Тема: **Разработка модуля обработки групповых запросов Modbus в среде Owen Logic**

СОДЕРЖАНИЕ

Список сокращений 3
Введение 3
1 Обзорная часть 3
1.1 Подход к построению среды разработки 5
1.1.1 Принципы DDD 5
1.1.2 Структура и компоненты на основе DDD 6
1.1.3 Преимущества применения DDD 7
1.1.4 Применение DDD для расширяемости 7
1.2 Анализ сред разработки 8
1.2.1 Codesys 8
1.2.2 TIA Portal (Siemens) 9
1.2.3 Trace Mode 9
1.2.4 Owen Logic 10
1.2.5 Сравнительная таблица сред разработки 11
2 Расчетно-конструкторская часть ..... -
2.1 Архитектура системы групповых запросов Modbus ..... -
2.1.1 Переменные в среде Owen Logic ..... -
2.1.2 Ограничение протокола Modbus ..... -
2.1.3 Требования к группировке переменных ..... -
2.1.4 Требования к пользовательскому интерфейсу ..... -
2.2 Архитектура модуля обработки групповых запросов ..... -
2.2.1 Диаграмма компонентов ..... -
2.2.2 Основные компоненты и их ответственность ..... -
2.3 Алгоритмы работы модуля ..... -
2.3.1 Алгоритм группировки переменных ..... -
2.3.2 Алгоритм проверки совместимости переменных ..... -
2.3.3 Алгоритм сортировки и упорядочивания переменных ..... -
2.3.4 Алгоритм слияния совместных групп ..... -
2.4 Обработка ограничений протоколов Modbus ..... -
2.4.1 Ограничения для протоколов RTU и ASCII ..... -

2.4.2 Расчет максимального размера запроса ..... -
2.5 Разработка тестовых сценариев ..... -
3 Экспериментальная часть ..... -
3.1 Реализация логики группировки ..... -
3.1.1 Класс ModbusRequestInfo ..... -
3.1.2 Класс CreateModbusRequestServicel ..... -
3.1.3 Класс ModbusRequestGroupBuilder ..... -
3.1.4 Структура ModbusRequestGroup ..... -
3.2 Реализация обработки ограничений протокола ..... -
3.2.1 Интерфейс IModbusRegisterLimitByProtocol ..... -
3.2.2 Реализация ограничения ModbusRegisterLimitByRtuProtocol ..... -
3.2.2 Реализация ограничения ModbusRegisterLimitByAsciiProtocol ..... -
3.3 Интеграция с пользовательским интерфейсом ..... -
3.3.1 Взаимодействие с GlobalVariableDictionary и IGeneratorFacade ..... -
3.3.2 Механизм анализа и сборки бинарных структур Bingo ..... -
3.4 Валидация запросов и проверка корректности переменных ..... -
3.4.1 Валидатор регистров Rs485RegistersPerRequestCountValidator ..... -
3.4.2 Цепочка валидаторов ValidatorChainProxy/ValidatorChainBaseProxy ..... -
3.4.3 Презентер устройства Rs485DevicePresenter ..... -
4 Тестирование модуля ..... -
4.1 Юнит-тестирование алгоритмов ..... -
4.2 Тестирование валидаторов RS-485 ..... -
4.3 Интеграционное тестирование ..... -
Заключение ..... -
Список использованных источников ..... -
ПРИЛОЖЕНИЕ А. Исходный код модуля группового опроса Modbus-устройств ..... -
ПРИЛОЖЕНИЕ Б. Руководство пользователя модуля группового опроса Modbus ..... -

Список сокращений

OL - OWEN Logic

FBD - Function Block Diagram

ST - Structured Text

DDD – Domain-Driven Design

ПЛК - Программируемый Логический Контроллер

Введение

В условиях современного промышленного производства автоматизация технологических процессов становится ключевым фактором, обеспечивающим повышение производительности, снижение затрат и улучшение качества работы. Одним из важнейших инструментов для решения этих задач является специализированное программное обеспечение для разработки алгоритмов управления различными устройствами. Одним из таких продуктов является **OL** — среда разработки, предназначенная для автоматизации управления устройствами компании «ОВЕН».

OL представляет собой мощный инструмент для проектирования и реализации алгоритмов управления с помощью графического языка программирования **FBD**, который соответствует международному стандарту **МЭК 61131-3**. Этот стандарт описывает методы программирования устройств автоматизации, таких как программируемые логические контроллеры (PLC), что позволяет создавать надежные и эффективные решения для управления различными устройствами и процессами.

Среда **OL** поддерживает широкий спектр функциональных возможностей, включая разработку алгоритмов для **программируемых реле** серий **ПР100**, **ПР200**, **ПР205**, а также интеграцию с **панелями оператора** и другими устройствами. Для взаимодействия с внешними системами **OL** использует такие протоколы связи, как **Modbus**, что позволяет seamlessly интегрировать систему в более сложные автоматизированные комплексы и сети. Кроме того, **OL** включает в себя инструменты для симуляции работы алгоритмов, что даёт возможность протестировать разрабатываемые решения до их внедрения на реальных устройствах, существенно ускоряя процесс разработки и устранения ошибок.

Документация к **OL**, опубликованная на официальном сайте компании «ОВЕН» [1], подробно описывает её функциональные возможности, включая работу с программируемыми реле серий **ПР100**, **ПР200** и **ПР205**, а также с панелями оператора. Эти материалы являются ценным ресурсом для пользователей, желающих эффективно использовать возможности **OL** и разрабатывать решения для автоматизации технологических процессов.

Современные системы автоматизации требуют не только высокой функциональности, но и удобства в использовании, что позволяет специалистам быстро и эффективно решать задачи. В этой связи **OL** представляет собой инструмент, ориентированный как на опытных разработчиков, так и на тех, кто только начинает работать в сфере автоматизации. Простота в освоении, гибкость и расширяемость системы

делают **OL** удобным выбором для проектирования решений в самых различных отраслях, от промышленности до сельского хозяйства.

Вместе с тем, с учетом быстрого развития технологий и появления новых потребностей пользователей, **OL** требует постоянного совершенствования. Усовершенствование функционала данной среды и добавление новых возможностей позволяет расширить её область применения, улучшить качество решений и ускорить процесс разработки. В рамках дипломной работы будет предложен анализ существующих функциональных возможностей **OL** с целью выявления направлений для дальнейшего улучшения и расширения её функционала.

1 ОБЗОРНАЯ ЧАСТЬ

1.1 Подход к построению среды разработки

Архитектура программного обеспечения играет ключевую роль в разработке и функционировании информационных систем. В случае с OL используется подход DDD, который ориентирован на создание гибких и масштабируемых систем, где бизнес-логика и предметная область играют главную роль в проектировании.

1.1.1 Принципы DDD

DDD — это подход к проектированию программных систем, который ставит в центр внимания предметную область и бизнес-логику [2]. Внутри OL DDD используется для упрощения разработки и возможного расширения функционала в будущем. Основные принципы DDD, которые применяются в системе, включают:

- **Моделирование предметной области**: Всё программное обеспечение строится с учётом требований и особенностей бизнес-логики, связанной с автоматизацией технологических процессов. Все компоненты системы проектируются в тесном взаимодействии с экспертами в предметной области, чтобы учесть все аспекты работы с программируемыми реле и панелями компании «ОВЕН».
- **Единый язык (Ubiquitous Language)**: В проектировании системы важно использовать общий язык, понятный как техническим специалистам, так и экспертам в области автоматизации. Это означает, что терминология и концепции, используемые в коде, документации и при общении с пользователями, должны быть одинаковыми, чтобы избежать недоразумений и упростить взаимодействие между всеми участниками разработки.
- **Разделение на контексты (Bounded Contexts)**: Система разделяется на различные **контексты**, каждый из которых охватывает определённую часть бизнес-логики. Это позволяет избежать излишней сложности и обеспечивать независимость различных компонентов. Например, контекст для работы с алгоритмами управления может быть отделён от контекста работы с базами данных или внешними системами. Это помогает лучше управлять зависимостями и облегчает поддержку и расширение системы.

1.1.2 Структура и компоненты на основе DDD

Для более детального описания архитектуры на основе принципов **DDD**, можно выделить несколько важных аспектов, которые помогут лучше понять роль каждого слоя и компонента. [3] Основные слои, которые составляют эту архитектуру:

- **Слой домена (Domain Layer)**: Этот слой включает в себя бизнес-логику и модели, которые отвечают за работу с программируемыми реле и панелями. В нем определяются основные сущности, такие как алгоритмы управления, параметры устройств и другие элементы.
- **Слой приложения (Application Layer)**: Этот слой взаимодействует с пользователем или другими приложениями, предоставляя интерфейсы для работы с бизнес-логикой. В нем находятся сервисы, которые управляют запросами, получают данные и передают их в слой домена
- **Инфраструктурный слой (Infrastructure Layer)**: Этот слой отвечает за взаимодействие с внешними системами и технологическими компонентами, такими как базы данных, внешние устройства, системы мониторинга и т.д.
- **Интерфейсы и внешние компоненты (Interfaces and External Components)**: Это взаимодействие с пользователем, другими приложениями или внешними сервисами, например, через графический интерфейс или командную строку.

Схематическое представление архитектуры DDD, представлено на рисунке 1.1.

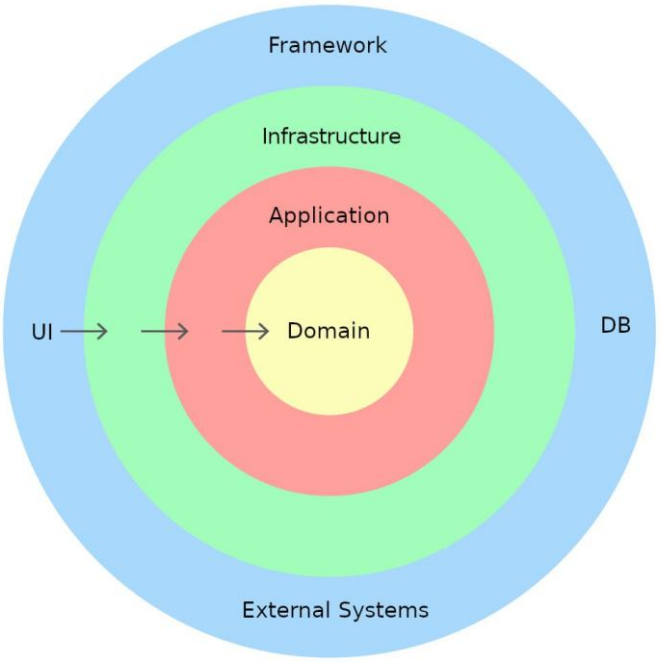

Рисунок 1.1 – Схематическое представление архитектуры DDD

1.1.3 Преимущества применения DDD

Использование **DDD** в OL позволяет достичь нескольких ключевых целей:

- **Модульность и независимость компонентов**: Разделение системы на контексты и использование чётких интерфейсов между ними позволяет создавать независимые компоненты, которые можно легко изменять, заменять или масштабировать.
- **Гибкость и масштабируемость**: Архитектура, основанная на DDD, позволяет гибко адаптировать систему под изменяющиеся требования. Это особенно важно для среды программирования, которая должна поддерживать разнообразие задач, связанных с управлением технологическими процессами.
- **Упрощение взаимодействия между разработчиками и бизнес-экспертами**: Единый язык, используемый в DDD, способствует лучшему пониманию предметной области и уменьшению недопонимания между участниками разработки. Это позволяет быстрее разрабатывать и внедрять новые функции и решать возникающие проблемы.

1.1.4 Применение DDD для расширяемости

С помощью **DDD** можно легко добавлять новый функционал и расширять программу. Это может быть особенно полезно в будущем, когда появятся новые типы устройств или технологии, которые потребуют интеграции в существующую систему. Благодаря ясному разделению на слои и контексты, расширение системы не будет нарушать её стабильность, а новые функции можно будет добавить с минимальными затратами.

1.2 Анализ сред разработки

Для выбора наиболее подходящей среды разработки для автоматизации процессов в промышленности важно учитывать несколько факторов: совместимость с оборудованием, поддерживаемые языки программирования, возможности интеграции с другими системами, а также требования по визуализации и отладке. В контексте автоматизации задач с использованием ПЛК и панелей управления, различные среды могут иметь свои преимущества в зависимости от потребностей проекта.

1.2.1 Codesys

Codesys является платформой для разработки программного обеспечения для программируемых логических контроллеров (ПЛК). Она поддерживает широкий спектр оборудования от различных производителей и предоставляет многоязыковую среду разработки, соответствующую стандарту IEC 61131-3 [4].

Ключевой функционал:

- Поддержка языков программирования FBD, LD, ST, IL и SFC.
- Мощные средства отладки, включая пошаговое выполнение и точечные остановки (breakpoints).
- Расширенная работа с коммуникациями: поддержка различных протоколов (Modbus, CANopen, BACnet и т.д.).
- Интеграция SCADA-систем для визуализации и мониторинга процессов.
- Широкая библиотека готовых функциональных блоков.
- Возможность управления системой с мобильных устройств.

Недостатки:

- Высокая сложность освоения для начинающих пользователей.

Codesys подходит для работы с многозадачными проектами и интеграции с разными производителями, но требует больше времени и ресурсов на настройку и изучение. Она также обладает высокой масштабируемостью и поддерживает интеграцию с современными технологиями, такими как IoT и Industry 4.0, что делает её подходящей для сложных и крупных проектов. Однако её сложность и потребность в обучении могут быть барьером для небольших компанийи.

1.2.2 TIA Portal (Siemens)

TIA Portal — это комплексная среда разработки от Siemens, предназначенная для программирования и конфигурирования устройств из линейки SIMATIC, включая ПЛК, HMI и SCADA [5].

Ключевой функционал:

- Единая среда разработки для всего оборудования Siemens.
- Поддержка языков стандарта IEC 61131-3, включая FBD, LD и ST.
- Визуализация процессов с помощью SCADA-систем.
- Расширенные возможности анализа производительности оборудования.
- Интеграция с системами управления энергопотреблением.

Недостатки:

- Ограниченная совместимость с оборудованием других производителей.
- Отсутствие встроенных симуляторов для быстрой отладки без подключения устройства.

TIA Portal является мощным инструментом для крупных предприятий, использующих оборудование Siemens, однако избыточна для более узких задач автоматизации.

1.2.3 Trace Mode

Trace Mode — это интегрированная система для автоматизации управления, мониторинга и сбора данных (SCADA/HMI), разработанная российской компанией AdAstrA [6].

Ключевой функционал:

- Создание человеко-машинных интерфейсов (HMI) и SCADA-систем.
- Поддержка большого количества протоколов связи, включая Modbus, OPC и BACnet.
- Интеграция с внешними базами данных для хранения параметров и результатов работы систем.
- Расширенные функции визуализации, включая 3D-графику.
- Инструменты аналитики для оценки эффективности работы системы.

Недостатки:

- Высокие системные требования.
- Сложность настройки для пользователей, не знакомых с системами SCADA.

Trace Mode ориентирован на создание SCADA-систем, а не на программирование логики работы реле и панелей. Его возможности визуализации и аналитики подходят для сложных систем мониторинга и управления.

1.2.4 Owen Logic

OWEN Logic — это специализированная среда разработки, предназначенная для программирования реле и панелей управления компании «ОВЕН». Программное обеспечение поддерживает два языка программирования — FBD и ST, что позволяет эффективно решать задачи автоматизации для оборудования этой компании.

Ключевой функционал**:**

- Поддержка языков программирования**:** FBD и ST, что соответствует стандарту IEC 61131-3.
- Интуитивно понятный интерфейс**:** Удобный графический интерфейс для создания и отладки программ.
- Возможности расширения**:** Программа позволяет создавать собственные функциональные блоки (макросы) и использовать их в других проектах.
- Бесплатное использование**:** Доступна бесплатно для пользователей, что делает её привлекательной для небольших и средних предприятий.
- Интеграция с системами управления и мониторинга**:** Возможность интеграции с внешними SCADA-системами для визуализации.

Недостатки**:**

- Ограниченная совместимость**:** Программное обеспечение предназначено исключительно для работы с оборудованием компании «ОВЕН».
- Ограниченные возможности визуализации**:** Отсутствие встроенных инструментов для создания сложных SCADA-интерфейсов.

OL подходит для автоматизации задач с использованием оборудования компании «ОВЕН», но ограничена в плане совместимости с другими производителями и возможностями визуализации. Для небольших предприятий это решение является оптимальным благодаря бесплатному доступу и простоте использования, но для более сложных проектов с интеграцией внешних систем или создания сложных SCADA-интерфейсов могут потребоваться дополнительные инструменты визуализации.

1.2.5 Сравнительная таблица сред разработки

Среда разработки	Поддержка языков	Совместимость с оборудованием	Стоимость лицензии	Поддержка протоколов связи	Отладочные средства
OWEN Logic	FBD, ST	ОВЕН (ПР100, ПР102, ПР200, ПР205, ИПП120)	Бесплатно	Modbus, CANopen	Пошаговая отладка
Codesys	FBD, LD, ST, IL, SFC	Широкая (множество производителей)	Платная	Modbus, CANopen, OPC, BACnet	Пошаговая отладка, точки остановки
TIA Portal	FBD, LD, ST	Siemens (SIMATIC)	Платная	Modbus, Profinet, OPC UA	Пошаговая отладка, точки остановки
Trace Mode	FBD, ST	Широкая (множество производителей)	Платная	Modbus, OPC, BACnet	Пошаговая отладка

2 Архитектура системы групповых запросов Modbus

2.1 Функциональные требования к модулю

Общие требования

Разрабатываемый модуль предназначен для формирования и управления групповыми Modbus-запросами (чтение/запись) для устройств платформы КС1 в среде OWEN Logic. Модуль должен:

- Объединять параметры в групповые запросы для повышения производительности взаимодействия с устройствами.
- Учитывать ограничения протоколов Modbus RTU и Modbus ASCII (максимальный размер запроса, ограничения по функциям).
- Поддерживать режимы: групповой (объединение переменных) и одиночный (без объединения).
- Предоставлять API для интеграции в существующую архитектуру OWEN Logic и UI-компоненты для настройки и отображения сформированных запросов.
- Обеспечивать валидацию переменных и визуальную выдачу ошибок/предупреждений в UI.
- Поддерживать гибкую политику разбиения больших групп на несколько физических сообщений (MB_Request) в соответствии с ограничениями протокола и пользовательскими настройками (макс. количество регистров).

2.1.1 Переменные в среде Owen Logic

Переменные в среде **Owen Logic** служат для хранения, передачи и обработки данных внутри проекта, а также для взаимодействия с внешними устройствами. Они являются основным элементом при построении схем управления, формировании Modbus-запросов и программировании интерфейсов визуализации.

Создание и управление переменными выполняется через **Таблицу переменных**, которая включает следующие категории:

- **Стандартные переменные** – используются в алгоритмах управления и визуализации.
- **Сервисные переменные** – обеспечивают внутренние функции прибора и не изменяются пользователем.
- **Сетевые переменные** – применяются для обмена данными между устройствами через интерфейсы связи (например, RS-485, Ethernet, Modbus).

Каждая переменная имеет следующие основные параметры:

- **Имя переменной** – уникальный идентификатор для отображения на схеме проекта.
- **Тип переменной** – определяет формат данных (булевский, целочисленный, с плавающей запятой).
- **Энергонезависимость** – при включении параметра значение сохраняется в энергонезависимой памяти (ПЗУ) устройства.
- **Значение по умолчанию** – используется при инициализации или при отсутствии связи с устройством.
- **Использование в проекте** – отображает, привязана ли переменная к блокам схемы.
- **Комментарий** – описание назначения переменной для удобства разработки.

На рисунке **2.1** представлена **таблица переменных** среды **Owen Logic**, отображающая структуру и категории переменных проекта.

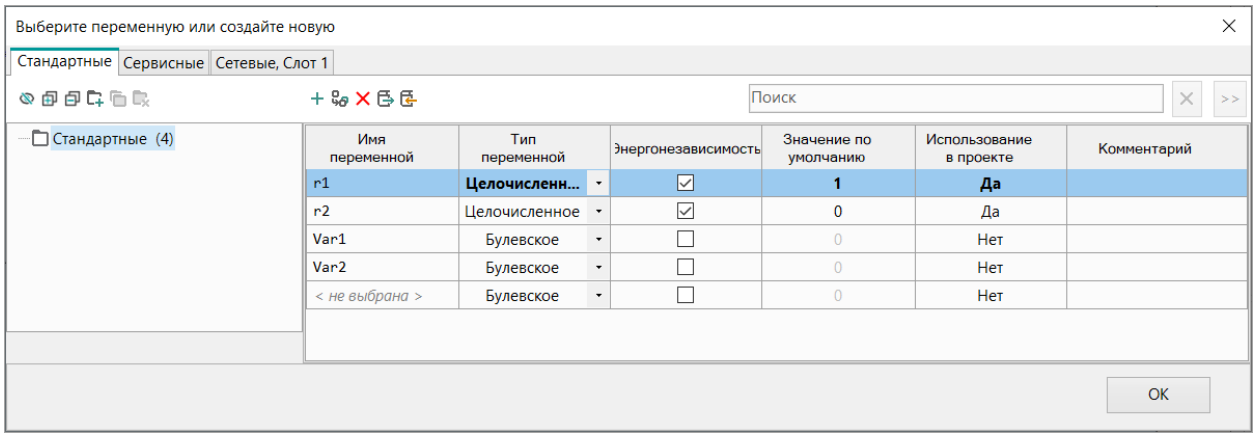

Рисунок 2.1 – Таблица переменных в среде Owen Logic

Типы переменных:

- **Булевский (Bool)** – принимает значения 0 (False) или 1 (True). Используется для логических условий и дискретных сигналов.

На рисунке **2.2** приведен пример отображения булевских переменных на схеме проекта.

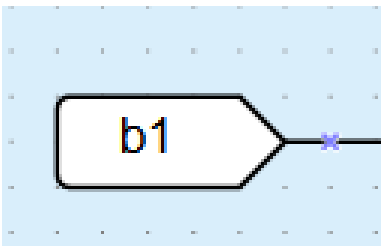

Рисунок 2.2 – Булевские переменные в Owen Logic

- **Целочисленный (Int/UInt)** – хранит значения целых чисел. Применяется для счетчиков, адресов, параметров, не требующих дробной части.

На рисунке **2.3** показан пример использования целочисленных переменных.

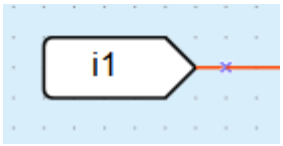

Рисунок 2.3 – Целочисленные переменные в Owen Logic

- **С плавающей запятой (Float/Real)** – хранит вещественные значения, используется для аналоговых сигналов, коэффициентов и вычислений.

На рисунке **2.4** представлен пример отображения переменных с плавающей запятой.

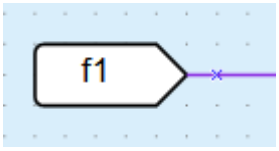

Рисунок 2.4 – Переменные с плавающей запятой в Owen Logic

На схеме Owen Logic линии, соединяющие переменные разных типов, имеют различный цвет:

– черный для булевских,

– красный для целочисленных,

– фиолетовый для вещественных.

Переменные могут быть связаны с регистрами внешних Modbus-устройств. В этом случае в таблице указывается **тип регистра (Holding, Input, Coil, Discrete)** и его **адрес**, что позволяет модулю обмена данными формировать соответствующие запросы чтения и записи.

Переменные интерфейса связи

В таблице переменных для каждого интерфейса связи создается отдельная вкладка, содержащая сетевые переменные. В заголовке вкладки указывается тип интерфейса (например, **RS-485**) и номер слота, к которому он подключён.

Режим Master.

В режиме Master таблица переменных содержит вкладки для каждого опрашиваемого устройства.

Для каждой переменной задаются параметры Modbus-связи — тип регистра (**Coil**, **Discrete Input**, **Holding**, **Input Register**), адрес, функция чтения или записи, а также направление

обмена.

Эти параметры используются модулем группировки при формировании запросов к Slave-устройствам.

На рисунке 2.5 показан пример таблицы переменных интерфейса связи в режиме Master.

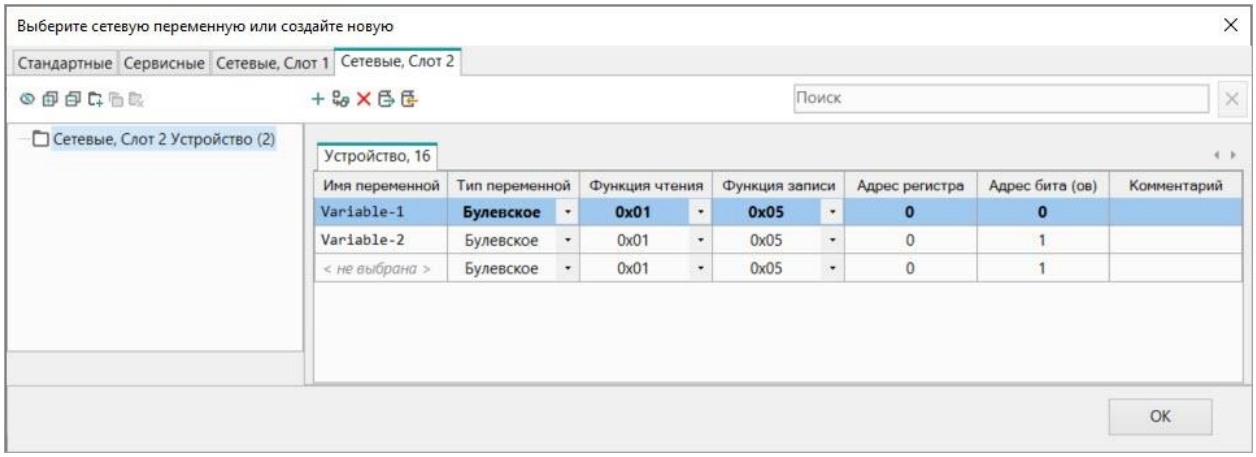

Рисунок 2.5 – Вкладка сетевых переменных в режиме Master

Режим Slave.

В режиме Slave таблица отображает переменные, значения которых предоставляются внешним Master-устройствам.

Такие переменные могут быть связаны с внутренними параметрами проекта и автоматически обновляются при изменении значений на схеме.

Это позволяет другим устройствам считывать актуальные данные в реальном времени.

На рисунке 2.6 приведен пример вкладки сетевых переменных в режиме Slave.

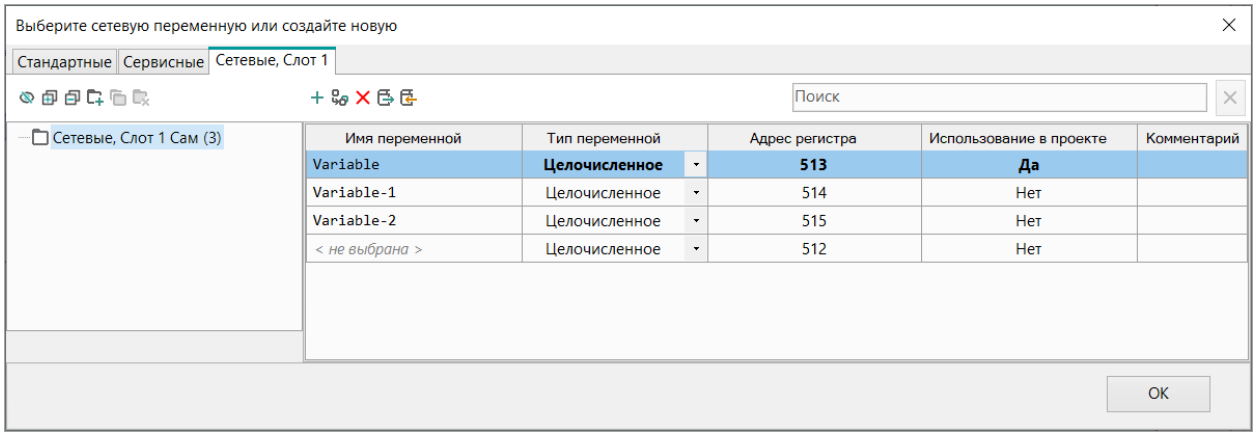

Рисунок 2.6 – Вкладка сетевых переменных в режиме Slave

2.1.1 Ограничение протокола Modbus

При формировании групповых запросов Modbus необходимо учитывать ряд ограничений, определяемых спецификацией протокола и особенностями аппаратной платформы КС1. Основным фактором, ограничивающим количество регистров в одном запросе, является размер кадра (буфера) передачи данных, который для устройств серии КС1 составляет 256 байт. Кроме того, пределы на количество регистров зависят от типа протокола (RTU или ASCII) и номера функции чтения/записи.

Максимальное количество регистров в запросе для различных комбинаций параметров приведено в таблице 2.2.

Таблица 2.2 – Ограничения на максимальное количество регистров в запросах Modbus

№ п/п	**Тип протокола**	**Номер функции**	**макс. кол-во регистров**	**Примечание**
1	RTU	0x03, 0x04	125	Ограничение на ввод параметра кол-во регистров в UI - 125.
2	RTU	0x10	123	Ограничение на ввод параметра кол-во регистров в UI - 125. Таким образом, если пользователь ввел кол-во регистров 125, то необходимо будет сформировать 2 посылки (структуры MB_Request): на 123 регистра и на 2 регистра.
3	RTU	0x01, 0x02	125	Максимальное количество бит/койлов в групповом запросе согласно спецификации Modbus 2000. Но у нас принято общее ограничение на ввод параметра кол-во регистров в UI - 125.
4	RTU	0x0F	125	Максимальное количество бит/койлов в групповом запросе согласно спецификации Modbus 1968. Но у нас принято общее ограничение на ввод параметра кол-во регистров в UI - 125.
5	ASCII	0x03, 0x04	61	Ограничение на ввод параметра кол-во регистров в UI - 125. Таким образом, если пользователь ввел кол-во регистров 125, то необходимо будет сформировать 3 посылки (структуры MB_Request): две посылки на 61 регистр и одну на 3 регистра.
6	ASCII	0x10	59	Ограничение на ввод параметра кол-во регистров в UI - 125. Таким образом, если пользователь ввел кол-во регистров 125, то необходимо будет сформировать 3 посылки (структуры MB_Request): две по 59 регистров и одну на 7 регистров.

7	ASCII	0x01, 0x02	125	Максимальное количество бит/койлов в групповом запросе согласно спецификации Modbus 984. Но у нас принято общее ограничение на ввод параметра кол-во регистров в UI - 125.
8	ASCII	0x0F	125	Максимальное количество бит/койлов в групповом запросе согласно спецификации Modbus 944. Но у нас принято общее ограничение на ввод параметра кол-во регистров в UI - 125.

2.1.2 Требования к группировке переменных

Перед формированием групповых запросов необходимо учитывать особенности протокола **Modbus**, в частности — структуру его областей данных и допустимые функции обмена.

Функции и области данных

При запросе Master обращается к одной из областей памяти Slave с помощью функции. Область памяти характеризуется типом хранимых значений (биты или регистры) и типом доступа (чтение или чтение/запись).

Области данных протокола Modbus

Область данных	Обозначение	Тип данных	Тип доступа
Coils (Регистры флагов)	0x	Булевый	Чтение/запись
Discrete Inputs (Дискретные входы)	1x	Булевый	Только чтение
Input Registers (Регистры ввода)	3x	Целочисленный	Только чтение
Holding Registers (Регистры хранения)	4x	Целочисленный	Чтение/запись

Каждая область памяти состоит из определенного (зависящего от конкретного устройства) количества ячеек. Каждая ячейка имеет уникальный адрес. Для конфигурируемых устройств производитель предоставляет карту регистров, в которой содержится информация о соответствии параметров устройства и их адресов. Для программируемых устройств пользователь формирует такую карту самостоятельно с помощью среды программирования. Существуют устройства, в которых сочетаются оба рассмотренных случая – у их карты регистров есть фиксированная часть и часть, которую пользователь может дополнить в соответствии со своей задачей.

В некоторых устройствах области памяти наложены друг на друга (например, 0x и 4x) – т. е. пользователь сможет обращаться разными функциями к одним и тем же регистрам.

Функция определяет операцию (чтение/запись) и область памяти, с которой эта операция будет произведена.

Основные функции протокола Modbus

Код функции	Имя функции	Выполняемая команда
1 (0x01)	Read Coil Status	Чтение значений из нескольких регистров флагов
2 (0x02)	Read Discrete Inputs	Чтение значений из нескольких дискретных входов
3 (0x03)	Read Holding Registers	Чтение значений из нескольких регистров хранения
4 (0x04)	Read Input Registers	Чтение значений из нескольких регистров ввода
5 (0x05)	Force Single Coil	Запись значения в один регистр флага
6 (0x06)	Preset Single Register	Запись значения в один регистр хранения
15 (0x0F)	Force Multiple Coils	Запись значений в несколько регистров флагов
16 (0x10)	Preset Multiple Registers	Запись значений в несколько регистров хранения

Опрос Slave-устройства

Опрос устройства может выполняться:

- **одиночным способом** — каждая переменная считывается отдельной командой;

- **групповым способом** — несколько переменных считываются одной командой при условии, что их адреса **расположены последовательно без разрывов**.

Групповой опрос позволяет **снизить трафик в сети и ускорить обмен**, но его применение ограничено типом данных и функцией Modbus.

Примечания и особенности формирования групповых запросов:

1. **Допустимые функции Modbus для групповых запросов.**
   Групповые запросы могут формироваться только для функций чтения 0x01, 0x02, 0x03, 0x04 и функций записи 0x0F, 0x10.
   Для функций записи одиночных битов или регистров (0x05, 0x06) группировка невозможна, так как данные функции работают только с одним адресом за операцию.
2. **Особенности опроса при режиме «Запись по изменению».**
   Если в группе переменных задано условие «запись по изменению», то при изменении значения хотя бы одной переменной из этой группы модуль формирует и отправляет единый запрос для всей группы.
   Такой подход обеспечивает консистентность данных в Slave-устройстве, но может приводить к увеличению частоты передачи сообщений при высокой изменчивости входных данных.
3. **Совместимость типов Real и Int32.**
   Переменные типов Real и Int32 могут объединяться в один групповой запрос, так как оба типа занимают по два Modbus-регистра (32 бита).
   Это позволяет повысить эффективность передачи данных, сохраняя корректное выравнивание адресов. Однако смешивание типов с разным количеством регистров (например, Int16 и Real) не допускается, поскольку нарушает непрерывность адресного пространства.

Для формирования группового запроса должны быть соблюдены все условия, перечисленные в таблице 2.1. Окно настроек опрашиваемого устройства представлено на рисунке.2.7.

Таблица 2.1 – Условия и примеры группировки переменных в Modbus-запросы

№	**Условие формирования группового запроса**	**Пример валидного условия**	**Пример невалидного условия**
1	Группировка только для переменных одного устройства	устройство адрес 16, var1, регистр 0; var2, регистр 2	разные адреса устройств (16 и 17)
2	Тип данных переменных должен быть одинаковым	var1:real (0), var2:real (2), var3:real (4)	var1:real, var2:int16
3	Адреса регистров/битов должны идти подряд без разрывов	var1:рег.0, var2:рег.2, var3:рег.4	var1:рег.0, var2:рег.3
4	Функции чтения/записи должны совпадать	все var1,var2: чтение 0x03	var1: чтение 0x03, var2: чтение 0x04
5	Условия опроса (период, команды) должны совпадать	var1,var2: период 100мс	var1: 100мс, var2: 200мс
6	Для опроса по команде «Запуск чтение» – одна командная переменная	var1,var2: чтение по команде var_b1	var1: var_b1, var2: var_b2
7	Для опроса по команде «Запуск записи» – одна командная переменная	var1,var2: запись по команде var_b1	var1: var_b1, var2: var_b2
8	Переменные статуса должны быть одинаковы или не заданы	var1,var2: статус var_i1	var1: статус var_i1, var2: не выбран

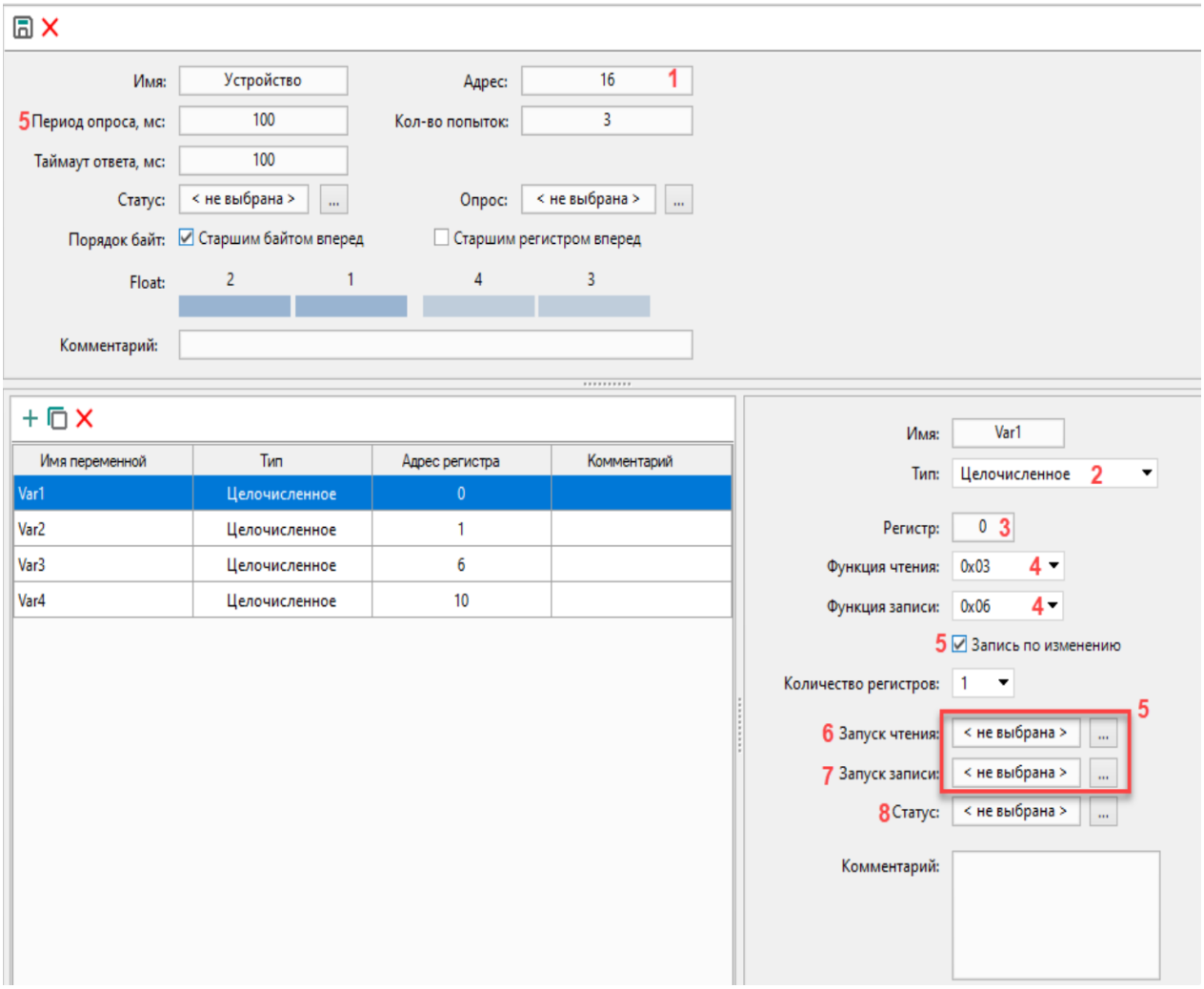

Рисунок 2.7 – Схема данных БД

Подумать о добавлении картинок с примерами валид и не валид

2.1.3 Требования к пользовательскому интерфейсу

Для реализации функционала групповых Modbus-запросов в среде OWEN Logic необходимо расширить окно настроек опрашиваемого устройства платформы КС1 за счёт добавления новых параметров (рис. 2.8).

В окно должны быть добавлены следующие элементы:

1. **Параметр «Протокол»**
   Тип элемента: выпадающий список.
   Возможные значения: **Modbus RTU / Modbus ASCII**.
   При выборе протокола выполняется динамическое обновление допустимого диапазона параметра *«Количество регистров в запросе»* в соответствии с ограничениями выбранного протокола.
   Значение по умолчанию: **Modbus RTU** (рис. 2.9).
2. **Параметр «Группировать запросы»**
   Тип элемента: выпадающий список (Да / Нет).
   Значение по умолчанию: **«Нет»**.
   При выборе значения **«Да»** активируется возможность задания дополнительных параметров группировки (рис. 2.10).
   Если значение параметра установлено в **«Нет»**, параметр *«Количество регистров в запросе»* становится **недоступным для редактирования** (рис. 2.11).
3. **Параметр «Количество регистров в запросе»**
   Диапазон допустимых значений:
   – для протокола Modbus RTU – от 1 до 125;
   – для протокола Modbus ASCII – от 1 до 61.
   Значение по умолчанию: 16.
   При вводе некорректного значения отображается пиктограмма с восклицательным знаком и тултип с сообщением о допустимом диапазоне значений:
   – сообщение о валидном диапазоне количества регистров в групповом запросе для протокола RTU (рис. 2.12).

Параметр «Количество регистров в запросе» напрямую связан с ограничениями протокола Modbus, приведёнными в таблице 2.2 (см. раздел 2.1.2). Изменение значения данного параметра влияет на разбиение групп переменных на отдельные физические запросы (MB_Request).

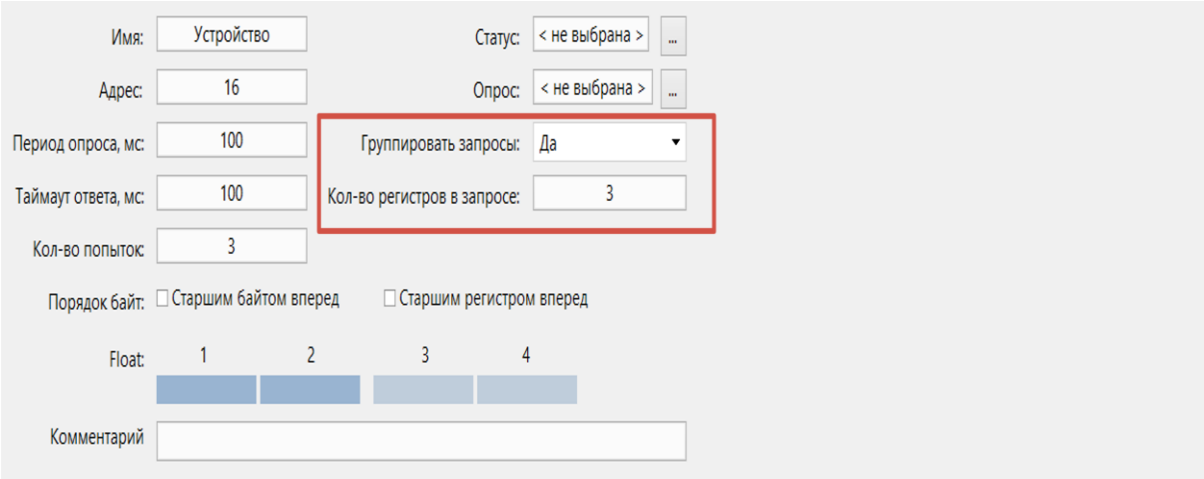

Рисунок 2.8 – Окно настроек опрашиваемого устройства с добавленными параметрами

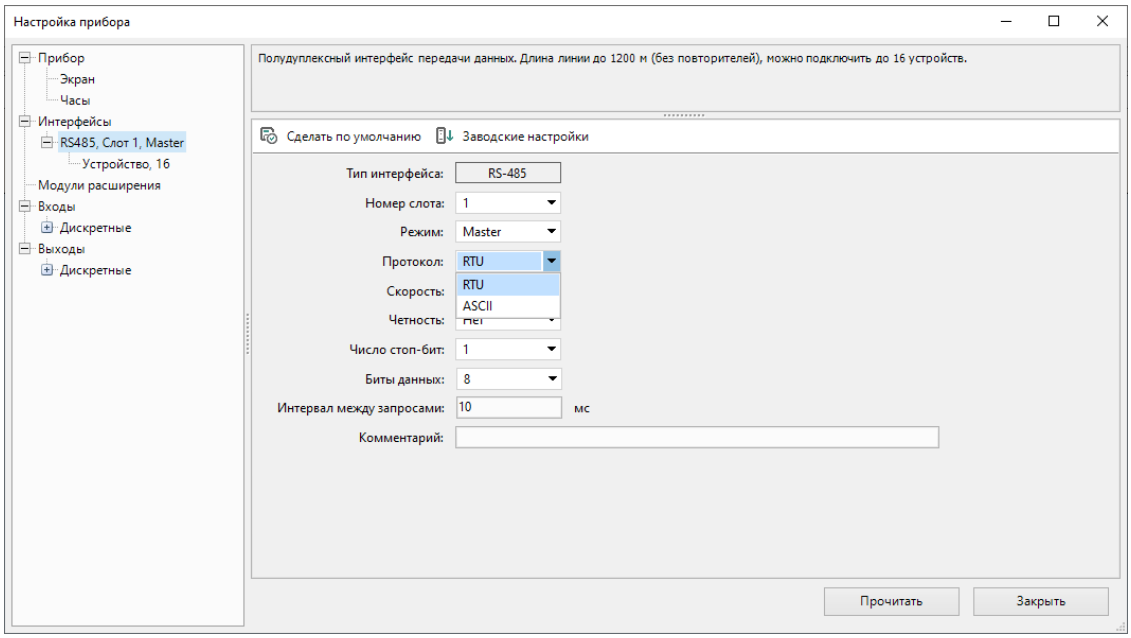

Рисунок 2.9 – Выбор протокола Modbus RTU/ASCII

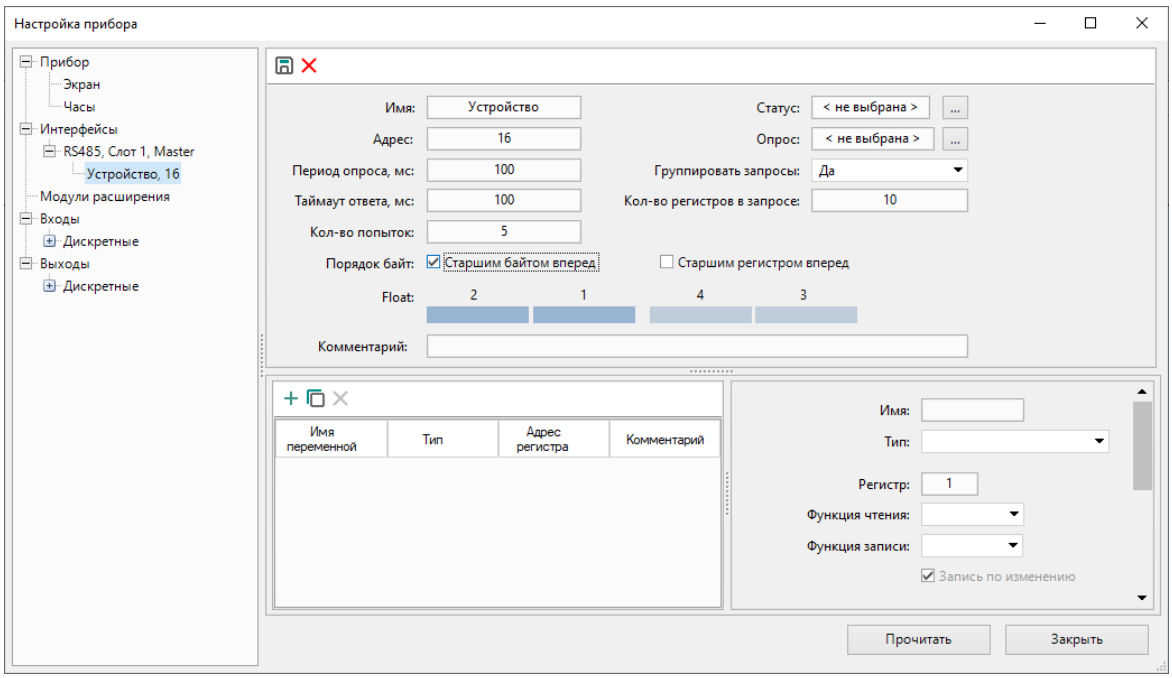

Рисунок 2.10 – Параметр «Группировать запросы»

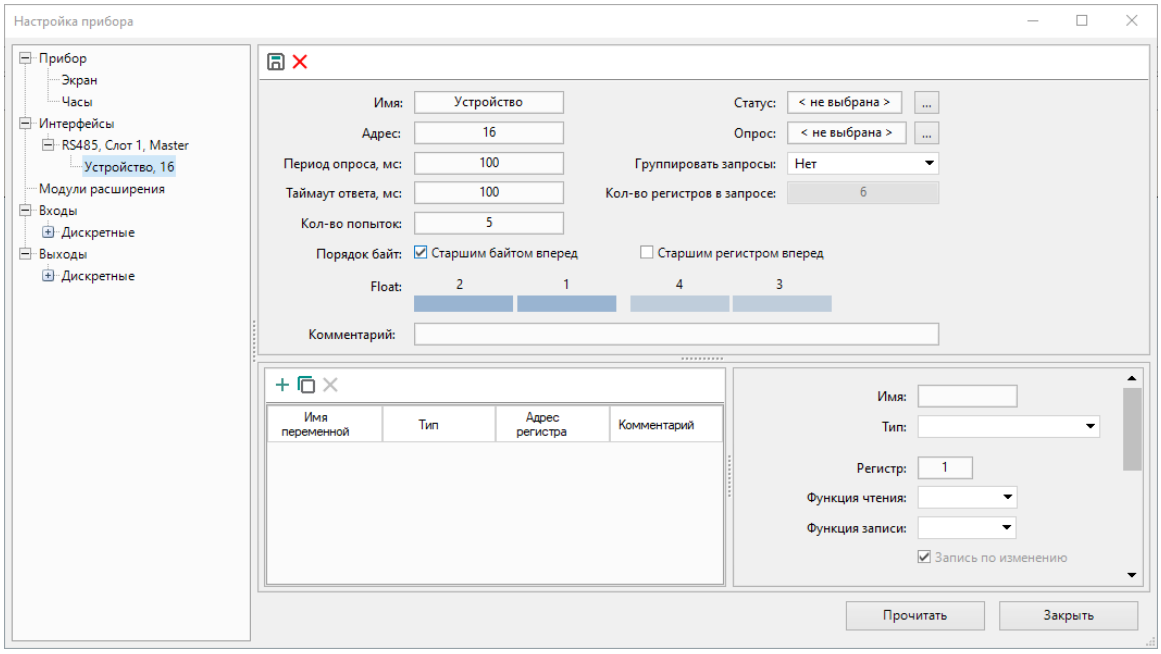

Рисунок 2.11 – Неактивное состояние параметра «Количество регистров в запросе» при значении «Группировать запросы = Нет»

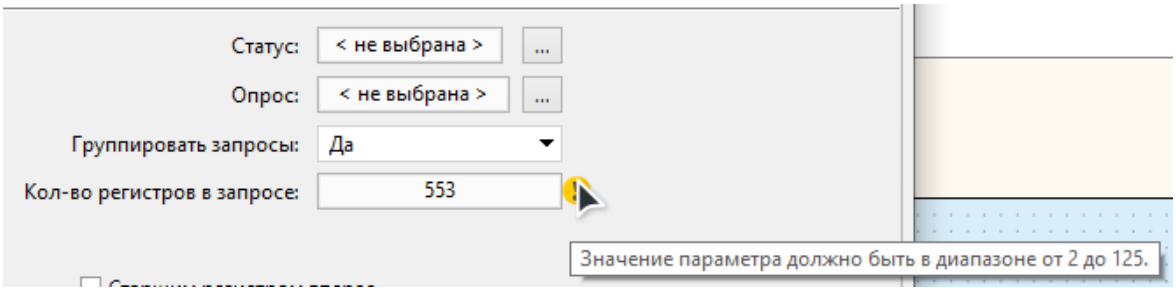

Рисунок 2.12 – Сообщение о недопустимом диапазоне значений для протокола Modbus

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] URL: https://owen.ru/documents

[2] URL: https://learn.microsoft.com/ru-ru/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/

[3] URL: https://learn.microsoft.com/ru-ru/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice

[4] URL: https://www.codesys.com/

[5] URL: https://www.siemens-pro.ru/soft/tia-portal.html

[6] URL: https://ru.wikipedia.org/wiki/Trace_mode