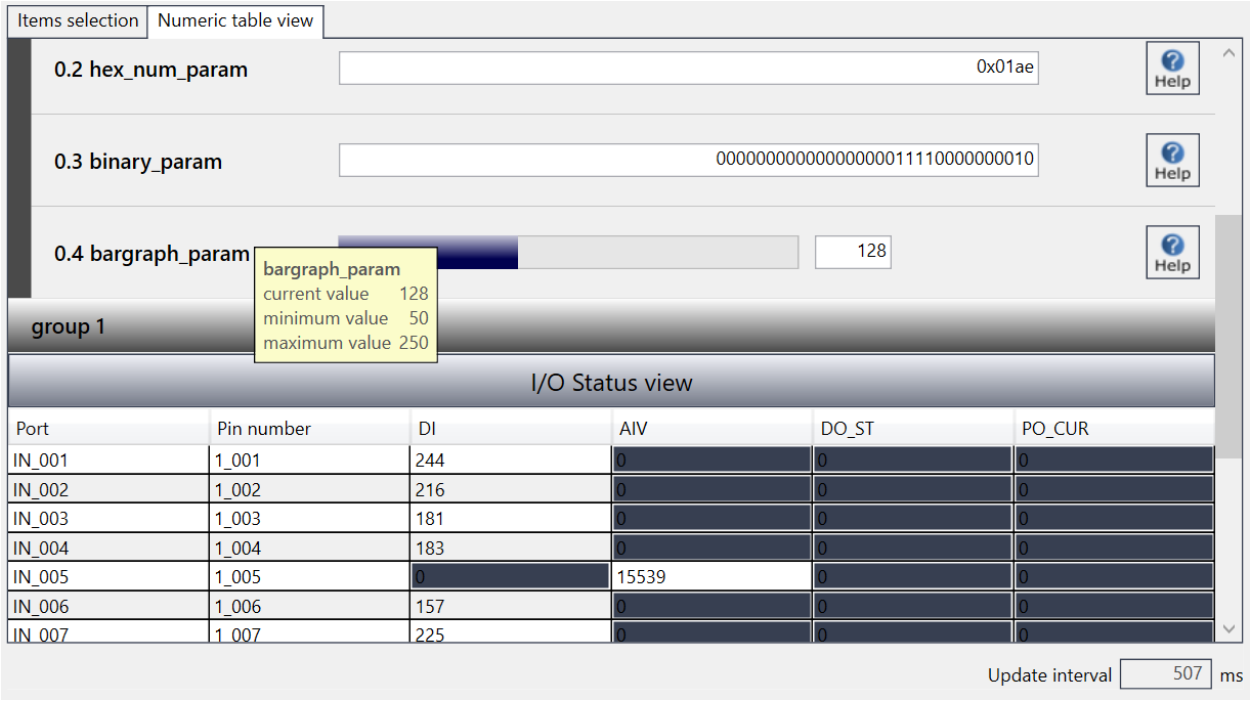


Рисунок 4.9 – Подвкладка Numeric table view

Открытая вкладка Error messages для просмотра полученных ошибок представлена на рисунке 4.10.

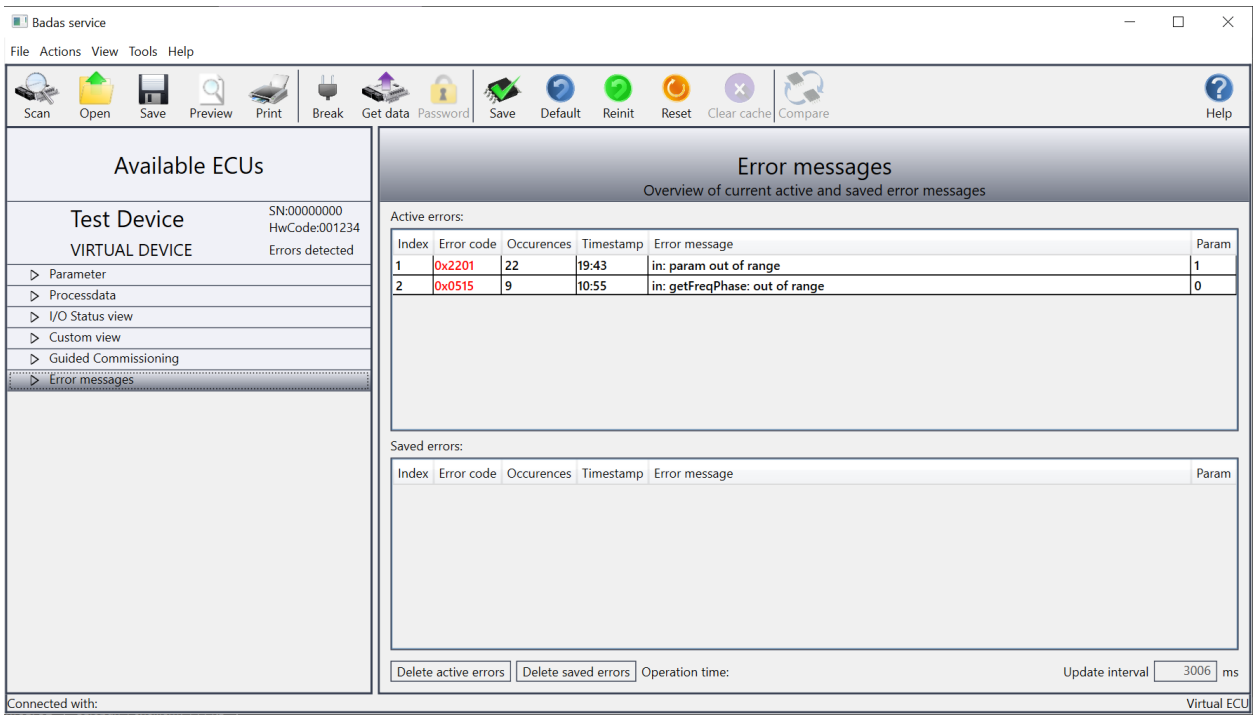


Рисунок 4.10 – Вкладка Error messages

Вкладка содержит таблицу Active errors (активные ошибки) и аналогичную ей таблицу Saved errors (сохраненные ошибки). В таблицах представлены данные ошибок: индекс (задается программно во время чтения данных ошибок), код ошибки, частота появления (количество), время появления, описание и параметр. Пользователь может очистить данные каждой из таблиц, интервал обновления данных вкладки по умолчанию равен 3000 мс.

Обновление данных вкладок Parameter, Processdata, I/O Status view и Error messages происходит посредством вызова соответствующих асинхронных методов модели представления открытой вкладки, зацикливающих однократное обновление данных вкладки и выставление задержки до момента, пока соответствующая вкладка открыта.

## 4.3 Реализация конвертеров значений

Конвертеры значений (value converter) позволяют преобразовать значение из источника привязки (Binding Source) к типу, который понятен приемнику привязки (Binding Target) и наоборот, т.к. не всегда два связываемых привязкой свойства могут иметь совместимые типы [10].

Команда кнопок выставления значения параметра вкладки Parameter в значение по умолчанию привязана к команде типа RelayCommand модели представления ParameterTreeViewModel и с помощью привязки MultiBinding [11] отправляет в качестве параметра 3 значения: индекс меню параметра, индекс подменю (группы) параметра и индекс самого параметра. Интерфейс ICommand, наследуемый классом RelayCommand, не позволяет создавать команды, принимающие в качестве первого параметра несколько объектов [12]. Для осуществления приема последовательности значений командой реализован конвертер MultiValueConverter, принимающий эту последовательность и преобразующий её в единый объект и наоборот. Исходный код MultiValueConverter представлен в листинге Б.1 приложения Б.

При выборе отображаемых вкладок на вкладке Custom view, их представления становятся доступными и видимыми на подвкладке Numeric table view и наоборот – становятся недоступными, если не выбраны. Для конвертации булевого значения в значение свойства Visibility (видимость) написан конвертер BoolVisibilityConverter, исходный код которого представлен в листинге Б.2 приложения Б.

Параметры вкладки Parameter могут иметь единицу измерения, поэтому для конвертации значения unitCode (порядковый номер единицы измерения) в текстовое обозначение единицы измерения для отображения в текстовом поле TextBlock параметра написан конвертер UnitcodeStringConverter, исходный код которого представлен в листинге Б.3 приложения Б.

Так как значение всех типов параметров вкладок Parameter и Processdata хранится в свойстве типа short, для конвертации значения типа short в значение типа параметра написаны конвертеры: ShortBoolConverter (конвертация short в bool и наоборот), BinaryConverter (конвертация short в бинарное представление и наоборот), HexValueConverter (конвертация short в шестнадцатеричное представление и наоборот), исходные коды которых представлены в листингах Б.4, Б.5 и Б.6 приложения Б соответственно.

### 4.4 Реализация нестандартных поведений элементов управления

Каждый элемент управления, предоставляемый фреймворком WPF, имеет свою собственную стандартную модель поведения. Поведение инкапсулирует общую часть функциональности элемента управления. Созданная функциональность затем может добавляться в элемент управления внутри любого приложения за счет подключения этого элемента к правильному поведению и установки свойств поведения [13].

Так, например, элемент управления Slider по умолчанию может инкрементировать свое значение при перетаскивании ползунка либо на 1, либо на длину одного шага, устанавливаемую свойством TickFrequency. Каждый отображаемый в интерфейсе параметр числового типа имеет свойство step, обозначающее значение, на которое изменится Value (значение) параметра после одной операции инкрементирования. В случае, если step не совпадает с интервалом, установленным свойством TickFrequency, и не равен 1, инкрементировать значение элемента Slider на величину step не получится. Для этого было реализовано нестандартное поведение для элемента Slider, которое при изменении значения Slider вычисляет количество полных шагов, на которое изменилось значение, и присваивает элементу Slider значение, равное старое_значение + количество_полных_шагов * step. Исходный код нестандартного поведения для элемента Slider представлен в листинге В.1 приложения В.

Стандартное поведение кнопки ToggleButton, имплементирующей значение параметра булевого типа, не дает возможности сделать кнопку одновременно доступной для нажатия и недоступной для изменения значения, что необходимо для взаимодействия с параметрами вкладки Processdata, но невозможности их изменения. Для этого было написано нестандартное поведение элемента ToggleButton, которое при нажатии на кнопку просто ничего не выполняет. Исходный код представлен в листинге В.2 приложения В.

Стандартный предоставляемый фреймворком WPF шаблон элемента DataGrid [14], используемый в качестве таблиц на вкладках I/O Status view и Error messages, содержит элемент ScrollViewer, внутри которого располагается всё содержимое (Content) элемента DataGrid. Таким образом, при использовании внешнего элемента ScrollViewer, при наведении курсора на элемент DataGrid и попытке пролистать содержимое, операция делегируется внешним ScrollViewer внутреннему ScrollViewer элемента DataGrid и интерфейс попытается пролистать именно содержимое внутреннего, даже если он отключен, что недопустимо, например, для подвкладки Numeric table view вкладки Custom view, в которой сразу несколько представлений находятся в одном общем элементе ScrollViewer и необходимо пролистать содержимое общего (внешнего) элемента ScrollViewer. Для этого было реализовано нестандартное поведение для элемента ScrollViewer, представленное в листинге В.3 приложения В.

### 4.5 Реализация поиска в дереве параметров и таблице портов

Для возможности поиска по дереву параметров и таблице портов, на соответствующих вкладках есть текстовое поле для набора строки (критерия поиска), вхождение которой требуется искать в свойстве Name меню, подменю и параметров вкладок Parameter и Processdata или в свойствах Port и PinNumber портов вкладки I/O Status view [15]. Свойству UpdateSourceTrigger привязки Binding текста поискового поля присваивается значение PropertyChanged. Таким образом, связанные значение поискового поля и свойство типа string модели представления изменяются синхронно, и, следовательно, поиск запускается каждый раз при изменении критерия поиска.

В классах ParameterMenu, ParameterSubmenu и Parameter, объекты которых составляют дерево параметров, реализовано булевое свойство IsMatch, обозначающее наличие полного или частичного совпадения критерия поиска со свойством Name, к которому односторонне

привязывается свойство Visibility каждого элемента дерева параметров, и метод ApplyCriteria, который проверяет наличие совпадения, запускает метод с одноименным названием для каждого дочернего элемента и выставляет значение свойства IsExpanded для родительского элемента, в случае обнаружения совпадения.

В классе IOPort, объекты которого составляют таблицу портов, реализовано аналогичное свойство IsMatch и метод ApplyCriteria, проверяющий наличие совпадения. Свойство Visibility каждого элемента (строки) таблицы портов привязывается к свойству IsMatch соответствующего объекта класса IOPort с помощью триггера данных DataTrigger [16].

### 4.6 Подключение предоставленного предприятием бэкенда

Так как интерфейс имеет много моделей и моделей представления, которые используют предоставленный предприятием функционал работы с контроллером, было решено создать статический класс-обработчик UDSHandler, имплементирующий методы добавленного в проект бэкенда и возвращающий необходимые данные, чтобы несущественно повысить оптимизацию и упростить написание методов моделей представления и парсеров данных.

# 5 ТЕСТИРОВАНИЕ ИНТЕРФЕЙСА

## 5.1 Методология тестирования

Для тестирования интерфейса был использован метод альфа-тестирования, которое обычно проводится на поздней стадии разработки продукта и включает имитацию реального использования продукта сотрудником или тестировщиком. Данный вид тестирования позволяет проверить работоспособность системы на соответствие требованиям и выявить неполадки для дальнейшего исправления.

## 5.2 Проведение процедуры тестирования

В рамках тестирования программа была запущена с подключенным через USB CAN адаптер контроллером с тестовой прошивкой. С помощью статического класса UDSHandler модели представлений получают данные из контроллера и преобразуют их в объекты классов моделей, выводимых в интерфейс. Проверка вывода данных в интерфейс показана на примере вкладки Parameter и представлена на рисунке 5.1.

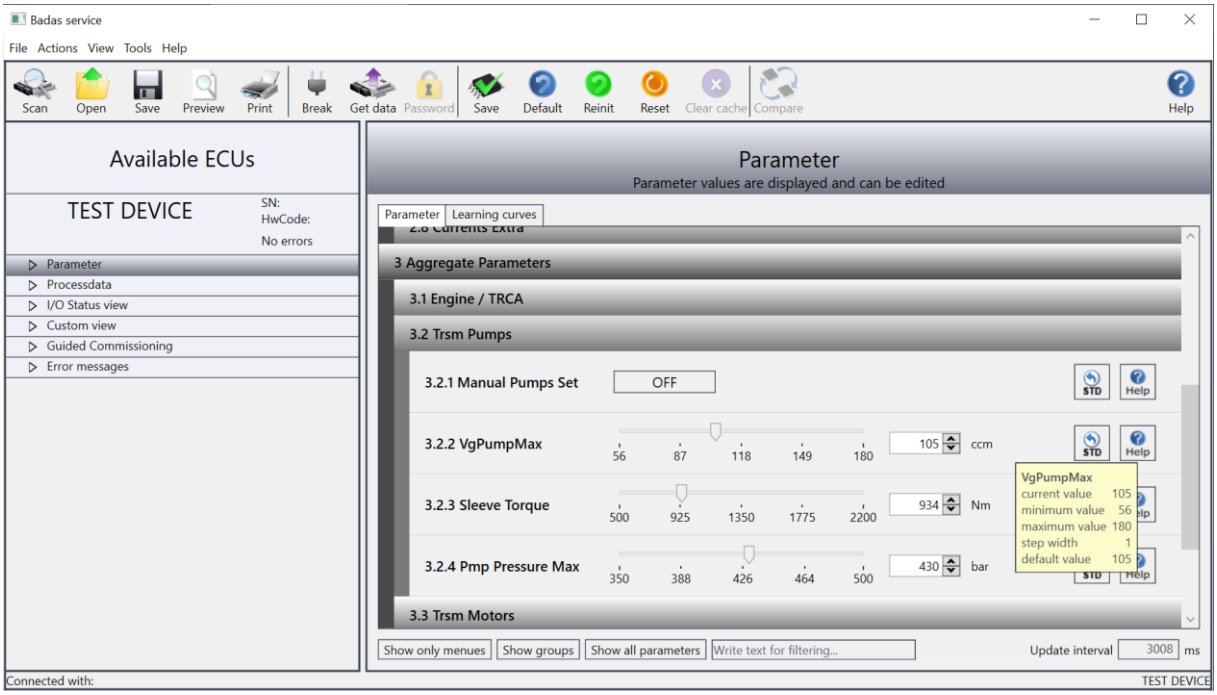


Рисунок 5.1 – Тестирование вывода на примере вкладки Parameter

Как видно на рисунке, параметры выводятся корректно, получают порядковые номера, корректно задаются деления элементов Slider, определяются единицы измерения.

Тестирование проверки правильности ввода [17] показано на рисунках 5.2 (параметр с введенным значением недопустимого типа до нажатия Enter или потери текстовым полем фокуса) и 5.3 (параметр после ввода значения недопустимого типа).


Рисунок 5.2 – Попытка ввода значения недопустимого типа


Рисунок 5.3 – Результат ввода значения недопустимого типа

Как видно из рисунка 5.3, текстовое поле значения параметра не позволяет задать недопустимое значение и принимает последнее правильное значение. Также, при попытке задать значение меньше минимального или больше максимального для данного параметра, будет выставлено минимальное или максимальное значение соответственно.

Проверка поведения элемента Slider представлена на рисунках 5.4 (значение слайдера до перетаскивания ползунка) и 5.5 (значение слайдера при попытке перетащить ползунок на небольшую величину, не кратную значению свойства step).

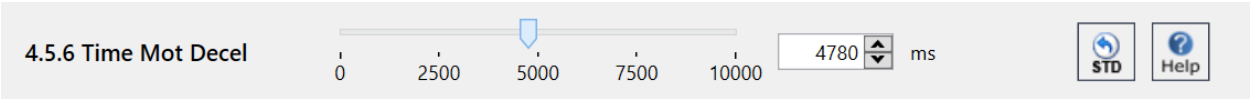


Рисунок 5.4 – Значение слайдера до перетаскивания ползунка


Рисунок 5.5 – Значение слайдера при попытке сдвинуть ползунок на величину, не кратную значению свойства step

Как видно из рисунка 5.5, слайдер не позволяет изменить значение на величину, не кратную значению свойства step параметра, и округляет

заданное пользователем путем перетаскивания ползунка значение в меньшую сторону.

Проверка установки значения параметра в значение по умолчанию показана на рисунках 5.6 (до установки в значение по умолчанию) и 5.7 (после установки значения по умолчанию нажатием кнопки STD).

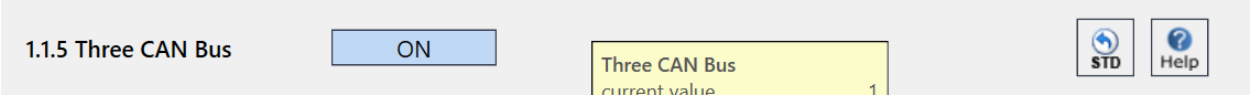


Рисунок 5.6 – Значение параметра до установки в значение по умолчанию

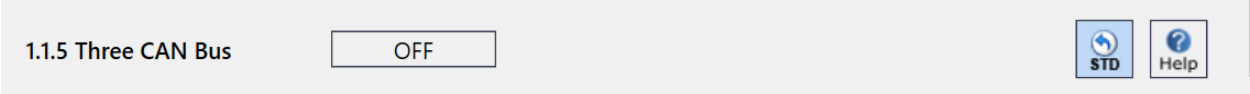


Рисунок 5.7 – Значение параметра после установки в значение по умолчанию

Тестирование поиска по дереву параметров показано на рисунке 5.8.

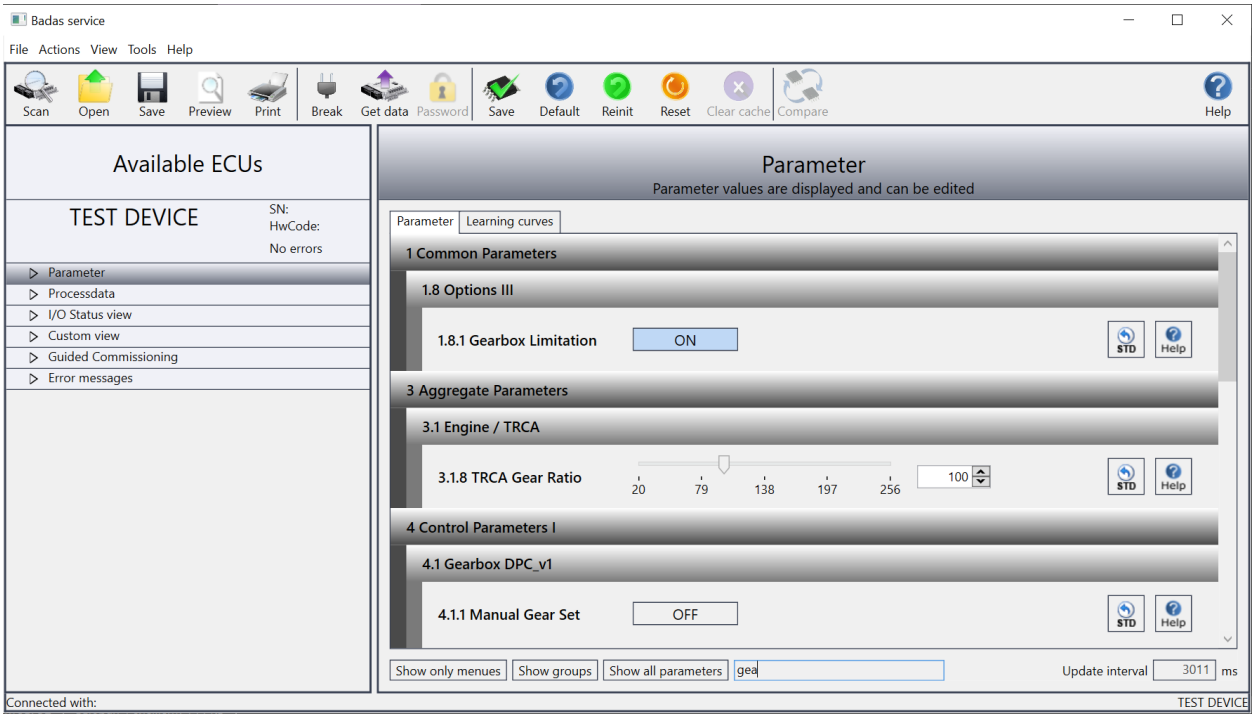


Рисунок 5.8 – Поиск в дереве параметров по заданной строке

Поиск работает правильно, при нахождении совпадений открываются все родительские элементы дерева, а элементы, не подходящие под критерий поиска, становятся невидимыми и вместе с родительскими элементами «исчезают» из текущего представления, данные дерева при этом не изменяются.

Также, было проведено тестирование кнопок Break, Get data, Save, Default, Reinit и Reset, которое доказало корректность их работы: на кнопку Break разрывается соединение с подключенным контроллером, становятся недоступны кнопки работы с контроллером и панель переключения вкладок, единственной доступной и автоматически активной вкладкой становится вкладка Available ECUs; на кнопку Get data вызываются все функции чтения данных из контроллера (основная информация, параметры, ошибки и др.) и обновляются данные каждой вкладки; на кнопку Save для каждого параметра вызывается функция сохранения текущего значения в EEPROM контроллера; на кнопку Default все значения параметров выставляются в значение по умолчанию; на кнопку Reinit значения параметров выставляются в значения, сохраненные в EEPROM; на кнопку Reset вызывается функция перезагрузки контроллера ResetECU, разрывается и заного устанавливается соединение с контроллером, вызываются все функции чтения данных, обновляются данные каждой вкладки.

# ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы был разработан графический пользовательский интерфейс программы диагностики и настройки электронных блоков управления самоходных машин, реализующий функционал предоставленного предприятием бэкенда и имеющий возможность для расширения.

Для достижения поставленной цели были решены следующие задачи:

1) выполнен аналитический обзор интерфейса главного аналога, программы BODAS-Service, изучены основные технологические решения в области разработки программ для персональных компьютеров и выбраны наиболее подходящие для реализации проекта;

2) сформированы основные функциональные и нефункциональные требования;

3) выполнено проектирование, согласно сформированным требованиям;

4) реализованы модели, представления и модели представлений;

5) проведено тестирование на стенде с тестовым блоком управления.

Перспективы развития проекта:

– построение графиков изменения значений параметров и портов, взаимодействие пользователя с графиками (масштабирование по осям, пауза и возобновление построения в реальном времени, удаление и добавление графиков на координатную плоскость и т.д.);

– создание, сохранение и загрузка готовых диагностических проектов;

– возможность выбора интерфейса подключения и языка контроллера;

– возможность выбирать конкретные параметры в деревьях параметров и порты в таблице I/O Status для отображения на вкладке Custom view. На данный момент доступен выбор только всех данных для каждой вкладки из трех возможных, отображаемых на вкладке Custom view.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Ютт, В. Е. Электронные системы управления ДВС и методы их диагностирования: Учебное пособие для вузов / В. Е. Ютт, Г. Е. Рузавин – М.: Горячая линия-Телеком, 2007. – 104 с.

2 BODAS-service Version 3.6 RE 95086 [Электронный ресурс] // Bosch Rexroth AG. URL: https://dc-br.resource.bosch.com/media/us/products_13/product_groups_1/mobile_hydraulics_4/pdfs_6/re95086.pdf (дата обращения: 12.02.23).

3 Windows Presentation Foundation [Электронный ресурс] // Microsoft Learn. URL: https://learn.microsoft.com/en-us/previous-versions/dotnet/netframework-3.0/ms754130(v=vs.85) (дата обращения: 13.02.23).

4 Gamma, E. Design Patterns / E. Gamma, R. Helm, R. Johnson, J. Vlissides // Addison-Wesley, – 2016. – 395 с.

5 Паттерны для новичков: MVC vs MVP vs MVVM [Электронный ресурс] // Habr. URL: https://habr.com/ru/articles/215605/ (дата обращения: 15.02.23).

6 Часть 2: MVVM: полное понимание (+WPF) [Электронный ресурс] // Habr. URL: https://habr.com/ru/articles/339538/ (дата обращения: 15.02.23).

7 Data binding overview (WPF .NET) [Электронный ресурс] // Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/?view=netdesktop-7.0 (дата обращения: 18.02.23).

8 Navigation with MVVM [Электронный ресурс] // Rachel Lim's Blog. URL: https://rachel53461.wordpress.com/2011/12/18/navigation-with-mvvm-2/ (дата обращения: 19.02.23).

9 Styles and templates (WPF .NET) [Электронный ресурс] // Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/controls/styles-templates-overview?view=netdesktop-7.0 (дата обращения: 18.02.23).

10 Форматирование значений привязки и конвертеры значений [Электронный ресурс] // METANIT. URL: https://metanit.com/sharp/wpf/11.3.php (дата обращения: 19.03.23).

11 Overview Of Multi Binding In MVVM - WPF [Электронный ресурс] // C# Corner. URL: https://www.c-sharpcorner.com/article/overview-of-multi-binding-in-mvvm-wpf/ (дата обращения: 19.03.2023).

12 RelayCommand attribute [Электронный ресурс] // Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/generators/relaycommand (дата обращения: 24.02.23).

13 Поведения [Электронный ресурс] // Professor Web. URL: https://professorweb.ru/my/WPF/binding_and_styles_WPF/level11/11_9.php (дата обращения: 25.03.23).

14 DataGrid Класс [Электронный ресурс] // Microsoft Learn. URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.controls.datagrid?view=windowsdesktop-7.0 (дата обращения: 15.03.23).

15 Searchable WPF TreeView [Электронный ресурс] // Codeproject. URL: https://www.codeproject.com/Articles/718022/Searchable-WPF-TreeView (дата обращения: 17.03.2023).

16 Trigger, DataTrigger & EventTrigger [Электронный ресурс] // WPF Tutorial. URL: https://wpf-tutorial.com/ru/93/стили/trigger-datatrigger-eventtrigger/ (дата обращения: 03.03.2023).

17 How to: Implement Binding Validation [Электронный ресурс] // Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/how-to-implement-binding-validation?view=netframeworkdesktop-4.8 (дата обращения: 01.04.2023).

18 Иерархические данные и HierarchicalDataTemplate [Электронный ресурс] // METANIT. URL: https://metanit.com/sharp/wpf/14.8.php (дата обращения: 17.03.23).

19 Macdonald, Pro Wpf in C# 2008: Windows Presentation Foundation with .Net 3.5, Second Edition / Matthew Macdonald. – 2008. – 1040 с.

20 WPF TreeView with multiple types [Электронный ресурс] // STACK OVERFLOW. URL: https://stackoverflow.com/questions/33242702/wpf-treeview-with-multiple-types (дата обращения: 17.03.23).

21 SliderWithTickLabels WPF Control [Электронный ресурс] // GitHub. URL: https://github.com/tdcosta100/SliderWithTickLabels (дата обращения: 06.04.23).

22 Default text for a SearchBox in WPF [Электронный ресурс] // STACK OVERFLOW. URL: https://stackoverflow.com/questions/384264/how-do-you-implement-default-text-for-a-search-box-in-wpf (дата обращения: 06.04.2023).

# ПРИЛОЖЕНИЯ

## ПРИЛОЖЕНИЕ А

## Шаблоны основных элементов управления

Листинг А.1 – Шаблон дерева параметров ParameterTreeView

```
<TreeView ScrollViewer.HorizontalScrollBarVisibility="Disabled"
    ScrollViewer.VerticalScrollBarVisibility="Disabled" BorderThickness="0" Padding="0"
    ItemsSource="{Binding ParametersTree.Values}" HorizontalContentAlignment="Stretch"
    ItemContainerStyle="{StaticResource TreeViewMenuItemStyle}">
    <TreeView.Resources>
      <HierarchicalDataTemplate DataType="{x:Type model:ParameterMenu}"
ItemsSource="{Binding Submenus.Values}" ItemContainerStyle="{StaticResource
TreeViewSubmenuItemStyle}">
        <StackPanel Style="{StaticResource ParameterMenuStackPanel}">
          <TextBlock Text="{Binding Name}"/>
        </StackPanel>
      </HierarchicalDataTemplate>
      <HierarchicalDataTemplate DataType="{x:Type model:ParameterSubmenu}"
ItemsSource="{Binding Parameters.Values}" ItemContainerStyle="{StaticResource
TreeViewParameterItemStyle}">
        <StackPanel Style="{StaticResource ParameterMenuStackPanel}">
          <TextBlock Text="{Binding Name}"/>
        </StackPanel>
      </HierarchicalDataTemplate>
      <DataTemplate DataType="{x:Type model:NumberParameter}">
        <RadioButton Style="{StaticResource ParameterTemplate}">
          <Grid>
            <Grid.ColumnDefinitions>
              <ColumnDefinition Width="200"/>
              <ColumnDefinition Width="*" MinWidth="400"/>
              <ColumnDefinition Width="115"/>
            </Grid.ColumnDefinitions>
            <TextBlock Grid.Column="0" Style="{StaticResource ParameterNameTextBlock}"/>
            <StackPanel Grid.Column="1" Style="{StaticResource NumberParameterStackPanel}">
              <slider:SliderWithTickLabels Foreground="{StaticResource SliderForeground}"
Style="{StaticResource DoubleNumberSliderWithTicks}" Minimum="{Binding minValue}"
Maximum="{Binding maxValue}" SmallChange="{Binding step}" LargeChange="{Binding step}"
TickFrequency="{Binding tickFrequency}" Value="{Binding Value}">
                <i:Interaction.Behaviors>
                  <mvvm:SnappingSlider/>
                </i:Interaction.Behaviors>
              </slider:SliderWithTickLabels>
              <xctk:ShortUpDown Increment="{Binding step}" UpdateValueOnEnterKey="True"
Value="{Binding Value, ValidatesOnExceptions=True}"/>
              <TextBlock Text="{Binding unitCode, Mode=OneWay, Converter={StaticResource
UnitcodeStringConverter}}"/>
            </StackPanel>
            <StackPanel Grid.Column="2" Orientation="Horizontal">
```

```
                <Button Style="{StaticResource ParameterSTDButton}"/>
                <Button Style="{StaticResource ParameterHelpButton}"/>
              </StackPanel>
            </Grid>
            <RadioButton.ToolTip>
              <ToolTip Style="{StaticResource ParameterTooltip}">
                <StackPanel>
                  <TextBlock Text="{Binding Name}" FontWeight="Medium"/>
                  <DockPanel>
                    <StackPanel DockPanel.Dock="Left">
                      <TextBlock Text="current value"/>
                      <TextBlock Text="minimum value"/>
                      <TextBlock Text="maximum value"/>
                      <TextBlock Text="step width"/>
                      <TextBlock Text="default value"/>
                    </StackPanel>
                    <StackPanel DockPanel.Dock="Right" Margin="5, 0, 0, 0">
                      <StackPanel.Resources>
                        <Style TargetType="{x:Type TextBlock}">
                          <Setter Property="HorizontalAlignment" Value="Right"/>
                        </Style>
                      </StackPanel.Resources>
                      <TextBlock Text="{Binding Value}"/>
                      <TextBlock Text="{Binding minValue}"/>
                      <TextBlock Text="{Binding maxValue}"/>
                      <TextBlock Text="{Binding step}"/>
                      <TextBlock Text="{Binding defaultValue}"/>
                    </StackPanel>
                  </DockPanel>
                </StackPanel>
              </ToolTip>
            </RadioButton.ToolTip>
          </RadioButton>
        </DataTemplate>
        <DataTemplate DataType="{x:Type model:HexNumberParameter}">
          <RadioButton Style="{StaticResource ParameterTemplate}">
            <Grid>
              <Grid.ColumnDefinitions>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="*" MinWidth="400"/>
                <ColumnDefinition Width="115"/>
              </Grid.ColumnDefinitions>
              <TextBlock Grid.Column="0" Style="{StaticResource ParameterNameTextBlock}"/>
              <StackPanel Grid.Column="1" Style="{StaticResource NumberParameterStackPanel}">
                <slider:SliderWithTickLabels Style="{StaticResource HexNumberSliderWithTicks}"
Minimum="{Binding minValue}" Maximum="{Binding maxValue}" TickFrequency="{Binding
tickFrequency}" Value="{Binding Value}">
                  <i:Interaction.Behaviors>
                    <mvvm:SnappingSlider/>
                  </i:Interaction.Behaviors>
                </slider:SliderWithTickLabels>
```

```
            <xctk:ShortUpDown Increment="{Binding step}" Value="{Binding Value,
ValidatesOnExceptions=True}" UpdateValueOnEnterKey="True" FormatString="X"
ParsingNumberStyle="HexNumber"/>
            <TextBlock Text="Hex" VerticalAlignment="Center" Margin="-10, 0"/>
          </StackPanel>
          <StackPanel Grid.Column="2" Orientation="Horizontal">
            <Button Style="{StaticResource ParameterSTDButton}"/>
            <Button Style="{StaticResource ParameterHelpButton}"/>
          </StackPanel>
        </Grid>
        <RadioButton.ToolTip>
          <ToolTip Style="{StaticResource ParameterTooltip}">
            <StackPanel>
              <TextBlock Text="{Binding Name}" FontWeight="Medium"/>
              <DockPanel>
                <StackPanel DockPanel.Dock="Left">
                  <TextBlock Text="current value"/>
                  <TextBlock Text="minimum value"/>
                  <TextBlock Text="maximum value"/>
                  <TextBlock Text="step width"/>
                  <TextBlock Text="default value"/>
                </StackPanel>
                <StackPanel DockPanel.Dock="Right" Margin="5, 0, 0, 0">
                  <StackPanel.Resources>
                    <Style TargetType="{x:Type TextBlock}">
                      <Setter Property="HorizontalAlignment" Value="Right"/>
                    </Style>
                  </StackPanel.Resources>
                  <TextBlock>
                    <TextBlock.Text>
                      <MultiBinding Converter="{StaticResource HexValueConverter}">
                        <Binding Path="Value"/>
                        <Binding Path="digits"/>
                      </MultiBinding>
                    </TextBlock.Text>
                  </TextBlock>
                  <TextBlock>
                    <TextBlock.Text>
                      <MultiBinding Converter="{StaticResource HexValueConverter}">
                        <Binding Path="minValue"/>
                        <Binding Path="digits"/>
                      </MultiBinding>
                    </TextBlock.Text>
                  </TextBlock>
                  <TextBlock>
                    <TextBlock.Text>
                      <MultiBinding Converter="{StaticResource HexValueConverter}">
                        <Binding Path="maxValue"/>
                        <Binding Path="digits"/>
                      </MultiBinding>
                    </TextBlock.Text>
```

```
                        </TextBlock>
                        <TextBlock>
                            <TextBlock.Text>
                                <MultiBinding Converter="{StaticResource HexValueConverter}">
                                    <Binding Path="step"/>
                                    <Binding Path="digits"/>
                                </MultiBinding>
                            </TextBlock.Text>
                        </TextBlock>
                        <TextBlock>
                            <TextBlock.Text>
                                <MultiBinding Converter="{StaticResource HexValueConverter}">
                                    <Binding Path="defaultValue"/>
                                    <Binding Path="digits"/>
                                </MultiBinding>
                            </TextBlock.Text>
                        </TextBlock>
                    </StackPanel>
                </DockPanel>
            </StackPanel>
        </ToolTip>
    </RadioButton.ToolTip>
</RadioButton>
</DataTemplate>
<DataTemplate DataType="{x:Type model:SwitchParameter}">
    <RadioButton Style="{StaticResource ParameterTemplate}">
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="*" MinWidth="400"/>
                <ColumnDefinition Width="115"/>
            </Grid.ColumnDefinitions>
            <TextBlock Grid.Column="0" Style="{StaticResource ParameterNameTextBlock}"/>
            <ToggleButton Grid.Column="1" Style="{StaticResource ParameterSwitchButton}"
IsChecked="{Binding Value, Converter={StaticResource ShortBoolConverter}}"/>
            <StackPanel Grid.Column="2" Orientation="Horizontal">
                <Button Style="{StaticResource ParameterSTDButton}"/>
                <Button Style="{StaticResource ParameterHelpButton}"/>
            </StackPanel>
        </Grid>
        <RadioButton.ToolTip>
            <ToolTip Style="{StaticResource ParameterTooltip}">
                <StackPanel>
                    <TextBlock Text="{Binding Name}" FontWeight="Medium"/>
                    <DockPanel>
                        <StackPanel DockPanel.Dock="Left">
                            <TextBlock Text="current value"/>
                            <TextBlock Text="minimum value"/>
                            <TextBlock Text="maximum value"/>
                            <TextBlock Text="switch mode"/>
                            <TextBlock Text="default value"/>
```

```
                    </StackPanel>
                    <StackPanel DockPanel.Dock="Right" Margin="5, 0, 0, 0">
                      <StackPanel.Resources>
                        <Style TargetType="{x:Type TextBlock}">
                          <Setter Property="HorizontalAlignment" Value="Right"/>
                        </Style>
                      </StackPanel.Resources>
                      <TextBlock Text="{Binding Value}"/>
                      <TextBlock Text="{Binding minValue}"/>
                      <TextBlock Text="{Binding maxValue}"/>
                      <TextBlock Text="{Binding Name}"/>
                      <TextBlock Text="{Binding defaultValue}"/>
                    </StackPanel>
                  </DockPanel>
                </StackPanel>
              </ToolTip>
            </RadioButton.ToolTip>
          </RadioButton>
        </DataTemplate>
        <DataTemplate DataType="{x:Type model:ListParameter}">
          <RadioButton Style="{StaticResource ParameterTemplate}">
            <Grid>
              <Grid.ColumnDefinitions>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="*" MinWidth="400"/>
                <ColumnDefinition Width="115"/>
              </Grid.ColumnDefinitions>
              <TextBlock Grid.Column="0" Style="{StaticResource ParameterNameTextBlock}"/>
              <StackPanel Grid.Column="1" Style="{StaticResource ListParameterStackPanel}">
                <ComboBox ItemsSource="{Binding ListEntries.Values}" SelectedIndex="{Binding
Value}"/>
                <TextBlock Text="Value: "/>
                <TextBox Text="{Binding Value, Mode=OneWay}"/>
              </StackPanel>
              <StackPanel Grid.Column="2" Orientation="Horizontal">
                <Button Style="{StaticResource ParameterSTDButton}"/>
                <Button Style="{StaticResource ParameterHelpButton}"/>
              </StackPanel>
            </Grid>
            <RadioButton.ToolTip>
              <ToolTip Style="{StaticResource ParameterTooltip}">
                <StackPanel>
                  <TextBlock Text="{Binding Name}" FontWeight="Medium"/>
                  <DockPanel>
                    <StackPanel DockPanel.Dock="Left">
                      <TextBlock Text="current value"/>
                      <TextBlock Text="minimum value"/>
                      <TextBlock Text="maximum value"/>
                      <TextBlock Text="default value"/>
                    </StackPanel>
                    <StackPanel DockPanel.Dock="Right" Margin="5, 0, 0, 0">
```

```
                        <StackPanel.Resources>
                         <Style TargetType="{x:Type TextBlock}">
                            <Setter Property="HorizontalAlignment" Value="Right"/>
                         </Style>
                        </StackPanel.Resources>
                        <TextBlock Text="{Binding Value}"/>
                        <TextBlock Text="0"/>
                        <TextBlock Text="{Binding maxValue}"/>
                        <TextBlock Text="{Binding defaultValue}"/>
                      </StackPanel>
                    </DockPanel>
                  </StackPanel>
                </ToolTip>
              </RadioButton.ToolTip>
            </RadioButton>
          </DataTemplate>
        </TreeView.Resources>
        <TreeView.Template>
          <ControlTemplate TargetType="{x:Type TreeView}">
            <ItemsPresenter/>
          </ControlTemplate>
        </TreeView.Template>
    </TreeView>
```

Листинг А.2 – Шаблон дерева параметров ProcessdataTreeView

```
    <TreeView ScrollViewer.HorizontalScrollBarVisibility="Auto"
      ScrollViewer.VerticalScrollBarVisibility="Visible" BorderThickness="0" Padding="0"
      ItemsSource="{Binding ProcessdataTree.Values}" HorizontalContentAlignment="Stretch"
      ItemContainerStyle="{StaticResource TreeViewMenuItemStyle}">
      <TreeView.Resources>
        <HierarchicalDataTemplate DataType="{x:Type model:ParameterSubmenu}"
ItemsSource="{Binding Parameters.Values}" ItemContainerStyle="{StaticResource
TreeViewParameterItemStyle}">
          <StackPanel Style="{StaticResource ParameterMenuStackPanel}">
            <TextBlock Text="{Binding Name}"/>
          </StackPanel>
        </HierarchicalDataTemplate>
        <DataTemplate DataType="{x:Type model:ProcessdataNumberParameter}">
          <RadioButton Style="{StaticResource ParameterTemplate}">
            <Grid>
              <Grid.ColumnDefinitions>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="*" MinWidth="400"/>
                <ColumnDefinition Width="115"/>
              </Grid.ColumnDefinitions>
              <TextBlock Grid.Column="0" Style="{StaticResource
ProcessdataParameterNameTextBlock}"/>
              <TextBox Grid.Column="1" Height="25" VerticalAlignment="Center" Text="{Binding
Value, Mode=OneWay}" TextAlignment="Right"/>
```

```
            <Button Grid.Column="2" Style="{StaticResource ParameterHelpButton}"
        HorizontalAlignment="Right" Margin="10, 0"/>
            </Grid>
            <RadioButton.ToolTip>
              <ToolTip Style="{StaticResource ParameterTooltip}">
                <StackPanel>
                  <TextBlock Text="{Binding Name}" FontWeight="Medium"/>
                  <DockPanel>
                    <StackPanel DockPanel.Dock="Left">
                      <TextBlock Text="current value"/>
                      <TextBlock Text="minimum value"/>
                      <TextBlock Text="maximum value"/>
                      <TextBlock Text="post comma digits"/>
                    </StackPanel>
                    <StackPanel DockPanel.Dock="Right" Margin="5, 0, 0, 0">
                      <StackPanel.Resources>
                        <Style TargetType="{x:Type TextBlock}">
                          <Setter Property="HorizontalAlignment" Value="Right"/>
                        </Style>
                      </StackPanel.Resources>
                      <TextBlock Text="{Binding Value}"/>
                      <TextBlock Text="{Binding minValue}"/>
                      <TextBlock Text="{Binding maxValue}"/>
                      <TextBlock Text="{Binding digits}"/>
                    </StackPanel>
                  </DockPanel>
                </StackPanel>
              </ToolTip>
            </RadioButton.ToolTip>
          </RadioButton>
        </DataTemplate>
        <DataTemplate DataType="{x:Type model:SwitchParameter}">
          <RadioButton Style="{StaticResource ParameterTemplate}">
            <Grid>
              <Grid.ColumnDefinitions>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="*" MinWidth="400"/>
                <ColumnDefinition Width="115"/>
              </Grid.ColumnDefinitions>
              <TextBlock Grid.Column="0" Style="{StaticResource
ProcessdataParameterNameTextBlock}"/>
              <mvvm:LockedToggleButton Grid.Column="1" Style="{StaticResource
ParameterSwitchButton}" IsChecked="{Binding Value, Mode=OneWay, Converter={StaticResource
ShortBoolConverter}}"/>
              <Button Grid.Column="2" Style="{StaticResource ParameterHelpButton}"
HorizontalAlignment="Right" Margin="10, 0"/>
            </Grid>
            <RadioButton.ToolTip>
              <ToolTip Style="{StaticResource ParameterTooltip}">
                <StackPanel>
                  <TextBlock Text="{Binding Name}" FontWeight="Medium"/>
```

```
                    <DockPanel>
                        <StackPanel DockPanel.Dock="Left">
                            <TextBlock Text="current value"/>
                            <TextBlock Text="minimum value"/>
                            <TextBlock Text="maximum value"/>
                            <TextBlock Text="switch mode"/>
                        </StackPanel>
                        <StackPanel DockPanel.Dock="Right" Margin="5, 0, 0, 0">
                            <StackPanel.Resources>
                                <Style TargetType="{x:Type TextBlock}">
                                    <Setter Property="HorizontalAlignment" Value="Right"/>
                                </Style>
                            </StackPanel.Resources>
                            <TextBlock Text="{Binding Value}"/>
                            <TextBlock Text="{Binding minValue}"/>
                            <TextBlock Text="{Binding maxValue}"/>
                            <TextBlock Text="{Binding Name}"/>
                        </StackPanel>
                    </DockPanel>
                </StackPanel>
            </ToolTip>
        </RadioButton.ToolTip>
    </RadioButton>
</DataTemplate>
<DataTemplate DataType="{x:Type model:BinaryNumberParameter}">
    <RadioButton Style="{StaticResource ParameterTemplate}">
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="*" MinWidth="400"/>
                <ColumnDefinition Width="115"/>
            </Grid.ColumnDefinitions>
            <TextBlock Grid.Column="0" Style="{StaticResource
ProcessdataParameterNameTextBlock}"/>
            <TextBox Grid.Column="1" Height="22" VerticalAlignment="Center"
TextAlignment="Right" IsReadOnly="True">
                <TextBox.Text>
                    <MultiBinding Converter="{StaticResource BinaryConverter}" Mode="OneWay">
                        <Binding Path="Value"/>
                        <Binding Path="digits"/>
                    </MultiBinding>
                </TextBox.Text>
            </TextBox>
            <Button Grid.Column="2" Style="{StaticResource ParameterHelpButton}"
HorizontalAlignment="Right" Margin="10, 0"/>
        </Grid>
        <RadioButton.ToolTip>
            <ToolTip Style="{StaticResource ParameterTooltip}">
                <StackPanel>
                    <TextBlock Text="{Binding Name}" FontWeight="Medium"/>
                    <DockPanel>
```

```
                <StackPanel DockPanel.Dock="Left">
                    <TextBlock Text="current value"/>
                    <TextBlock Text="minimum value"/>
                    <TextBlock Text="maximum value"/>
                    <TextBlock Text="number of digits"/>
                </StackPanel>
                <StackPanel DockPanel.Dock="Right" Margin="5, 0, 0, 0">
                    <StackPanel.Resources>
                        <Style TargetType="{x:Type TextBlock}">
                            <Setter Property="HorizontalAlignment" Value="Right"/>
                        </Style>
                    </StackPanel.Resources>
                    <TextBlock>
                        <TextBlock.Text>
                            <MultiBinding Converter="{StaticResource BinaryConverter}">
                                <Binding Path="Value"/>
                                <Binding Path="digits"/>
                            </MultiBinding>
                        </TextBlock.Text>
                    </TextBlock>
                    <TextBlock>
                        <TextBlock.Text>
                            <MultiBinding Converter="{StaticResource BinaryConverter}">
                                <Binding Path="minValue"/>
                                <Binding Path="digits"/>
                            </MultiBinding>
                        </TextBlock.Text>
                    </TextBlock>
                    <TextBlock>
                        <TextBlock.Text>
                            <MultiBinding Converter="{StaticResource BinaryConverter}">
                                <Binding Path="maxValue"/>
                                <Binding Path="digits"/>
                            </MultiBinding>
                        </TextBlock.Text>
                    </TextBlock>
                    <TextBlock Text="{Binding digits}"/>
                </StackPanel>
            </DockPanel>
        </StackPanel>
    </ToolTip>
  </RadioButton.ToolTip>
 </RadioButton>
</DataTemplate>
<DataTemplate DataType="{x:Type model:HexNumberParameter}">
 <RadioButton Style="{StaticResource ParameterTemplate}">
  <Grid>
   <Grid.ColumnDefinitions>
    <ColumnDefinition Width="200"/>
    <ColumnDefinition Width="*" MinWidth="400"/>
    <ColumnDefinition Width="115"/>
```

```
                </Grid.ColumnDefinitions>
                <TextBlock Grid.Column="0" Style="{StaticResource
ProcessdataParameterNameTextBlock}"/>
                <TextBox Grid.Column="1" Height="22" VerticalAlignment="Center"
TextAlignment="Right" IsReadOnly="True">
                    <TextBox.Text>
                        <MultiBinding Converter="{StaticResource HexValueConverter}"
Mode="OneWay">
                            <Binding Path="Value"/>
                            <Binding Path="digits"/>
                        </MultiBinding>
                    </TextBox.Text>
                </TextBox>
                <Button Grid.Column="2" Style="{StaticResource ParameterHelpButton}"
HorizontalAlignment="Right" Margin="10, 0"/>
            </Grid>
            <RadioButton.ToolTip>
                <ToolTip Style="{StaticResource ParameterTooltip}">
                    <StackPanel>
                        <TextBlock Text="{Binding Name}" FontWeight="Medium"/>
                        <DockPanel>
                            <StackPanel DockPanel.Dock="Left">
                                <TextBlock Text="current value"/>
                                <TextBlock Text="minimum value"/>
                                <TextBlock Text="maximum value"/>
                                <TextBlock Text="number of digits"/>
                            </StackPanel>
                            <StackPanel DockPanel.Dock="Right" Margin="5, 0, 0, 0">
                                <StackPanel.Resources>
                                    <Style TargetType="{x:Type TextBlock}">
                                        <Setter Property="HorizontalAlignment" Value="Right"/>
                                    </Style>
                                </StackPanel.Resources>
                                <TextBlock>
                                    <TextBlock.Text>
                                        <MultiBinding Converter="{StaticResource HexValueConverter}">
                                            <Binding Path="Value"/>
                                            <Binding Path="digits"/>
                                        </MultiBinding>
                                    </TextBlock.Text>
                                </TextBlock>
                                <TextBlock>
                                    <TextBlock.Text>
                                        <MultiBinding Converter="{StaticResource HexValueConverter}">
                                            <Binding Path="minValue"/>
                                            <Binding Path="digits"/>
                                        </MultiBinding>
                                    </TextBlock.Text>
                                </TextBlock>
                                <TextBlock>
                                    <TextBlock.Text>
```

```
                        <MultiBinding Converter="{StaticResource HexValueConverter}">
                            <Binding Path="maxValue"/>
                            <Binding Path="digits"/>
                        </MultiBinding>
                    </TextBlock.Text>
                </TextBlock>
                <TextBlock Text="{Binding digits}"/>
            </StackPanel>
        </DockPanel>
    </StackPanel>
</ToolTip>
</RadioButton.ToolTip>
</RadioButton>
</DataTemplate>
<DataTemplate DataType="{x:Type model:BargraphParameter}">
<RadioButton Style="{StaticResource ParameterTemplate}">
<Grid>
<Grid.ColumnDefinitions>
<ColumnDefinition Width="200"/>
<ColumnDefinition Width="*" MinWidth="400"/>
<ColumnDefinition Width="115"/>
</Grid.ColumnDefinitions>
<TextBlock Grid.Column="0" Style="{StaticResource
ProcessdataParameterNameTextBlock}"/>
<StackPanel Grid.Column="1" Orientation="Horizontal">
<ProgressBar Foreground="{StaticResource BargraphParameterBackgroundBrush}"
Width="300" Height="22" Minimum="{Binding minValue}" Maximum="{Binding maxValue}"
Value="{Binding Value, Mode=OneWay}"/>
<TextBox Margin="10, 0" Width="50" Height="22" Text="{Binding Value,
Mode=OneWay}" TextAlignment="Right" IsReadOnly="True"/>
</StackPanel>
<Button Grid.Column="2" Style="{StaticResource ParameterHelpButton}"
HorizontalAlignment="Right" Margin="10, 0"/>
</Grid>
<RadioButton.ToolTip>
<ToolTip Style="{StaticResource ParameterTooltip}">
<StackPanel>
<TextBlock Text="{Binding Name}" FontWeight="Medium"/>
<DockPanel>
<StackPanel DockPanel.Dock="Left">
<TextBlock Text="current value"/>
<TextBlock Text="minimum value"/>
<TextBlock Text="maximum value"/>
</StackPanel>
<StackPanel DockPanel.Dock="Right" Margin="5, 0, 0, 0">
<StackPanel.Resources>
<Style TargetType="{x:Type TextBlock}">
<Setter Property="HorizontalAlignment" Value="Right"/>
</Style>
</StackPanel.Resources>
<TextBlock Text="{Binding Value}"/>
```

```
                        <TextBlock Text="{Binding minValue}"/>
                        <TextBlock Text="{Binding maxValue}"/>
                      </StackPanel>
                    </DockPanel>
                  </StackPanel>
                </ToolTip>
              </RadioButton.ToolTip>
            </RadioButton>
          </DataTemplate>
        </TreeView.Resources>
        <TreeView.Template>
          <ControlTemplate TargetType="{x:Type TreeView}">
            <ItemsPresenter/>
          </ControlTemplate>
        </TreeView.Template>
      </TreeView>
```

Листинг А.3 – Шаблон таблицы портов IOStatusTableView

```
    <DataGrid HorizontalContentAlignment="Stretch" ScrollViewer.PanningMode="None"
HorizontalAlignment="Stretch" Background="{StaticResource TabItemBackgroundColor}"
AlternatingRowBackground="{StaticResource TabBackgroundColor}"
HorizontalScrollBarVisibility="Disabled" VerticalScrollBarVisibility="Disabled"
AutoGenerateColumns="False" ItemsSource="{Binding IOStatusList}" CanUserAddRows="False"
CanUserDeleteRows="False" CanUserResizeRows="False" HeadersVisibility="Column"
IsReadOnly="True" CanUserResizeColumns="False" BorderThickness="0">
        <DataGrid.Columns>
          <DataGridTextColumn Width="*" MinWidth="75" Header="Port" Binding="{Binding Port}"/>
          <DataGridTextColumn Width="*" MinWidth="75" Header="Pin number" Binding="{Binding
PinNumber}"/>
          <DataGridTextColumn Width="*" MinWidth="75" Header="DI" Binding="{Binding DI}"/>
          <DataGridTextColumn Width="*" MinWidth="75" Header="AIV" Binding="{Binding AIV}"/>
          <DataGridTextColumn Width="*" MinWidth="75" Header="DO__ST" Binding="{Binding
DO_ST}"/>
          <DataGridTextColumn Width="*" MinWidth="75" Header="PO__CUR" Binding="{Binding
PO_CUR}"/>
        </DataGrid.Columns>
        <DataGrid.RowStyle>
          <Style TargetType="{x:Type DataGridRow}">
            <Style.Triggers>
              <DataTrigger Binding="{Binding IsMatch}" Value="False">
                <Setter Property="Visibility" Value="Collapsed"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </DataGrid.RowStyle>
        <DataGrid.CellStyle>
          <Style TargetType="{x:Type DataGridCell}">
            <Style.Triggers>
              <DataTrigger Binding="{Binding Path=Content.Text, RelativeSource={RelativeSource
Self}}" Value="0">
```

```
                <Setter Property="Background" Value="{StaticResource DefaultBorderBrush}"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </DataGrid.CellStyle>
    </DataGrid>
```

Листинг А.4 – Шаблон таблицы ошибок ErrorTableView

```
    <DataGrid HorizontalScrollBarVisibility="Disabled" VerticalScrollBarVisibility="auto"
AutoGenerateColumns="False" ItemsSource="{Binding ActiveErrors}" CanUserResizeColumns="False"
CanUserAddRows="False" CanUserDeleteRows="False" CanUserResizeRows="False"
HeadersVisibility="Column">
        <DataGrid.Columns>
          <DataGridTextColumn Header="Index" Binding="{Binding Key}"/>
          <DataGridTemplateColumn Header="Error code">
            <DataGridTemplateColumn.CellTemplate>
              <DataTemplate>
                <TextBlock Text="{Binding Value.ErrorCode, Converter={StaticResource
HexCodeConverter}}" Foreground="{Binding Value.ErrorCode, Converter={StaticResource
CodeColorConverter}}"/>
              </DataTemplate>
            </DataGridTemplateColumn.CellTemplate>
          </DataGridTemplateColumn>
          <DataGridTextColumn Header="Occurences" Binding="{Binding Value.Occurence}"/>
          <DataGridTemplateColumn Header="Timestamp">
            <DataGridTemplateColumn.CellTemplate>
              <DataTemplate>
                <TextBlock Text="{Binding Value.Timestamp, Converter={StaticResource
TimestampConverter}}"/>
              </DataTemplate>
            </DataGridTemplateColumn.CellTemplate>
          </DataGridTemplateColumn>
          <DataGridTextColumn Width="*" MinWidth="200" Header="Error message"
Binding="{Binding Value.Description}"/>
          <DataGridTextColumn Header="Param" Binding="{Binding Value.Parameter}"/>
        </DataGrid.Columns>
        <DataGrid.CellStyle>
          <Style TargetType="{x:Type DataGridCell}">
            <Setter Property="FontWeight" Value="Medium"/>
          </Style>
        </DataGrid.CellStyle>
    </DataGrid>
```

# ПРИЛОЖЕНИЕ Б

## Конвертеры значений

### Листинг Б.1 – Исходный код MultiValueConverter

```
    public class MultiValueConverter : IMultiValueConverter
    {
        public object Convert(object[] values, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            return values.Clone();
        }

        public object[] ConvertBack(object value, Type[] targetTypes, object parameter,
System.Globalization.CultureInfo culture)
        {
            return (object[])value;
        }
    }
```

### Листинг Б.2 – Исходный код BoolVisibilityConverter

```
    public class BoolVisibilityConverter : IValueConverter
    {
        public object? Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            return (value is bool)
                ? (((bool)value == true) ? Visibility.Visible : Visibility.Collapsed)
                : null;
        }
        public object ConvertBack(object value, Type targetTypes, object parameter,
CultureInfo culture)
        {
            return (value is Visibility) ? (((Visibility)value == Visibility.Visible) ?
true : false) : Visibility.Visible;
        }
    }
```

### Листинг Б.3 – Исходный код UnitcodeStringConverter

```
    public class UnitcodeStringConverter : IValueConverter
    {
        public object? Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            if (value is byte)
            {
                switch ((byte)value)
                {
                    case 0:
                        return "";
                    case 1:
                        return "V";
                    case 2:
                        return "mA";
                    case 3:
                        return "rpm";
                    case 4:
                        return "ccm";
                    case 5:
                        return "Nm";
                    case 6:
                        return "bar";
```

```
                    case 7:
                        return "ms";
                    case 8:
                        return "s";
                    case 9:
                        return "hrs";
                    case 10:
                        return "min";
                    case 11:
                        return "km";
                    case 12:
                        return "%";
                    case 13:
                        return "km/h";
                    case 14:
                        return "°C";
                    default:
                        return "";
                }
            }
            else return "";
        }
        public object ConvertBack(object value, Type targetTypes, object parameter,
CultureInfo culture)
        {
            if (value is string){
                switch ((string)value)
                {
                    case "":
                        return 0;
                    case "V":
                        return 1;
                    case "mA":
                        return 2;
                    case "rpm":
                        return 3;
                    case "ccm":
                        return 4;
                    case "Nm":
                        return 5;
                    case "bar":
                        return 6;
                    case "ms":
                        return 7;
                    case "s":
                        return 8;
                    case "hrs":
                        return 9;
                    case "min":
                        return 10;
                    case "km":
                        return 11;
                    case "%":
                        return 12;
                    case "km/h":
                        return 13;
                    case "°C":
                        return 14;
                    default:
                        return "";
                }
            }
            else return 0;
        }
    }
```

Листинг Б.4 – Исходный код ShortBoolConverter

```
    public class ShortBoolConverter : IValueConverter
    {
        public object? Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            return (value is short)
                ? (((short)value == 1) ? true : false)
                : null;
        }

        public object ConvertBack(object value, Type targetTypes, object parameter,
CultureInfo culture)
        {
            return (value is bool) ? (((bool)value == true) ? 1 : 0) : 0;
        }
    }
```

Листинг Б.5 – Исходный код BinaryConverter

```
    public class BinaryConverter : IMultiValueConverter
    {
        public object? Convert(object[] values, Type targetType, object parameter,
CultureInfo culture)
        {
            return (values[0] is short) ? System.Convert.ToString((short)values[0],
2).PadLeft(System.Convert.ToInt32(values[1]), '0') : null;
        }

        public object[] ConvertBack(object value, Type[] targetTypes, object parameter,
CultureInfo culture)
        {
            return new object[] { System.Convert.ToInt16((string)value, 2),
((string)value).Length };
        }
    }
```

Листинг Б.6 – Исходный код HexValueConverter

```
    public class HexValueConverter : IMultiValueConverter
    {
        public object? Convert(object[] values, Type targetType, object parameter,
CultureInfo culture)
        {
            return (values[0] is short && values[1] is ushort)
                ? ("0x" + System.Convert.ToString((short)values[0],
16).PadLeft(System.Convert.ToInt32((ushort)values[1]), '0'))
                : (values[0] is double && values[1] is ushort)
                    ? ("0x" + System.Convert.ToString((int)((double)values[0]),
16).PadLeft(System.Convert.ToInt32((ushort)values[1]), '0'))
                    : null;
        }

        public object[] ConvertBack(object value, Type[] targetTypes, object parameter,
CultureInfo culture)
        {
            return new object[] { System.Convert.ToInt16(((string)value).Remove(0, 2),
16), (ushort)((string)value).Remove(0, 2).Length };
        }
    }
```

# ПРИЛОЖЕНИЕ В

## Нестандартные поведения элементов управления

Листинг В.1 – Поведение слайдера

```
    public class SnappingSlider : Behavior<SliderWithTickLabels.SliderWithTickLabels>
    {
        protected override void OnAttached()
        {
            base.OnAttached();
            AssociatedObject.ValueChanged += AssociatedObjectValueChanged;
        }

        protected override void OnDetaching()
        {
            base.OnDetaching();
            AssociatedObject.ValueChanged -= AssociatedObjectValueChanged;
        }

        private void AssociatedObjectValueChanged(object sender,
System.Windows.RoutedPropertyChangedEventArgs<double> e)
        {
            double steps = Math.Round((e.NewValue - AssociatedObject.Minimum) /
AssociatedObject.SmallChange);
            if (steps < 0)
                return;
            double newValue = AssociatedObject.Minimum + steps *
AssociatedObject.SmallChange;
            if (newValue > AssociatedObject.Maximum)
                return;
            AssociatedObject.Value = newValue;
        }
    }
```

Листинг В.2 – Поведение кнопки ToggleButton для параметров булевого типа вкладки Processdata

```
    public class LockedToggleButton : ToggleButton
    {
        protected override void OnToggle() { }
    }
```

Листинг В.3 – Поведение ScrollViewer

```
    public class TopMouseScrollPriorityBehavior
    {
        public static bool GetTopMouseScrollPriority(DependencyObject obj)
        {
            return (bool)obj.GetValue(TopMouseScrollPriorityProperty);
        }

        public static void SetTopMouseScrollPriority(DependencyObject obj, bool value)
        {
            obj.SetValue(TopMouseScrollPriorityProperty, value);
        }

        public static readonly DependencyProperty TopMouseScrollPriorityProperty =
            DependencyProperty.RegisterAttached("TopMouseScrollPriority", typeof(bool),
typeof(TopMouseScrollPriorityBehavior), new PropertyMetadata(false, OnPropertyChanged));

        private static void OnPropertyChanged(DependencyObject d,
DependencyPropertyChangedEventArgs e)
        {
```

```
            var scrollViewer = d as ScrollViewer;
            if (scrollViewer == null)
                throw new
InvalidOperationException($"{nameof(TopMouseScrollPriorityBehavior)}.{nameof(TopMouseScro
llPriorityProperty)} can only be applied to controls of type {nameof(ScrollViewer)}");
            if (e.NewValue == e.OldValue)
                return;
            if ((bool)e.NewValue)
                scrollViewer.PreviewMouseWheel += ScrollViewer_PreviewMouseWheel;
            else
                scrollViewer.PreviewMouseWheel -= ScrollViewer_PreviewMouseWheel;
        }

        private static void ScrollViewer_PreviewMouseWheel(object sender,
System.Windows.Input.MouseWheelEventArgs e)
        {
            var scrollViewer = (ScrollViewer)sender;
            scrollViewer.ScrollToVerticalOffset(scrollViewer.VerticalOffset - e.Delta);
            e.Handled = true;
        }
    }
```