**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по дисциплине «Производственная практика НИР»**

**Тема: Создание графического настройщика для приборов тепловычислителей (СПТ96Х, СПТ76Х, СПЕ)**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8303 |  | Кабанов Н.С. |
| Руководитель |  | Морозов С.М. |

Санкт-Петербург

2023

**ЗАДАНИЕ**

**НА НАУЧНО-ИССЛЕДОВАТЕЛЬСКУЮ РАБОТУ**

|  |  |  |
| --- | --- | --- |
| Студент Кабанов Н.С. |  |  |
| Группа 8303 |  |  |
| Тема работы: Создание графического настройщика для приборов тепловычислителей (СПТ96Х, СПТ76Х, СПЕ). | | |
| Исходные данные:  Формулирование постановки задачи, создание функционала вывода данных и дальнейшего экспорта их и обзор литературы | | |
| Содержание пояснительной записки:  «Содержание», «Основные термины», «Введение», «Постановка задачи»,  «Результаты работы в весеннем семестре», «План работы на осенний семестр»,  «Список использованных источников» | | |
| Предполагаемый объем пояснительной записки: Не менее 15 страниц. | | |
| Дата выдачи задания: |  |  |
| Дата сдачи реферата: 18.12.2023г. |  |  |
| Дата защиты реферата: |  |  |
| Студент |  | Кабанов Н.С. |
| Руководитель |  | Морозов С.М. |

**АННОТАЦИЯ**

В данной работе разработан функционал вывода необходимых элементов навигации по параметрам прибора, сделан интерфейс ввода значений с возможностью дополнения структуру и так же реализован функционал экспорта данных для дальнейшей записи настроечной БД в прибор такого так СПТ962. Графический настройщик сделан в среде Windows Form с помощью языка программирования C#.

**SUMMARY**

In this work, a functionality has been developed for displaying the necessary navigation elements according to the device parameters, an interface for entering values has been created with the ability to add structure, and a functionality for exporting data has also been implemented for further recording of the configuration database into a device such as SPT962. The graphical customizer is made in the Windows Form environment using the C# programming language.

**СОДЕРЖАНИЕ**

[ОСНОВНЫЕ ТЕРМИНЫ 5](#_Toc153914361)

[ВВЕДЕНИЕ 7](#_Toc153914362)

[ПОСТАНОВКА ЗАДАЧИ 8](#_Toc153914363)

[1. РЕЗУЛЬТАТЫ РАБОТЫ В ВЕСЕННЕМ СЕМЕСТРЕ 9](#_Toc153914364)

[1.1. Анализ существующих решений навигации, отображения данных и выгрузки данных 9](#_Toc153914365)

[1.1. Оценка их возможностей и функционала 11](#_Toc153914366)

[1.2. Реализация навигации, окна отображения значений 12](#_Toc153914367)

[1.4. Реализация метода выгрузки данных из DataGridView 17](#_Toc153914368)

[1.5. Реализация метода загрузки данных в окно отображения значений. 19](#_Toc153914369)

[1.6. Дополнительный функционал программы 21](#_Toc153914370)

[2. ПЛАН РАБОТЫ НА ОСЕННИЙ СЕМЕСТР 23](#_Toc153914371)

[ЗАКЛЮЧЕНИЕ 24](#_Toc153914372)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25](#_Toc153914373)

[Приложение А 26](#_Toc153914374)

# ОСНОВНЫЕ ТЕРМИНЫ

* DataGridView – это компонент пользовательского интерфейса, предоставляемый библиотекой Windows Forms для работы с табличными данными в приложениях, написанных на языке программирования C#. Этот компонент представляет собой гибкую и удобную таблицу, которая позволяет отображать, редактировать и взаимодействовать с данными в виде сетки. [1].
* TreeView – элемент управления (control), который используется для отображения иерархических данных в виде древовидной структуры. Он позволяет пользователю легко взаимодействовать с иерархией элементов. [3].
* XML — стандартный формат для обмена данными, который используется для хранения и передачи информации в структурированном виде. XML используется в различных областях программирования, включая язык C# и технологии Windows Forms. [8].
* Настроечная БД — созданный файл в формате XML, который необходимо загружать в программу “Конфигуратор” для дальнейшей интеграции с прибором.
* LINQ — технология запросов, интегрированная непосредственно в язык программирования C#. Она предоставляет удобный и выразительный способ выполнения запросов и манипуляций с данными независимо от их источника. В контексте Windows Forms LINQ может использоваться для более удобного и читаемого кода при работе с коллекциями данных, базами данных, XML и другими источниками данных. [6].
* Сериализация — это процесс преобразования объектов в формат XML, который можно сохранить или передать по сети. [2].
* Десериализация — это процесс преобразования данных из формата, в котором они были сохранены или переданы (например, в формате XML), обратно в объекты или структуры данных в программе.
* XmlSerializer — это класс, предоставляющий функциональность сериализации и десериализации объектов в и из формата XML. Он является частью пространства имен System.Xml.Serialization и предоставляет удобный способ работы с XML-данными. [4].
* Обработчики событий в Windows Forms (WinForms) представляют собой механизм для обработки событий, таких как щелчок мыши, нажатие клавиш, изменение текста и другие взаимодействия пользователя с графическим интерфейсом приложения. Каждый элемент управления в WinForms может генерировать события, такие как Click, KeyPress и т.д. Для обработки этих событий нужно создать обработчики, которые будут вызываться при возникновении событий.
* Click – это обработчик события щелчка мыши на элементе управления, таком как кнопка. Это событие происходит, когда пользователь кликает (нажимает и отпускает) левой кнопкой мыши на элементе. Обработчик Click позволяет привязать определенные действия к этому событию.
* Load – это событием формы, которое используется для выполнения дополнительных действий при загрузке формы, таких как начальная инициализация элементов управления.
* After\_Select – это обработчик события для реагирования на выбор узла в элементе управления TreeView. Это событие срабатывает после того, как пользователь выбрал узел в дереве. Это часто используется для выполнения действий, связанных с выбранным узлом, например, отображения информации о выбранном элементе или взаимодействия с другими элементами формы на основе выбора.

# ВВЕДЕНИЕ

В настоящее время отрасль энергетики и теплоснабжения сталкивается с постоянными вызовами в области повышения эффективности и оптимизации процессов учета тепловой энергии.

Целью моей университетской практики является разработка графического настройщика для приборов тепловычислителей. Этот проект направлен на улучшение управления и настройки данных устройств, предоставляя более удобный и интуитивно понятный интерфейс для технических специалистов и операторов систем теплоснабжения.

Создание графического настройщика не только способствует повышению эффективности эксплуатации тепловычислителей, но также открывает новые возможности для внедрения современных технологий в области учета тепловой энергии. Разработка удобного и инновационного инструмента для настройки приборов станет важным шагом в совершенствовании систем теплоснабжения и обеспечении более эффективного управления энергоресурсами.

Важной составляющей программы является удобный механизм навигации по параметрам прибора и практичность ввода необходимых значений для каждого параметра. Программа должна корректно выводить необходимые данные в специализированное окно и структурированно выгружать информацию. Причиной разработки графического настройщика стало устаревшее и не до конца практичное решение – программа "Конфигуратор". С появлением новых сотрудников возникла необходимость в обновлении программного обеспечения, поскольку долгий процесс обучения становится непрактичным. Понятная и структурированная программа должна решить данную проблему.

# ПОСТАНОВКА ЗАДАЧИ

**Актуальность:**

Актуальность создания графического настройщика для приборов тепловычислителей (СПТ96Х, СПТ76Х, СПЕ) заключается в неотложной необходимости совершенствования и упрощения процессов управления и настройки этих устройств. Развитие современных технологий в области учета тепловой энергии требует новых подходов к настройке приборов, обеспечивая техническим специалистам и операторам более удобный и эффективный инструмент управления теплоснабжением. Создание графического настройщика становится важным шагом в повышении функциональности и адаптации этих приборов к современным стандартам и требованиям отрасли.

**Объектом исследования** являются отображение навигации и окна для ввода значений и так же выгрузка заполненных данных в настроечную БД для компании НПФ “Логика”.

**Предметом исследования** являются подходы к реализации навигации, отображения окна, выгрузка БД.

**Целью работы** является создание методов навигации, отображения окна ввода данных и выгрузка настроечной БД.

Для достижения цели необходимо выполнить следующие задачи:

* + Анализ существующих решений навигации, отображения данных и выгрузки данных;
  + Оценка их возможностей и функционала;
  + Реализация навигации, окна отображения значений;
  + Реализация механизма выгрузки параметров.
  + Реализация метода загрузки данных в окно отображения значений.

# РЕЗУЛЬТАТЫ РАБОТЫ В ВЕСЕННЕМ СЕМЕСТРЕ

# Анализ существующих решений навигации, отображения данных и выгрузки данных

Изначально способом отображения навигации, было распределение параметров по группам и скрытие их в кнопки, после нажатия кнопки отображаются вложенные в него элементы.

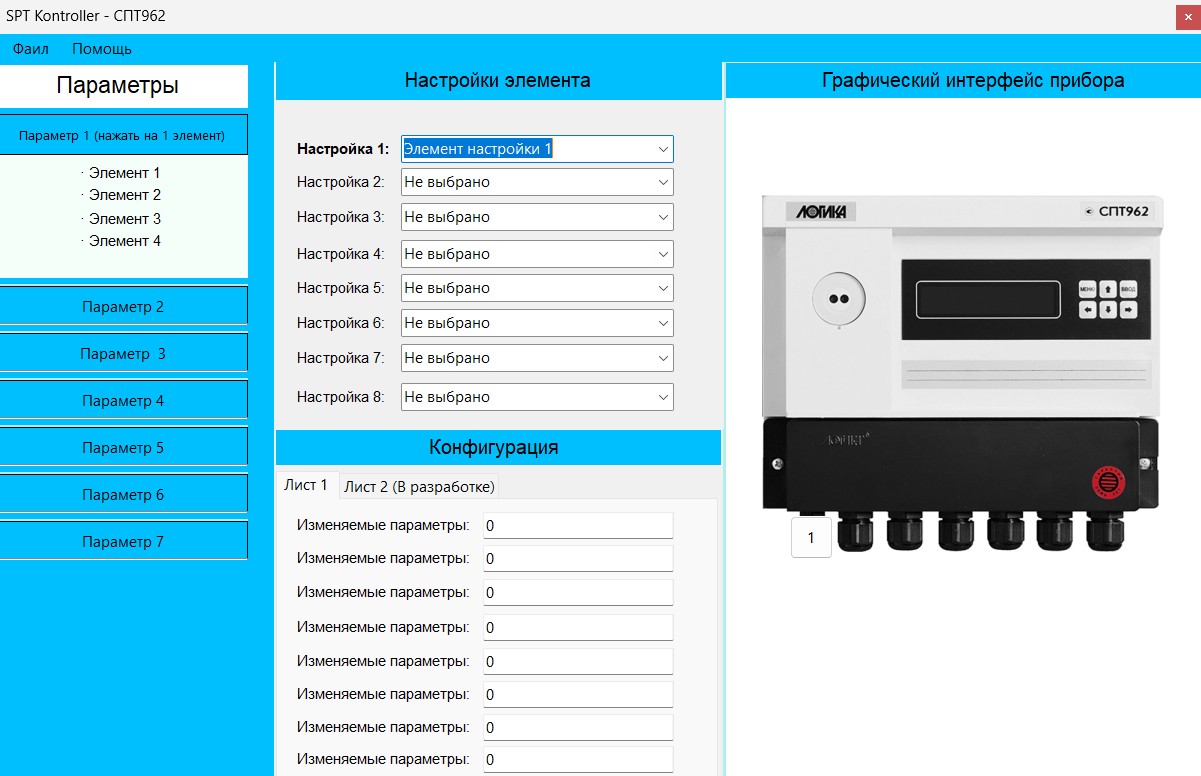


Рисунок 1 – Изначальный экран настройки приборов

Пересмотрев данный подход, пришли к отказу от данного решения, причиной этого стало, слишком большой массив параметров в особенности для приборов СПТ962 и СПТ963. Такое количество значений могло вызвать трудности в дальнейшей работе с программой и отсутствие удобства технического специалиста по настройке приборов.

Заменой навигации стал Treeview, с расширенной структурой ветвей. Основными преимуществами данного подхода стала гибкость настройки Treeview и возможность интеграции с выводом данных в окно для заполнения.

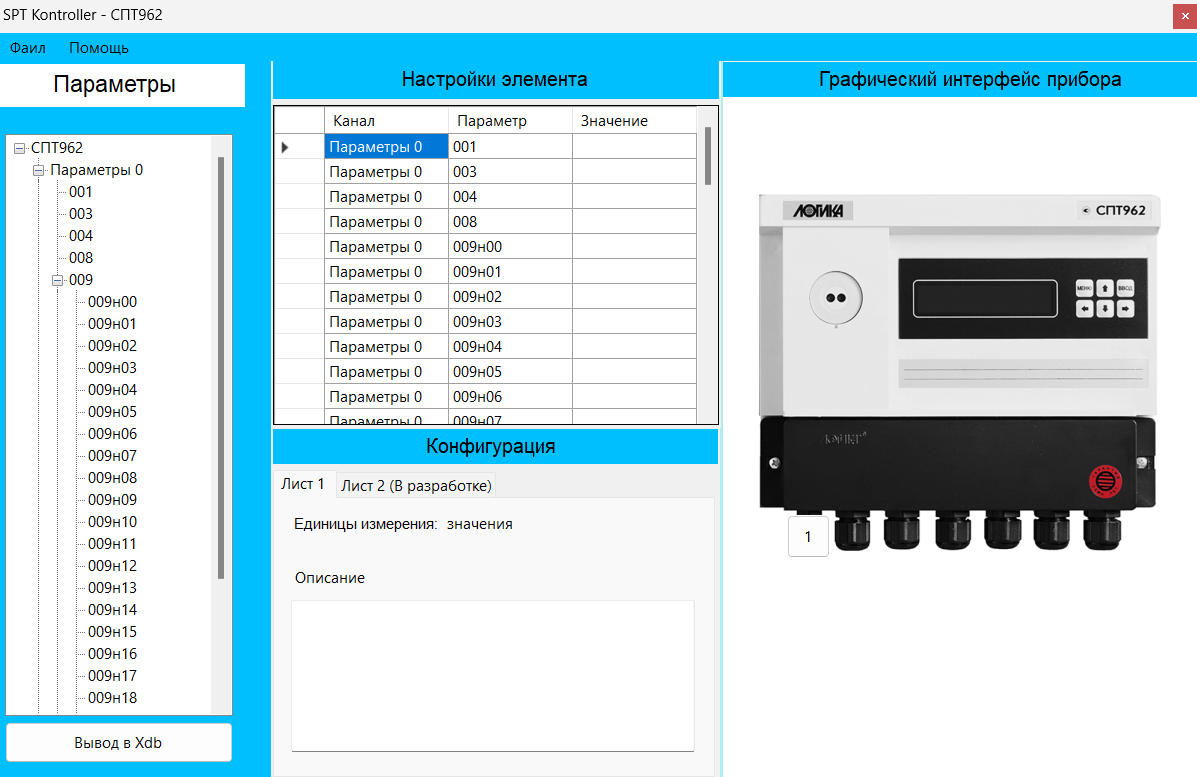


Рисунок 2 – Новый экран настройки прибора СПТ962

Анализируя возможные подходу к выводу информации столкнулись с несколькими проблемами, необходим был гибкий инструментарий, который может полностью отвечать требованиям. Выбор был остановлен на DataGridView. Так как он гибко интегрируется с TreeView.

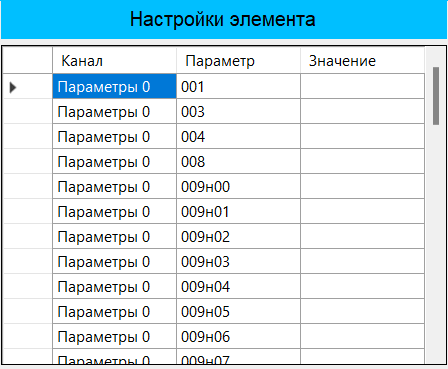


Рисунок 3 – DataGridView в окне настройки прибора СПТ962

Следующей задачей было анализ вариантов исполнения метода выгрузки данных. По техническому заданию необходимостью было выгружать данные в Настроечную БД через XML. Так как дальше БД должно быть загружаться в программу “Конфигуратор” для дальнейшего соединения с прибором и загрузка в него параметров.

Так же необходимо было выгружать данные в формате. xdb, чтобы программа “Конфигуратор” могла принять корректно данные.

# Оценка их возможностей и функционала

**Основные характеристики TreeView в C#:**

* TreeView отображает данные в виде древовидной структуры, где каждый узел может содержать подузлы, создавая иерархию.
* Каждый узел TreeView представляет элемент данных, который может быть представлен пользователю. Эти элементы могут быть текстовыми метками или даже произвольными элементами управления.
* TreeView обычно предоставляет методы для добавления, удаления и изменения узлов и их содержимого. Это обеспечивает динамичное управление данными в древовидной структуре.
* Как и многие элементы управления, TreeView генерирует события, такие как выбор узла, раскрытие или сворачивание узла. Эти события позволяют реагировать на действия пользователя.
* TreeView предоставляет множество параметров для настройки внешнего вида элементов, таких как цвета, шрифты, стили и многое другое.

**Основные возможности DataGridView включают в себя:**

* DataGridView может отображать данные из различных источников, таких как массивы, списки, базы данных и другие. Он предоставляет удобный способ отображения табличной информации.
* Пользователи могут редактировать ячейки прямо внутри DataGridView. Это делает компонент полезным для создания интерактивных приложений, где требуется редактирование данных в таблице.
* DataGridView предоставляет встроенные возможности сортировки и фильтрации данных, что облегчает работу с большими объемами информации.
* Компонент генерирует различные события, которые могут быть использованы для обработки пользовательских действий, таких как изменение данных или выделение ячеек.

**Основные характеристики использования XML в C# и Windows Forms:**

* XML позволяет представлять данные в виде структурированного текста с использованием тегов и атрибутов. Это делает его удобным для хранения разнообразных данных, таких как конфигурационные файлы, настройки приложений, или данные приложений.
* XML часто используется для обмена данными между различными приложениями и платформами. Это обеспечивает стандартизированный формат, который может быть легко интерпретирован и обработан различными системами.
* В C# и Windows Forms XML часто используется для сериализации (преобразования объектов в формат XML) и десериализации (восстановления объектов из формата XML) данных. Это удобно для сохранения состояния приложения или обмена данными между компонентами.
* XML может быть использован для создания конфигурационных файлов, в которых хранятся параметры и настройки приложений. Это облегчает изменение настроек без изменения исходного кода программы.
* В приложениях Windows Forms XML может быть использован для хранения данных, которые необходимо сохранить между сеансами работы приложения, для передачи данных между формами или компонентами приложения, а также для других задач, связанных с обработкой структурированных данных.

# Реализация навигации, окна отображения значений

Создано TreeView, которое заполнили параметрами, так же добавили для него метод treeView1\_AfterSelect, который далее будет необходим для отображения элементов в DataGridView в зависимости от ветви TreeView.

Следующим этапом было создание DataGridView из ветвей TreeView. Создается DataTable, который представляет собой объект, что хранит данные в виде таблицы с колонками и строками. Обозначаются колонки “Канал”, “Параметр”, “Значение”.

Так же создается TagList с именем \_xmlModel, который представляет собой структуру данных, описывающую иерархию каналов, параметров и значений. Объект \_xmlModel удобно использовать для представления иерархической структуры данных.

После мы получаем корневые узлы TreeView, циклом проходим их все и заполняем значениями корневой узел. Далее нам необходимо получить все дочерние элементы. Этот процесс состоит из этапов:

* Получение всех дочерних узлов для текущего дочернего узла.
* Запуск цикла по дочерним узлам текущего дочернего узла, в котором проверяется наличие детей.
* Создание и заполнение объекта TagGroup, если есть дети:
* Печать отладочного сообщения.
* Создание нового объекта TagGroup.
* Заполнение свойств объекта TagGroup.
* Добавление объекта TagGroup к списку тегов (Tag) текущего канала.
* Обход дочерних узлов текущего дочернего узла (TagGroup).
* Получение всех дочерних узлов для текущего дочернего узла (TagGroup).
* Запуск цикла по дочерним узлам текущего дочернего узла (TagGroup), в котором создаются и заполняются объекты TagGroupTag.
* Добавление объектов TagGroupTag к списку тегов (Tag) текущего объекта TagGroup.
* Создание и заполнение объекта Tag, если нет детей:
* Создание нового объекта Tag.
* Заполнение свойств объекта Tag.
* Добавление объекта Tag к списку тегов (Tag) текущего канала.
* Добавление данных в DataTable:
* Добавление строк в DataTable для каждого созданного объекта Channel, TagGroup и Tag. Эти строки включают информацию о каналах, параметрах и значениях, которые затем будут использоваться для заполнения DataGridView.

Следующим этапом создается функция GetChildNodes, которая используется для получения всех дочерних узлов для указанного узла типа TreeNode в TreeView. После определения этой функции, код устанавливает источник данных для DataGridView с использованием объекта dataTable. Рассмотрим подробнее данную функцию:

* Определение функции GetChildNodes – эта функция создана для удобства получения всех дочерних узлов для указанного узла TreeNode.
* Создается пустой список childNodes, предназначенный для хранения всех дочерних узлов.
* С использованием цикла foreach перебираются все узлы в коллекции node.Nodes.
* Каждый узел добавляется в список childNodes.
* Список childNodes возвращается после завершения цикла.

В конце указывается источник данных для DataGridView и он является равным dataTable, который содержит данные о каналах, параметрах и значениях.

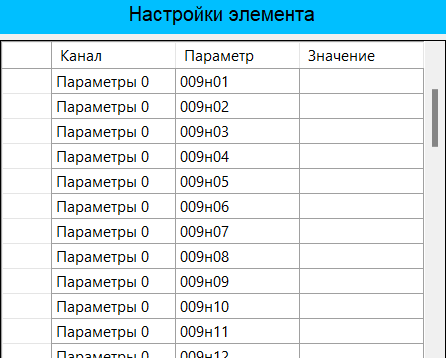


Рисунок 4 – Перенос TreeView в DataGridView

Так же в дополнении есть методы, которые срабатывают в обработчике событий при загрузке формы:

* DataGridViewAutoSizeColumnsMode.Fill – метод, который определяет ширину столбцов в DataGridView. Она будет автоматически подстраиваться под размеры контейнера. В данном случае установлено значение Fill, что означает, что столбцы будут заполнять все доступное пространство в горизонтальном направлении.
* Вызывается метод ConvertTreeViewIntoDataGridView, который, реализует перенос данных из TreeView в DataGridView.

Заключительным этапом является отображение данных о выбранном узле и его дочерних узлах в dataGridView1 после выбора узла в TreeView в методе, который мы создали treeView1\_AfterSelect. Он состоит из следующий операций:

* dataGridView1.Columns.Add("Column1","Канал")- добавляет колонку с именем Column1 и заголовком "Канал" в dataGridView1. Аналогично добавляются еще две колонки с заголовками "Параметр" и "Значение".
* string parent = e.Node.Text - получает текст выбранного узла TreeView и сохраняет его в переменной parent.
* dataGridView1.Rows.Add(parent, "") - добавляет новую строку в dataGridView1 с данными о parent в первой колонке.
* Foreach (var childNode in e.Node.Nodes.Cast<TreeNode>()) - запускает цикл по всем дочерним узлам выбранного узла TreeView.
* AddChildNodeToDataGridView(parent, childNode) - вызывает метод AddChildNodeToDataGridView для каждого дочернего узла.
* Удаление первой строки, если есть дочерние узлы:

if (dataGridView1.Rows.Count > 2) - проверяет, есть ли дочерние узлы, добавленные в dataGridView1.

* dataGridView1.Rows.RemoveAt(0) - если есть дочерние узлы, удаляет первую строку, которая содержит информацию о родительском узле.

Из обработчика событий treeView1\_AfterSelect вызывается метод AddChildNodeToDataGridView, который представляет собой рекурсивную функцию, которая добавляет данные о дочерних узлах TreeView в DataGridView. Ее код состоит из:

* dataGridView1.Rows.Add(parent, node.Text)- добавляет новую строку в dataGridView1 с двумя значениями: parent и node.Text, parent представляет текст родительского узла, переданного из предыдущей функции treeView1\_AfterSelect, node.Text представляет текст текущего дочернего узла.
* foreach (var childNode in node.Nodes.Cast<TreeNode>())- запускает цикл по всем дочерним узлам текущего узла node.
* AddChildNodeToDataGridView(parent, childNode) - рекурсивно вызывает саму себя для каждого дочернего узла childNode. Таким образом, происходит обход всех дочерних узлов текущего узла и их добавление в dataGridView1.

Этот метод выполняет глубокий обход дочерних узлов и добавляет их в dataGridView1, начиная с текущего узла и рекурсивно обрабатывая все его поддеревья. Таким образом, для каждого узла в TreeView добавляется соответствующая строка в DataGridView.

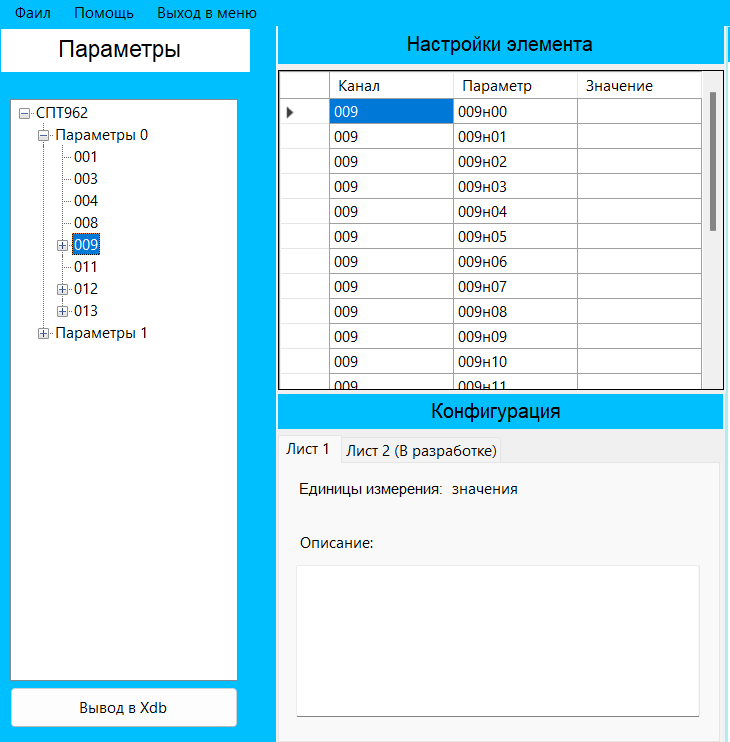


Рисунок 5 – DataGridView при выборе определенной ветви в TreeView

# 1.4. Реализация метода выгрузки данных из DataGridView

Создан метод CreateXdb, который начинается с перебора циклом всех строк в DataGridView1, получение значения из первого столбца текущей строки и сохранение в переменной channel. Аналогично получаются значения из второго и третьего столбцов и сохраняются в переменных parametr и value соответственно.

Далее используется LINQ-запрос для поиска узла в структуре \_xmlModel. Если находится узел с заданными значениями channel и parametr, то он сохраняется в переменную result.

Если найден узел (result не равен null), то его значение (Value) обновляется значением из value. В случае, когда узел с заданными значениями channel и parametr не существует, то предполагается, что это узел с детьми типа TagGroup. Происходит поиск такого узла, затем находится объект типа Tag внутри этого TagGroup с нужным Id, и его значение обновляется значением из value.

Следующий пункт – это создание XmlSerializer и сериализация в файл. Создается экземпляра XmlSerializer для сериализации объектов типа TagList, пустое пространство имен для устранения namespace в XML и диалоговое окно для сохранения файла. В конце создается FileStream для создания нового файла XML, и модель \_xmlModel сериализуется в этот файл с использованием XmlSerializer.

Так же был создан отдельный файл “XmlModel.cs”, который представляет собой определение классов для сериализации и десериализации данных XML. Атрибуты XmlAttribute используются для маппинга свойств класса на атрибуты XML-элементов. Код public string Ordinal {get; set;} – это свойство для атрибута Ordinal и так же со всеми остальными классами. Эти классы предназначены для использования с сериализатором XML (XmlSerializer), чтобы объекты могли быть преобразованы в XML-документ и обратно.

Последним этапом выгрузки данных является обработчик событий для кнопки при нажатии срабатывает метод CreateXdb и запускается процесс экспорта из DataGridView.

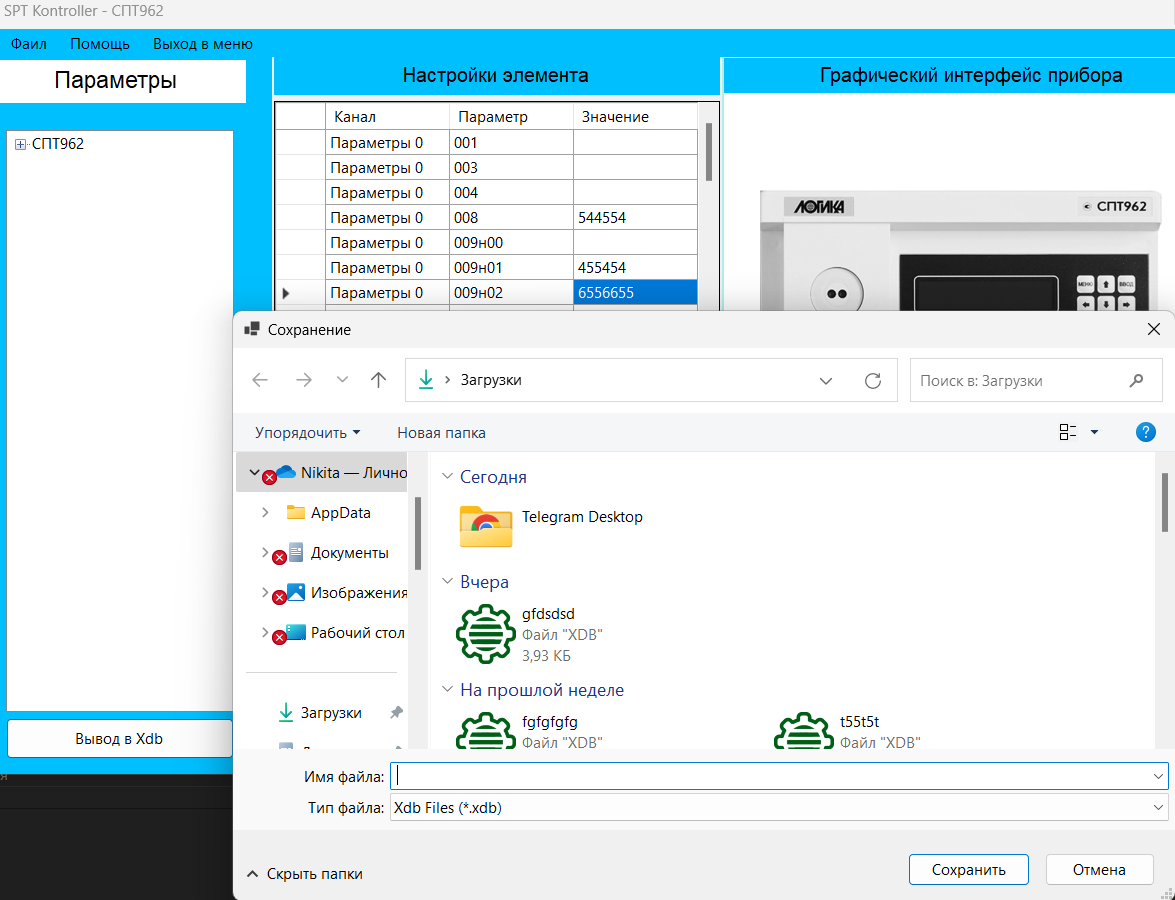


Рисунок 6 – нажатие на кнопку “Выгрузка Xdb” и запуска метода CreateXdb

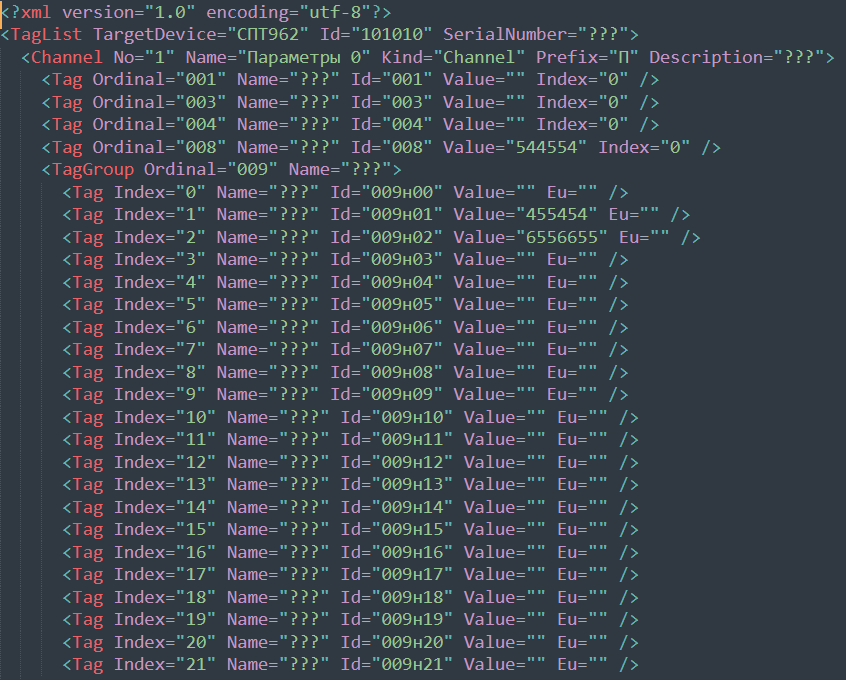


Рисунок 7 – выгруженный файл с расширением .xdb заполненный данными

# 1.5. Реализация метода загрузки данных в окно отображения значений.

Был реализован метод загрузки данных из файла .xdb, который будет заполнять столбцы и значения в DataGridView. Сначала происходит очистка DataGridView от предыдущих данных, чтобы гарантировать чистоту при новой загрузке.

После создается новый объект DataTable с тремя колонками: "Канал", "Параметр", "Значение", который будет использоваться для заполнения DataGridView. Далее происходит десериализация данных из XML-файла в объект \_xmlModel с использованием XmlSerializer. Так же реализован проверка по значению в TargetDevice, чтоб оно соответствовало “СПТ962”, если нет, то выводится сообщения об ошибке. Затем происходит итерация по объекту \_xmlModel, и для каждого тега (Tag) или группы тегов (TagGroup) добавляются строки в DataTable и в виде источника данных для DataGridView1 устанавливается dataTable.

Для работы данного метода был добавлен обработчик событий “Click” для элемента меню “Открыть проект”, который создает диалоговое окно и сделан фильтр по расширению файла “.xdb”.

Открывается диалоговое окно для выбора файла. Если пользователь выбирает файл и нажимает "OK", то выполняется метод LoadXdb с выбранным путем к файлу в качестве аргумента.

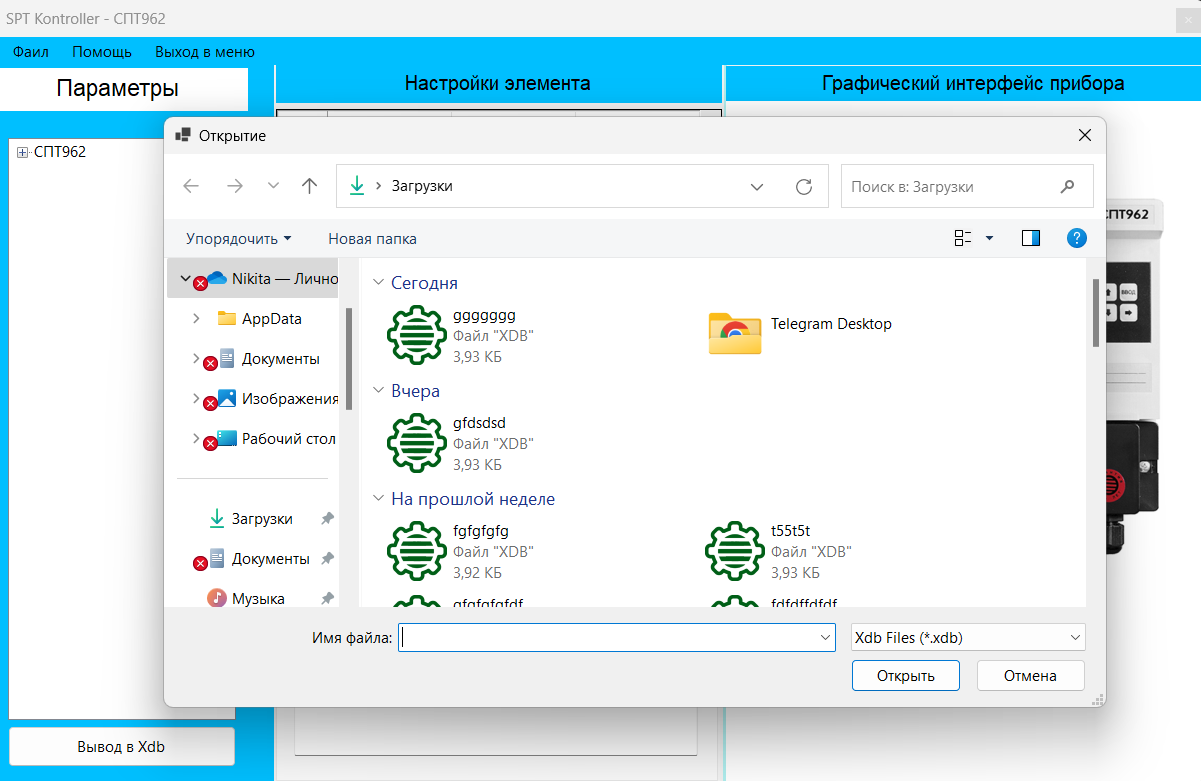


Рисунок 8 – диалоговое окно после нажатия на элемент меню “Открыть проект”

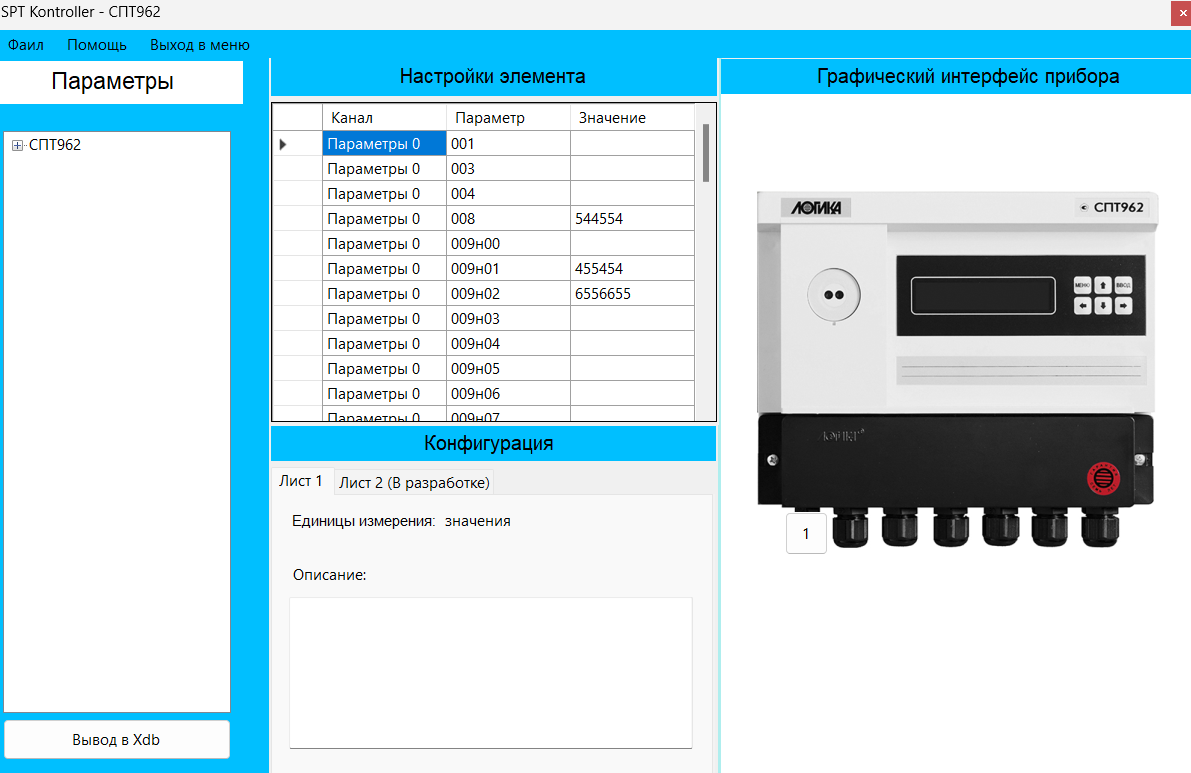


Рисунок 9 – загруженные данные из файла отображаются в DataGridView в столбце “Значения”

# 1.6. Дополнительный функционал программы

В данной работе так же был реализован дополнительный функции, которые необходимы для каждой программы, такие как:

Сделана новая форма для настройки прибора СПТ963. Она приведена полностью к общей стилистике приложения и создано TreeView. Реализован DataGridView, который переносит данные из TreeView.

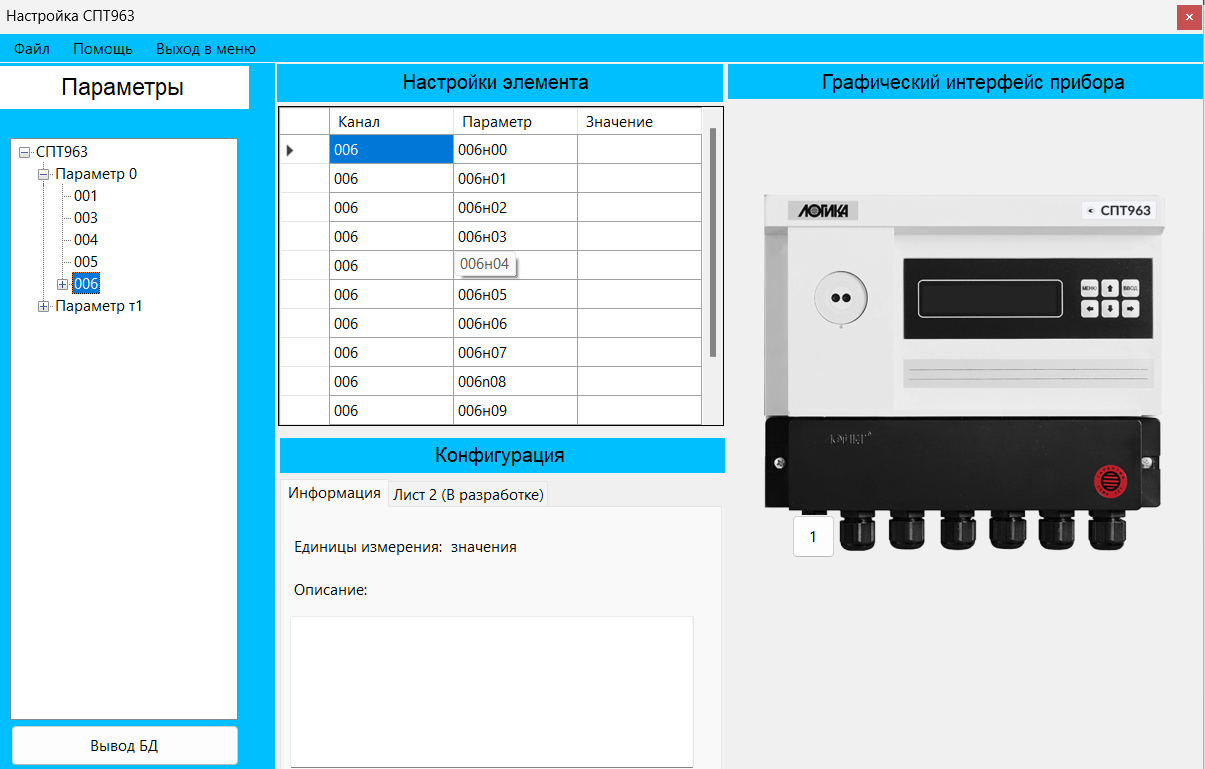


Рисунок 10 – форма для настройки прибора СПТ963

Подключение обработчиков событий “Click” для каждого элемента меню во всех формах. Сделан выход из приложения, создание БД для СПТ 962 и СПТ963.

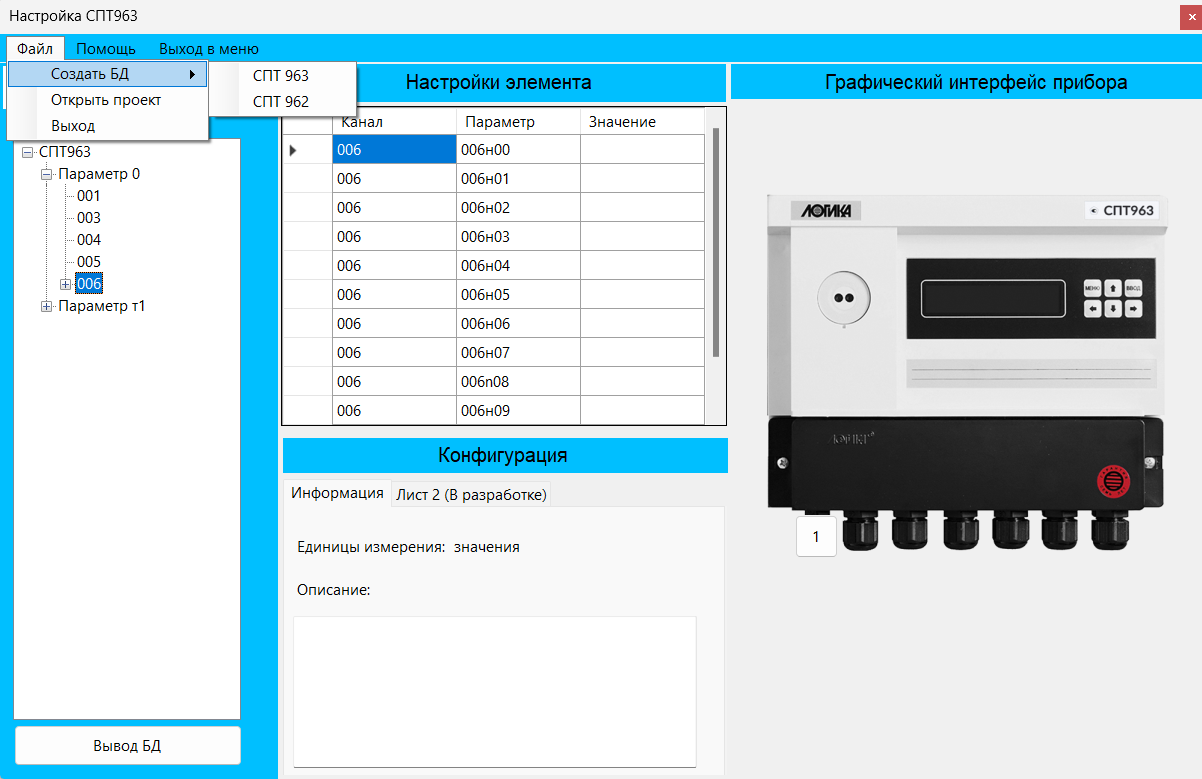


Рисунок 11 – элементы меню, к которым подключены обработчики событий

Создано модальное окно, которое является отдельной формой. Сделано оно для отображения справки о приложении. Включает в себя текстовую информацию об компании разработчике и пользовательском соглашении.

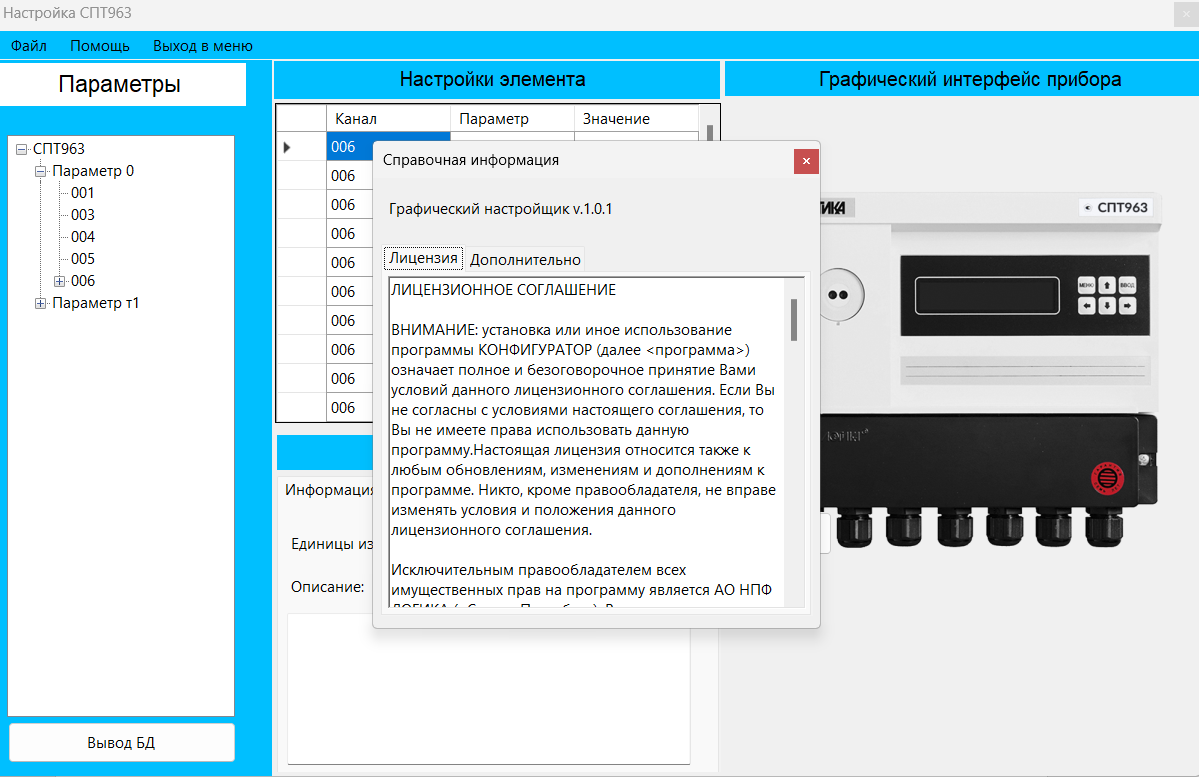


Рисунок 12 – модальное окно справки

Реализован отдельный элемент навигации “Выход в меню” для перехода в главную форму (начальную).



Рисунок 13 – элемент навигации “Выход в меню”

# ПЛАН РАБОТЫ НА ОСЕННИЙ СЕМЕСТР

В ходе проведенной работы был разработан функционал навигации, отображения данных из TreeView в DataGridView, выгрузка параметров в XML, загрузка данных из файла и дополнительные функции для форм.

План на осенний семестр состоит из:

* Доработка DataGridView, чтобы значения заполненных параметров, сохранялись при переходе по ветвям TreeView.
* Разработка метода для хранения данных Name, для каждого элемента TreeView, которые будут передаваться в XML для выгрузки.
* Создание метода для отображения “Единиц измерения”, для каждого элемента из DataGridView при клике на строку параметра.
* Модификация метода загрузки проекта, чтоб независимо от формы он корректно получал путь файла и принимала данные форма в зависимости от TargetDevice.
* Прикрепление к методам “Выгрузки данных”, “Открытие проекта”

# ЗАКЛЮЧЕНИЕ

В рамках разработки графического настройщика для тепловычислителей был создан функционал, направленный на улучшение процессов управления и настройки приборов. Новый инструмент предоставляет техническим специалистам и операторам более удобный и интуитивно понятный интерфейс, современный и эффективный метод взаимодействия с приборами.

В ходе разработки были выполнены следующие задачи:

* Произведен анализ существующих решений в области навигации, отображения данных и выгрузки настроечных данных, что позволило выявить сильные и слабые стороны существующих решений.
* Реализованы механизмы навигации по параметрам прибора и отображения значений, обеспечивающие легкость в использовании и понимание структуры приборов.
* Разработан механизм выгрузки параметров в формате XML, что позволяет обновлять настроечную базу данных на приборах семейства СПТ962.
* Реализован механизм загрузки данных из настроечной базы данных, что позволяет оперативно вносить изменения в интерфейс и структуру приборов.

Важным моментом стало обеспечение удобного механизма навигации по параметрам прибора и практичности ввода значений для каждого параметра. Программа корректно выводит необходимые данные в специализированное окно и структурированно выгружает информацию.

Таким образом, создание графического настройщика не только способствует повышению эффективности эксплуатации тепловычислителей, но также открывает новые перспективы для внедрения современных технологий в области учета тепловой энергии. Разработка удобного инструмента для настройки приборов станет важным шагом в совершенствовании систем теплоснабжения и обеспечении более эффективного управления энергоресурсами.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Windows Form [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-7.0> (дата обращения: 14.05.2023)
2. Сериализация XML [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/dotnet/standard/serialization/introducing-xml-serialization (дата обращения: 20.11.2023)
3. Windows Forms. Программирование на C# [Электронный ресурс].

— URL: <http://csharpcoding.org/category/windows-forms/> (дата обращения: 11.10.2023).

1. Сериализация в XML. XmlSerializer [Электронный ресурс]. — URL: https://metanit.com/sharp/tutorial/6.4.php".(дата обращения: 02.12.2023).
2. Полное руководство по языку программирования С# 6.0 и платформе .NET 4.6 [Электронный ресурс]. — Режим доступа: http://metanit.com/ sharp/tutorial/ (дата обращения: 10.12.2016).
3. Абрамян М. Э. А13 Технология LINQ на примерах. Практикум с использованием электронного задачника Programming Taskbook for LINQ. – М. : ДМК Пресс, 2014. – 326 с. : ил. ISBN 978-5-94074-981-3
4. Петцольд Ч. Программирование для Microsoft Windows на С#. Пер. с англ.— М.: Издательско-торговый дом «Русская Редакция», 2002. – 576с.
5. Дальви Д. XML для разработчиков-профессионалов .NET/ Д. Дальви. М.: Лори, 2003 - 656 с.
6. МальцевА. И. Алгоритмы и рекурсивные функции.— 2-е изд.— М.: Наука. Гл. ред. физ.-мат. лит., 1986.— 368 с.
7. Шилдт Г. Полное руководство С# 4.0 [Текст]: учебное пособие / Г. Шилдт — пер. с англ. Берштейн И. В. — Москва: Вильямс, 2012.— 1051 с.

# Приложение А

Ссылка на проект: <https://github.com/shoker-droid/spt-kontroller-ver.1.0.3.git>