

НАСТРОЙКА ОБМЕНА ДААННЫМИ ПО ПРОТОКОЛУ MODBUS НА КОНТРОЛЛЕРАХ СЕРИИ REGUL RX00

Руководство пользователя

DPA-302.1

Версия документа 1.3

Версия ПО 1.6.5.0

Июнь 2021

История изменений руководства пользователя

Версия руководства пользователя	Описание изменения
1.2	<p>Добавлена история изменений руководства пользователя.</p> <p>Modbus Serial Master: расширен диапазон пользовательских функций ModbusUserRequest2.</p> <p>Дополнительно по тексту внесены небольшие изменения с уточняющей информацией</p>
1.3	<p><i>Раздел «Настройка Modbus TCP»:</i> описана возможность задавать номер порта TCP в диапазоне от 256 до 65535.</p> <p><i>Раздел «Настройка Serial Master»:</i> описана возможность экспортировать/импортировать каналы modbus в/из файла с расширением *.xml.</p> <p><i>Раздел «Настройка Serial Slave»:</i> описана возможность в процессе работы изменять адрес устройства при помощи свойства SlaveID.</p> <p>Дополнительно по тексту внесены небольшие изменения с уточняющей информацией</p>

АННОТАЦИЯ

Настоящий документ содержит сведения о настройке обмена данными по протоколу Modbus на промышленных логических контроллерах серии Regul RX00. Настройка осуществляется с помощью программного обеспечения Epsilon LD.

Данное руководство предназначено для эксплуатационного персонала и инженеров-проектировщиков АСУ ТП, которые должны:

- иметь, как минимум, среднее техническое образование;
- приступить к работе только после изучения данного руководства.


Обновление информации в Руководстве

Производитель ООО «Прософт-Системы» оставляет за собой право изменять информацию в настоящем Руководстве и обязуется публиковать более новые версии с внесенными изменениями. Обновленная версия Руководства доступна для скачивания на официальном сайте Производителя: <https://www.prosoftsystems.ru/>.


Для своевременного отслеживания выхода новой версии Руководства рекомендуется оформить подписку на обновление документа. Для этого необходимо на сайте Производителя: <https://www.prosoftsystems.ru/> во вкладке «Документация» под иконками документов кликнуть на кнопку «Подписаться на обновления» и оставить свои контактные данные.

В руководстве присутствуют знаки с предупреждающей и поясняющей информацией. Каждый знак обозначает следующее:

ПРЕДУПРЕЖДАЮЩИЕ ЗНАКИ

	<p>ВНИМАНИЕ! Здесь следует обратить внимание на способы и приемы, которые необходимо в точности выполнять во избежание ошибок при эксплуатации или настройке.</p>
---	--

ИНФОРМАЦИОННЫЕ ЗНАКИ

	<p>ИНФОРМАЦИЯ Здесь следует обратить внимание на <u>важную</u> информацию</p>
---	--

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	4
ВВЕДЕНИЕ.....	5
Общие сведения.....	5
Перечень рекомендуемых документов	5
Начало работы	5
Принцип добавления устройств в конфигурацию контроллера	6
НАСТРОЙКА MODBUS ASCII И MODBUS RTU	10
Добавление последовательного порта в конфигурацию контроллера	10
Добавление порта	10
Настройка параметров порта.....	11
Настройка Modbus Serial Master	11
Настройка Modbus Serial Slave	18
НАСТРОЙКА MODBUS TCP	24
Настройка Modbus TCP Master	24
Настройка Modbus TCP Slave	26
ПРИЛОЖЕНИЕ А.....	28

ВВЕДЕНИЕ

Общие сведения

Программное обеспечение контроллера Regul позволяет сконфигурировать его как в качестве Modbus Master, так и в качестве Modbus Slave. Эти реализации независимы друг от друга, то есть по одной из линий связи контроллер может являться «мастером», а по другой – «слэйвом».

В качестве Modbus Master контроллер опрашивает slave-устройства по последовательной линии RS-232|485 (режимы Modbus RTU|ASCII), либо по Ethernet (режим Modbus TCP). В качестве Modbus Slave контроллер также работает в трех режимах (RTU|ASCII|TCP).

Поддерживаются следующие функции протокола Modbus:

- чтение и запись состояний реле - Read Coils (0x01), Write Single Coil (0x05), Write Multiple Coils (0x0F);
- чтение дискретных входов - Read Discrete Inputs (0x02);
- чтение и запись регистров хранения - Read Holding Registers (0x03), Write Single Register (0x06), Write Multiple Registers (0x10);
- чтение входных регистров - Read Input Register (0x04).

Перечень рекомендуемых документов

Для получения дополнительной информации по настройке других параметров контроллеров серии Regul RX00 в среде разработки Epsilon LD рекомендуется ознакомиться со следующими документами:

- Программное обеспечение Epsilon LD. Руководство пользователя;
- Regul R600. Системное руководство;
- Regul R500. Системное руководство;
- Regul R400. Системное руководство;
- Regul R200. Системное руководство.

Начало работы

Установите на компьютер программное обеспечение **Epsilon LD**. Описание процесса установки программы, а также инструкции по работе с программой приведены в документе «Программное обеспечение Epsilon LD. Руководство пользователя». Программа установки и документация доступны на сайте www.prosoftsystems.ru.

Запустите программу **Epsilon LD**. Создайте или откройте проект. Убедитесь, что в проекте есть контроллер, который будет участвовать в обмене данными по протоколу Modbus. Если

контроллер отсутствует, добавьте его с помощью Мастера конфигурации Regul (см. документ «Программное обеспечение Epsilon LD. Руководство пользователя»).

Принцип добавления устройств в конфигурацию контроллера

Алгоритм добавления устройств для настройки интерфейсов представлен на рисунке 1

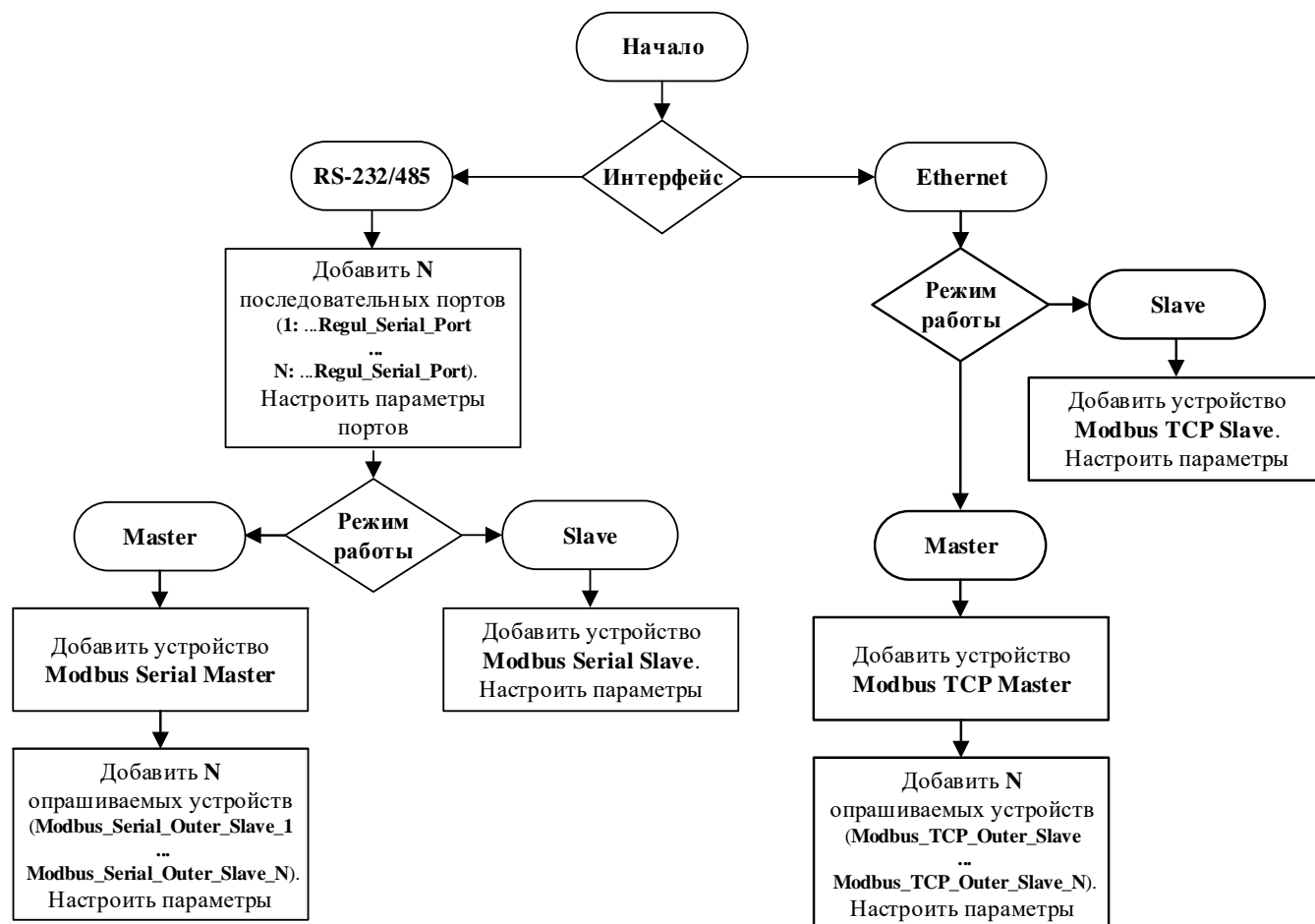


Рисунок 1 – Алгоритм настройки

Для добавления устройства поместите курсор на его название (в окне дерева устройств), нажмите правую кнопку мыши. В появившемся контекстном меню выберите пункт **Добавить устройство...** (Рисунок 2).

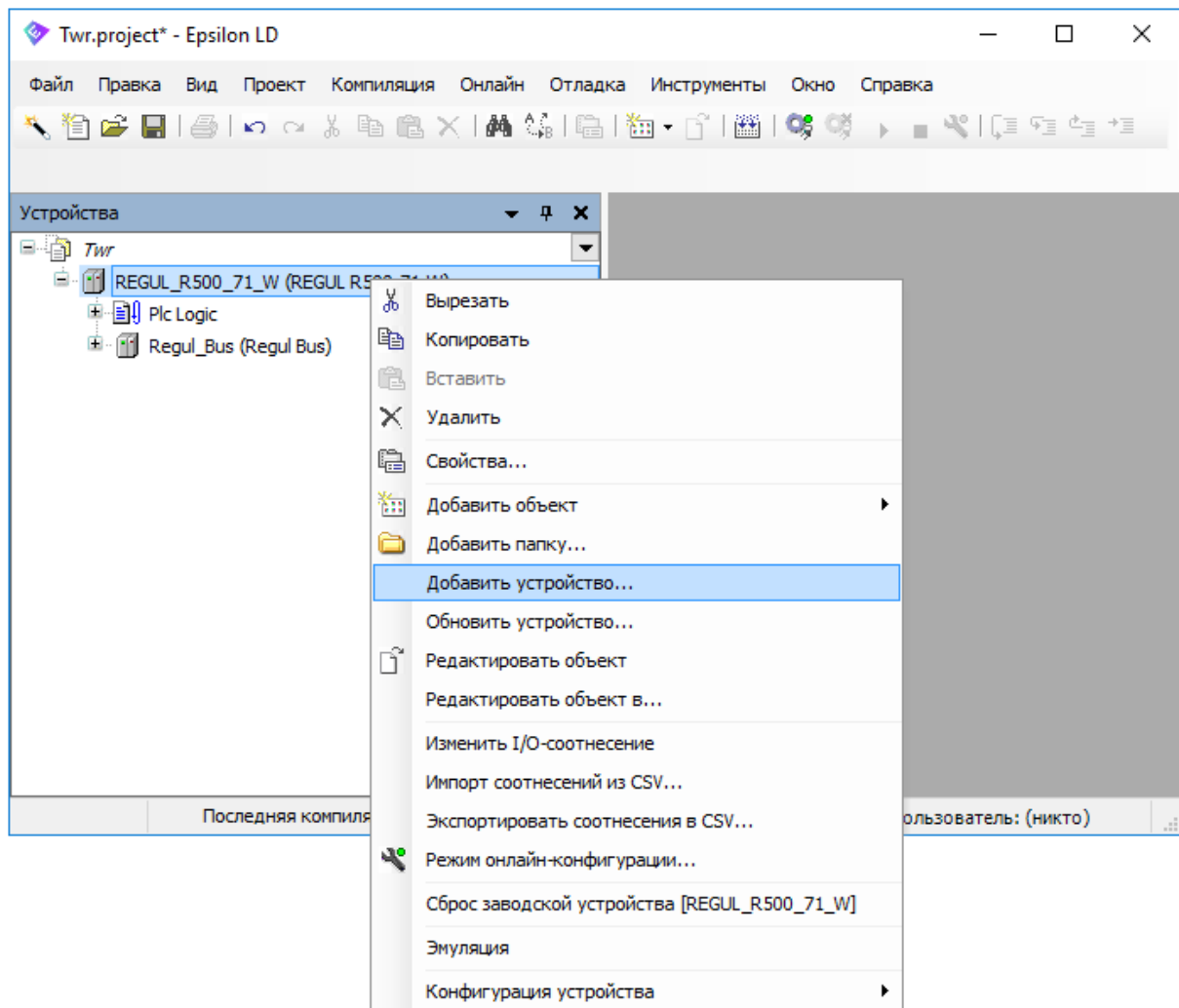


Рисунок 2 – Контекстное меню

Откроется окно **Добавить устройство**, где по умолчанию отображается тот список устройств, который в данный момент доступен пользователю для вставки, например, при добавлении крейта – список крейтов, при размещении модулей в крейте – список модулей, для настройки Modbus – последовательный порт, slave- и master-устройства (Рисунок 3).

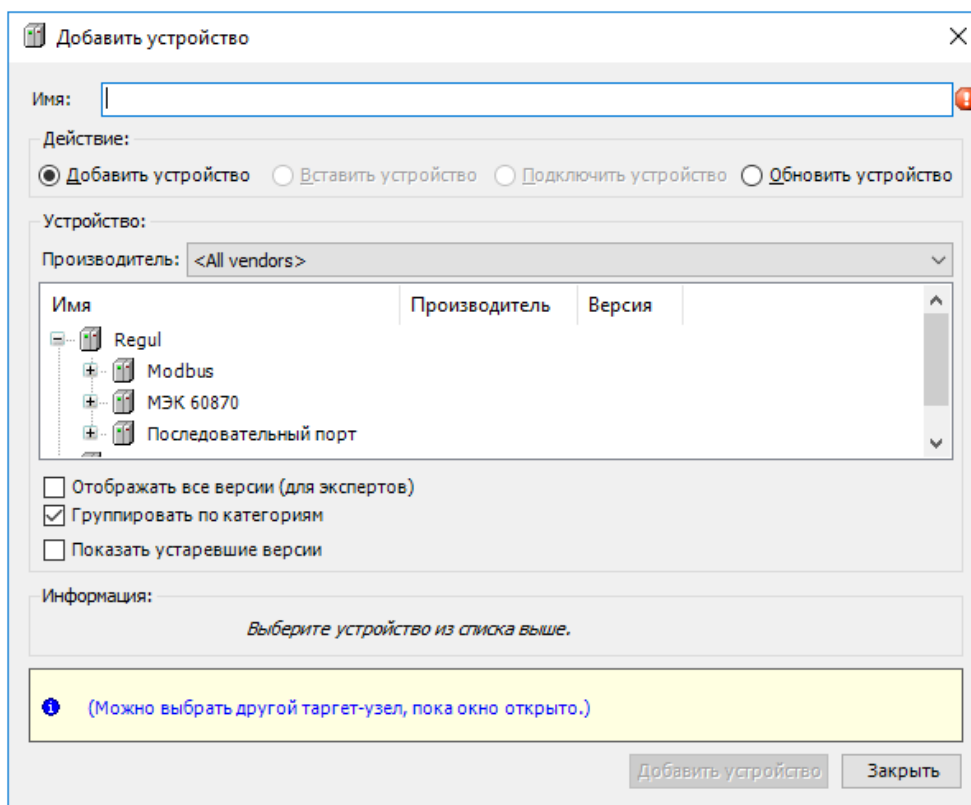


Рисунок 3 – Окно «Добавить устройство»

Выберите нужное устройство, нажмите кнопку **Добавить устройство** или дважды щелкните левой кнопкой мыши. Выбранное устройство появится в проекте в дереве устройств.

В дереве устройств к контроллеру требуется подключать (в зависимости от ситуации) не только устройства, соответствующие реально существующим модулям, но и виртуальные устройства, у которых нет аппаратного эквивалента. Так, например, Modbus-устройства являются виртуальными и фактически предназначены для настройки параметров МЭК-библиотеки, автоматически загружаемой в проект при подключении конкретного устройства.

При настройке различных режимов Modbus-устройства подключаются к соответствующему последовательному порту или непосредственно к контроллеру. На рисунке 4 показаны примеры конфигураций при настройке различных режимов Modbus:

- вариант 1 – Modbus Serial Master, где каналом связи является встроенный последовательный порт контроллера;
- вариант 2 – Modbus Serial Slave, канал связи - также встроенный последовательный порт;
- вариант 3 – Modbus Serial Master на последовательном порту коммуникационного модуля;
- вариант 4 – Modbus Serial Slave на последовательном порту коммуникационного модуля;
- вариант 5 – Modbus TCP Master;
- вариант 6 – Modbus TCP Slave.

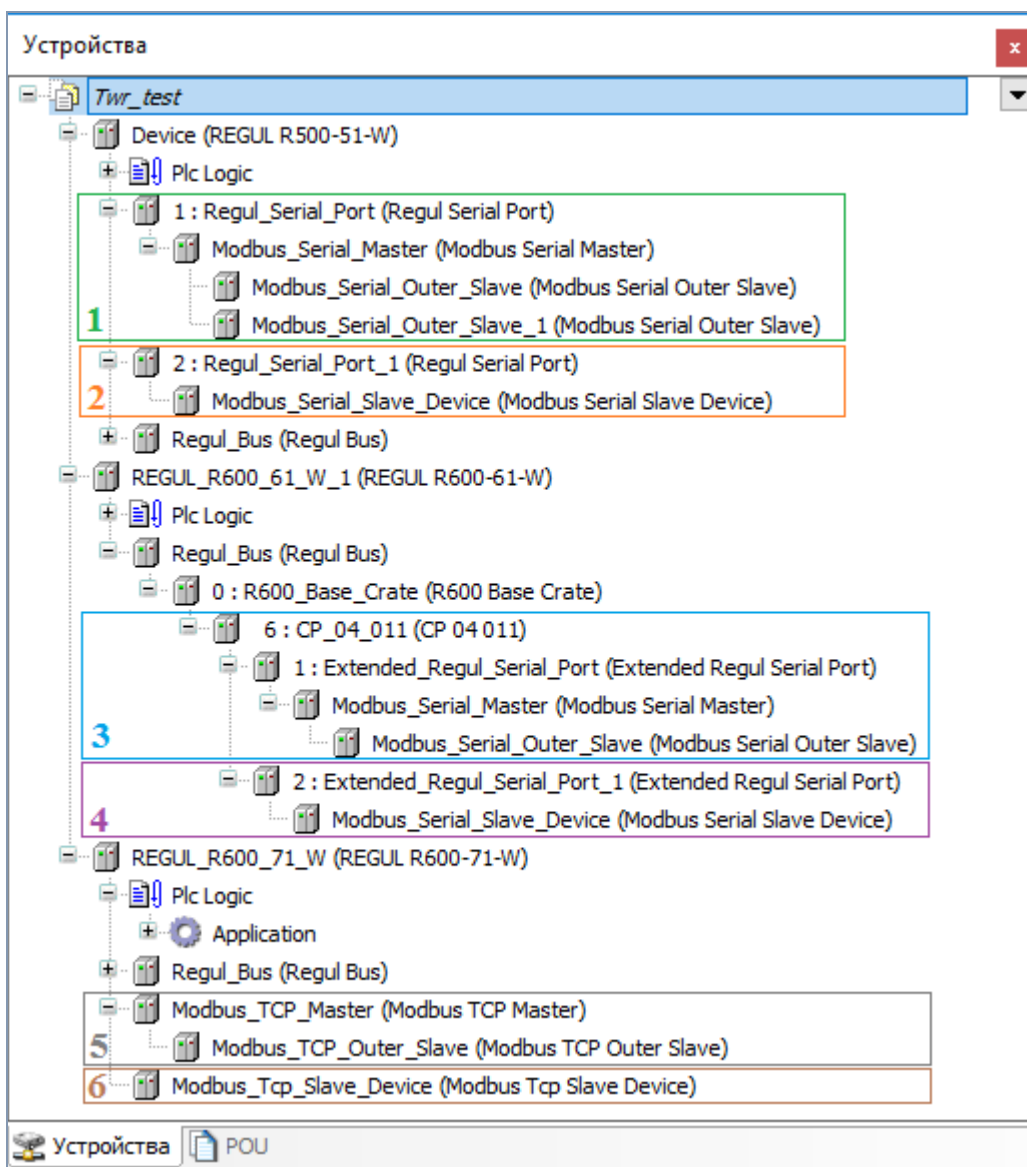





Рисунок 4 – Конфигурация устройств для различных режимов Modbus



ИНФОРМАЦИЯ

Начиная с версии СПО 1.6.5.0, при выборе устройства Modbus доступны варианты драйвера с дополнением OS:

-  TCP Modbus Slave
 -  Modbus Tcp Slave "Prosoft-Systems" Ltd. 1.6.5.0
 -  Modbus Tcp Slave OS "Prosoft-Systems" Ltd. 1.6.5.0 Устройство Modbus Tcp Slave. Драйвер на уровне ОС.

Данный драйвер реализован на уровне операционной системы, что позволяет распределить нагрузку от его работы при использовании многоядерных процессоров

НАСТРОЙКА MODBUS ASCII И MODBUS RTU

Добавление последовательного порта в конфигурацию контроллера

Перед настройкой Modbus Serial в конфигурацию контроллера необходимо добавить последовательный порт.

Добавление порта

Если для обмена данными будет использоваться какой-либо из портов RS-485/232 модуля центрального процессора, то в конфигурацию нужно добавить к контроллеру устройство Regul Serial Port (*Regul* → *Последовательный порт* → *Regul Serial Port*). Количество возможных устройств данного типа ограничивается количеством физических портов RS-485 (RS-232) на модуле центрального процессора.

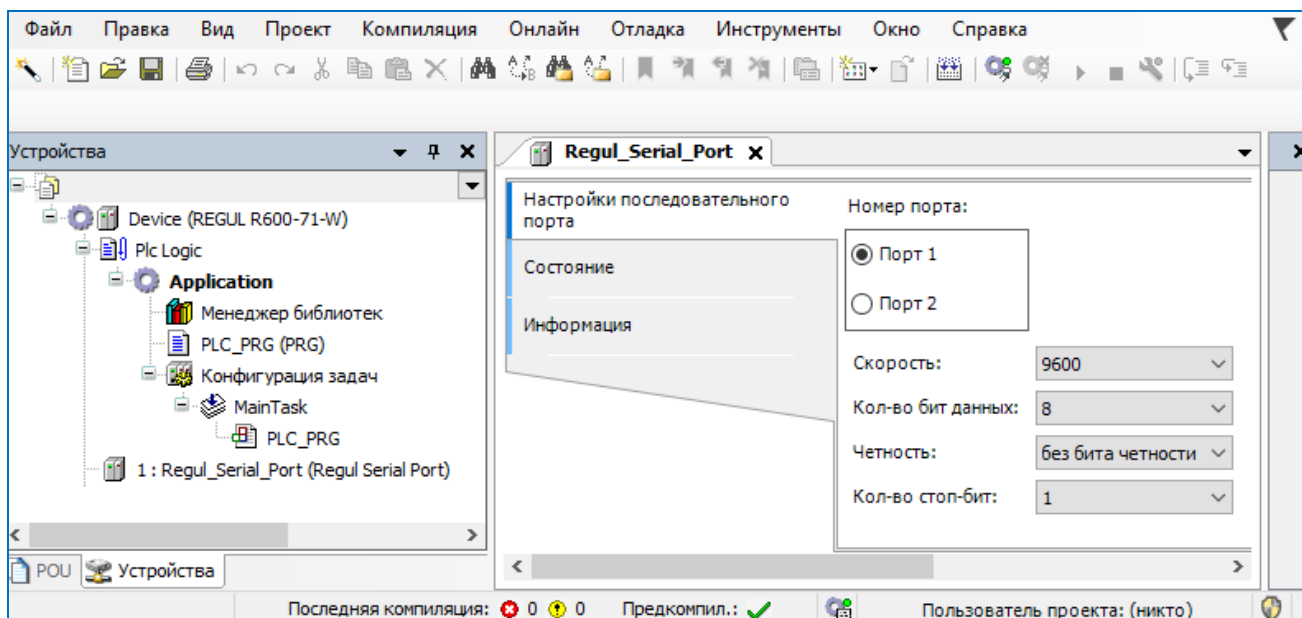


Рисунок 5 – Настройки последовательного порта Regul Serial Port

Если для обмена данными будет использоваться порт модуля коммуникационного процессора, то в конфигурацию нужно добавить к модулю коммуникационного процессора устройство Extended Regul Serial Port (*Regul* → *Последовательный порт* → *Extended Regul Serial Port*). Количество возможных дочерних устройств ограничивается моделью модуля.

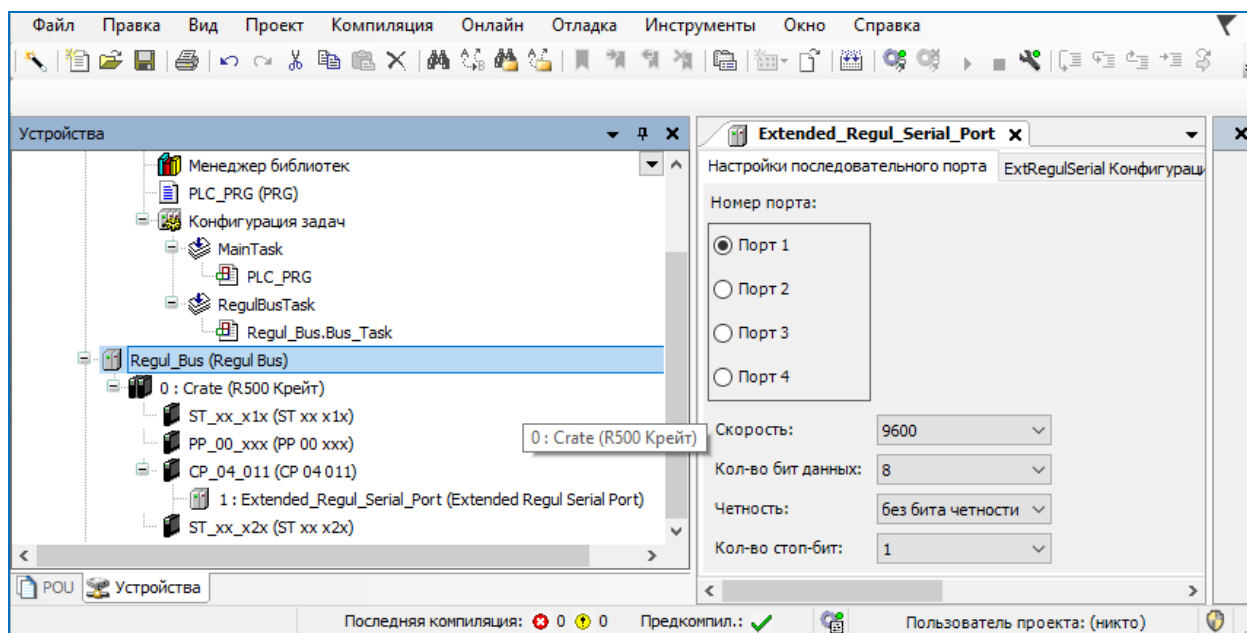


Рисунок 6 – Настройки последовательного порта Extended Regul Serial Port

Настройка параметров порта

Двойным щелчком по названию порта откройте главную вкладку параметров порта. Перейдите на внутреннюю вкладку **Настройки последовательного порта** (Рисунок 5, Рисунок 6). Установите переключатель в поле **Номер порта** (совпадает с номером, указанным на модуле). Далее, выбирая значение в раскрывающемся списке, установите следующие параметры:

- **Скорость** – значения: 1200, 4800, 9600, 19200, 38400, 57600, 115200;
- **Кол-во бит данных** – значения: 8, 7, 6, 5;
- **Четность** – значения: без бита четности, проверка на нечетность, проверка на четность;
- **Кол-во стоп-бит** – 1 или 2.

Настройка Modbus Serial Master

В случае, когда контроллер будет использоваться как ведущее, опрашивающее устройство, необходима настройка Modbus Serial Master. При настройке данного режима требуется задать параметры slave-устройства, которое будет опрашиваться контроллером. Кроме того, требуется описать набор данных, который будет запрашиваться по Modbus.

Добавьте устройство **Modbus Serial Master** к последовательному порту Regul Serial Port или Extended Regul Serial Port (*Regul* → *Modbus* → *Serial Modbus Master* → *Modbus Serial Master*). Двойным щелчком по названию устройства **Modbus Serial Master** откройте вкладку параметров (Рисунок 7). Установка флажка в поле **Отладочный режим (Debug mode)** включает отладочный режим (в журнал контроллера будет записываться дополнительная информация о работе).

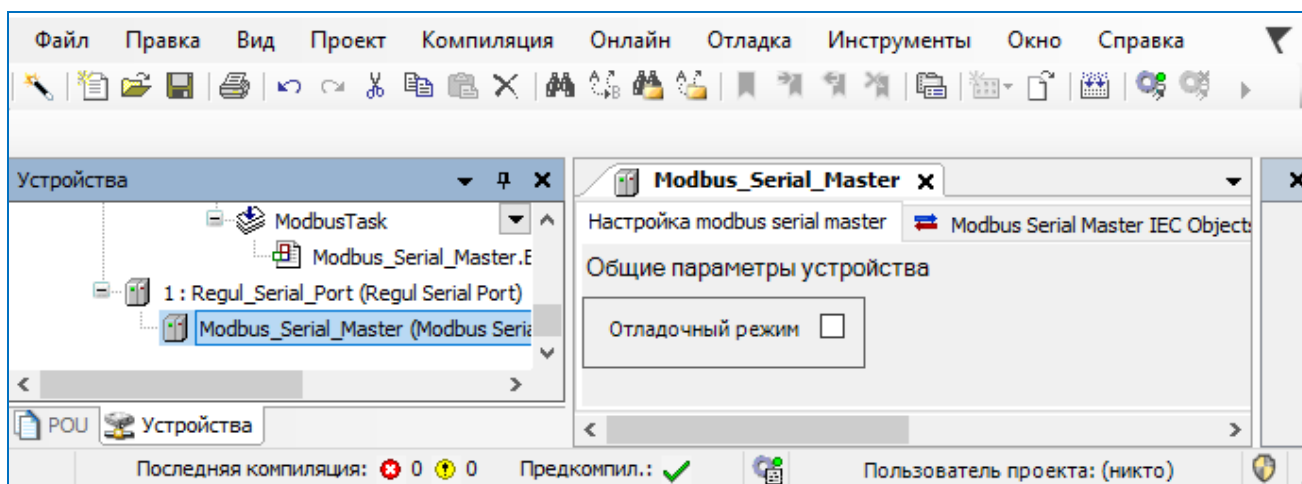


Рисунок 7 – Настройка параметров modbus serial master

Далее к устройству **Modbus Serial Master** нужно подключить одно или несколько внешних slave-устройств (outer slaves), которые будут опрашиваться контроллером (*Regul* → *Modbus* → *Serial Modbus Master* → *Modbus Serial Outer Slave*). Максимальное количество ограничено диапазоном доступных адресов Modbus. Двойным щелчком по названию устройства **Modbus Serial Outer Slave** откройте вкладку параметров. По умолчанию открывается первая внутренняя вкладка **Настройка modbus serial outer slave** (Рисунок 8).

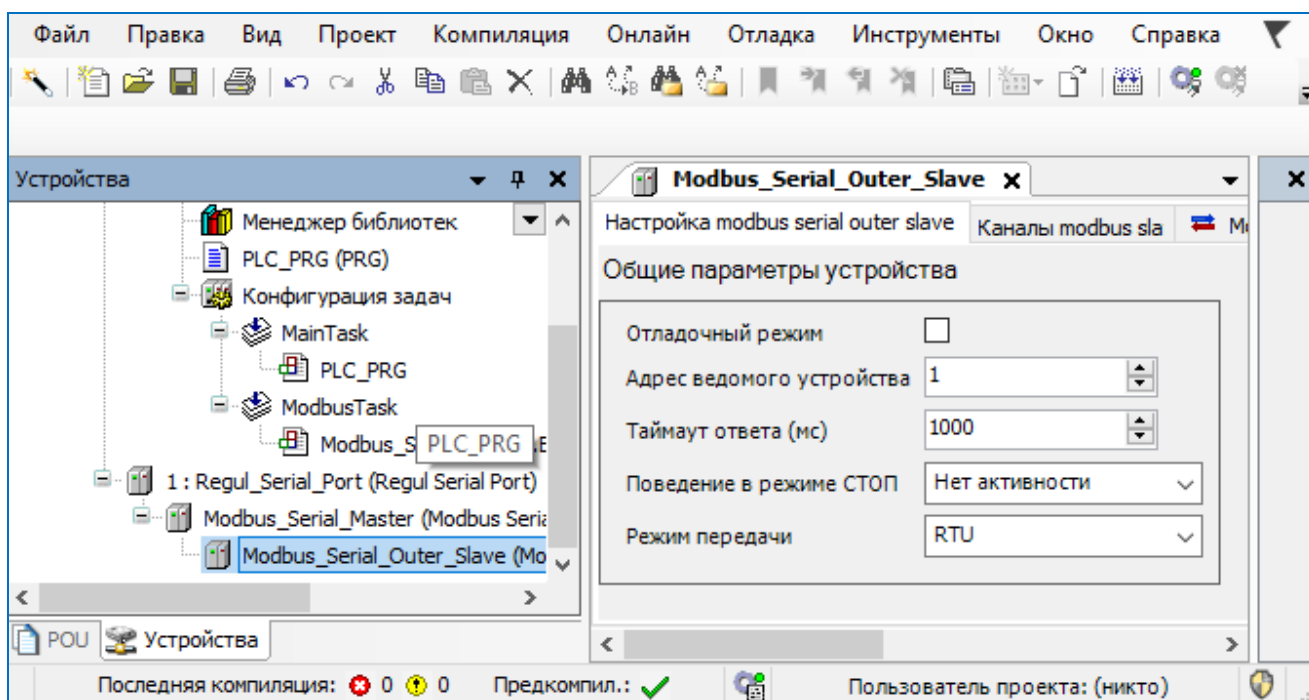


Рисунок 8 – Настройка параметров modbus serial outer slave

Установка флажка в поле **Отладочный режим (Debug mode)** включает отладочный режим.

Установите значения следующих параметров:

- **Адрес ведомого устройства (Slave ADR)** – это slave-адрес устройства согласно протоколу Modbus. Каждое из устройств (outer slave) на общей шине опроса (serial line) должно иметь уникальный адрес в диапазоне от 1 до 247;

- **Таймаут ответа (мс) (Response timeout)** – время ожидания ответа на запрос. По умолчанию установлено значение 1000 мс;
- **Поведение в режиме СТОП (Behavior in STOP mode)** – поведение в режиме *STOP*. Указывает, что делать, если переключатель RUN|STOP контроллера переведен в положение *STOP*: вариант *Нет активности (No activity)* означает прекращение опроса, вариант *Нормальная работа (Normal work)* означает продолжение работы в обычном режиме (рабочий цикл Modbus не зависит от Run|Stop контроллера);
- **Режим передачи (Serial Mode)** – режим опроса. Возможные значения: *ASCII* или *RTU*.



ИНФОРМАЦИЯ

Предусмотрена возможность самостоятельно активировать «поведение в режиме STOP» в программном коде. Это может потребоваться, например, в резервированной системе, если требуется, чтобы ведомый контроллер не вел опрос. При добавлении устройства Modbus также создается экземпляр соответствующего одноименного функционального блока. Для активации режима требуется в программе у экземпляра функционального блока Modbus Serial Master (не slave!) свойству `ActivateStopBehavior` присвоить значение `TRUE`:

`Modbus_Serial_Master.ActivateStopBehavior:=TRUE;`

После этого все подключенные к данному мастеру slave-устройства перейдут в STOP-режим работы

Далее требуется описать данные, которые контроллер будет запрашивать у slave-устройств. Перейдите на внутреннюю вкладку **Каналы modbus slave**. (Рисунок 9)

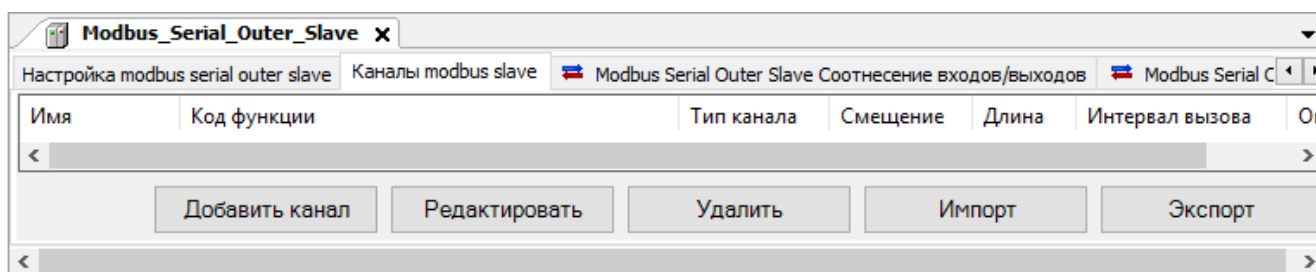


Рисунок 9 - Вкладка «Каналы modbus slave»

Каждый запрос Modbus представляется в виде «канала modbus slave». Чтобы задать описание нового канала, нажмите кнопку **Добавить канал**, откроется диалоговое окно (Рисунок 10). Кнопка **Редактировать** открывает эту же форму, но в режиме редактирования существующего канала. Кнопка **Удалить** удаляет описание указанного канала Modbus.

В процессе работы может возникнуть необходимость восстановить параметры, указанные ранее. Это возможно, если прежние настройки были экспортированы в файл. Кнопки **Экспорт/Импорт** позволяют сохранять/загружать информацию в/из файл с расширением `*.xml`. Для сохранения в файл нажмите кнопку **Экспорт**, откроется окно **Экспортировать каналы modbus**. Определите папку, в которой будет храниться этот файл, нажмите кнопку **Сохранить**. Чтобы открыть ранее сохраненные файлы нажмите кнопку **Импорт**, найдите на компьютере файлы типа `...mb_outer_channels.xml`.

Рисунок 10 – Добавление канала Modbus slave

Установите значения следующих параметров:

- **Имя** – понятное наименование (human readable);
- **Код функции** – номер функции Modbus, используемой для чтения/записи данных
- **Тип канала** – если выбрать значение *Таймер (Timer)*, то данный запрос будет выполняться с периодичностью, указанной в параметре **Интервал вызова (ms)**, при выборе значения *Триггер (Trigger)* запрос будет выполняться по событию (далее потребуются указать переменную, которая инициирует событие);
- **Смещение** – адрес первого запрашиваемого элемента, согласно протоколу Modbus;
- **Длина** – количество запрашиваемых элементов (регистров, либо флагов, в зависимости от кода функции);
- **Интервал вызова (ms)** – периодичность опроса данного канала, в миллисекундах;
- **Описание** – опционально, текстовое описание.

Пользовательские функции в соответствии со стандартом задаются в следующем диапазоне: 65 – 72 (0x41 – 0x48) и 100 – 110 (0x64 – 0x6E). Пример создания программы для выполнения пользовательских функций приведен в приложении А.

На рисунке 11 приведено два примера: добавление канала для чтения 32 дискретных значений с периодичностью 1000 мс и добавление команды для установки единственного флага (single coil) с номером 10 по событию.

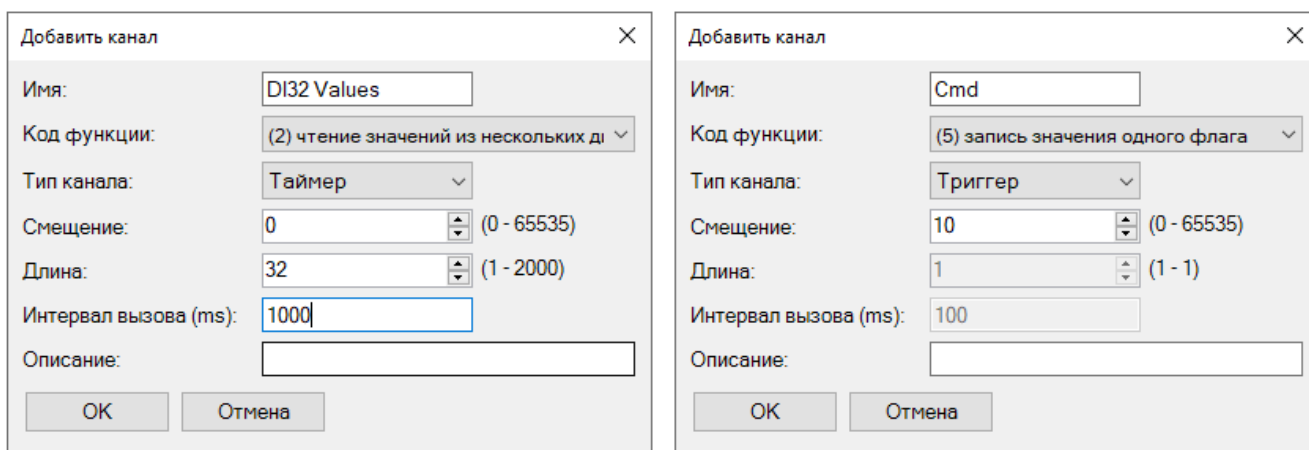


Рисунок 11 – Примеры каналов Modbus slave

Добавленные каналы на вкладке **Каналы modbus slave** выглядят следующим образом:

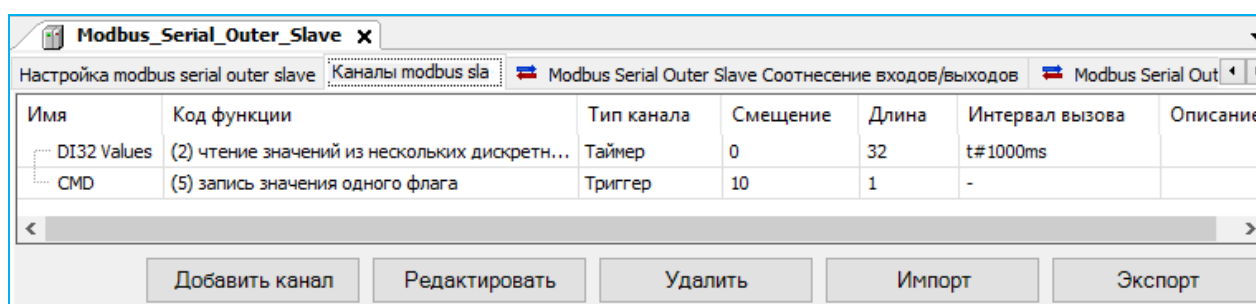


Рисунок 12 – Пример вкладки «Каналы modbus slave»

Далее перейдите на вкладку **Modbus Serial Outer Slave Соотнесение входов/выходов**. Параметр **Соединение (Connect)** показывает состояние связи с устройством, исходя из результата последнего запроса (нет ответа – нет связи) (Рисунок 13).

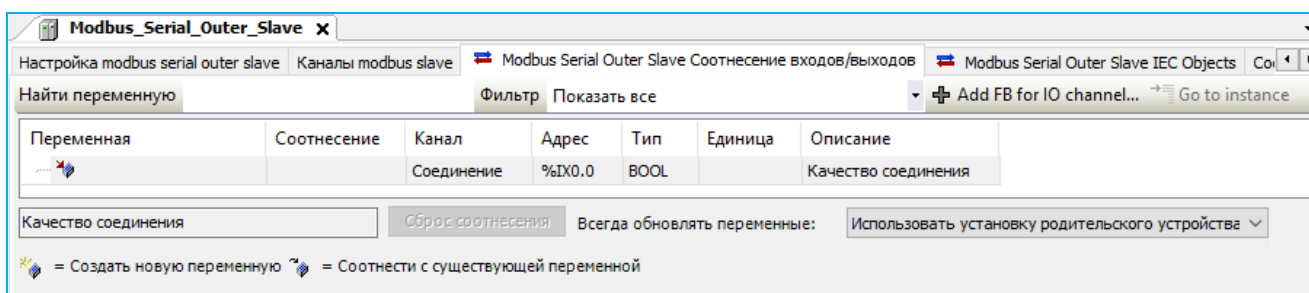


Рисунок 13 – Параметр «Соединение» отображает состояние связи с устройством

Для того, чтобы данные, получаемые от slave-устройства, использовать в программе контроллера, применяется инструмент **Соотнесение входов/выходов**. Он позволяет сопоставить данные каналов Modbus пользовательским переменным программы контроллера.

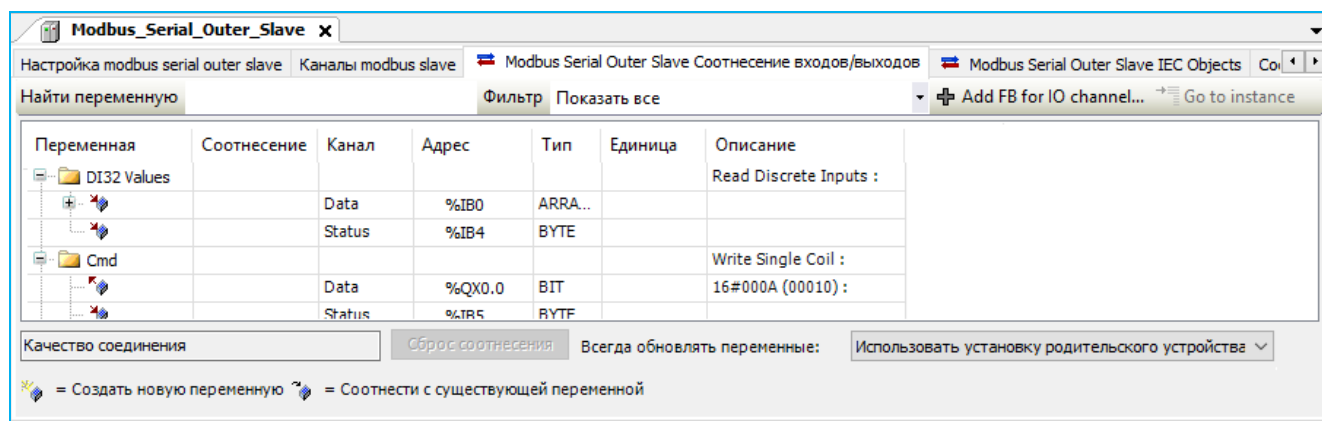


Рисунок 14 – Вкладка «Modbus Serial Outer Slave Соотнесение входов/выходов»



ИНФОРМАЦИЯ

В терминологии среды разработки каналом является любой входной или выходной параметр устройства. Чтобы не возникало путаницы с каналами Modbus, каналы, перечисленные в таблице на рисунке 14, в данном разделе называются параметрами

Каждый канал Modbus имеет несколько таких параметров:

- **Data** – собственно передаваемый блок данных Modbus. Представлен в виде массива элементов типа *WORD* (по сути регистров – для регистровых каналов), либо в виде массива байт (для битовых/дискретных каналов). Среда разработки осуществляет побитное копирование: для входов – из блока данных Modbus в переменную контроллера, для выходов – из переменных в драйвер Modbus;
- **Status** – показывает текущее состояние канала. Возможные значения описываются перечислением *ChannelStatus*, объявленным в библиотеке *PsModbusSerialMaster*. Чтобы отслеживать состояние канала в программе контроллера, его можно связать с переменной типа *BYTE*;
- **Trigger** – появляется при соответствующей настройке в поле **Тип канала**;



ВНИМАНИЕ!

При связывании переменной с блоком данных Modbus (параметр *Data*) обязательно выполнение следующего условия – размер переменной должен быть достаточным, т.е. быть больше или равным размеру блока данных

Ниже приведен пример объявления переменных для связывания с блоком данных Modbus.



```

VAR_GLOBAL
ai8:ARRAY [1..8] OF REAL;      //8*4bytes reals=32bytes=16registers
di32:UDINT;                    //double unsigned int
cmd_val: BOOL;                 //command value
cmd_trig: BOOL;                //command trigger
END_VAR
    
```

При объявлении перечисления состояния канала в библиотеке *PsModbusSerialMaster* используются следующие статусы канала (Таблица 1):

Таблица 1 – Статусы канала в библиотеке PsModbusSerialMaster

Название	Initial	Комментарий
Unreliable	0	Недостоверный
InProcess		Идет выполнение цикла «запрос-ответ»
Timeout		Ошибка – нет ответа
Ok		Данные – достоверны
ErrorInResponse		Ошибка – ошибка в ответе
UserReqArg		Ошибка – некорректный аргумент в пользовательском запросе
StopReading		Процесс чтения канала был остановлен

Чтобы связать параметр ввода/вывода с переменной, на вкладке **Modbus Serial Outer Slave Соотнесение входов/выходов** (Рисунок 14) дважды щелкните в строке нужного канала. Появится кнопка , открывающая окно **Ассистент ввода** (Рисунок 15). Найдите нужную переменную. Если установлен флажок в поле **Структурированный вид**, то раскрывайте списки с помощью кнопки . Если флажок снят и переменные представлены одним большим списком, для удобства поиска воспользуйтесь фильтром.

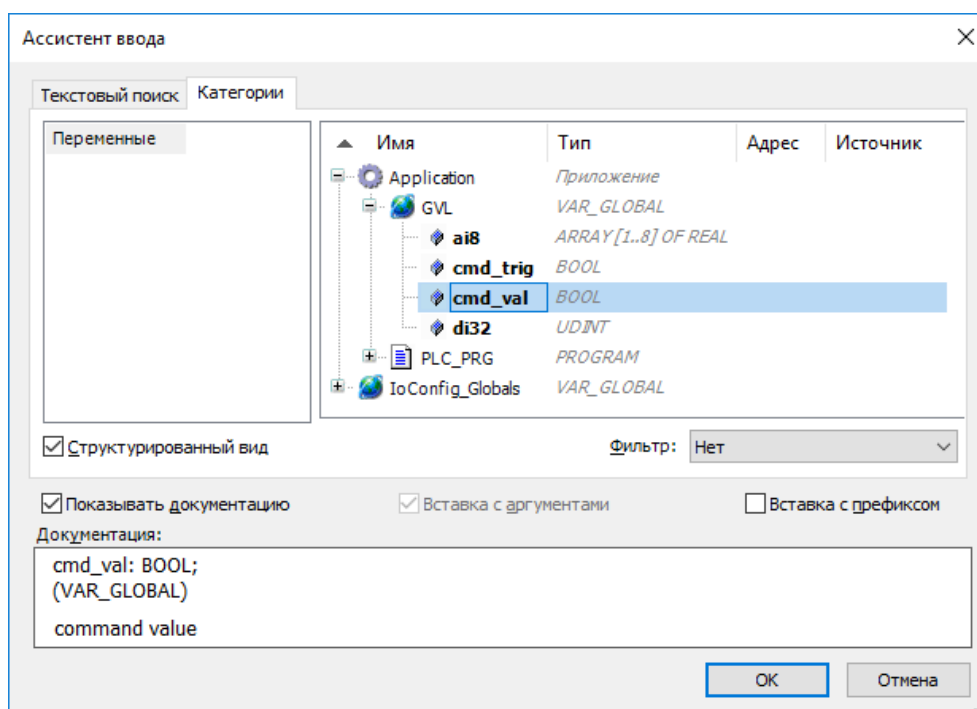


Рисунок 15 – Диалоговое окно «Ассистент ввода»

Вкладка **Modbus Serial Outer Slave Соотнесение входов/выходов**, где уже выполнена привязка каналов к переменным, выглядит следующим образом (Рисунок 16):

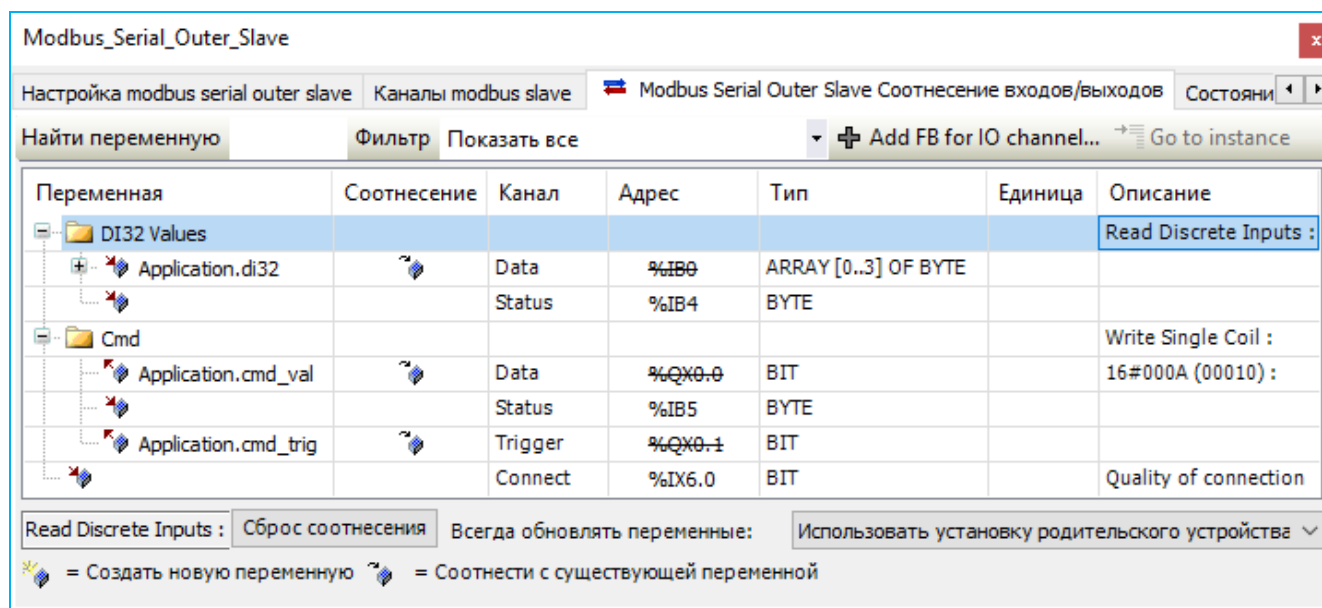


Рисунок 16 – Вкладка «Modbus Serial Outer Slave Соотнесение входов/выходов»

При опросе по событию следует связать параметр **Trigger** с переменной типа *BOOL*. Событием будет переключение переменной из состояния *FALSE* в состояние *TRUE*. Драйвер Modbus не осуществляет сброс триггера, для повторения запроса следует самостоятельно перевести переменную в состояние *FALSE* и в следующем цикле задачи контроллера вновь инициировать событие.

Настройка Modbus Serial Slave

В случае, когда контроллер будет являться slave-устройством, необходима настройка Modbus Serial Slave.

Добавьте устройство **Modbus Serial Slave** к последовательному порту Regul Serial Port или Extended Regul Serial Port (*Regul* → *Modbus* → *Serial Modbus Slave* → *Modbus Serial Slave*). Двойным щелчком по названию устройства **Modbus Serial Slave** откройте вкладку параметров. По умолчанию открывается первая внутренняя вкладка **Настройка modbus serial slave** (Рисунок 17).

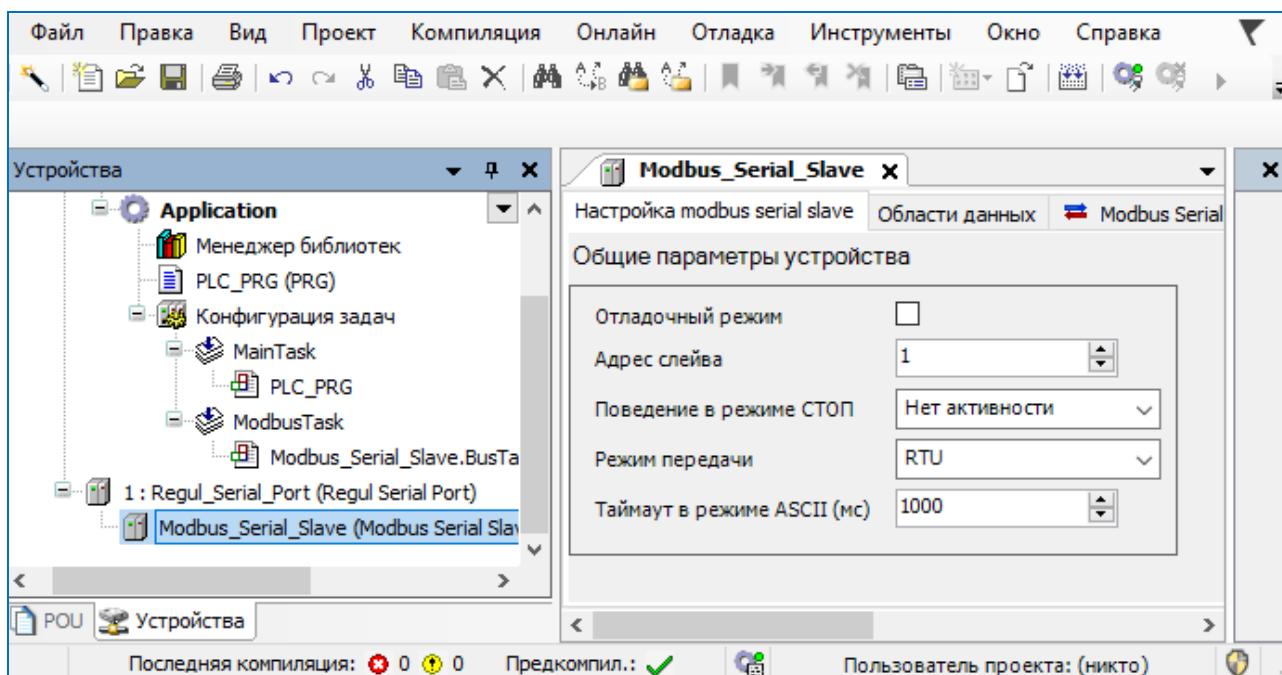


Рисунок 17 – Настройка параметров modbus serial slave

Установка флажка в поле **Отладочный режим (Debug mode)** включает отладочный режим (в журнал контроллера будет записываться дополнительная информация о работе).

Установите значения следующих параметров:

- **Адрес слейва (Slave ADR)** – это slave-адрес, назначенный контроллеру согласно протоколу Modbus;



ИНФОРМАЦИЯ

Начиная с версии СПО 1.6.4.0, появилась возможность в процессе работы изменять адрес устройства при помощи свойства SlaveID, например:

```
Modbus_Serial_Slave.SlaveID := newSlaveId;
```

- **Поведение в режиме стоп (Behavior in STOP mode)** – поведение в режиме *STOP*. Указывает, что делать, если переключатель RUN|STOP контроллера переведен в положение *STOP* – вариант *Нет активности (No activity)* означает, что контроллер прекращает отвечать на запросы, вариант *Нормальная работа (Normal work)* означает продолжение работы в обычном режиме (рабочий цикл Modbus не зависит от Run|Stop контроллера). При выборе значения *Генерация исключения (Exception)* ответ на любой запрос Modbus будет содержать код ошибки 06 (*Устройство занято*);
- **Режим передачи (Serial Mode)** – режим опроса. Возможные значения: *ASCII* или *RTU*.
- **Таймаут в режиме ASCII (мс) (ASCII Mode Timeout)** – время ожидания конца сообщения (перевод строки и возврат каретки – LF, CR) в миллисекундах.



ИНФОРМАЦИЯ

Предусмотрена возможность самостоятельно активировать «поведение в режиме STOP» в программном коде. Для активации режима требуется в программе у экземпляра устройства Modbus Serial Slave свойству ActivateStopBehavior присвоить значение TRUE:

```
Modbus_Serial_Slave_Device.ActivateStopBehavior:=TRUE;
```

После этого slave-устройство перейдет в STOP-режим работы

Перейдите на внутреннюю вкладку **Области данных modbus slave**, где размещается описание области данных Modbus, доступной для запроса со стороны внешних Master-устройств (Рисунок 18).

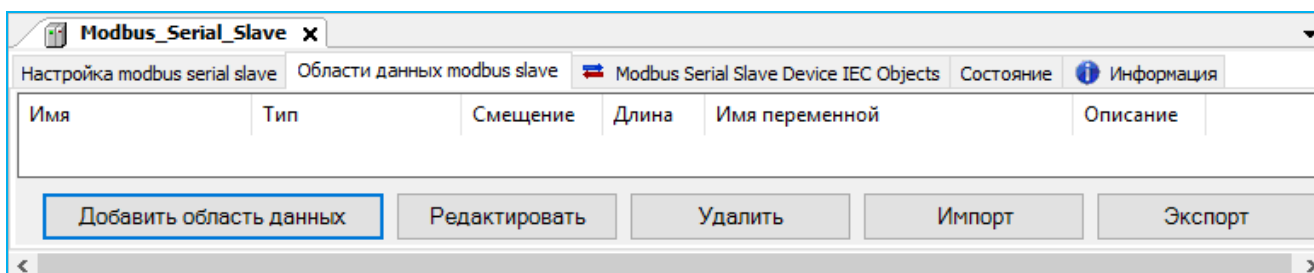


Рисунок 18 – Вкладка «Области данных Modbus slave»

Чтобы задать описание новой области нажмите кнопку **Добавить область данных**, откроется диалоговое окно (Рисунок 19).

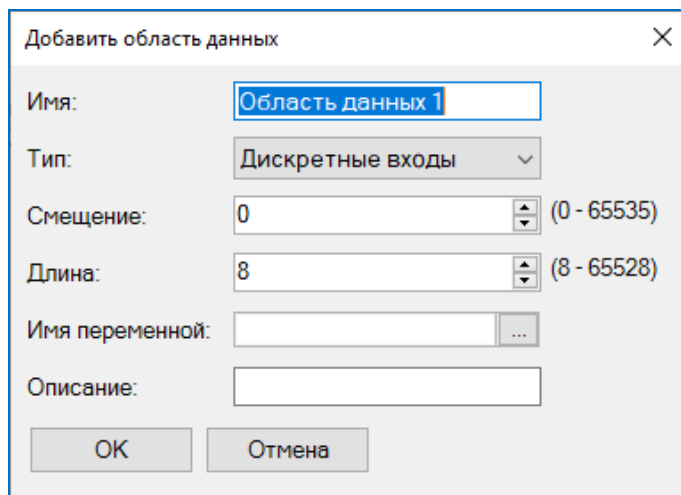


Рисунок 19 – Добавление области данных Modbus slave



Кнопка **Редактировать** открывает эту же форму, но в режиме редактирования существующей области данных. Кнопка **Удалить** удаляет описание указанной области данных Modbus Slave.

В процессе работы может возникнуть необходимость восстановить параметры, указанные ранее. Это возможно, если прежние настройки были экспортированы в файл. Кнопки **Экспорт/Импорт** позволяют сохранять/загружать информацию в/из файл с расширением ***.xml**. Для сохранения в файл нажмите кнопку **Экспорт**, откроется окно **Экспортировать**

каналы modbus. Определите папку, в которой будет храниться этот файл, нажмите кнопку **Сохранить**. Чтобы открыть ранее сохраненные файлы нажмите кнопку **Импорт**, найдите на компьютере файлы типа ... *mb_direct_channels.xml*.

Установите значения следующих параметров:

- **Имя** – понятное наименование (human readable);
- **Тип** – тип данных Modbus (*Дискретные входы (Discrete Inputs)*, *Регистры флагов (Coils)*, *Регистры ввода (Input Registers)*, *Регистры хранения (Holding Registers)*);
- **Смещение** – адрес первого элемента, доступного для запроса, согласно протоколу Modbus;
- **Длина** – количество элементов, доступных для запроса (регистров либо флагов, в зависимости от типа данных Modbus);
- **Имя переменной** – здесь указывается переменная (программы контроллера), в которой хранятся передаваемые данные, размер переменной должен быть не менее размера объявленной области данных Modbus (поле **Длина**);
- **Описание** – опционально, текстовое описание.

Для заполнения поля **Имя переменной** нажмите в этом поле кнопку , открывающую окно **Ассистент ввода**. Найдите нужную переменную. Если установлен флажок в поле **Структурированный вид**, то раскрывайте списки с помощью кнопки . Если флажок снят и переменные представлены одним большим списком, для удобства поиска воспользуйтесь фильтром.

На рисунке 20 приведен пример, когда создано 4 области данных, по одной для каждого типа данных Modbus. Размер каждой области данных – 2000 элементов. Для областей данных типа *Регистры флагов (Coils)* и *Дискретные входы (Discrete Inputs)* элементом данных является бит, соответственно размер связываемой переменной в байтах вычисляется по формуле: $(\text{длина канала}-1)/8+1$. Для областей данных типа *Регистры ввода (Input Registers)* и *Регистры хранения (Holding Registers)* элементом данных является регистр, по размеру соответствующий типу *WORD* в среде разработки. В нашем примере с областями данных типа *Регистры флагов* и *Дискретные входы* связаны массивы *coils_area* и *di_area* соответственно, размером по 250 байт (Рисунок 21). С областями данных типа *Регистры ввода* и *Регистры хранения* связаны массивы *ir_area* и *hr_area* соответственно, по 2000 элементов *WORD* (Рисунок 21).

Modbus_Serial_Slave x					
Настройка modbus serial slave		Области данных modbus slave		Modbus Serial Slave Device IEC Objects	
Состояние Информация					
Имя	Тип	Смещение	Длина	Имя переменной	Описание
Область данных 1	Дискретные входы	0	2000	PLC_PRG.di_area	
Область данных 2	Регистры флагов	0	2000	PLC_PRG.coils_area	
Область данных 3	Регистры ввода	0	2000	PLC_PRG.ir_area	
Область данных 4	Регистры хранения	0	2000	PLC_PRG.hr_area	

Добавить область данных
 Редактировать
 Удалить
 Импорт
 Экспорт

Рисунок 20 – Примеры областей данных Modbus Slave



ИНФОРМАЦИЯ

Привязка переменных реализована на уровне библиотеки PsModbusSerialSlave, а не с использованием механизма I/O Mapping (Соотнесение входов/выходов), как в предыдущих версиях ПО

```

PLC_PRG
1  PROGRAM PLC_PRG
2  VAR
3  //Вариант работы 1 - создаем области в M-памяти, связываем их с каналами modbus slave
4  coils_area AT %MW0:ARRAY [0..249] OF BYTE; //область адресации реле (coils)
5  di_area AT %MW125:ARRAY [0..249] OF BYTE; //область адресации дискретных входов (DI)
6  hr_area AT %MW250:ARRAY [1..2000] OF WORD; //область адресации хранимых регистров (HR)
7  ir_area AT %MW2250:ARRAY [1..2000] OF WORD; //область адресации входных регистров (coils)
8  //переменные размещаем по адресам, соответствующим ранее указанным M-областям
9  coils1 AT %MW0: D16_VALUE_T; //coils test
10 coils2 AT %MW124: D16_VALUE_T;
11 di1 AT %MW125: D16_VALUE_T; //di test
12 di2 AT %MW249: D16_VALUE_T;
13 hr1 AT %MW250: ARRAY [1..8] OF REAL:= [1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8]; //hr test
14 hr2 AT %MW2234: ARRAY [1..8] OF REAL:= [1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8];
15 ir1 AT %MW2250: ARRAY [1..8] OF REAL:= [1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8]; //ir test
16 ir2 AT %MW4234: ARRAY [1..8] OF REAL:= [1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8];
17
18 //Вариант работы 2 - создаем массивы для размещения переменных различных типов,
19 //связываем их с каналами modbus slave
20 reals_area: ARRAY [1..2000] OF REAL;
21 dint_area: ARRAY [1..2000] OF DINT;
22 discretets_area: ARRAY [1..250] OF BYTE;
23 //переменные объявляем как ссылки на элементы соответствующих массивов
24 real1: REFERENCE TO REAL:=reals_area[1];
25 real2000: REFERENCE TO REAL:=reals_area[2000];
26 dint1: REFERENCE TO DINT:=dint_area[1];
27 dint2000: REFERENCE TO DINT:=dint_area[2000];
    
```

Рисунок 21 – Объявление переменных для связи с каналами Modbus. Использование M-памяти и ссылок (REFERENCE)

При работе с данными, передаваемыми по Modbus, можно применять стратегии, описанные ниже.

Первая стратегия – использование M-памяти контроллера. Для каждого типа данных Modbus создается по одной переменной с размером, достаточным для размещения всех передаваемых значений. Переменная объявляется с директивой AT и указанием на адрес в M-памяти. Далее следует создать по одному каналу Modbus для каждого типа данных (*Дискретные входы,*

Регистры флагов, Регистры ввода, Регистры хранения), связывая их с ранее объявленными переменными. Эти переменные выполняют только роль буферов данных, доступных на чтение и на запись. Собственно, данные, которые требуется передавать по Modbus, могут иметь любой тип среды разработки и должны объявляться также с директивой AT, размещаясь по адресам, попадающим в соответствующий буфер. Такой подход реализован в примере, представленном на рисунках 20 и 21 (помечен как «Вариант работы 1»). Массивы coils_area, di_area, hr_area, ir_area являются буферами данных, переменные coils1, coils2, di1, di2, hr1, hr2, ir1, ir2 – тестовые элементы данных, размещенные в указанных буферах.

Вторая стратегия – связывание канала со структурой (тип данных STRUCT). При объявлении структуры в среде разработки нужно задать в ней все поля, которые предназначены для выдачи по Modbus, далее создать канал и связать его с переменной – экземпляром этой структуры.

```

1 //объявление структуры
2 {attribute 'pack_mode' := '1'}
3     TYPE MODBUS_DATA_T :
4     STRUCT
5         ai8: ARRAY [0..7] OF REAL;
6         counter1, counter2: DINT;
7         b0,b1,b2,b3,b4,b5,b6,b7:BIT;
8     END_STRUCT
9     END_TYPE
10 //объявление переменной для связи с каналом Modbus
11     mb_data : MODBUS_DATA_T;

```

```

1 //реализация
2 mb_data.ai8[0]:=15.6;
3 mb_data.counter1:= mb_data.counter1+1;
4 mb_data.b1:=0; mb_data.b5:=1;
5

```

Рисунок 22 – Пример связывания канала со структурой

НАСТРОЙКА MODBUS TCP

Настройка Modbus TCP Master

В случае, когда контроллер будет использоваться как ведущее, опрашивающее устройство, необходима настройка Modbus TCP Master. При настройке данного режима требуется задать параметры slave-устройства, которое будет опрашиваться контроллером. Кроме того, требуется описать набор данных, который будет запрашиваться по Modbus.

Добавьте устройство **Modbus TCP Master** к контроллеру (*Regul* → *Modbus* → *TCP Modbus Master* → *Modbus TCP Master*). Двойным щелчком по названию устройства **Modbus TCP Master** откройте вкладку параметров (Рисунок 23).

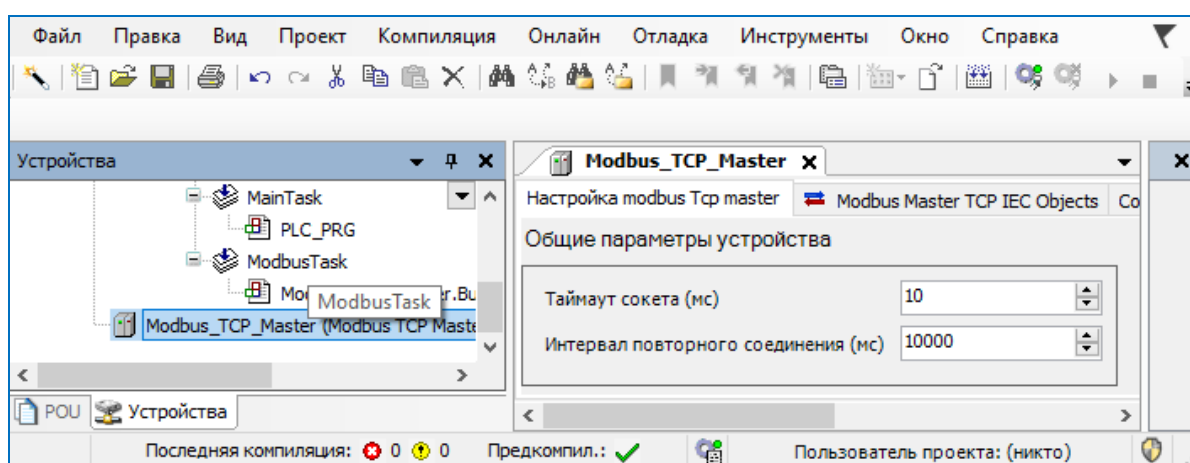


Рисунок 23 – Общие параметры Modbus TCP Master

К общим параметрам устройства относятся параметры, менять значения, которых обычно не требуется:

- **Таймаут сокета (мс) (Socket Timeout)** – это время ожидания в миллисекундах для операции select ();
- **Интервал повторного соединения (мс) (Reconnect interval)** – если нет TCP-соединения, по истечении данного временного интервала (в миллисекундах) произойдет попытка переустановить TCP-соединение.

Далее к устройству **Modbus TCP Master** нужно подключить одно или несколько внешних slave-устройств (outer slaves), которые будут опрашиваться контроллером (*Regul* → *Modbus* → *TCP Modbus Master* → *Modbus TCP Outer Slave*). Двойным щелчком по названию устройства **Modbus TCP Outer Slave** откройте вкладку параметров. По умолчанию открывается первая внутренняя вкладка **Настройка modbus Tcp outer slave** (Рисунок 24).

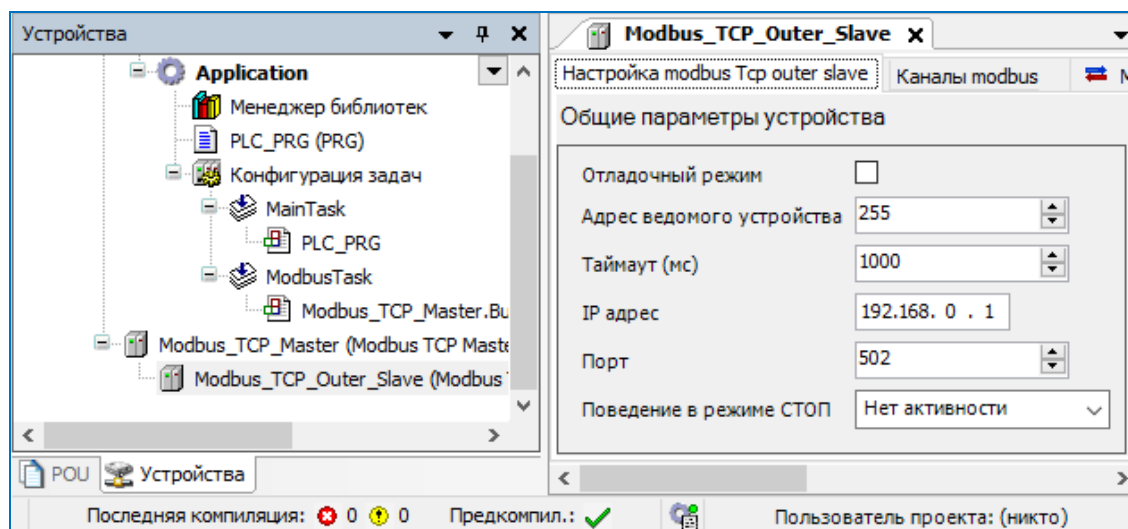


Рисунок 24 – Настройка параметров modbus Tcp outer slave

Установка флажка в поле **Отладочный режим (Debug mode)** включает отладочный режим (в журнал контроллера будет записываться дополнительная информация о работе).

Установите значения следующих параметров:

- **Адрес ведомого устройства (Unit ID)** – это адрес опрашиваемого устройства. Он может требоваться для некоторых специфичных устройств, которым необходимо явное указание адреса Modbus Slave;
- **Таймаут (мс) (Timeout)** – время ожидания ответа на запрос. По умолчанию установлено значение 1000 мс;
- **IP адрес (IP Address)** – адрес slave-устройства;
- **Порт (Port)** – номер открытого на Slave порта TCP для подключения (в диапазоне от 256 до 65535);
- **Поведение в режиме СТОП (Behavior in STOP mode)** – поведение в режиме *STOP*. Указывает, что делать, если переключатель RUN|STOP контроллера переведен в положение *STOP* – вариант *Нет активности (No activity)* означает прекращение опроса, вариант *Нормальная работа (Normal work)* означает продолжение работы в обычном режиме (рабочий цикл Modbus не зависит от Run|Stop контроллера), вариант *Закреть соединение (Close Connection)* означает, что соединение будет закрыто.



ИНФОРМАЦИЯ

Предусмотрена возможность самостоятельно активировать «поведение в режиме STOP» в программном коде. Для активации режима требуется в программе у соответствующего одноименного функционального блока Modbus TCP Master свойству ActivateStopBehavior присвоить значение TRUE:

```
Modbus_Tcp_Master.ActivateStopBehavior:=TRUE;
```

После этого все подключенные к данному мастеру slave-устройства перейдут в STOP-режим работы

Далее требуется описать данные, которые контроллер будет запрашивать у slave-устройств. Эта процедура и дальнейшие настройки полностью аналогичны тем, что описаны в разделе **Настройка Modbus Serial Master**.

Настройка Modbus TCP Slave

В случае, когда контроллер будет являться slave-устройством, необходима настройка Modbus TCP Slave.

Добавьте устройство **Modbus Tcp Slave** к контроллеру (*Regul* → *Modbus* → *TCP Modbus Slave* → *Modbus Tcp Slave*). Двойным щелчком по названию устройства **Modbus Tcp Slave** откройте вкладку параметров. По умолчанию открывается первая внутренняя вкладка **Настройка modbus Tcp slave** (Рисунок 25).

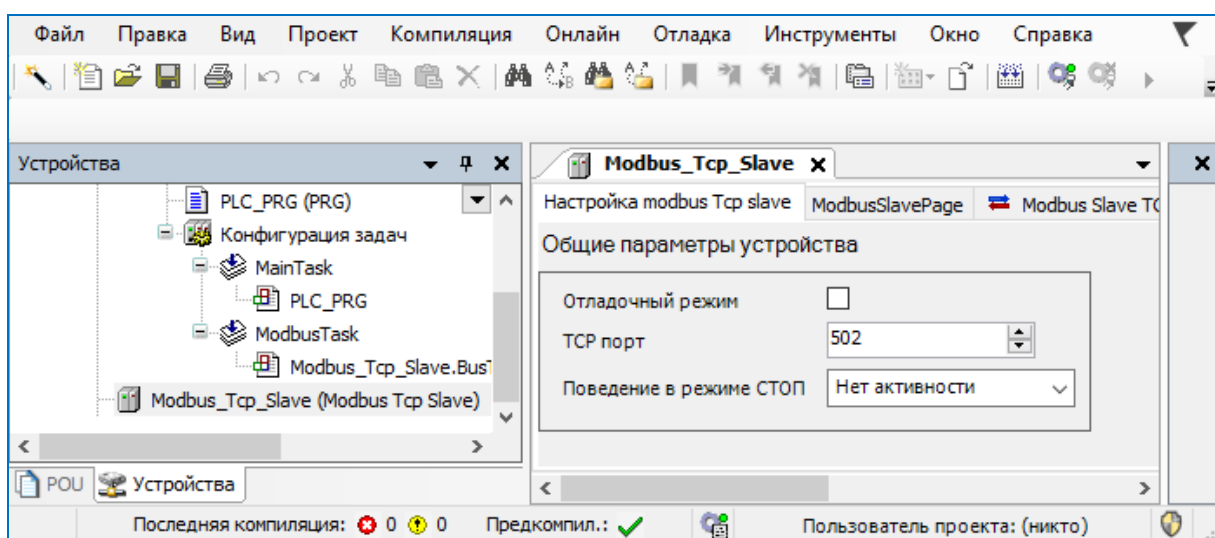


Рисунок 25 – Настройка параметров Настройка modbus Tcp slave

Установка флажка в поле **Отладочный режим (Debug mode)** включает отладочный режим (в журнал контроллера будет записываться дополнительная информация о работе).

Установите значения следующих параметров:

- **ТСП порт (TCP Port)** – номер порта, прослушиваемого драйвером Modbus (в диапазоне от 256 до 65535);
- **Поведение в режиме СТОП (Behavior in STOP mode)** – поведение в режиме *STOP*. Указывает, что делать, если переключатель RUN|STOP контроллера переведен в положение *STOP* – вариант *Нет активности (No activity)* означает прекращение опроса, вариант *Нормальная работа Normal work* означает продолжение работы в обычном режиме (рабочий цикл Modbus не зависит от Run|Stop контроллера), вариант *Закрывать соединение Close Connection* означает, что соединение будет закрыто, попытки установки TCP-соединения будут отклонены. При выборе значения *Генерация исключения (Exception)* ответ на любой запрос Modbus будет содержать код ошибки 06 (*Устройство занято*).



ИНФОРМАЦИЯ

Предусмотрена возможность самостоятельно активировать «поведение в режиме STOP» в программном коде. Для активации режима требуется в программе у экземпляра устройства Modbus TCP Slave свойству ActivateStopBehavior присвоить значение TRUE:

```
Modbus_TCP_Slave_Device.ActivateStopBehavior:=TRUE;
```

После этого slave-устройство перейдет в STOP-режим работы

Каждое установленное соединение с ПЛК, в случае бездействия, будет закрыто по таймауту. Автоматическое отключение неиспользуемого соединения произойдет через 60 секунд.



ИНФОРМАЦИЯ

Для поддержания соединения отправляйте как минимум одну команду с интервалом менее 60 секунд

Настройки областей данных Modbus Slave и приемы работы с ними полностью идентичны тем, что описаны в разделе **Настройка Modbus Serial Slave**.

ПРИЛОЖЕНИЕ А

Пример создания программы для пользовательских функций приведен ниже.

```
END_VAR
PROGRAM PLC_PRG
VAR
    MASTER_DATA2: ARRAY [0..9] OF WORD;
    mdb_request2: PsIoDrvModbusSerialMaster.ModbusUserRequest2;
END_VAR
-----
mdb_request2(
    xExecute := TRUE,
    refModbusOuterSlave := Modbus_Serial_Outer_Slave,
    byFC := 16#6e,
    //pSendBuf, //M: Буфер отправляемых данных (PDU запроса без номера функции)
    //uiSendDataSize, //M: размер отправляемого блока данных
    pRecvBuf := MASTER_DATA2[0], //M: Выходной Буфер данных (без номера функции)
    uiRecvBufSize := SIZEOF(MASTER_DATA2) //M: размер выходного буфера (просто
    размер выделенного массива байт)
);

IF(mdb_request2.xDone) THEN
    mdb_request2(xExecute := FALSE);
END_IF

IF(mdb_request2.xError) THEN
    mdb_request2(xExecute := FALSE);
END_IF

mdb_request2.Status;
```