



# **CODESYS V3.5**

**Настройка обмена по протоколу Modbus**



**Руководство пользователя**

03.09.2019

версия 2.0

## Оглавление

<b>1</b>	<b>Цель документа. Способы работы с Modbus в CODESYS V3.5</b>	<b>6</b>
<b>2</b>	<b>Общие сведения</b>	<b>7</b>
2.1	Основные сведения об интерфейсе RS-485	7
2.2	Основные сведения о протоколе Modbus	7
2.3	Нумерация COM-портов в CODESYS	10
2.4	Особенности работы с модулями Mx110	11
<b>3</b>	<b>Шаблоны модулей Mx110 и Mx210</b>	<b>13</b>
3.1	Установка шаблонов модулей в среду CODESYS	13
3.2	Пример: СПК1хх [M01] + модули Mx110	15
3.3	Пример: СПК1хх [M01] + модули Mx210	23
3.4	Диагностика и управление обменом	30
3.5	Библиотеки Mx Assistant	32
<b>4</b>	<b>Стандартные средства конфигурирования</b>	<b>34</b>
4.1	Общая методика конфигурирования интерфейсов	34
4.2	Настройка контроллера в режиме Modbus Serial Master	35
4.3	Настройка контроллера в режиме Modbus RTU Slave	42
4.4	Настройка контроллера в режиме Modbus TCP Master	46
4.5	Настройка контроллера в режиме Modbus TCP Slave	51
4.6	Диагностика и управление обменом	54
4.7	Компоненты Modbus и конфигурация задач	60
4.8	Преобразование данных для передачи по Modbus	61
4.8.1	Использование объединений (UNION)	61
4.8.2	Использование указателей	65

4.9	Пример: СПК1хх [M01] (Modbus RTU Master) + модули Mx110 .....	68
4.10	Пример: СПК1хх [M01] (Modbus RTU Slave) + MasterOPC Universal Modbus Server .....	80
4.11	Пример: СПК1хх [M01] (Modbus TCP Master) + модули Mx210 .....	91
4.12	Пример: СПК1хх [M01] (Modbus TCP Slave) + MasterOPC Universal Modbus Server .....	102
<b>5</b>	<b>Библиотека OwenCommunication .....</b>	<b>113</b>
5.1	Основная информация.....	113
5.2	Установка библиотеки .....	113
5.3	Добавление библиотеки в проект CODESYS .....	114
5.4	Структуры и перечисления .....	115
5.4.1	Перечисление ERROR .....	115
5.4.2	Перечисление COM_PARITY.....	116
5.4.3	Перечисление COM_STOPBIT .....	116
5.4.4	Перечисление MB_FC.....	116
5.4.5	Структура MB_REQ_INFO.....	117
5.5	ФБ настройки интерфейсов .....	118
5.5.1	ФБ COM_Control .....	118
5.5.2	ФБ TCP_Client .....	119
5.6	ФБ протокола Modbus .....	120
5.6.1	ФБ MB_SerialRequest .....	120
5.6.2	ФБ MB_SerialSlave .....	122
5.6.3	ФБ MB_TcpRequest.....	124
5.6.4	ФБ MB_TcpSlave .....	126
5.7	ФБ нестандартных протоколов.....	128
5.7.1	ФБ UNM_SerialRequest.....	128
5.7.2	ФБ UNM_TcpRequest.....	130
5.7.3	ФБ UNM_UdpRequest .....	132
5.8	Функции и ФБ преобразования данных .....	134
5.8.1	ФБ DWORD_TO_WORD2 .....	134
5.8.2	ФБ REAL_TO_WORD2.....	135
5.8.3	Функция WORD2_TO_DWORD .....	136
5.8.4	Функция WORD2_TO_REAL.....	137
5.8.5	Функция SWAP_DATA .....	138
5.9	Примеры.....	139
5.9.1	СПК1хх [M01] (Modbus RTU Master) + модули Mx110 .....	139
5.9.2	СПК1хх [M01] (Modbus RTU Slave) + MasterOPC Universal Modbus Server .....	149
5.9.3	СПК1хх [M01] (Modbus TCP Master) + модули Mx210 .....	157
5.9.4	СПК1хх [M01] (Modbus TCP Slave) + MasterOPC Universal Modbus Server .....	167
5.9.5	СПК1хх [M01] (Modbus TCP Slave) – чтение файлов с помощью 20 функции Modbus 175	

<b>6</b>	<b>FAQ</b>	<b>182</b>
6.1	Что делать, если не удастся наладить обмен по Modbus?	182
6.2	Каким образом считать/передать значение с плавающей точкой (REAL)?	183
6.3	Каким образом считать/передать отрицательное значение (INT)?	183
6.4	Вопросы по стандартным средствам конфигурирования	183
6.4.1	Какие версии компонентов рекомендуются к использованию?	183
6.4.2	Modbus Serial Master: как реализовать чтение/запись по триггеру?	184
6.4.3	Modbus RTU Slave: почему принятые значения сбрасываются в 0?	185
6.4.4	Можно ли менять данные holding регистров из программы?	185
6.4.5	Как произвести диагностику обмена в программе?	185
6.4.6	Как расшифровываются пиктограммы статуса обмена?	185
6.5	Вопросы по библиотеке OwenCommunication	186
6.5.1	В примерах работы с библиотекой используются действия. Как добавить их в проект?	186
6.5.2	Позволяет ли библиотека организовать опрос с более высокой частотой по сравнению со стандартными средствами конфигурирования?	186
<b>Приложение А. Рекомендуемые версии компонентов Modbus</b>		<b>187</b>
<b>Приложение Б. Листинги примеров</b>		<b>188</b>
<b>Б1</b>	<b>Листинг примера из п. 5.9.1</b>	<b>188</b>
Б.1.1	Код программы PLC_PRG_ST	188
Б.1.2	Код действия COM1	189
Б.1.3	Код действия COM2	190
<b>Б2</b>	<b>Листинг примера из п. 5.9.2</b>	<b>192</b>
<b>Б3</b>	<b>Листинг примера из п. 5.9.3</b>	<b>194</b>
Б.3.1	Код программы PLC_PRG_ST	194
Б.3.2	Код действия MV210_101	195
Б.3.3	Код действия MK210_301	196
<b>Б4</b>	<b>Листинг примера из п. 5.9.4</b>	<b>198</b>
<b>Б5</b>	<b>Листинг примера из п. 5.9.5</b>	<b>200</b>



## 1 Цель документа. Способы работы с Modbus в CODESYS V3.5

Настоящее руководство описывает настройку обмена данными по протоколу **Modbus** для контроллеров ОВЕН, программируемых в среде **CODESYS V3.5**. Предполагается, что читатель обладает базовыми навыками работы с **CODESYS**, поэтому общие вопросы (например, создание и загрузка проектов) в данном документе не рассматриваются – они подробно описаны в документах **CODESYS V3.5. Первый старт** и **CODESYS V3.5. FAQ**, которые доступны на сайте [ОВЕН](#) в разделе [CODESYS V3/Документация](#). Документ соответствует версии CODESYS **3.5.11.5** или выше.

В зависимости от квалификации и потребностей пользователя имеется возможность выбрать наиболее подходящий для него способ организации связи по протоколу **Modbus**:

1. Для начинающих пользователей, работающих с модулями [Mx110](#) и Mx210 – **шаблоны модулей**. Шаблоны представляют собой сконфигурированные компоненты **CODESYS**, добавляемые в проект несколькими кликами мыши, для которых следует указать только сетевой адрес модуля.

Преимущества	Ограничения
Простота использования	Строго заданная конфигурация регистров без возможности редактирования
Быстрое создание проекта	
Нет необходимости в дополнительном программировании	
Не надо разбираться с картами регистров модулей	

2. Для обычных пользователей – **стандартные средства конфигурирования CODESYS**. С их помощью можно достаточно просто настроить обмен с любым устройством.

Преимущества	Ограничения
Возможность создания произвольной конфигурации	Требуются знания спецификации протокола Modbus
Возможность выбора регистров и функций для опроса	
Ручная настройка таймаутов и задержек для обеспечения корректной работы специфических устройств	Ограниченные возможности по управлению обменом
Нет необходимости в дополнительном программировании	Ограничения по числу опрашиваемых устройств, каналов и т. д.

3. Для профессионалов – библиотека **OwenCommunication**. Библиотека позволяет настроить обмен с любым устройством, но, в отличие от стандартных средств (пп. 2), лишена их ограничений и предоставляет дополнительный функционал.

Преимущества	Ограничения
Практически неограниченные возможности для работы с протоколом Modbus	Требуются хорошие навыки программирования и знание спецификации протокола Modbus
Создание любых конфигураций и методов опроса устройств	
Удобно при программировании модульных систем (когда в разные промежутки времени в работе находится разное оборудование)	Библиотека поддерживается только на контроллерах ОВЕН (затрудняется процесс переноса ПО на другие устройства)

## 2 Общие сведения

### 2.1 Основные сведения об интерфейсе RS-485

1. Интерфейс [RS-485](#) подразумевает использование исключительно топологии типа «шина» (топологии типа «звезда» и «кольцо» не описаны в стандарте).
2. В сети может присутствовать только одно master-устройство, которое отправляет запросы и принимает ответы от подчиненных slave-устройств. Slave-устройства не могут являться инициаторами обмена.
3. Число slave-устройств в сегменте сети не должно превышать **32**. Сегменты могут быть соединены повторителями (например, [ОВЕН АС5](#)), но следует учитывать что так как опрос всех устройств происходит последовательно, то время одного полного цикла опроса может значительно увеличиться. Общее ограничение числа slave-устройств в сети для протокола Modbus – **247**.
4. На первом и последнем устройстве в сети рекомендуется устанавливать согласующий резистор (терминатор) с сопротивлением **120 Ом**.
5. Для линий связи RS-485 необходимо использовать экранированный кабель с витой парой, предназначенный для промышленного интерфейса RS-485 с волновым сопротивлением **120 Ом** (например, КИПЭВ). Экран кабеля должен быть соединен с функциональной землей только в одной точке.
6. Некоторые устройства имеют встроенные резисторы подтяжки интерфейса RS-485. Информация и рекомендации по их использованию приведены в руководстве по эксплуатации на соответствующие приборы.

### 2.2 Основные сведения о протоколе Modbus

[Modbus](#) – открытый коммуникационный протокол, основанный на архитектуре **Master-Slave** (ведущий-ведомый). Спецификация протокола доступна на сайте [Modbus Organization](#).

**Master** (мастер, ведущее устройство) является инициатором обмена и может считывать и записывать данные в slave-устройства.

**Slave** (слэйв, подчиненное устройство) отвечает на запросы master-устройства, но не может самостоятельно инициировать обмен.

Существуют две основные реализации протокола:

1. **Modbus Serial** для передачи данных с использованием последовательных интерфейсов [RS-232/RS-485](#);
2. **Modbus TCP** для передачи данных через сети [TCP/IP](#).

**Modbus Serial** имеет два режима передачи данных:

1. **Modbus RTU** (передача данных в двоичном виде);
2. **Modbus ASCII** (передача данных в виде ASCII символов).

В случае использования протокола **Modbus** поверх интерфейса **RS-232/RS-485** в сети может присутствовать только одно master-устройство и несколько (до **247**) slave-устройств. Адрес **0** используется для широковещательной рассылки (команд записи, которую получают все slave-устройства).

В сети **Modbus TCP** нет явного ограничения на количество master- и slave-устройств. Кроме того, устройство может одновременно выполнять функции master и slave. В сети могут также существовать специальные шлюзы (**gateway**) для объединения сетей **Modbus Serial** и **Modbus TCP**.

Запрос master-устройства к slave-устройству содержит:

- **Slave ID** (адрес slave-устройства);
- **Код функции**, применяемой к slave-устройству;
- **Данные** – адрес первого регистра и их количество (в случае записи – также записываемые значения).
- Контрольную сумму.

Ответ slave-устройства имеет схожую структуру.

Запрос master-устройства представляет собой обращение к одной из **областей памяти** slave-устройства с помощью определенной **функции**. **Область памяти** характеризуется типом хранящихся в ней значений (биты/регистры) и типом доступа (только чтение/чтение и запись). Стандарт Modbus определяет 4 области памяти:

**Таблица 2.1 – Области данных протокола Modbus**

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

Каждая область памяти состоит из определенного (зависящего от конкретного устройства) количества ячеек. Каждая ячейка имеет уникальный адрес. Для конфигурируемых устройств (таких как ТРМ, ПЧВ и т. д.) производитель предоставляет **карту регистров**, в которой содержится информация об адресах и типах параметров устройства. Для программируемых устройств пользователь формирует такую карту самостоятельно с помощью среды разработки. Существуют устройства, в которых сочетаются оба рассмотренных случая – у их карты регистров есть фиксированная часть, которую пользователь может дополнить в соответствии со своей задачей (но адреса ячеек не должны пересекаться).



**ПРИМЕЧАНИЕ**

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

**Функция** определяет операцию (чтение/запись) и область памяти, с которой эта операция будет произведена. Ниже приведен список наиболее часто используемых функций:

**Таблица 2.2 – Основные функции протокола Modbus**

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





**ПРИМЕЧАНИЕ**

Нельзя смешивать понятия области памяти и функции. У начинающих пользователей часто возникают проблемы при работе с **input** и **holding** регистрами, поскольку **область памяти holding** регистров имеет обозначение **4x**, а **функция чтения** holding регистров – **0x03** (может интуитивно показаться, что идентификатор области памяти и код функции должны совпадать – но на практике это не так).

Ниже приведен фрагмент карты регистров для модуля аналогового ввода [ОВЕН MB110-8A](#). В ней для каждого параметра указан адрес и тип данных (тип данных определяет число ячеек памяти, занимаемых параметром). В таблице не упомянуто, в какой области памяти расположены параметры – но в примечании указано, что обращаться к ним необходимо функциями **0x03** и **0x04** – значит области памяти **4x** и **3x** в устройстве наложены друг на друга.

**Окончание таблицы В.4**

Параметр	Тип	Адрес регистра	
		(hex)	(dec)
Положение десятичной точки в целом значении для входа 2 (знач. DP)	int16	0006	6
Целое значение измерение входа 2 со смещением точки	int16	0007	7
Статус измерения входа 2 (код исключительной ситуации)	int16	0008	8
Циклическое время измерения входа 2	int16	0009	9
Измерение входа 2 в представлении с плавающей точкой	Float32	000A,000B	10,11
...			
Положение десятичной точки в целом значении для входа 8 (знач. DP)	int16	002A	42
Целое значение измерение входа 8 со смещением точки	int16	002B	43
Статус измерения входа 8 (код исключительной ситуации)	int16	002C	44
Циклическое время измерения входа 8	int16	002D	45
Измерение входа 8 в представлении с плавающей точкой	Float32	002E,002F	46,47

**Примечания**  
 1 Все регистры только на чтение, чтение регистров осуществляется командами 03 или 04 (прибор поддерживает обе команды).  
 2 При передаче 4-х байтных значений (тип Float 32) старшее слово передается в регистре с меньшим номером.

**Рисунок 2.1 – Фрагмент карты регистров модуля MB110-8A**

В различных документах идентичные обозначения могут иметь разный смысл в зависимости от контекста. Например, префикс **0x** часто используют как указание на шестнадцатеричную систему счисления, поэтому в одном случае **0x30** может обозначать «30-й бит области памяти **coils**», а в другом – «адрес 30 в шестнадцатеричной (HEX) системе счисления» (такой адрес может относиться к любой области памяти).

Другой пример необходимости уточнения контекста – принцип адресации регистров. В некоторых случаях в адреса битов/регистров закладывается префикс области памяти, в которых они находятся, например – **30101** (цифра **3** указывает на **input регистры**), **40202** (цифра **4** указывает на **holding регистры**). Обычно подразумевается, что адрес **30001** соответствует нулевому input регистру, а **40001** – нулевому holding регистру. То есть при опросе упомянутых регистров (**30101** и **40202**) в настройках master-устройства следует указать **input регистр** с адресом **100** и **holding регистр** с адресом **201**.

В то же время существуют устройства, для которых адрес **40202** может являться адресом любой области памяти (например, **input регистр** номер **40202**).

Система обозначений для адресов битов/регистров slave-устройств зависит от конкретного производителя (в некоторых случаях – даже от конкретного документа), в связи с чем от пользователя требуется четкое понимание контекста используемых обозначений и повышенное внимание к примечаниям и сноскам.

Ниже приведен пример различных вариантов обозначений для **holding регистра** с адресом **39**:

- регистр **4x39**
- регистр **39**, чтение осуществляется функцией **03**
- регистр **0x27**, чтение осуществляется функцией **0x03**
- регистр **40040**

Запрос к slave-устройству может быть **одиночным** или **групповым**. В случае **одиночных запросов** master-устройство считывает каждый из параметров slave-устройства отдельной командой. В случае **группового опроса** master-устройство считывает одной командой сразу несколько параметров, адреса которых в карте регистров расположены строго последовательно и не имеют разрывов. Групповой опрос позволяет уменьшить трафик в сети и время, затрачиваемое на опрос устройства, но в некоторых случаях его применение невозможно (или возможно с ограничениями) из-за индивидуальных особенностей устройства.

## 2.3 Нумерация COM-портов в CODESYS

Во время настройки интерфейсов RS-232/RS-485 в **CODESYS** следует указывать номера портов. Для контроллеров OWEN эти номера приведены в таргет-файле устройства в узле **Device** на вкладке **Информация**:

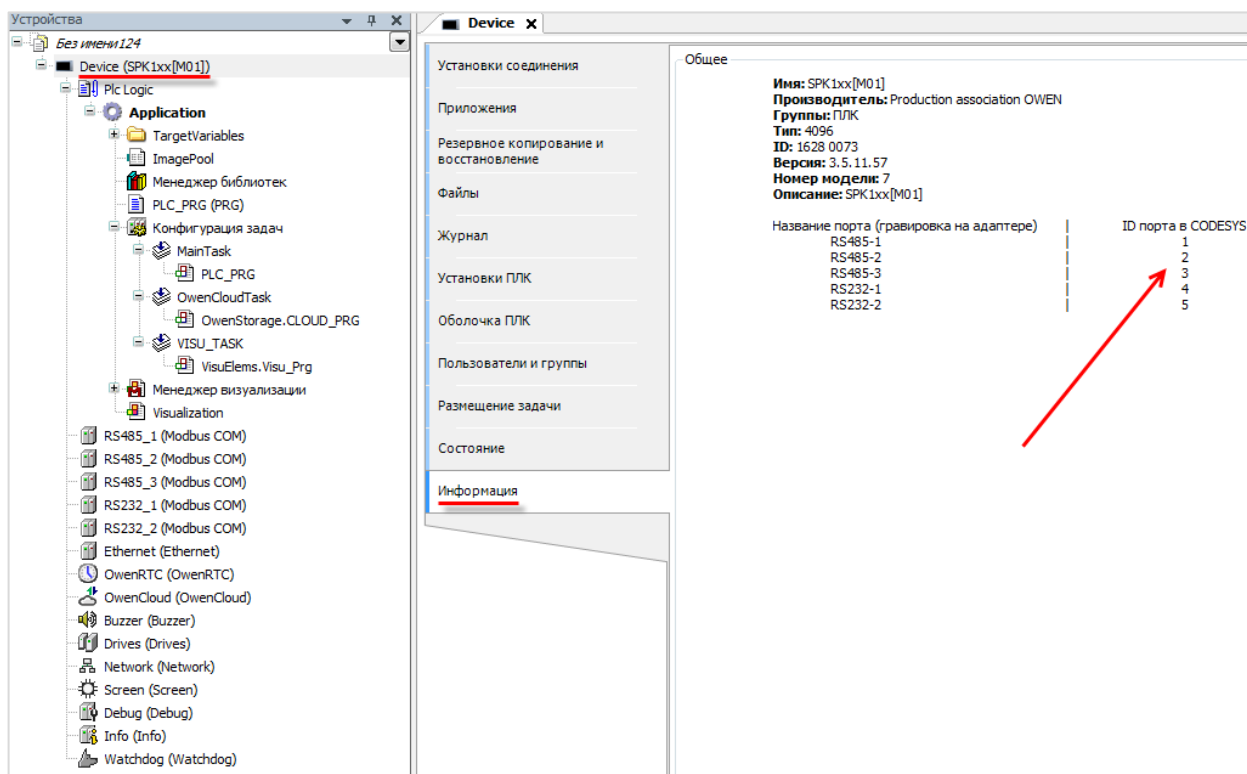


Рисунок 2.2 – Информация о нумерации COM-портов в таргет-файле

## 2.4 Особенности работы с модулями Mx110

Перед тем, как подключать модули Mx110 к контроллеру, их следует сконфигурировать с помощью программы **Конфигуратор Mx110**. Программа доступна на сайте [ОВЕН](#) на странице любого из модулей.

Для подключения к модулю следует указать его сетевые настройки. Если настройки неизвестны, то необходимо сбросить настройки на заводские (процесс сброса описан в руководстве по эксплуатации на модуль) и подключиться с помощью кнопки **Заводские сетевые настройки**.

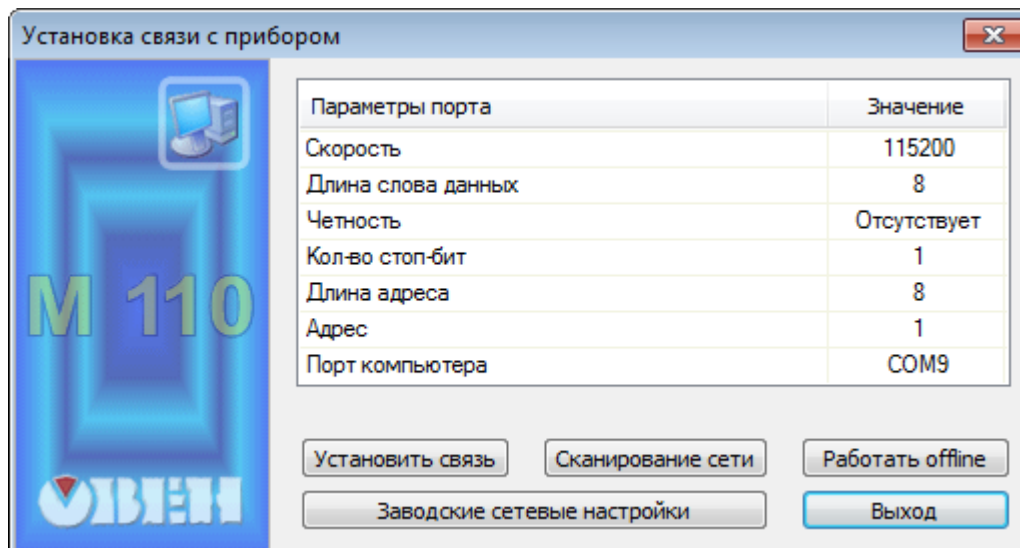


Рисунок 2.3 – Подключение к модулю Mx110 с помощью программы Конфигуратор Mx110

В конфигураторе задаются сетевые настройки модулей и параметры входов/выходов.



### ПРИМЕЧАНИЕ

В пределах одной сети все модули должны иметь одинаковые сетевые настройки, за исключением адресов.

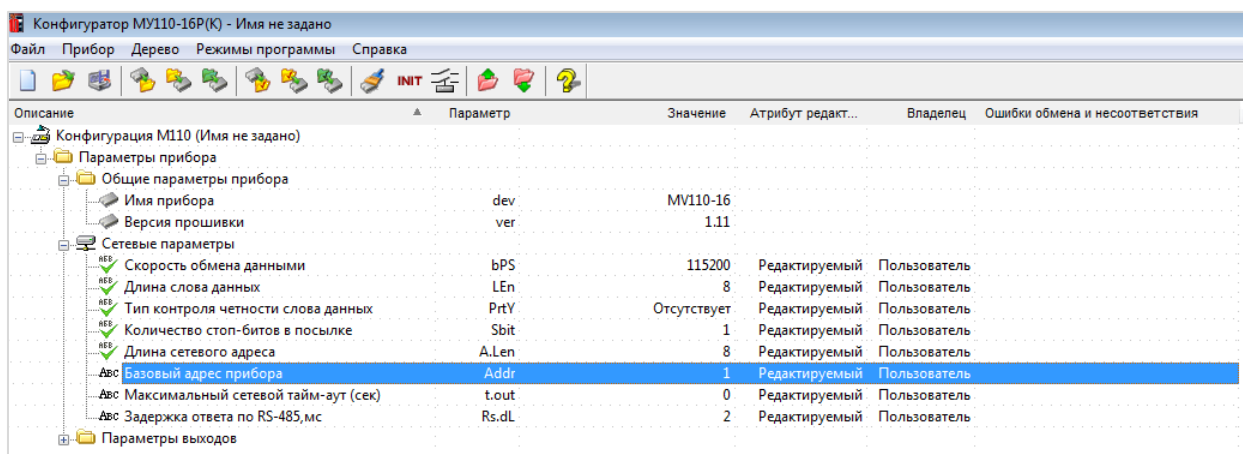


Рисунок 2.4 – Настройка модуля с помощью конфигуратора

Конфигурирование модулей происходит по протоколу **ОВЕН**. В связи с особенностями протокола во время конфигурирования каждый модуль занимает количество адресов, равное количеству его каналов. То есть в случае попытке настраивать модули, уже находящиеся в сети и имеющие последовательные адреса (1, 2, 3), могут возникнуть ошибки. Если предполагается, что в будущем

может потребоваться перенастройка модулей, то следует изначально задавать адреса модулей с промежутками, равными числу каналов в модулях.

Например, для связки МВ110-8А – МВ110-16Д – МВ110-16Р можно выбрать адреса 1 – 9 – 25.

После изменения настроек модуля через **Конфигуратор Мх110** следует перезагрузить его по питанию.

### 3 Шаблоны модулей Mx110 и Mx210

#### 3.1 Установка шаблонов модулей в среду CODESYS

Шаблоны представляют собой уже сконфигурированные slave-устройства с фиксированными картами регистров. Их настройка крайне проста и сводится только к выбору адреса модуля и привязки переменных к нужным каналам. Компания OWEN предоставляет шаблоны для модулей [Mx110](#), преобразователей частоты [ПЧВ](#) (шаблоны ПЧВ входят в состав пакета шаблонов Mx110) и модулей Mx210.

Для работы с шаблонами требуется установить в среду программирования соответствующий пакет. В настоящем руководстве описывается работа с шаблонами версии **3.5.11.4**. Работа с более старыми версиями шаблонов описана в предыдущих версиях руководства.

Пакеты доступны на сайте компании [OWEN](#) в разделе [CODESYS V3/Библиотеки и компоненты](#).

Для установки пакета в **CODESYS** в меню **Инструменты** следует выбрать пункт **Менеджер пакетов**, после чего указать путь к файлу пакета и нажать **Установить**.

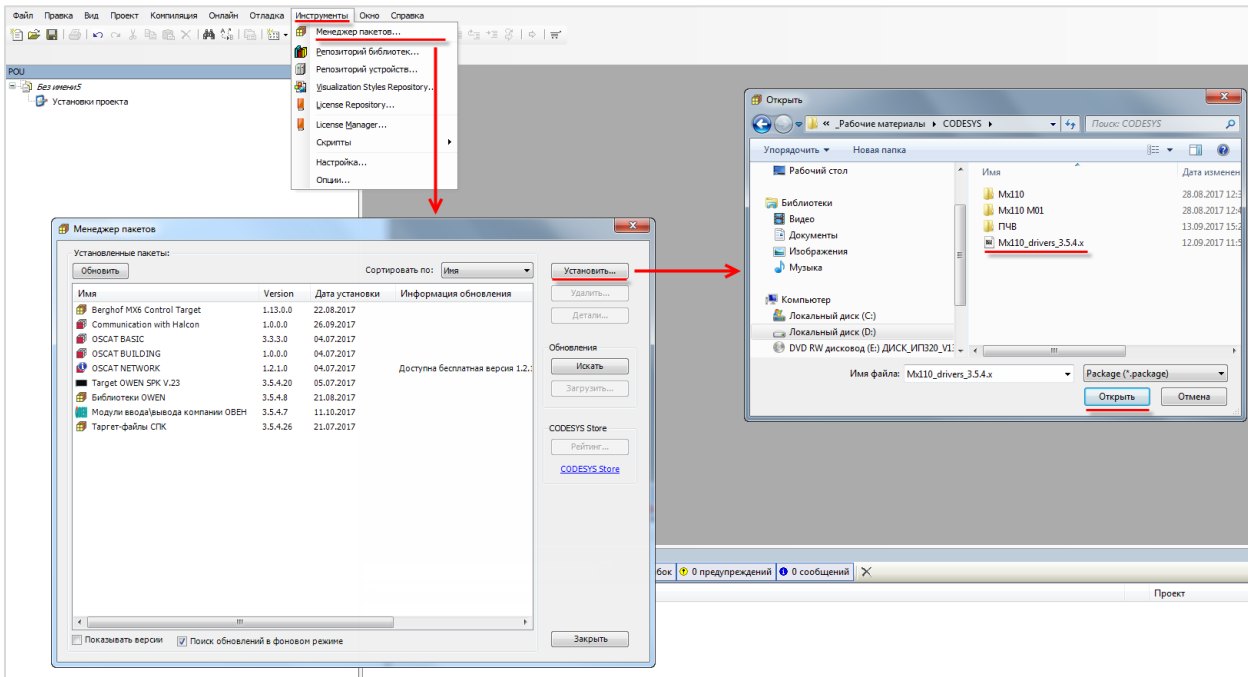


Рисунок 3.1.1 – Установка пакета шаблонов в среду CODESYS

В появившемся диалоговом окне следует выбрать пункт **Полная установка**, после чего нажать кнопку **Next**:

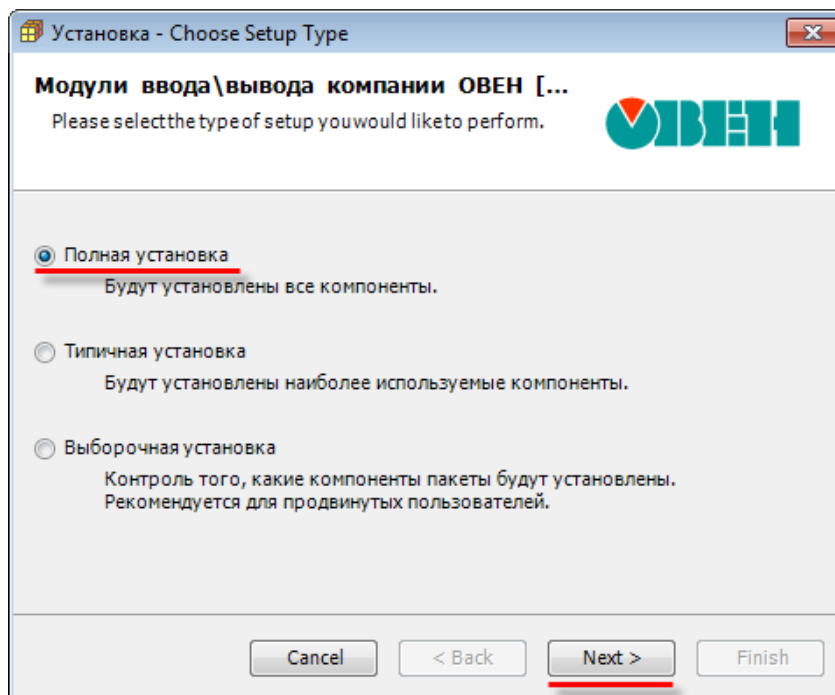


Рисунок 3.1.2 – Начало установки шаблонов модулей

После завершения установки следует закрыть диалоговое окно с помощью кнопки **Finish**:

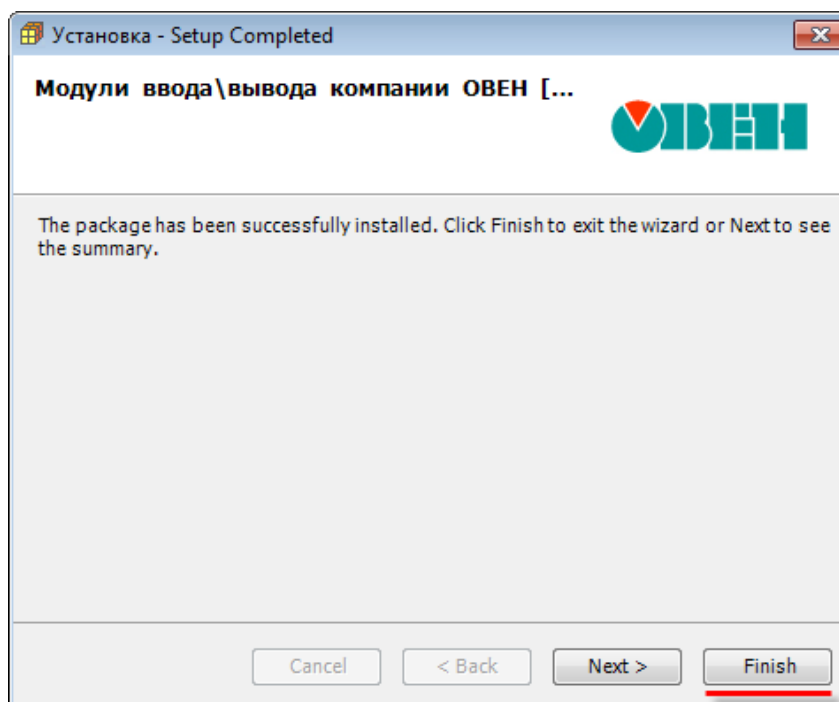


Рисунок 3.1.3 – Завершение установки шаблонов модулей

### 3.2 Пример: СПК1хх [M01] + модули Mx110

В качестве примера будет рассмотрена настройка обмена с модулями [Mx110](#) (MB110-8A, MB110-16Д, МУ110-16P) с использованием **шаблонов**. В примере используются шаблоны версии **3.5.11.4**.

**Реализуемый алгоритм:** если значение первого аналогового входа модуля **MB110-8A** превышает **30** и при этом первый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **МУ110-16P** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).

Структурная схема примера приведена на рисунке ниже:

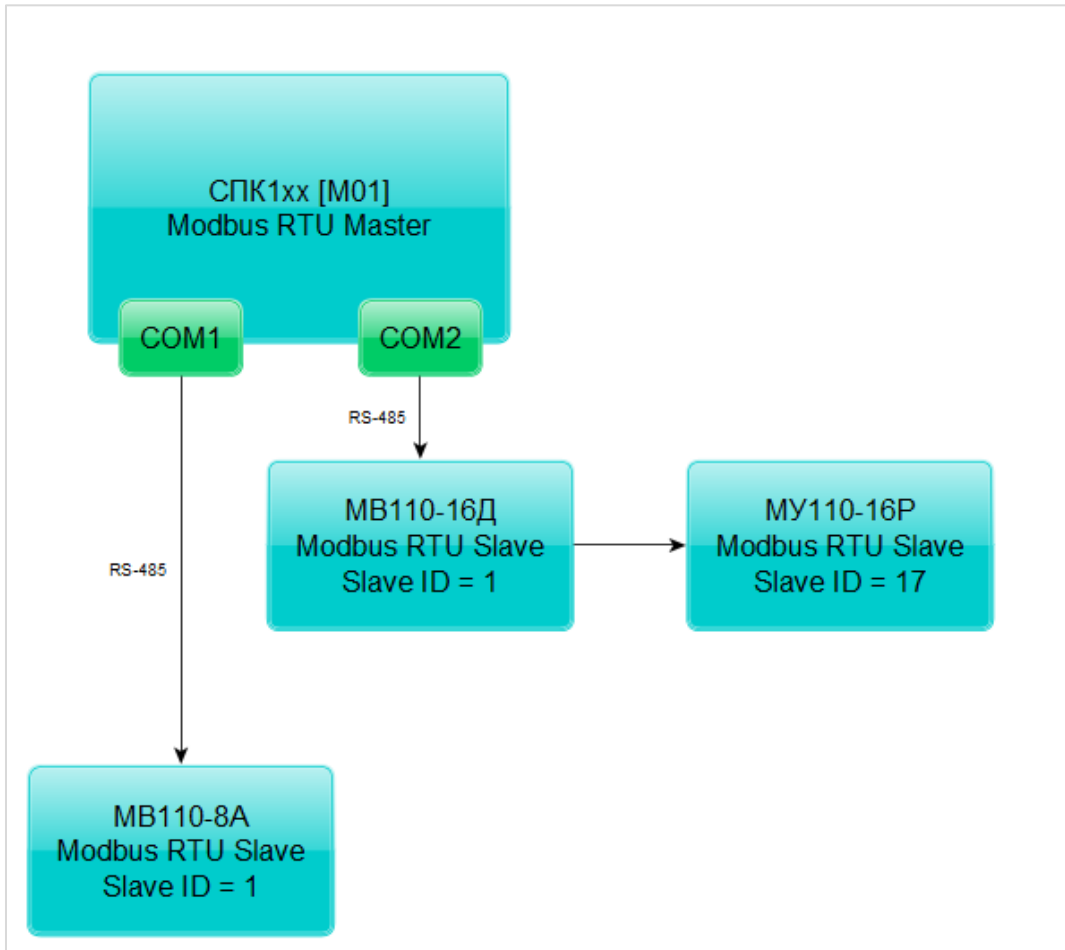


Рисунок 3.2.1 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел **Device** – **Обновить устройство**).

Пример доступен для скачивания: [Example\\_TemplatesMx110\\_3511v1.projectarchive](https://www.projectarchive.com/Example_TemplatesMx110_3511v1/)

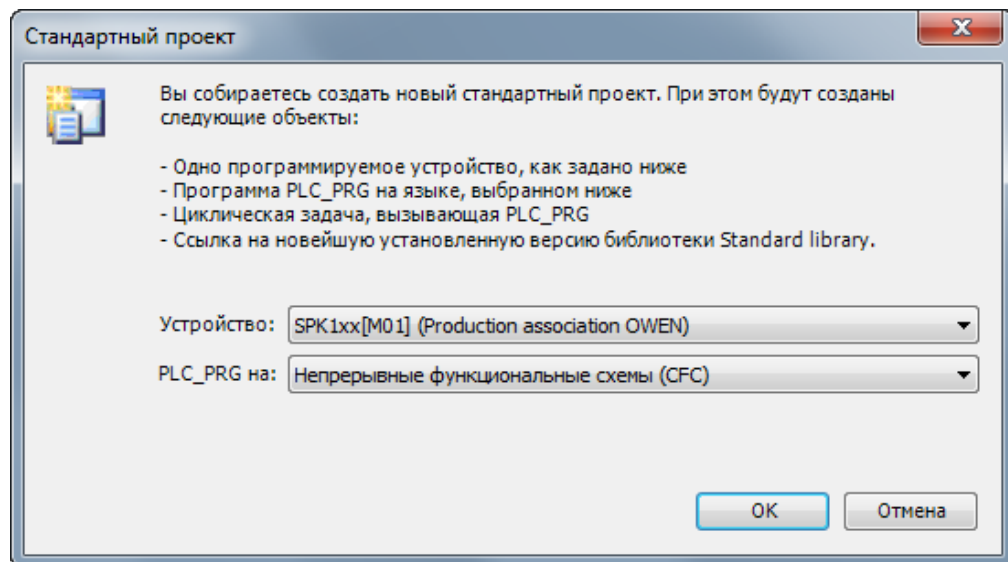
Сетевые параметры модулей приведены в таблице ниже:

**Таблица 3.2.1 – Сетевые параметры модулей Mx110**

Параметр	MB110-8A	MB110-16Д	MU110-16P
COM-порт контроллера, к которому подключен модуль	COM1	COM2	
ID COM-порта	1	2	
Адрес модуля	1	1	17
Скорость обмена	115200		
Количество бит данных	8		
Контроль четности	Отсутствует		
Количество стоп-бит	1		

Для настройки обмена следует:

1. Настроить модули **Mx110** с помощью программы **Конфигуратор Mx110** в соответствии с таблицей 3.2.1. Подключить модули к COM-портам контроллера в соответствии с [рисунком 3.2.1](#).
2. Установить пакет шаблонов модулей **Mx110** в CODESYS (см. [п. 3.1.1](#)).
3. Создать новый проект **CODESYS** с программой **PLC\_PRG** на языке **CFC**:



**Рисунок 3.2.2 – Создание проекта CODESYS**

4. Объявить в программе следующие переменные:

```

1  PROGRAM PLC_PRG
2  VAR
3      rAnalogInput1:    REAL;    // 1-й вход модуля MB110-8A
4      xDiscreteInput1:  BOOL;    // 1-й вход модуля MB110-16Д
5      xDiscreteOutput1: BOOL;    // 1-й выход модуля MU110-16P
6  END_VAR
7

```

**Рисунок 3.2.3 – Объявление переменных в программе PLC\_PRG**

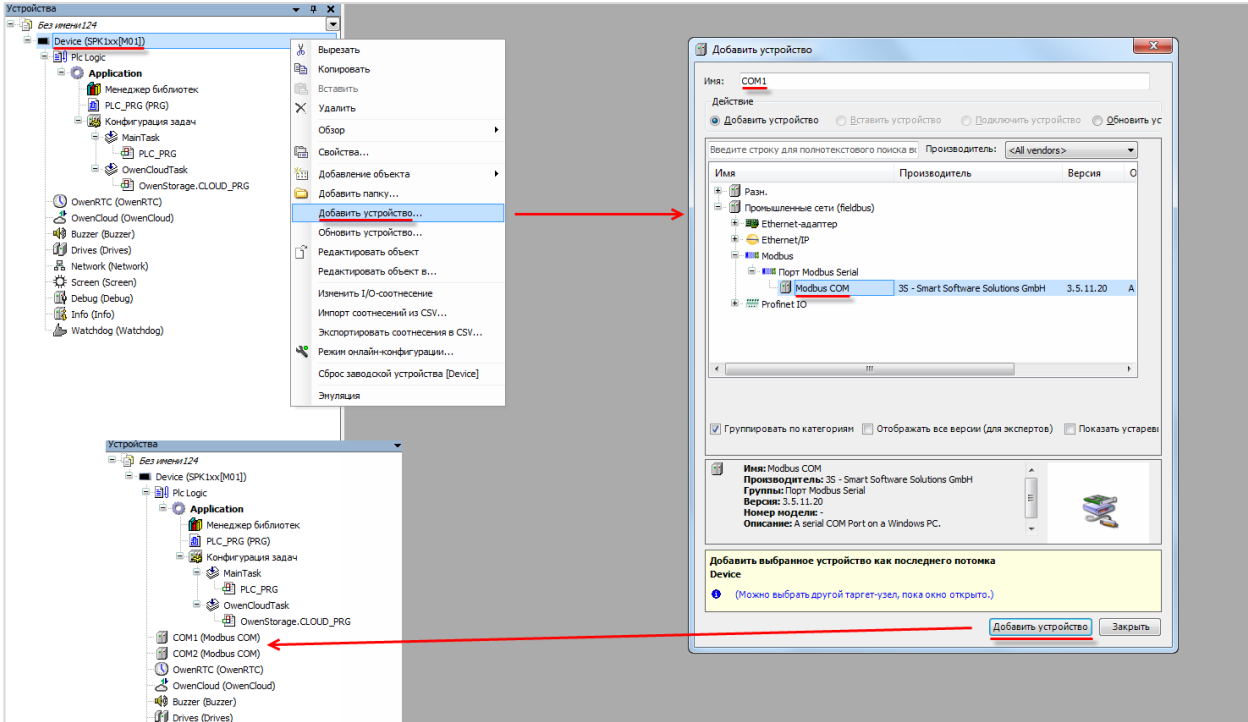


5. Добавить в проект два компонента **Modbus COM** с названиями **COM1** и **COM2**.



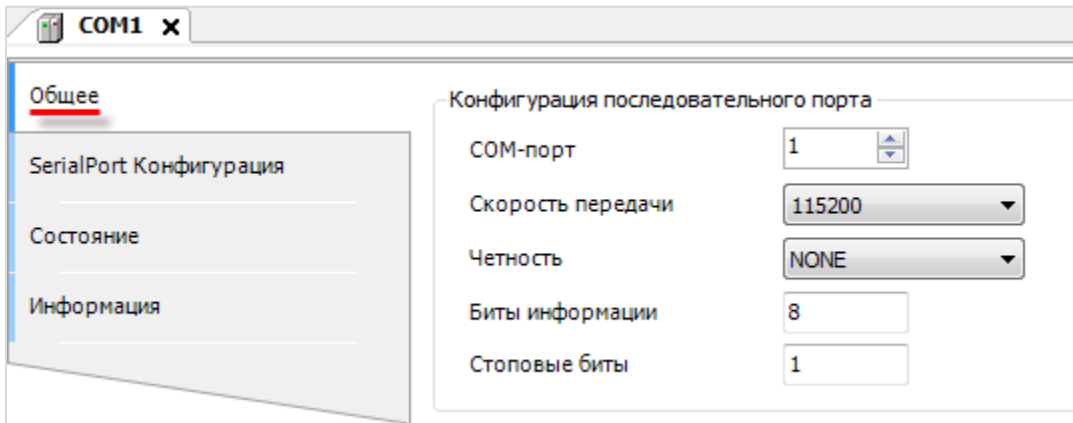
**ПРИМЕЧАНИЕ**

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).



**Рисунок 3.2.4 – Добавление компонента Modbus COM**

В конфигурации COM-портов следует указать [номера COM-портов](#) и сетевые настройки в соответствии с [таблицей 3.2.1](#):



**Рисунок 3.2.5 – Настройки COM-порта COM1**

6. В каждый из COM-портов добавить компонент **Modbus Master**.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

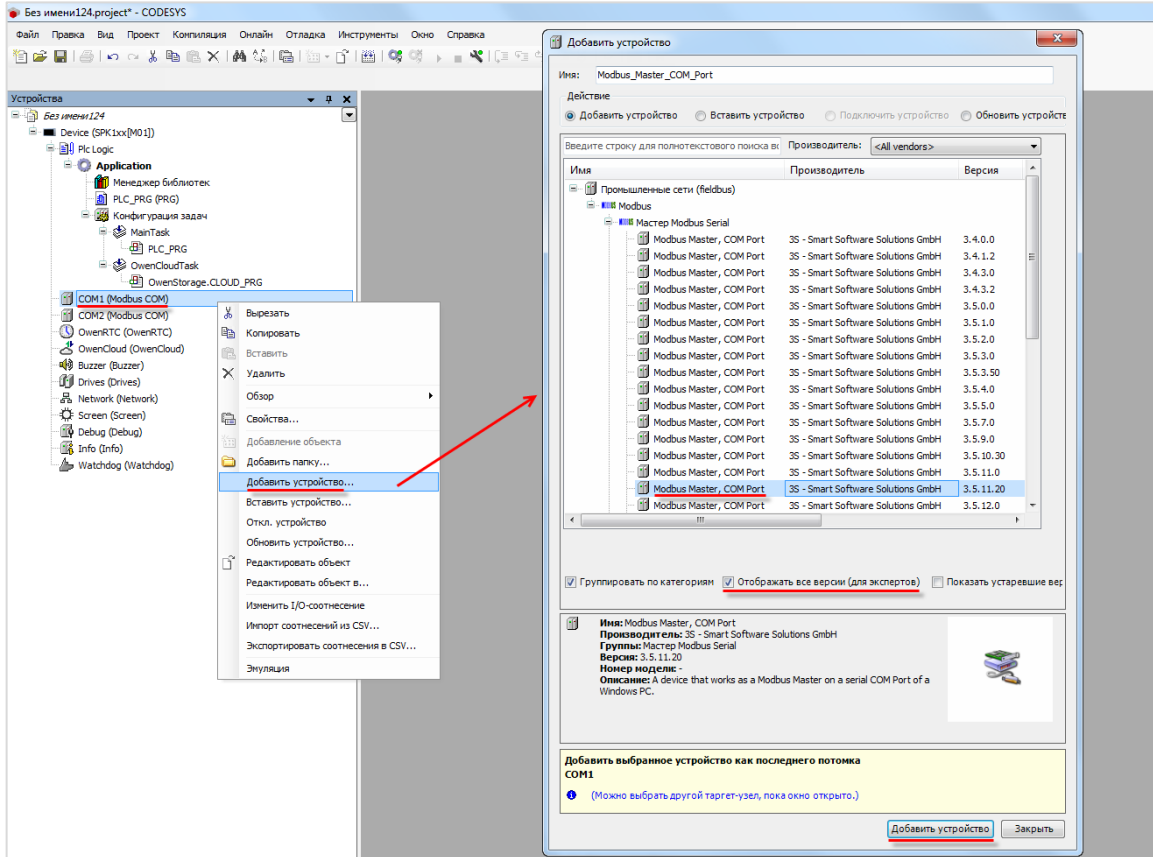


Рисунок 3.2.6 – Добавление компонента Modbus Master

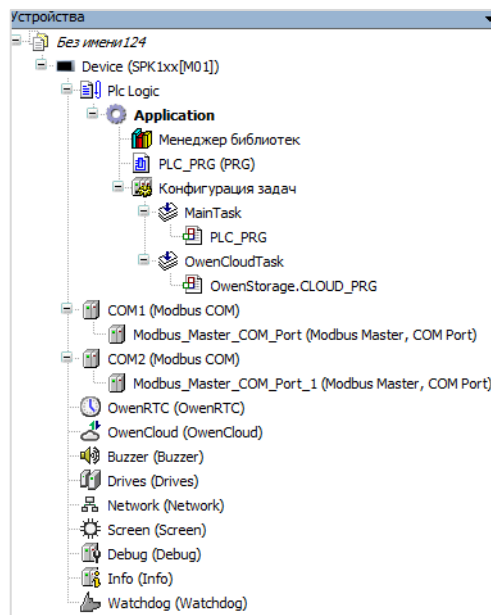


Рисунок 3.2.7 – Внешний вид дерева проекта после добавления Modbus Master

### 3. Шаблоны модулей Mx110 и Mx210

В настройках компонентов на вкладке **Общее** следует установить галочку **Автоперезапуск соединения**. В параметре **Время между фреймами** установить значение **20 мс**.

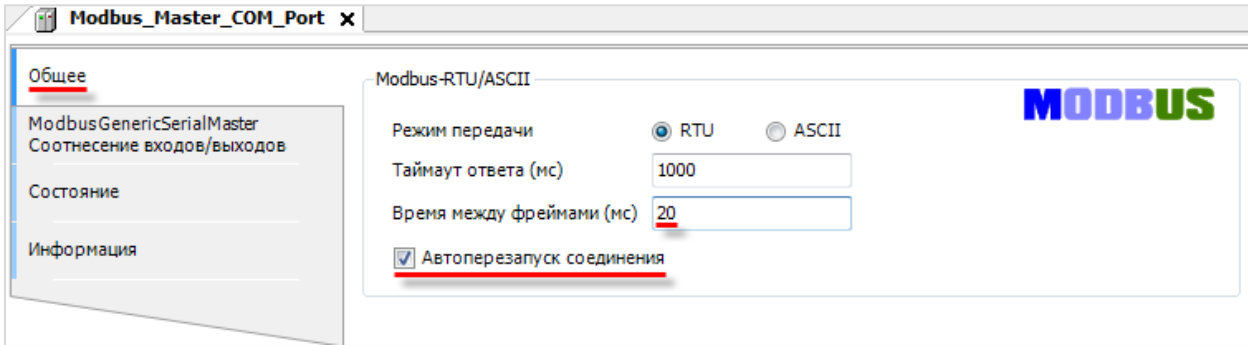


Рисунок 3.2.8 – Настройки компонентов Modbus Master

7. В компонент **Modbus Master** порта **COM1** следует добавить шаблон модуля **MB110-8A**, а в **Modbus Master** порта **COM2** – **MB110-16Д** и **MU110-16P**.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

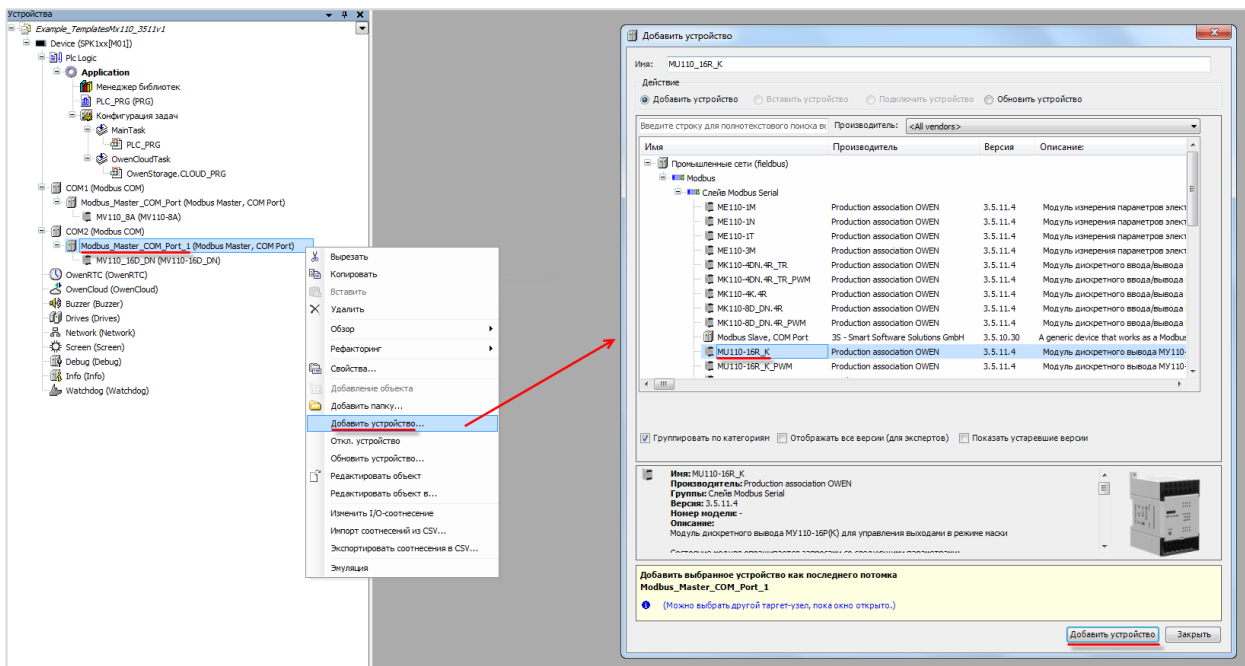


Рисунок 3.2.9 – Добавление шаблонов модулей в проект CODESYS

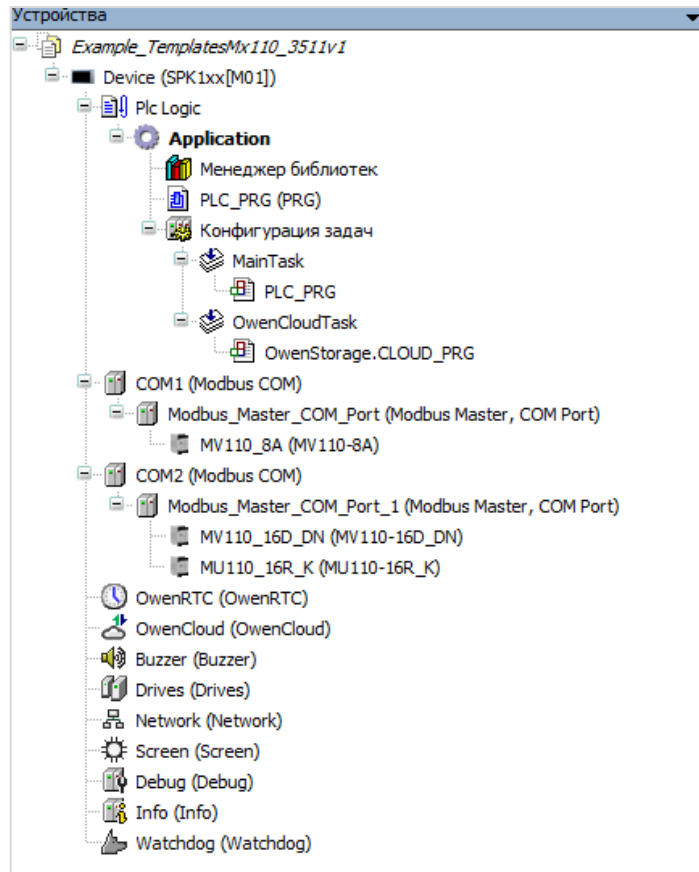


Рисунок 3.2.10 – Внешний вид дерева проекта после добавления шаблонов модулей

В настройках шаблонов следует указать адреса модулей согласно [таблице 3.2.1](#) (MB110-8A – адрес 1, MB110-16Д – адрес 1, МУ110-16Р – адрес 17):

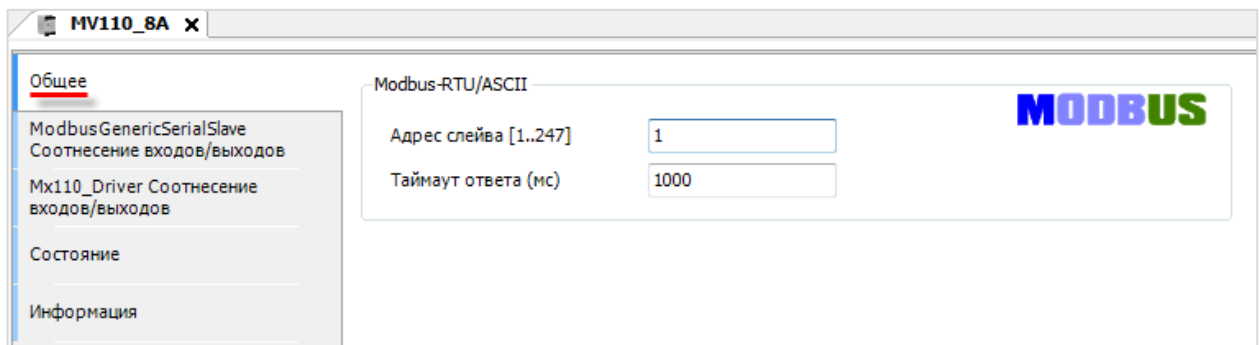


Рисунок 3.2.11 – Настройка шаблона модуля MV110\_8A

8. На вкладке **Mx110\_Driver Соотнесение входов/выходов** привязать переменные программы к каналам шаблонов в соответствии с данной таблицей:

Таблица 3.2.2 – Привязка переменных к каналам шаблонов

Переменная программы	Модуль	Канал
rAnalogInput1	MV110_8A	Вход 1/Измеренное значение
xDiscreteInput1	MV110_16D_DN	Маска входов/Вход 1
xDiscreteOutput1	MU110_16R_K	Маска выходов/Выход 1

### 3. Шаблоны модулей Mx110 и Mx210

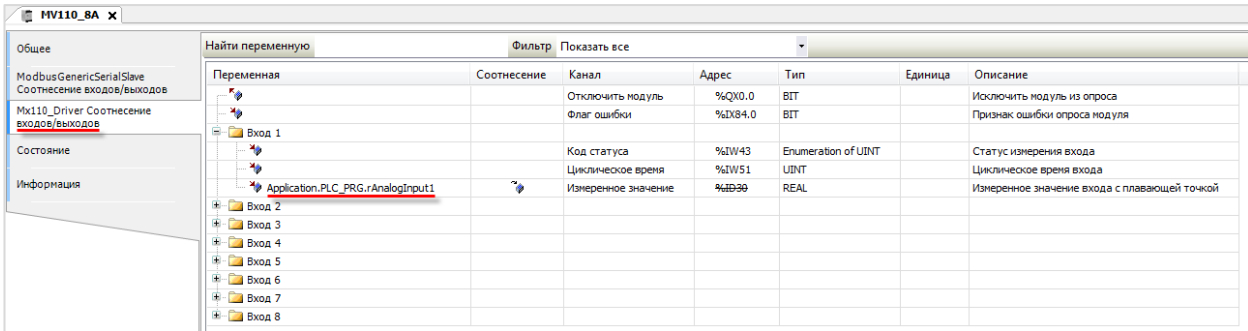


Рисунок 3.2.12 – Привязка переменной к шаблону модуля MB110-8A

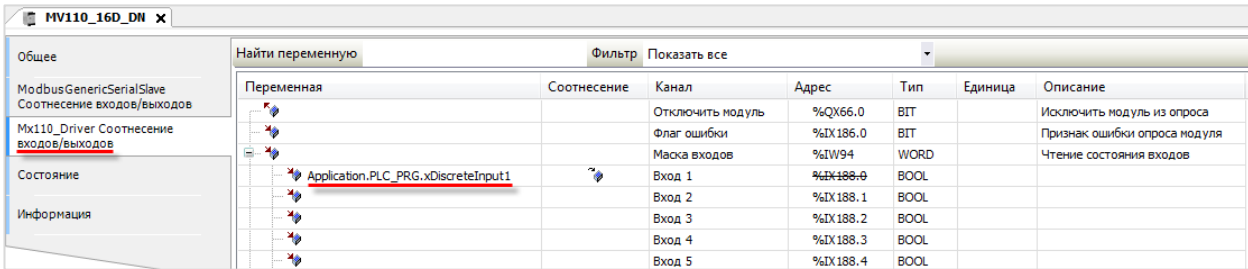


Рисунок 3.2.13 – Привязка переменной к шаблону модуля MB110-16Д

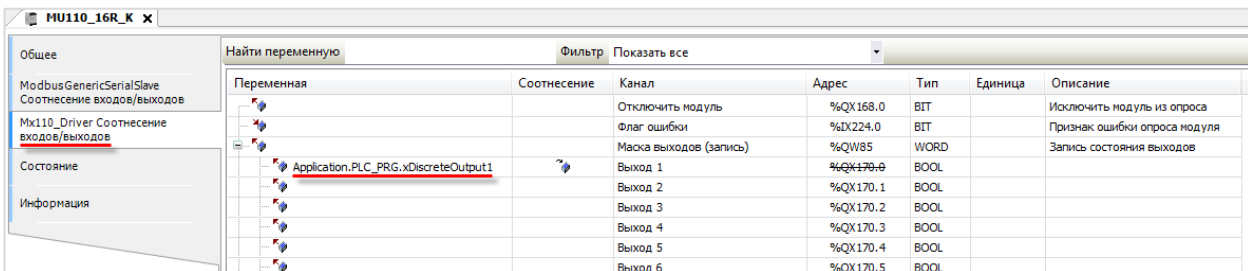


Рисунок 3.2.14 – Привязка переменной к шаблону модуля МУ110-16Р

9. Код программы PLC\_PRG будет выглядеть следующим образом:

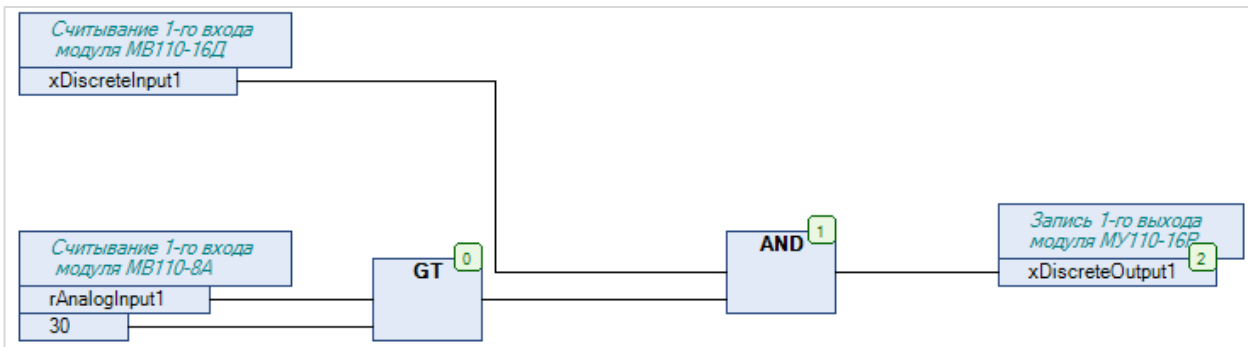


Рисунок 3.2.15 – Код программы на языке CFC

10. Загрузить проект в контроллер и запустить его.

В переменной **rAnalogInput1** будет отображаться текущее значение первого аналогового входа модуля **MB110-8A**. В переменной **xDiscreteInput1** будет отображаться текущее значение первого дискретного входа модуля **MB110-16Д**.

Если значение **rAnalogInput1** превысит **30** и при этом значение **xDiscreteInput1** будет равно **TRUE**, то в переменную **xDiscreteOutput1** будет записано значение **TRUE**, что приведет к замыканию первого дискретного выхода модуля **МУ110-16Р**. Если одно из условий перестанет выполняться, то выход будет разомкнут.

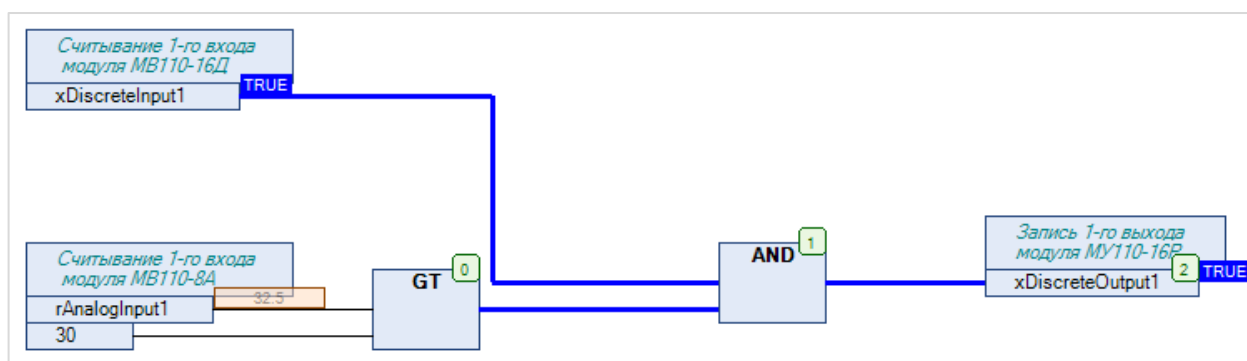


Рисунок 3.2.16 – Выполнение программы в режиме Online

**ПРИМЕЧАНИЕ**

Более подробная информация о настройках компонентов **ModbusCom** и **Modbus Master** приведена в [п. 4.2](#).

### 3.3 Пример: СПК1хх [M01] + модули Mx210

В качестве примера будет рассмотрена настройка обмена с модулями Mx210 (MB210-101 и МК210-301) с использованием **шаблонов**. В примере используются шаблоны версии **3.5.11.4**.

**Реализуемый алгоритм:** если значение первого аналогового входа модуля **MB210-101** превышает **30** и при этом первый дискретный вход модуля **МК210-301** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **МК210-301** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).

Структурная схема примера приведена на рисунке ниже:

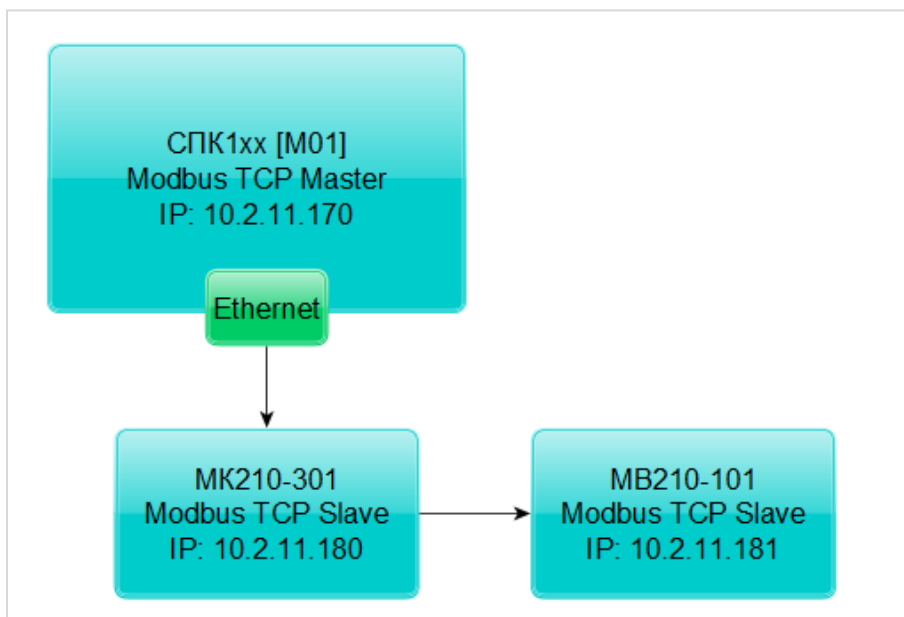


Рисунок 3.3.1 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел **Device** – **Обновить устройство**).

Пример доступен для скачивания: [Example\\_TemplatesMx210\\_3511v1.projectarchive](http://Example_TemplatesMx210_3511v1.projectarchive)

Видеоверсия примера доступна по [ссылке](#).

Сетевые параметры устройств приведены в таблице ниже:

Таблица 3.3.1 – Сетевые параметры устройств

Параметр	СПК1хх [M01]	МК210-301	MB210-101
IP-адрес	10.2.11.170	10.2.11.180	10.2.11.181
Маска подсети	255.255.0.0		
IP-адрес шлюза	10.2.1.1		

Для настройки обмена следует:

1. Настроить модули **Mx210** с помощью программы **ОВЕН Конфигуратор** в соответствии с таблицей 3.3.1 (см. руководство **Mx210. Примеры настройки обмена**). Подключить модули к контроллеру.
2. Установить пакет шаблонов модулей **Mx210** в CODESYS (см. [п. 3.1.1](#)).
3. Создать новый проект **CODESYS** с программой **PLC\_PRG** на языке **CFC**:

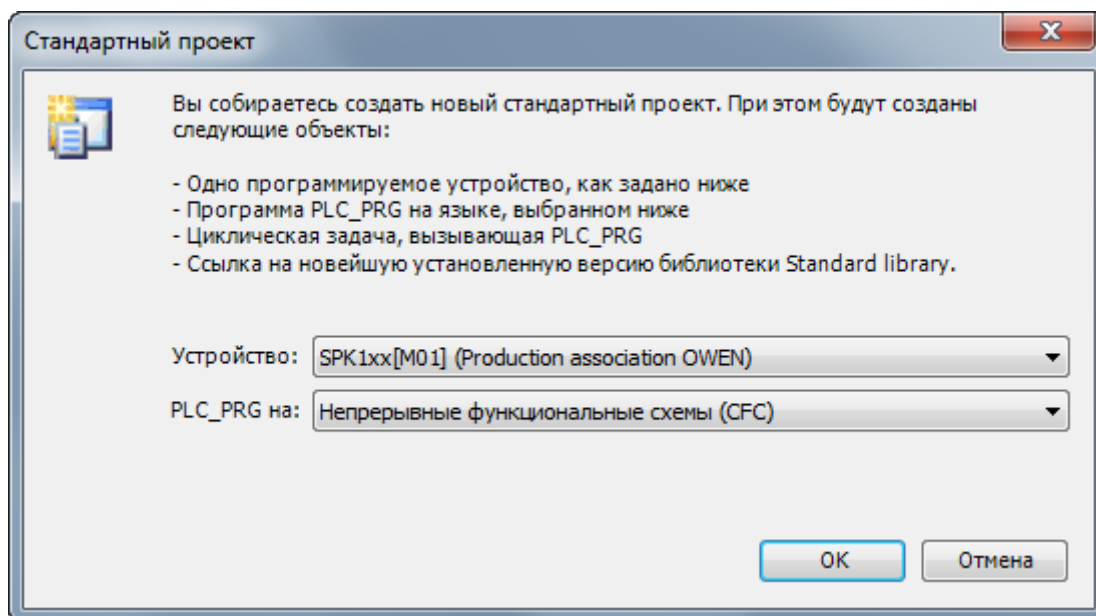


Рисунок 3.3.2 – Создание проекта CODESYS

4. Объявить в программе следующие переменные:

```

1  PROGRAM PLC_PRG
2  VAR
3      rAnalogInput1:    REAL;    // 1-й вход модуля MB210-101
4      xDiscreteInput1:  BOOL;    // 1-й вход модуля MK201-301
5      xDiscreteOutput1: BOOL;    // 1-й выход модуля MK201-301
6  END_VAR

```

Рисунок 3.3.3 – Объявление переменных в программе PLC\_PRG



5. Добавить в проект компонент **Ethernet**.



**ПРИМЕЧАНИЕ**

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

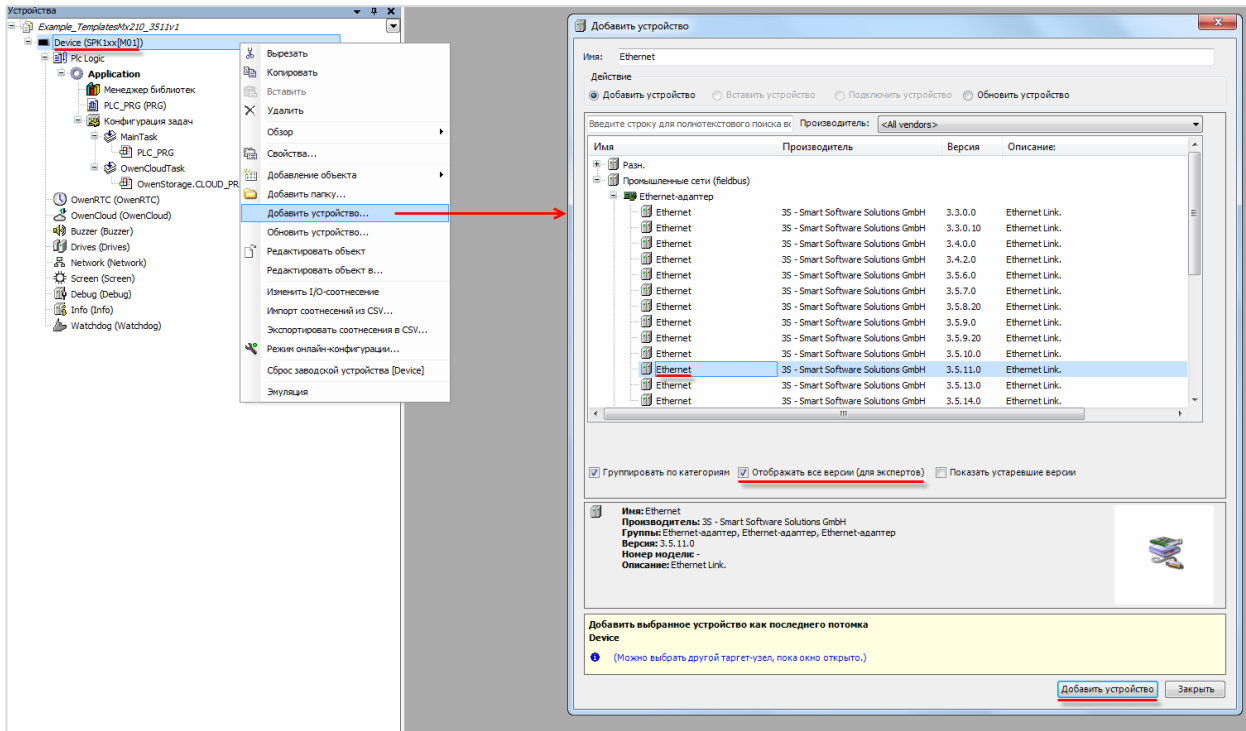


Рисунок 3.3.4 – Добавление компонента Ethernet

Затем следует установить соединение с контроллером, не загружая в него проект (**Device – Установка соединения – Сканировать сеть**) и в компоненте **Ethernet** на вкладке **Конфигурация Ethernet** выбрать нужный интерфейс.

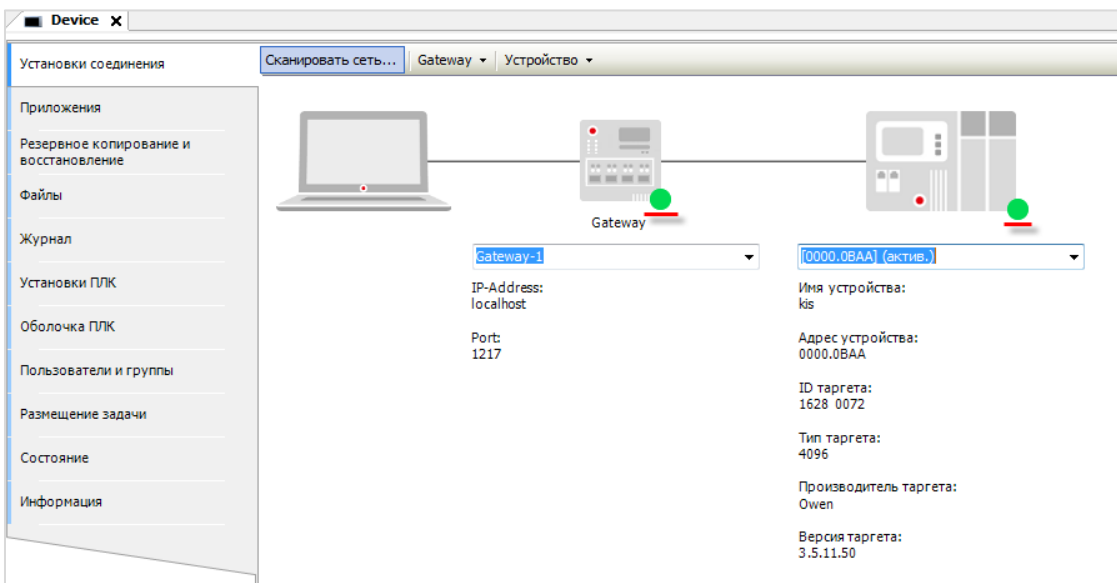


Рисунок 3.3.5 – Подключение к контроллеру

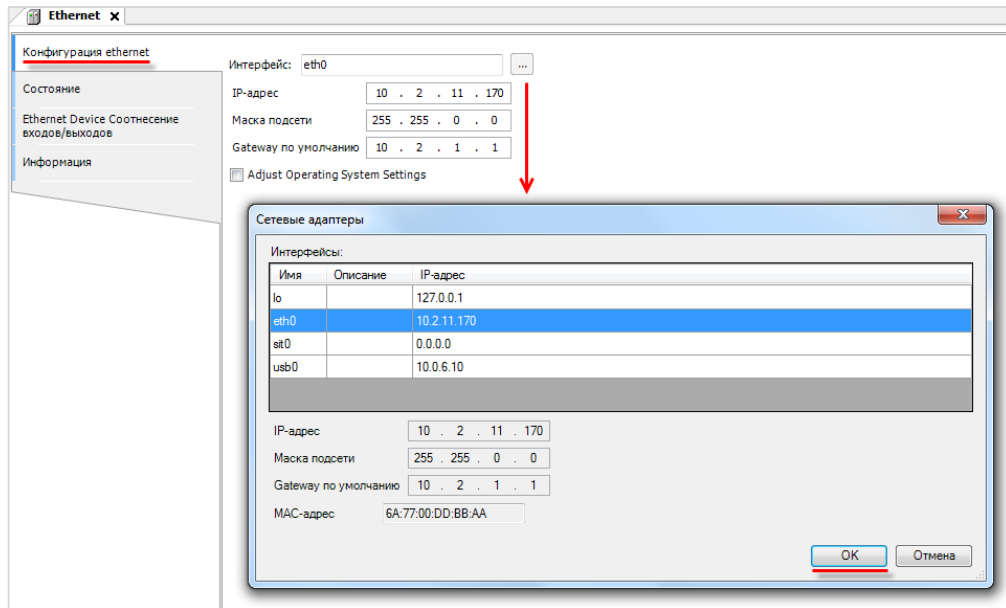


Рисунок 3.3.6 – Выбор используемого интерфейса

**ПРИМЕЧАНИЕ**

Настройки интерфейса задаются в конфигураторе контроллера (см. документ **CODESYS V3.5. FAQ**).

**6. В компонент Ethernet добавить компонент Modbus TCP Master.****ПРИМЕЧАНИЕ**

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

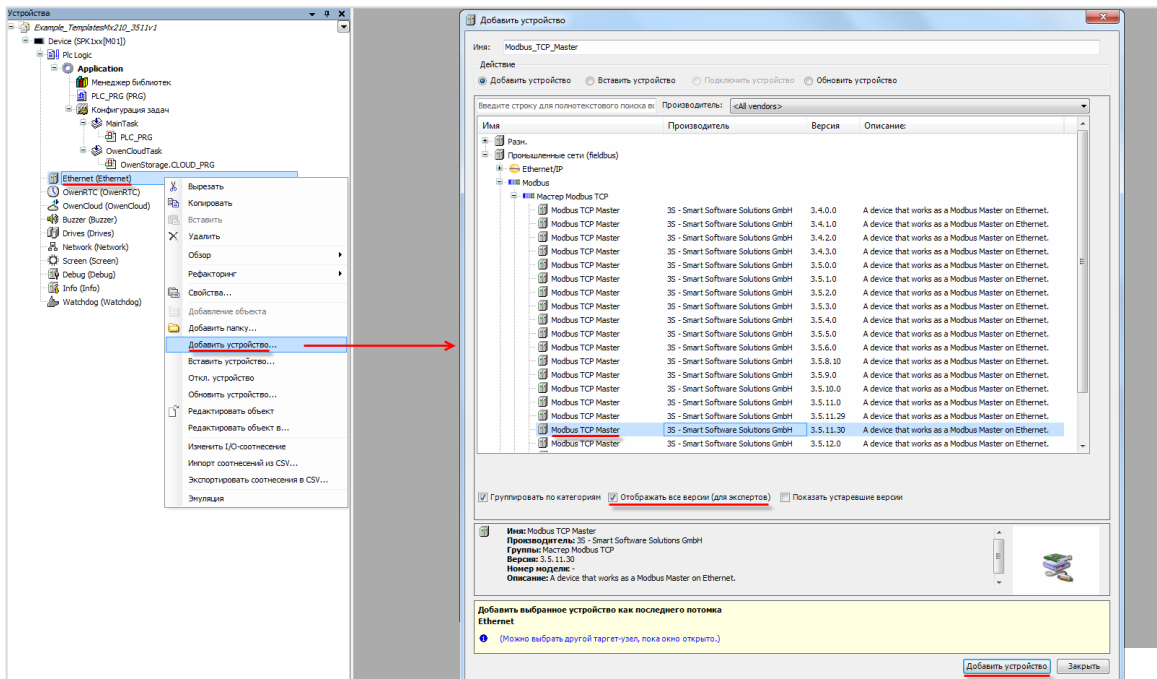


Рисунок 3.3.7 – Добавление компонента Modbus TCP Master

### 3. Шаблоны модулей Mx110 и Mx210

В настройках компонента вкладке **Общее** следует установить галочку **Автоподключение**.

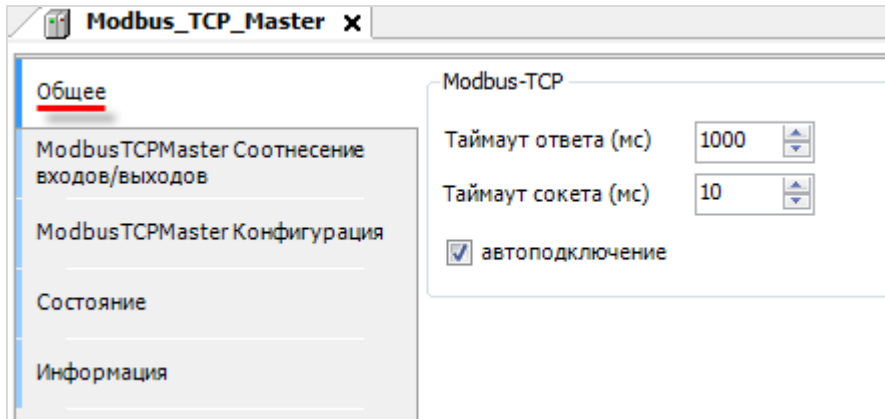


Рисунок 3.3.8 – Настройки компонента Modbus TCP Master

7. В компонент **Modbus TCP Master** следует добавить шаблоны модулей **MK210-301** и **MB210-101**.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

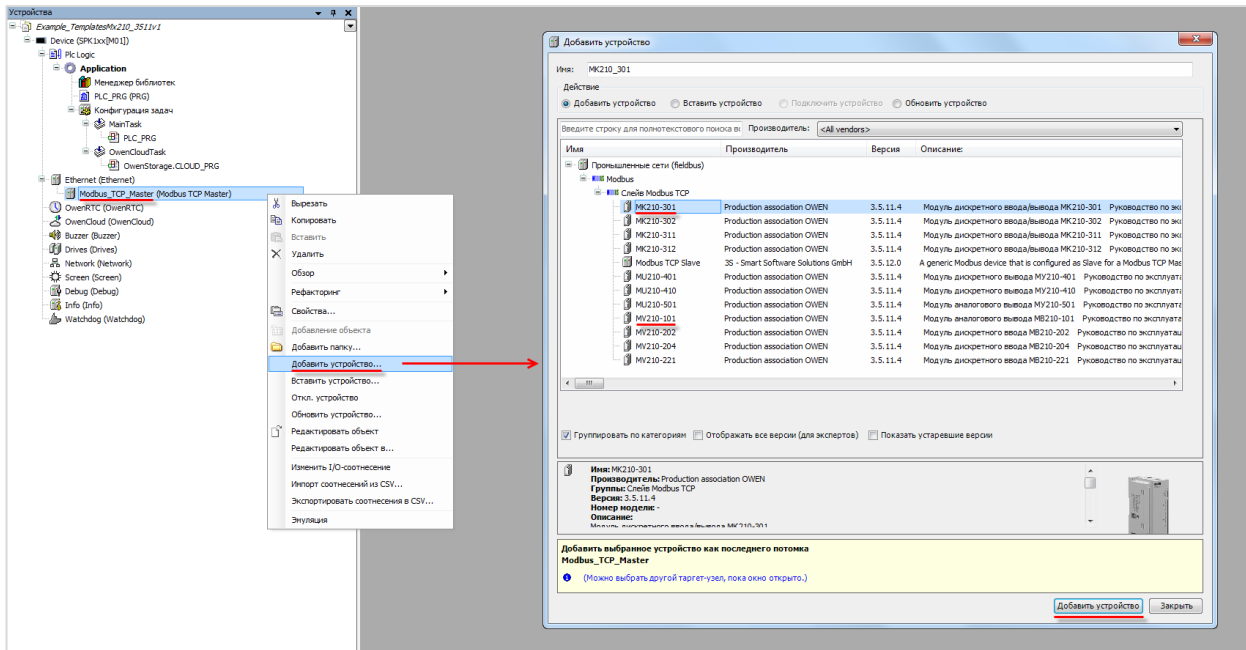


Рисунок 3.3.9 – Добавление шаблонов модулей в проект CODESYS

В настройках модулей следует указать их IP-адреса согласно [таблице 3.3.1](#) (MK210-301 – **10.2.11.180**, MB210-101 – **10.2.11.181**).

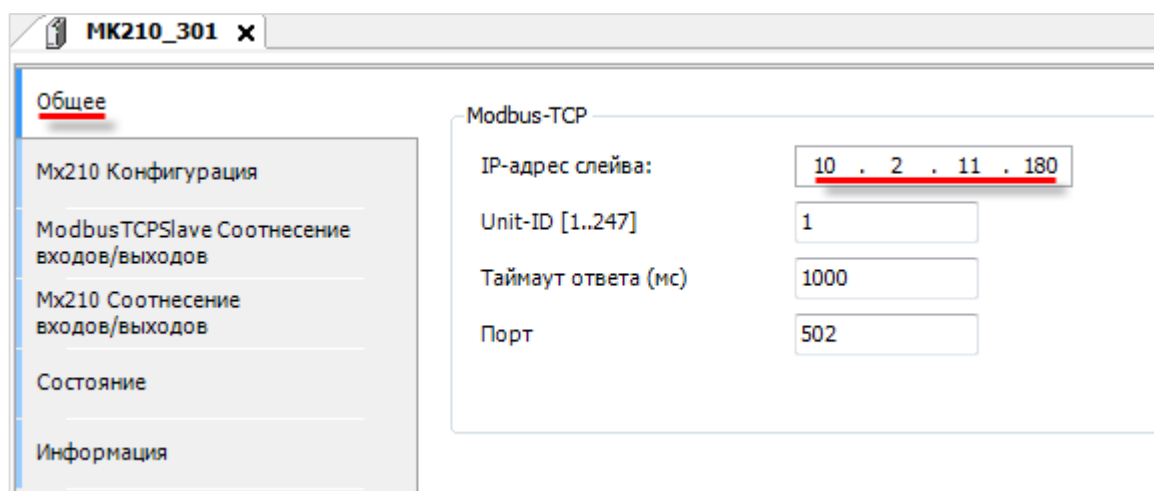


Рисунок 3.3.10 – Настройка шаблона модуля MK210\_301

8. На вкладке **Mx210 Соотнесение входов/выходов** привязать переменные программы к каналам шаблонов в соответствии с данной таблицей:

Таблица 3.3.2 – Привязка переменных к каналам шаблонов

Переменная программы	Модуль	Канал
rAnalogInput1	MV210_101	Входы/Вход 1/Измеренное значение
xDiscreteInput1	MK210_301	Входы/Битовая маска входов/Вход 1
xDiscreteOutput1		Выходы/Битовая маска выходов (запись)/Выход 1

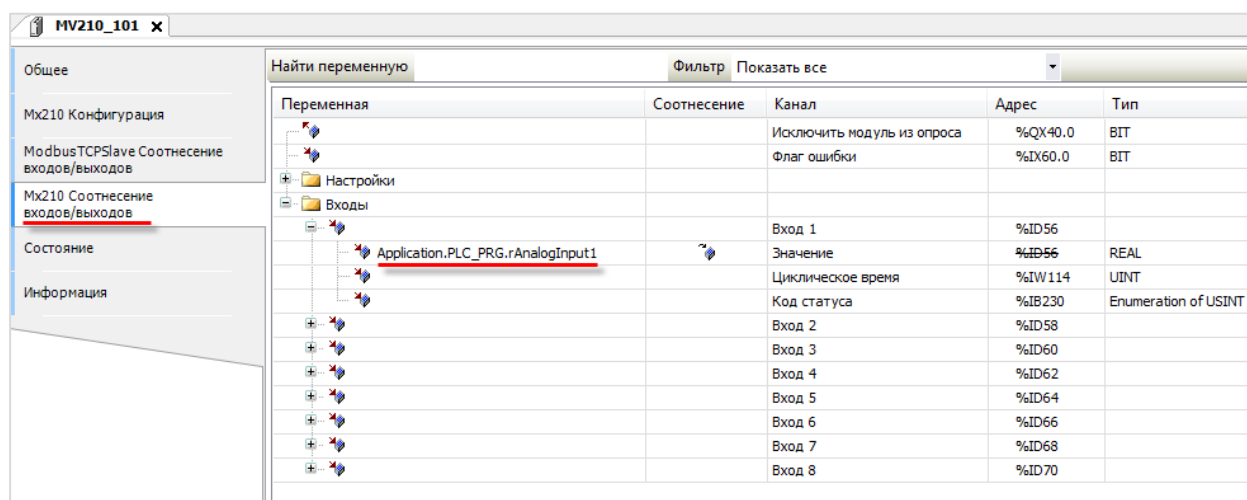


Рисунок 3.3.11 – Привязка переменной к шаблону модуля MV210-101

**ПРИМЕЧАНИЕ**

Конфигурационные параметры модулей доступны для изменения на вкладке **Mx210 Конфигурация**. В частности, на этой вкладке задается минимальный период опроса модулей и режим записи выходов (для модулей с выходами) – циклически или по изменению.

### 3. Шаблоны модулей Mx110 и Mx210

Переменная	Соотнесение	Канал	Адрес	Тип	Единица
		Исключить модуль из опроса	%QX2.0	BIT	
		Флаг ошибки	%IX0.0	BIT	
Входы					
Application.PLC_PRG.xDiscreteInput1		Битовая маска входов	%IB1	BYTE	
		Вход 1	%IX1.0	BOOL	
		Вход 2	%IX1.1	BOOL	
		Вход 3	%IX1.2	BOOL	
		Вход 4	%IX1.3	BOOL	
		Вход 5	%IX1.4	BOOL	
		Вход 6	%IX1.5	BOOL	
Выходы					
		Битовая маска выходов (чтение)	%QB2	BYTE	
		Битовая маска выходов (запись)	%QB3	BYTE	
Application.PLC_PRG.xDiscreteOutput1		Выход 1	%QX3.0	BOOL	
		Выход 2	%QX3.1	BOOL	
		Выход 3	%QX3.2	BOOL	
		Выход 4	%QX3.3	BOOL	
		Выход 5	%QX3.4	BOOL	
		Выход 6	%QX3.5	BOOL	
		Выход 7	%QX3.6	BOOL	
		Выход 8	%QX3.7	BOOL	

Рисунок 3.3.12 – Привязка переменной к шаблону модуля MK210-301

9. Код программы PLC\_PRG будет выглядеть следующим образом:

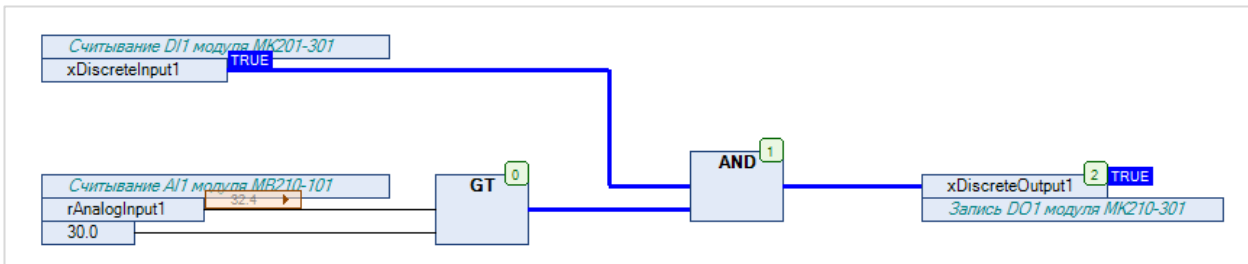


Рисунок 3.3.13 – Код программы на языке CFC

10. Загрузить проект в контроллер и запустить его.

В переменной **rAnalogInput1** будет отображаться текущее значение первого аналогового входа модуля **MB210-101**. В переменной **xDiscreteInput1** будет отображаться текущее значение первого дискретного входа модуля **MK210-301**.

Если значение **rAnalogInput1** превысит 30 и при этом значение **xDiscreteInput1** будет равно **TRUE**, то в переменную **xDiscreteOutput1** будет записано значение **TRUE**, что приведет к замыканию первого дискретного выхода модуля **MK210-301**. Если одно из условий перестанет выполняться, то выход будет разомкнут.



**ПРИМЕЧАНИЕ**

Более подробная информация о настройках компонентов **Ethernet** и **Modbus TCP Master** приведена в [п. 4.4](#).

### 3.4 Диагностика и управление обменом

Шаблон каждого модуля содержит каналы **Флаг ошибки** и **Исключить модуль из опроса**.

Канал **Флаг ошибки** принимает значение **TRUE** в случае ошибки обмена (например, ответ от модуля не пришел или из-за действия помех на линию связи пришел некорректный ответ).

Канал **Исключить модуль из опроса** позволяет остановить опрос модуля: пока канал имеет значение **TRUE**, то модуль не опрашивается.

Шаблоны модулей **Mx210** включают в себя переменные диагностики. Для их использования следует в нужном месте программы ввести имя модуля из дерева проекта с постфиксом **\_OwenDriver**, поставить точку и из выпадающего списка выбрать нужную переменную:

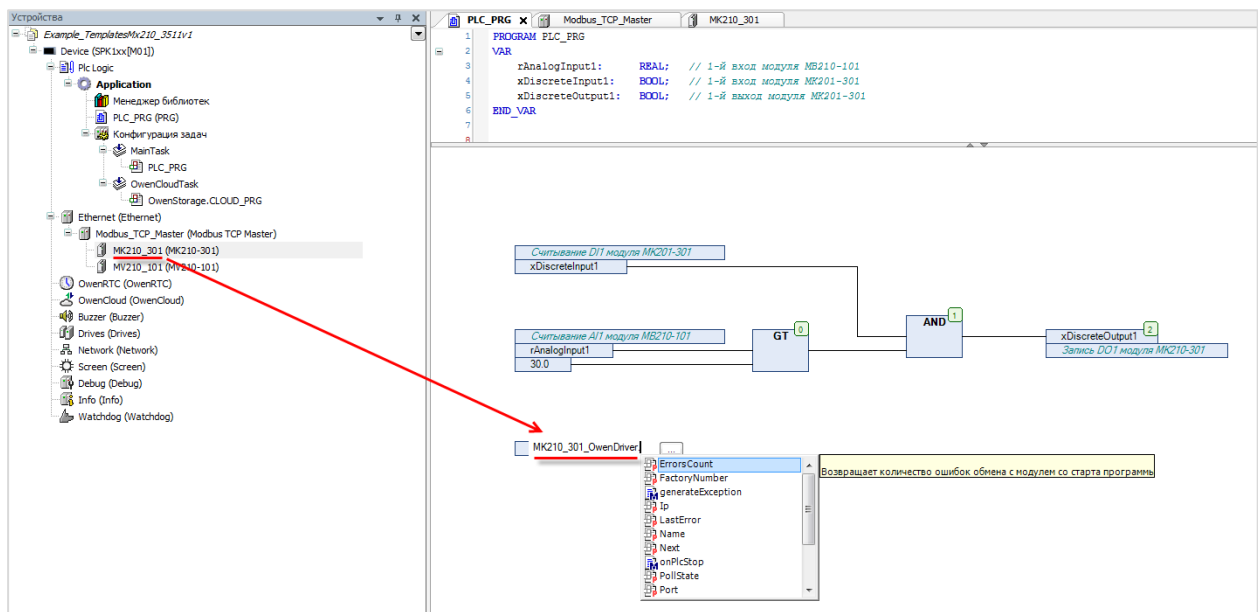


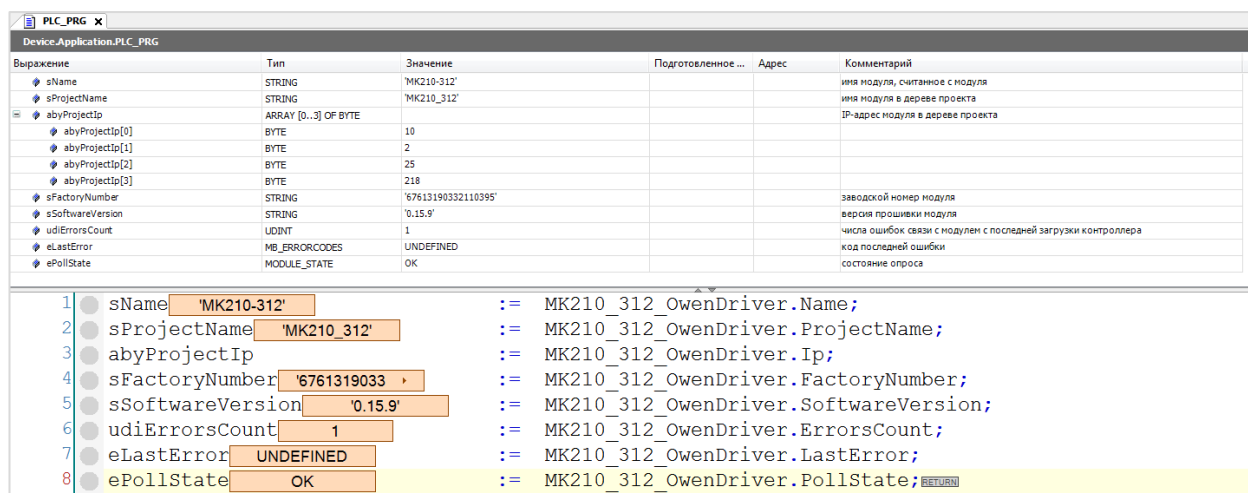
Рисунок 3.4.1 – Переменные диагностики модулей Mx210

Таблица 3.4.1 – Список переменных диагностики шаблонов модулей Mx210

Название	Тип	Описание
ErrorsCount	UDINT	Количество ошибок связи с модулем с момента последней загрузки контроллера
FactoryNumber	STRING	Заводской номер модуля
Ip	ARRAY [0..3] OF BYTE	IP-адрес модуля, заданный в шаблоне
LastError	<a href="#">IoDrvModbusTCP.MB_ErrorCodes</a>	Код последней ошибки обмена
Name	STRING	Имя модуля, заданное в его прошивке
PollState	IoDrvMx210.MODULE_STATE	Состояние обмена с модулем
Port	UINT	Порт Modbus TCP, заданный в шаблоне
ProjectName	STRING	Имя модуля в дереве проекта
SoftwareVersion	STRING	Версия прошивки модуля

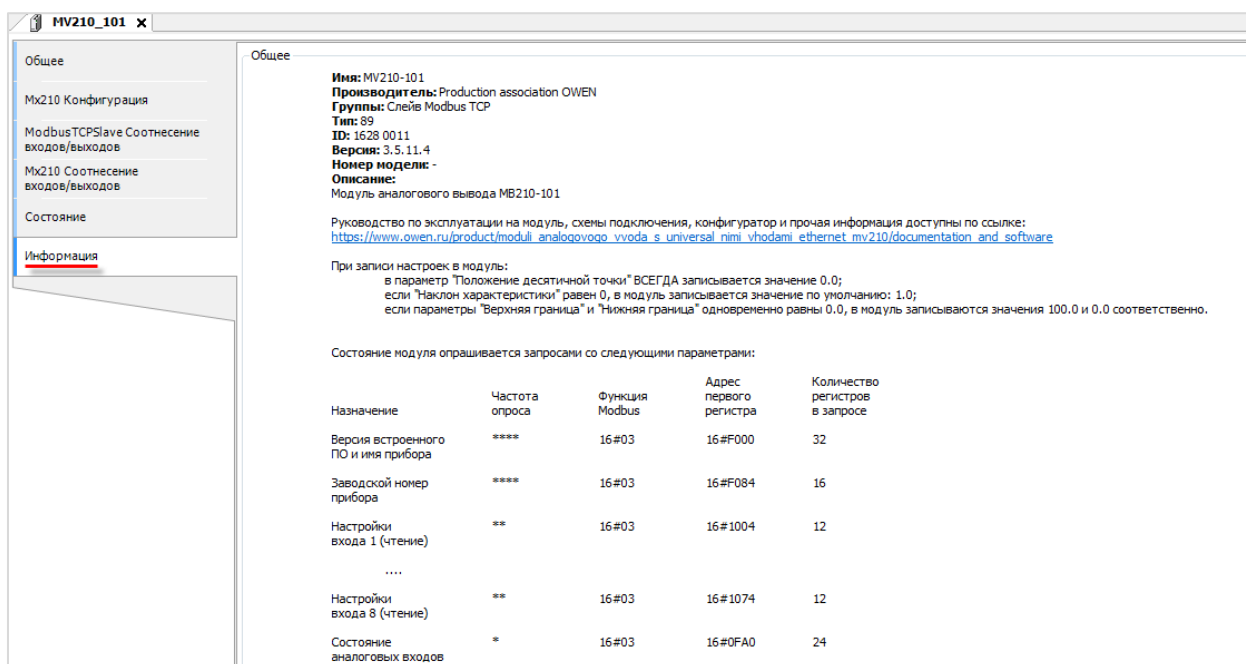
Параметры **FactoryNumber**, **Name** и **SoftwareVersion** считываются при каждом возобновлении обмена с модулем (то есть при физической замене одного модуля на другой идентичной модели – значения параметров будут актуализированы).

### 3. Шаблоны модулей Mx110 и Mx210



**Рисунок 3.4.2 – Использование переменных диагностики**

В процессе отладки может потребоваться информация о внутреннем устройстве шаблона (используемые коды функций, количество регистров в запросе и т. д.). Эти сведения приведены на вкладке **Информация**:



**Рисунок 3.4.3 – Информация об устройстве шаблона**

Информация о переменных диагностики стандартных компонентов (**Modbus Master, Modbus Slave** и т. д.) приведена в [п. 4.6](#).

### 3.5 Библиотеки Mx Assistant

При добавлении в проект шаблонов модулей аналогового ввода и вывода в **Менеджере библиотек** будет автоматически добавлена библиотека **Mx110 Assistant** (для шаблонов модулей **Mx110**) или **Mx210 Assistant** (для шаблонов модулей **Mx210**).

В случае необходимости библиотека может быть добавлена в проект вручную (**Менеджер библиотек – Добавить библиотеку**).

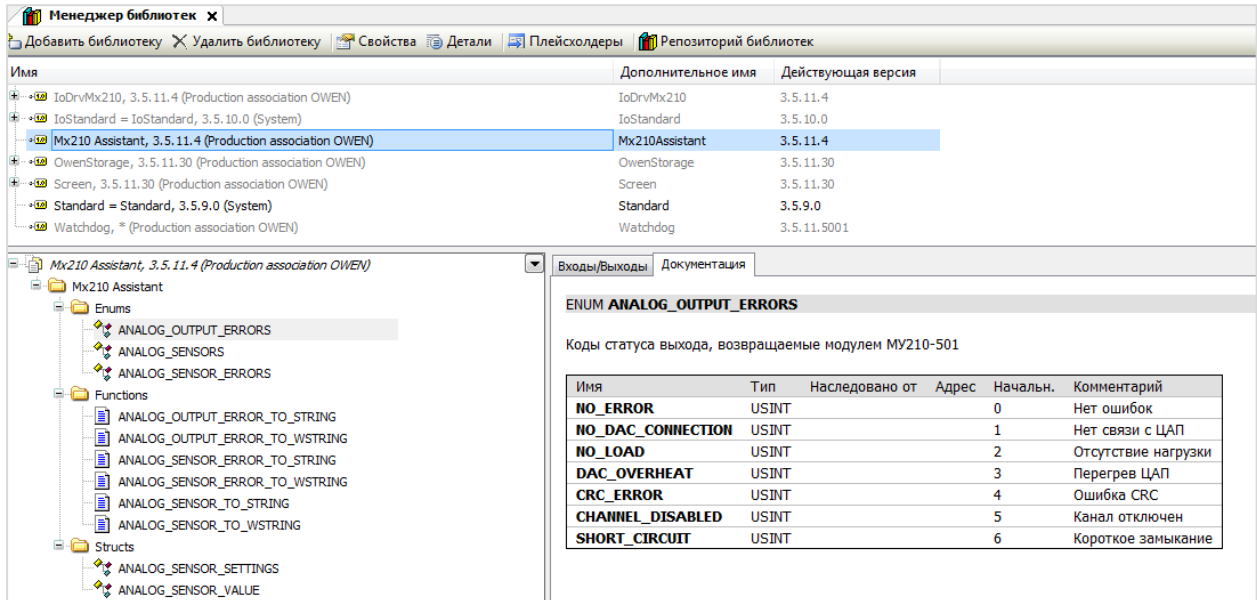


Рисунок 3.5.1 – Библиотека Mx210 Assistant в Менеджере библиотек

Таблица 3.5.1 – Содержимое библиотеки Mx110 Assistant

Название	Тип	Описание
MV_SENSOR_ERROR	Перечисление	Код статуса аналогового входа модуля <b>MB110-2A/2AC/8A/8AC</b> . Если зафиксирована ошибка обмена (канал <b>Флаг ошибки</b> имеет значение <b>TRUE</b> ), то будет возвращено значение <b>MODBUS_ERROR</b>
MvStatusToString	Функция	Функция преобразует код статуса в строку типа <b>STRING</b>
MvStatusToWstring	Функция	Функция преобразует код статуса в строку типа <b>WSTRING</b>

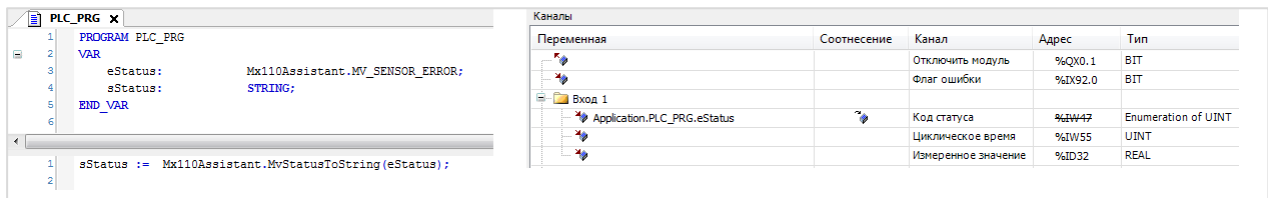


Рисунок 3.5.2 – Пример работы с функциями библиотеки Mx110 Assistant



Таблица 3.5.2 – Содержимое библиотеки Mx210 Assistant

Название	Тип	Описание
ANALOG_SENSOR_VALUE	Структура	Структура параметров аналогового входа модуля <b>MB210-101</b>
ANALOG_SENSOR_SETTINGS	Структура	Структура настроек аналогового входа модуля <b>MB210-101</b>
ANALOG_SENSORS	Перечисление	Код типа датчика для аналогового входа модуля <b>MB210-101</b>
ANALOG_SENSOR_ERRORS	Перечисление	Код статуса аналогового входа модуля <b>MB210-101</b> . Если зафиксирована ошибка обмена (канал <b>Флаг ошибки</b> имеет значение <b>TRUE</b> ), то будет возвращено значение <b>FIELDBUS_ERROR</b>
ANALOG_OUTPUT_ERRORS	Перечисление	Код статуса аналогового выхода модуля <b>MY210-501</b>
ANALOG_SENSOR_TO_STRING	Функция	Функция преобразует код типа датчика в строку типа <b>STRING</b>
ANALOG_SENSOR_TO_WSTRING	Функция	Функция преобразует код типа датчика в строку типа <b>WSTRING</b>
ANALOG_SENSOR_ERROR_TO_STRING	Функция	Функция преобразует код статуса аналогового входа в строку типа <b>STRING</b>
ANALOG_SENSOR_TO_ERROR_WSTRING	Функция	Функция преобразует код статуса аналогового входа в строку типа <b>WSTRING</b>
ANALOG_OUTPUT_ERROR_TO_STRING	Функция	Функция преобразует код статуса аналогового выхода в строку типа <b>STRING</b>
ANALOG_OUTPUT_ERROR_TO_WSTRING	Функция	Функция преобразует код статуса аналогового выхода в строку типа <b>WSTRING</b>

## 4 Стандартные средства конфигурирования

### 4.1 Общая методика конфигурирования интерфейсов

Настройка обмена в **CODESYS** состоит из следующих действий:

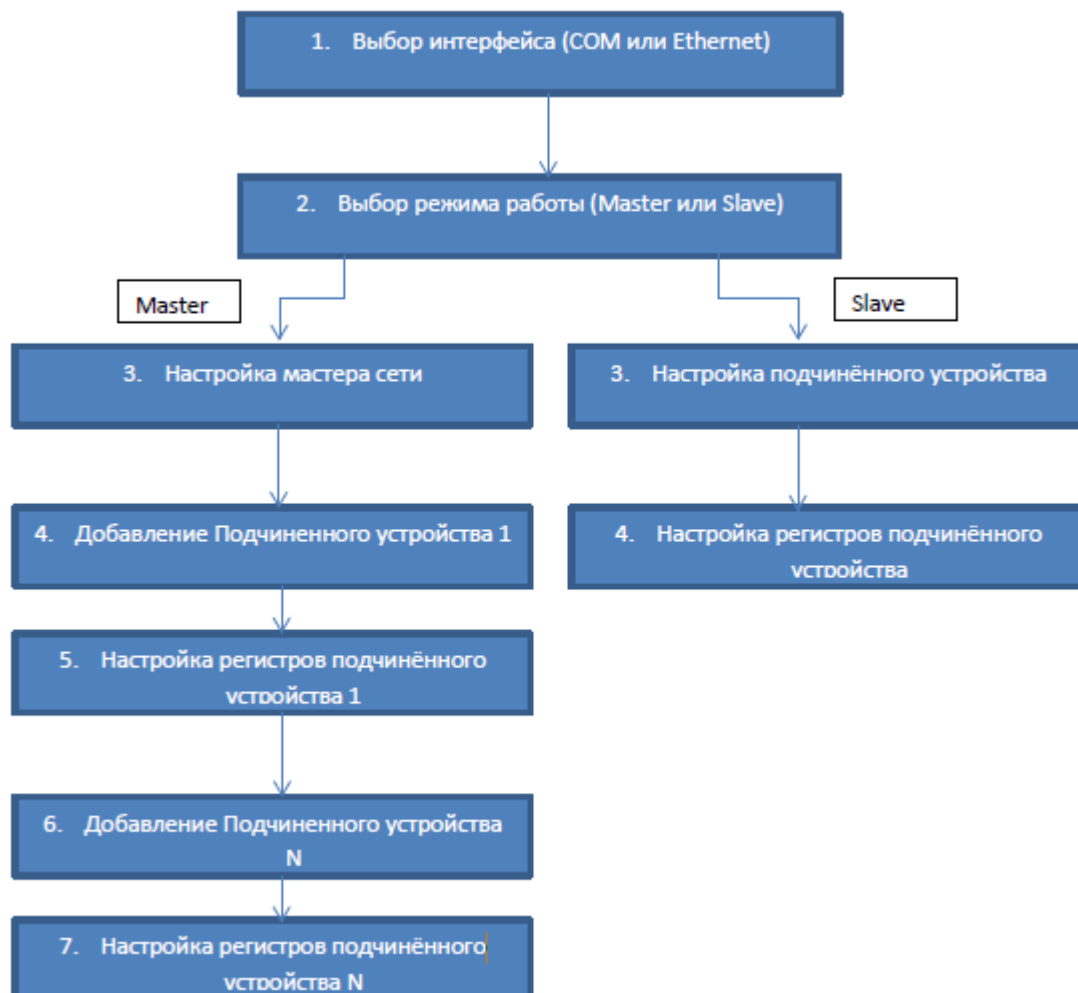


Рисунок 4.1.1 – Последовательность конфигурирования Modbus в CODESYS

Сначала следует добавить и настроить интерфейс. Затем выбрать режим работы интерфейса – **Master** или **Slave** (режим работы представляется отдельным компонентом). Если интерфейс работает в режиме мастера, то следует добавить все slave-устройства и указать для них адреса и опрашиваемые/записываемые регистры. Если интерфейс работает в режиме slave, то достаточно привязать к его регистрам нужные переменные.

## 4.2 Настройка контроллера в режиме Modbus Serial Master

Для настройки контроллера в режиме **Modbus Serial Master** следует:

1. Нажать **ПКМ** на компонент **Device** и **добавить** компонент **Modbus COM**, расположенный во вкладке **Промышленные сети/Modbus/Порт Modbus Serial**.



### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

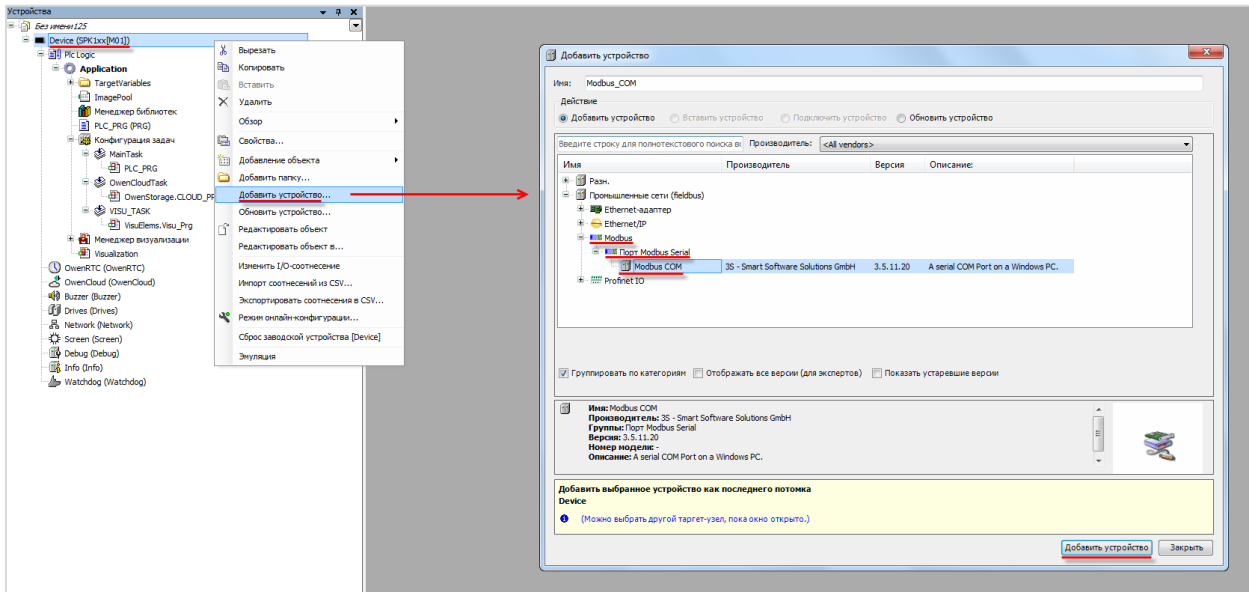


Рисунок 4.2.1 – Добавление компонента Modbus COM

В настройках компонента на вкладке **Общее** следует указать [номер COM-порта](#) контроллера и его сетевые настройки.

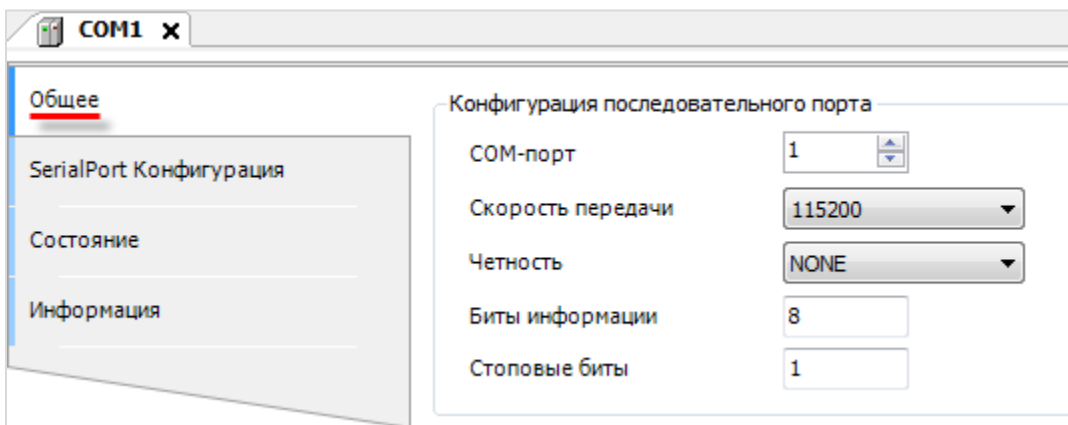


Рисунок 4.2.2 – Настройки компонента Modbus COM

Таблица 4.2.1 – Настройки компонента Modbus COM

Параметр	Описание
COM-порт	Идентификатор используемого COM-порта (см. <a href="#">п. 2.3</a> )
Скорость передачи	Скорость передачи данных в бодах, возможные значения: <b>1200/2400/4800/9600/19200/38400/57600/115200</b>
Четность	Режим контроля четности: <b>NONE</b> – отсутствует, <b>EVEN</b> – проверка на четность, <b>ODD</b> – проверка на нечетность
Биты информации	Число бит данных. Возможные значения: <b>7</b> или <b>8</b>
Стоповые биты	Число стоповых бит. Возможные значения: <b>1</b> или <b>2</b>

2. Нажать ПКМ на компонент **Modbus COM** и добавить компонент **Modbus Master**, расположенный во вкладке **Промышленные сети/Modbus/Мастер Modbus Serial**.



### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

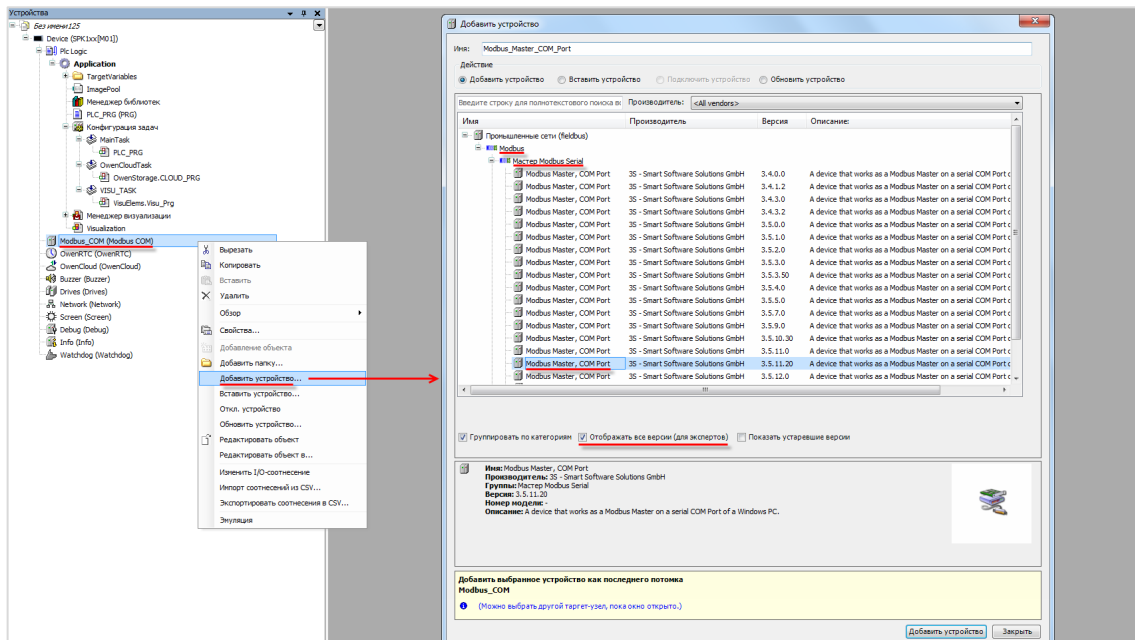


Рисунок 4.2.3 – Добавление компонента Modbus Master

В настройках компонента на вкладке **Общие** следует задать настройки мастера.

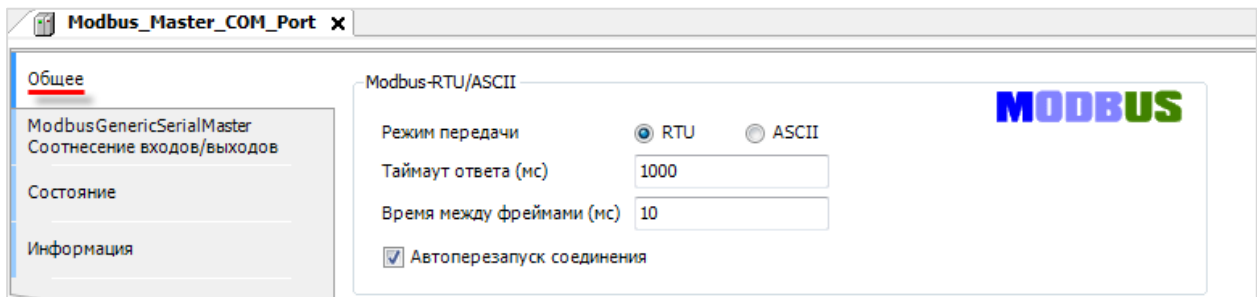


Рисунок 4.2.4 – Настройки компонента Modbus Master

Таблица 4.2.2 – Настройки компонента Modbus Master

Параметр	Описание
Режим передачи	Тип протокола обмена: <b>Modbus RTU</b> или <b>Modbus ASCII</b>
Таймаут ответа	Время (в мс), в течение которого master ожидает ответа slave-устройства. В случае отсутствия ответа по истечению этого времени master делает паузу на <b>время между фреймами</b> и переходит к опросу следующего slave-устройства. Значение, введенное здесь, будет по умолчанию использоваться для всех slave-устройств. На вкладке <b>Конфигурация Modbus Slave</b> (см. <a href="#">рисунок 4.2.6</a> ) для каждого устройства можно задать индивидуальный таймаут отклика
Время между фреймами	Время (в мс) между получением ответа от slave-устройства и началом опроса следующего. Чем выше скорость, тем меньшим может быть это значение (на скорости 115200 бит/с – 5–20 мс). Некоторые устройства в течение определенного времени (например, <b>ОВЕН СМ12</b> – на 50 мс) удерживают линию связи после ответа, поэтому в данном случае не имеет смысла выставлять время между фреймами меньше, чем это значение. При работе с модулями <b>Mx110</b> рекомендуется использовать значение <b>20 мс</b>
Автоперезапуск соединения	В случае отсутствия галочки не ответившее slave-устройство исключается из дальнейшего опроса. <b>Настоятельно рекомендуется</b> всегда включать эту опцию

3. Нажать **ПКМ** на компонент **Modbus Master** и добавить компонент **Modbus Slave**, расположенный во вкладке **Промышленные сети/Modbus/Слейв Modbus Serial**. Число компонентов должно соответствовать числу slave-устройств, подключенных к COM-порту. Максимальное возможное количество slave-устройств для одного мастера – **32**.



**ПРИМЕЧАНИЕ**

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

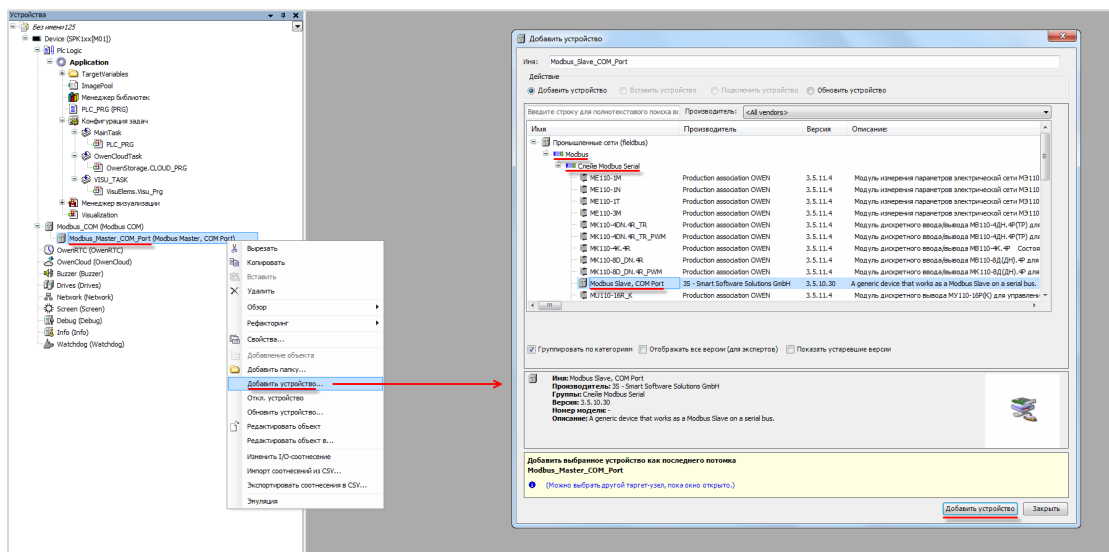
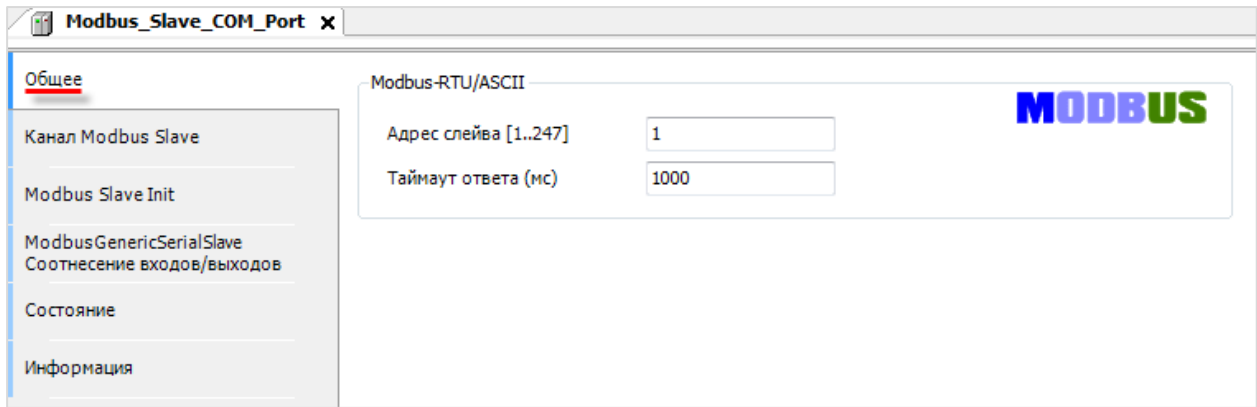


Рисунок 4.2.5 – Добавление компонента Modbus Slave



**Рисунок 4.2.6 – Настройки компонента Modbus Slave, вкладка Общее**

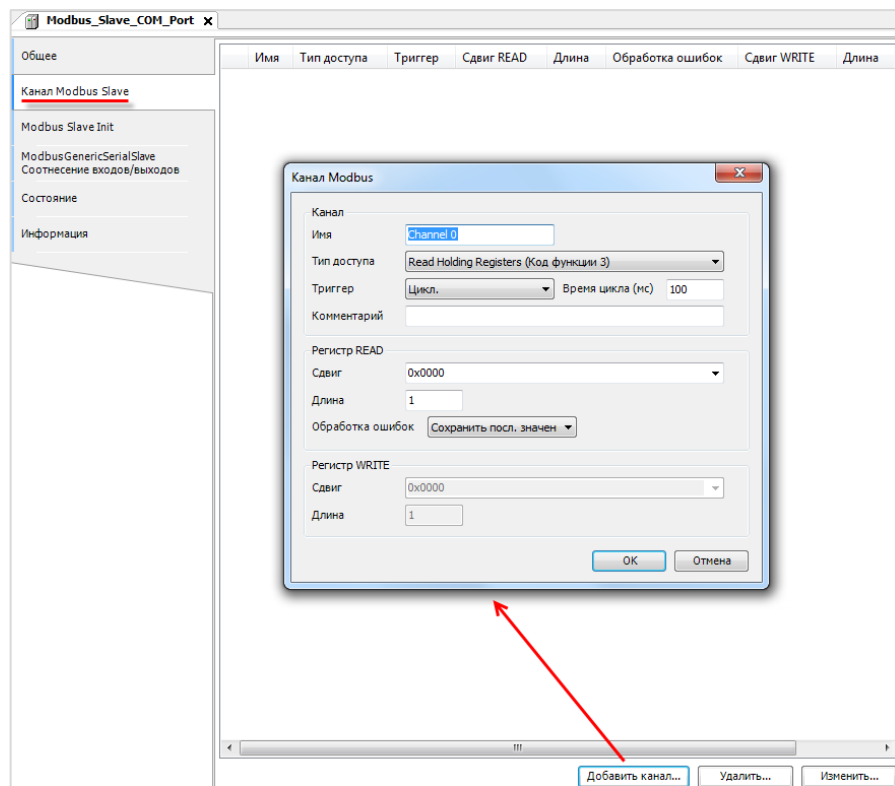
В настройках компонента на вкладке **Общее** следует указать адрес slave-устройства. В случае необходимости можно указать индивидуальный таймаут ответа – он будет иметь приоритет по сравнению с таймаутом, установленным в настройках **Modbus Master** (см. [рисунок 4.2.4](#)).



**ПРИМЕЧАНИЕ**

Диапазон доступных адресов slave-устройств – **1...247**. Широковещательная рассылка через адрес **0** не поддерживается.

На вкладке **Канал Modbus Slave** происходит добавление каналов slave-устройства. Канал является структурной единицей обмена, определяющей тип и число последовательно расположенных регистров slave-устройства и применяемую к ним операцию (чтение/запись). Максимальное число каналов для одного устройства – **100**. Для создания нового канала следует нажать кнопку **Добавить канал**, после чего определить его настройки:



**Рисунок 4.2.7 – Добавление канала Modbus Slave**

Таблица 4.2.3 – Параметры канала Modbus Slave

Параметр	Описание
Имя	Название канала
Тип доступа	Функция, применяемая к регистрам slave-устройства (см. <a href="#">таблицу 2.2</a> );
Триггер	Тип обращения к регистрам slave-устройства: циклически, по переднему фронту заданной логической переменной или <a href="#">из кода приложения</a>
Время цикла	Период опроса канала slave-устройства (для триггера типа <b>циклический</b> ). Должен быть кратен времени цикла задачи, к которой привязан обмен. Также время цикла должно выбираться в зависимости от опрашиваемого устройства – например, для модулей <b>МВ110-8А</b> время обновления данных одного канала для термопары типа ТХК составляет 0.4 секунды и, соответственно, разумное время цикла в секундах равно произведению 0.4 на число используемых каналов.
Комментарий	Описание канала
Сдвиг	Номер регистра или первого из последовательности регистров (для операций группового чтения/записи), к которым применяется заданная функция. Можно вводить как в десятичном, так и в шестнадцатеричном виде (например, <b>0x00FF</b> или <b>16#00FF</b> )
Длина	Количество регистров, к которым применяется заданная функция (для операций группового чтения/записи)
Обработка ошибок	Операция, выполняемая со значениями канала при возникновении ошибки обмена (только для считываемых регистров) – сохранение последнего значения или обнуление

Ниже приведен пример конфигурации двух каналов **Modbus Slave**:

The screenshot shows a software window titled "Modbus\_Slave\_COM\_Port" with a tree view on the left containing "Общее", "Канал Modbus Slave", "Modbus Slave Init", "ModbusGenericSerialSlave", "Соотнесение входов/выходов", "Состояние", and "Информация". The main area displays a table with the following data:

Имя	Тип доступа	Триггер	Сдвиг READ	Длина	Обработка ошибок	Сдвиг WRITE	Длина	Комментарий
0 Channel 0	Read Holding Registers (Код функции 03)	Цикл., t≠100ms	16#0000	2	Сохранить посл. значение			
1 Channel 1	Write Multiple Registers (Код функции 16)	Передний фронт				16#000A	1	

Рисунок 4.2.8 – Пример настройки каналов Modbus Slave

В данном случае master-устройство каждые **100 мс** будет опрашивать нулевой и первый holding регистры slave-устройства и по переднему фронту триггерной переменной записывать значение в десятый (16#000A=10#10) holding регистр slave-устройства.

На вкладке **Modbus Slave Init** можно указать команды записи, однократно выполняемые при запуске проекта.

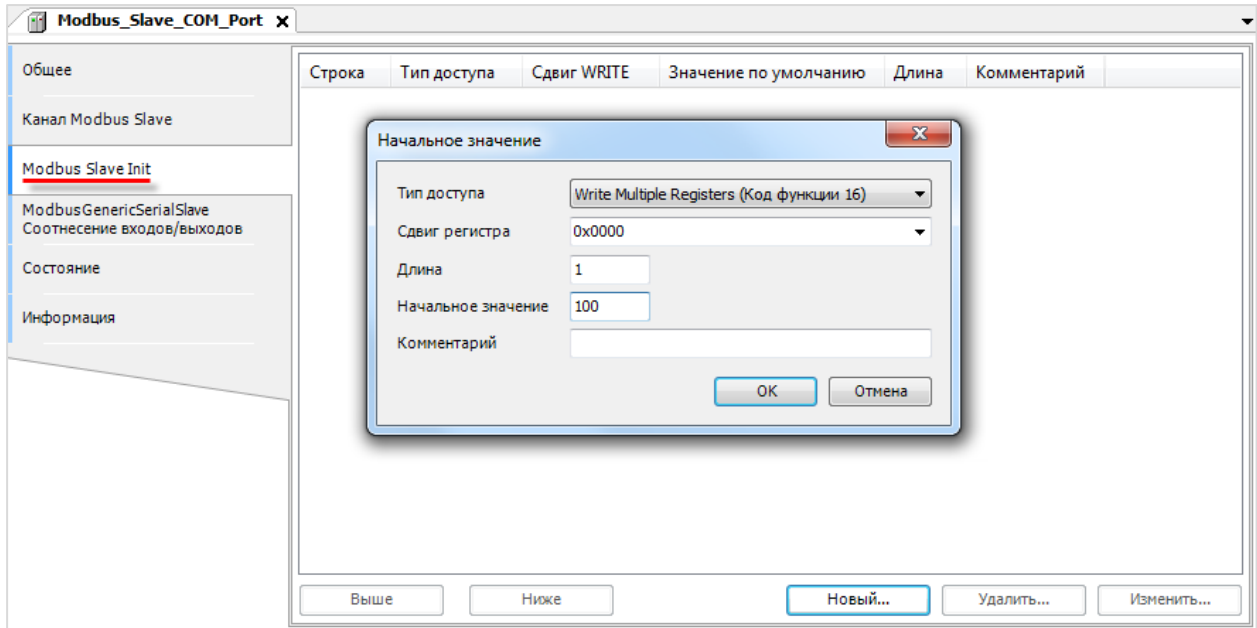


Рисунок 4.2.9 – Настройки вкладки Modbus Slave Init

На вкладке **ModbusGenericSerialSlave** **Соотнесение входов/выходов** осуществляется привязка переменных программы к каналам Modbus. Стандарт Modbus определяет использование двух типов данных: **BOOL** и **WORD**. Пользователь должен привязать к каждому регистру канала переменную соответствующего типа, либо привязать непосредственно к каналу массив переменных соответствующего типа. К каждому из битов **WORD** переменной можно также привязать **BOOL** переменную (для считываемых переменных эта привязка не исключает привязку WORD переменной, для записываемых – исключает).

**ПРИМЕЧАНИЕ**

Для корректного обновления данных в компоненте во вкладке **Всегда обновлять переменные** следует установить значение **Включено 2 (Всегда в задаче цикла шины)**.

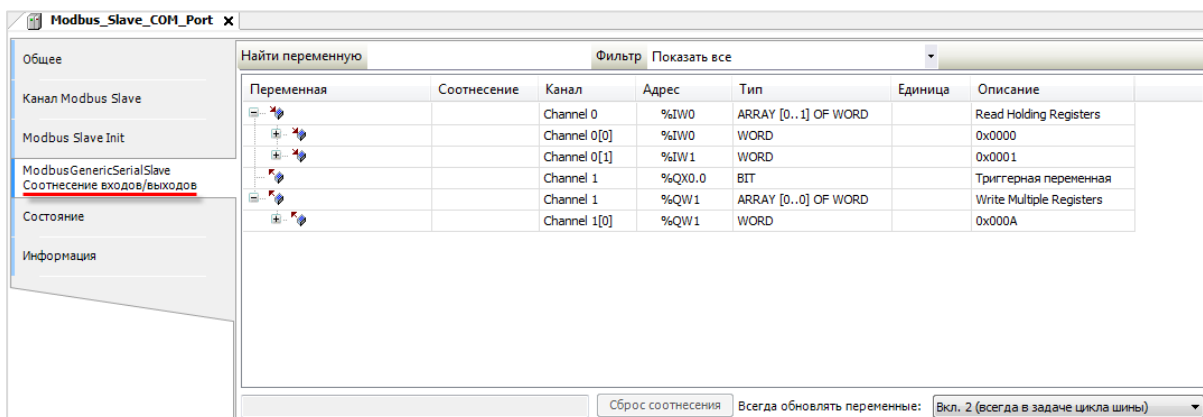


Рисунок 4.2.10 – Настройки вкладки ModbusGenericSerialSlave Соотнесение входов/выходов



#### 4. Стандартные средства конфигурирования

Для привязки переменных следует два раза нажать **ЛКМ** на ячейку столбца **Переменная**, после чего выбрать необходимую переменную проекта с помощью **Ассистента ввода** (или ввести имя переменной вручную):

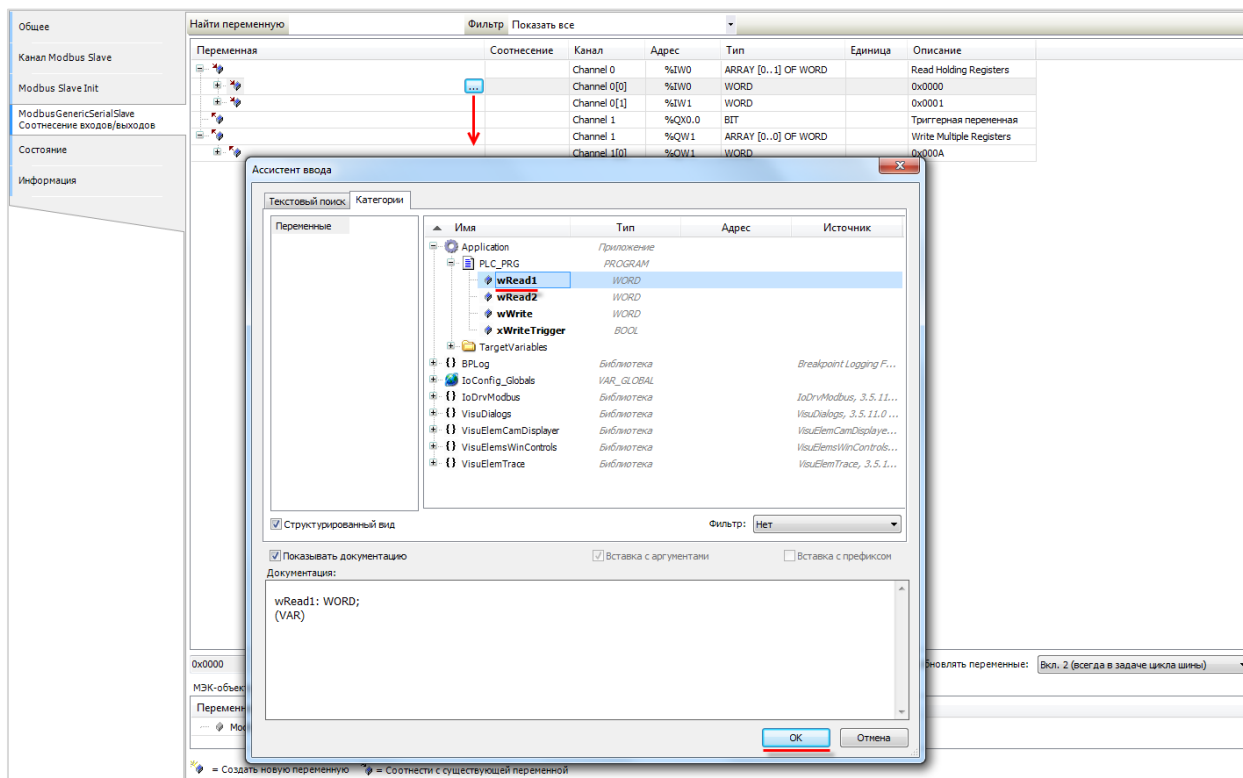


Рисунок 4.2.11 – Привязка переменных программы к каналам Modbus Slave

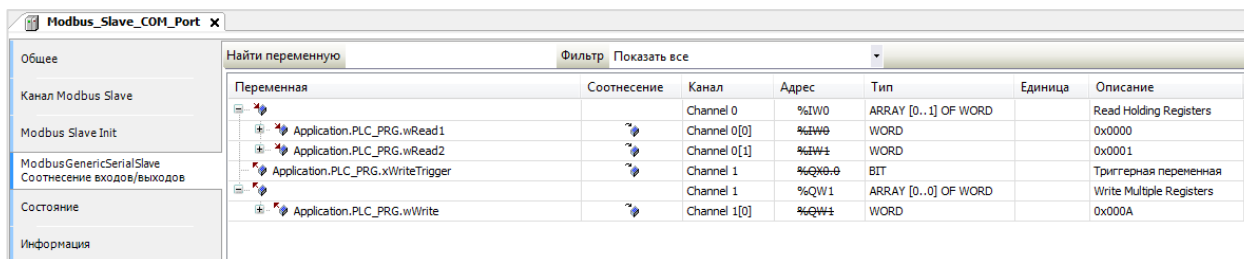


Рисунок 4.2.12 – Привязка переменных программы к каналам Modbus Slave

Пример настройки контроллера в режиме **Modbus Serial Master** для опроса модулей **Mx110** приведен в [п. 4.9](#).

### 4.3 Настройка контроллера в режиме Modbus RTU Slave

Для настройки контроллера в режиме **Modbus RTU Slave** следует:

1. Нажать **ПКМ** на компонент **Device** и добавить компонент **Modbus COM**, расположенный во вкладке **Промышленные сети/Modbus/Порт Modbus Serial**.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

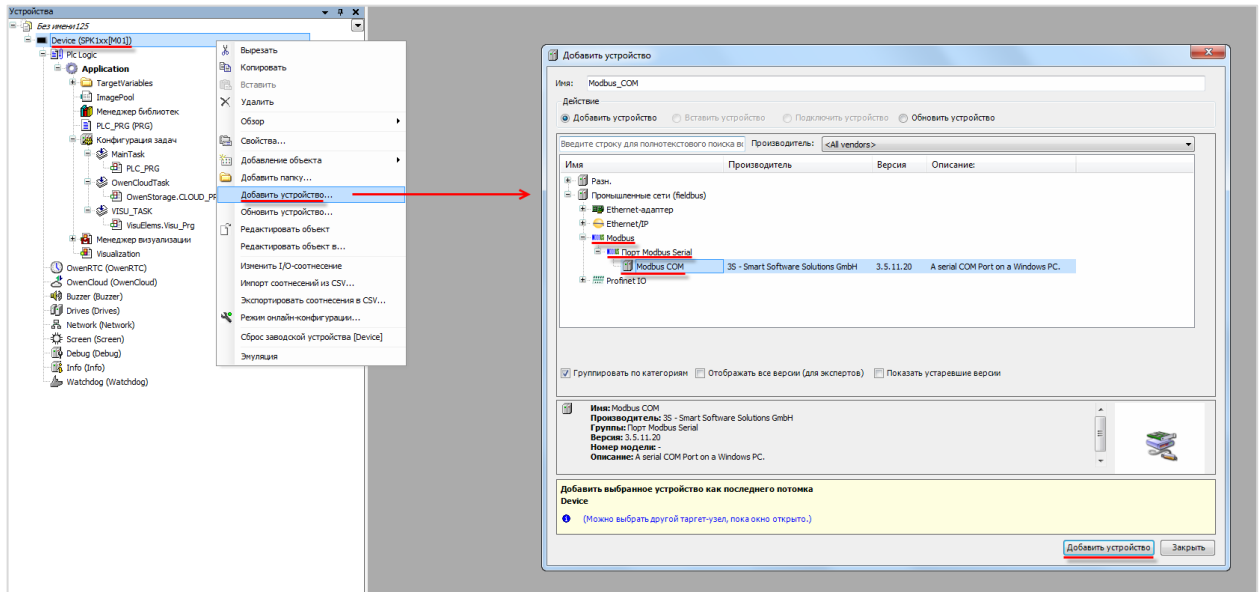


Рисунок 4.3.1 – Добавление компонента Modbus COM

Настройки компонента описаны в [п. 4.2 \(пп. 1\)](#).

2. Нажать **ПКМ** на компонент **Modbus COM** и добавить компонент **Modbus Serial Device**, расположенный во вкладке **Промышленные сети/Modbus/Устройство Modbus Serial**.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

#### 4. Стандартные средства конфигурирования

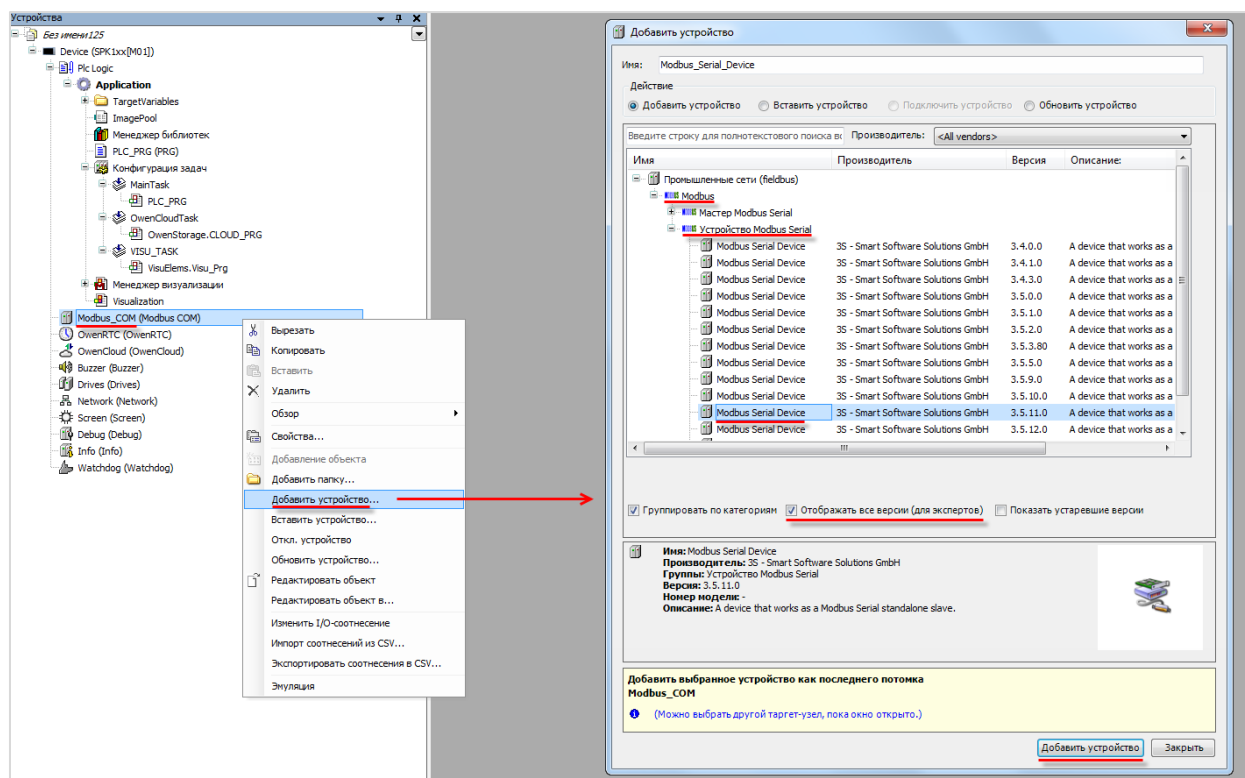


Рисунок 4.3.2 – Добавление компонента Modbus Serial Device

На вкладке **Modbus Serial Device** следует указать настройки slave-устройства:

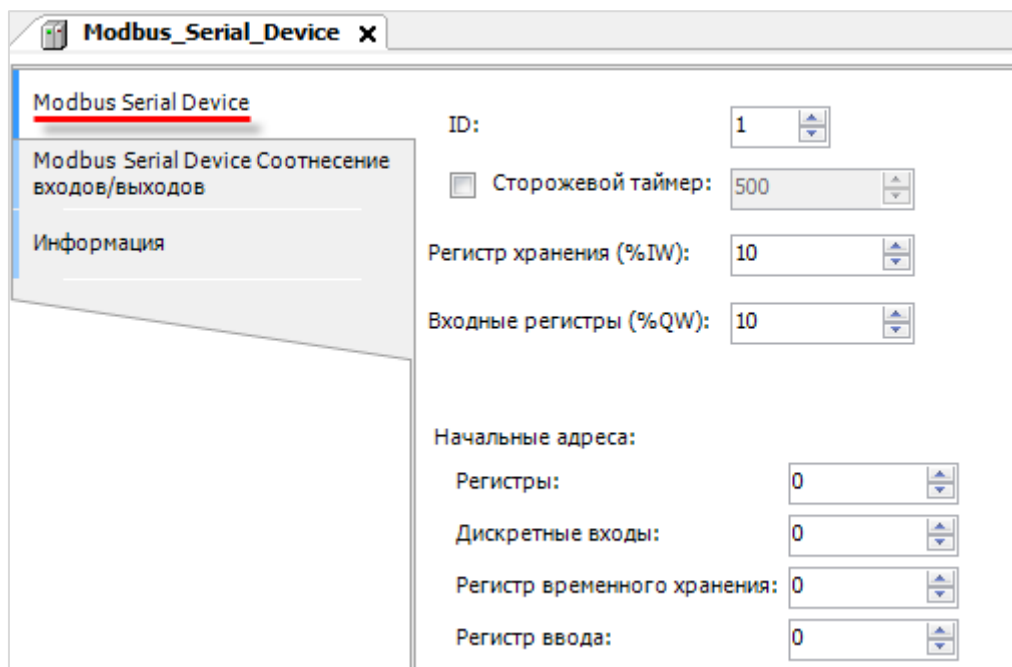


Рисунок 4.3.3 – Настройки компонента Modbus Serial Device



#### ПРИМЕЧАНИЕ

Компонент не поддерживает протокол **Modbus ASCII**. В случае необходимости работы по этому протоколу следует использовать ФБ [MB\\_SerialSlave](#) из библиотеки **OwenCommunication**.

Таблица 4.3.1 – Настройки компонента Modbus Serial Device

Параметр	Описание
ID	Адрес ( <b>Slave ID</b> ) контроллера в рамках выбранного COM-порта
Сторожевой таймер	Время ожидания (в мс) запроса от master-устройства. Если за это время запроса не приходит, то данные в регистрах обнуляются. В случае отсутствия галочки обнуления данных не происходит
Регистр хранения	Количество <b>holding</b> регистров для данного slave-устройства ( <b>2...500</b> )
Входные регистры	Количество <b>input</b> регистров для данного slave-устройства ( <b>2...500</b> )
Начальные адреса	Начальный адрес для каждой <a href="#">области памяти</a> Modbus. В случае получения запроса к регистру, адрес которого меньше, чем адрес начального регистра, контроллер вернет ошибку <b>02 (ILLEGAL_DATA_ADDRESS)</b>

**ПРИМЕЧАНИЕ**

В компоненте используется модель данных, в которой области памяти битов и регистров являются общими: область **coils** наложена на область **holding регистров**, а область **discrete inputs** наложена на область **input регистров**. При этом области памяти holding регистров и input регистров являются независимыми.

**ПРИМЕЧАНИЕ**

Holding регистры обозначаются как каналы типа **Входы (%I)**. Input регистры обозначаются как каналы типа **Выходы (%Q)**.

**ПРИМЕЧАНИЕ**

Адреса регистров могут быть определены по числу в квадратных скобках в столбце **Канал** (в случае использования начальных адресов по умолчанию). **Пример:** Входы[3] – holding регистр с адресом 3, Выходы[4] – input регистр с адресом 4. Адреса бит вычисляются по формуле: *адрес бита = номер регистра · 16 + номер бита в регистре*.

**ПРИМЕЧАНИЕ**

Holding регистры и биты (coils) могут быть изменены со стороны master-устройства и не могут быть изменены со стороны программы контроллера. В случае необходимости изменения этих данных из программы следует использовать ФБ [MB\\_SerialSlave](#) из библиотеки **OwenCommunication**.

**ПРИМЕЧАНИЕ**

В версиях ниже **3.5.14.0** компонент не поддерживает функцию **05 (Write Single Coil)**.

На вкладке **Modbus Serial Slave Соотнесение входов/выходов** осуществляется привязка переменных программы к регистрам slave-устройства. Стандарт **Modbus** определяет использование двух типов данных: **BOOL** и **WORD**. Пользователь должен привязать к каждому регистру канала переменную соответствующего типа либо привязать непосредственно к каналу массив переменных соответствующего типа. К каждому из битов **WORD** переменной можно также привязать **BOOL** переменную (для holding регистров эта привязка не исключает привязку **WORD** переменной, для input регистров – исключает).

**ПРИМЕЧАНИЕ**

Для корректного обновления данных во вкладке **Всегда обновлять переменные** следует установить значение **Включено 2 (Всегда в задаче цикла шины)**.

#### 4. Стандартные средства конфигурирования

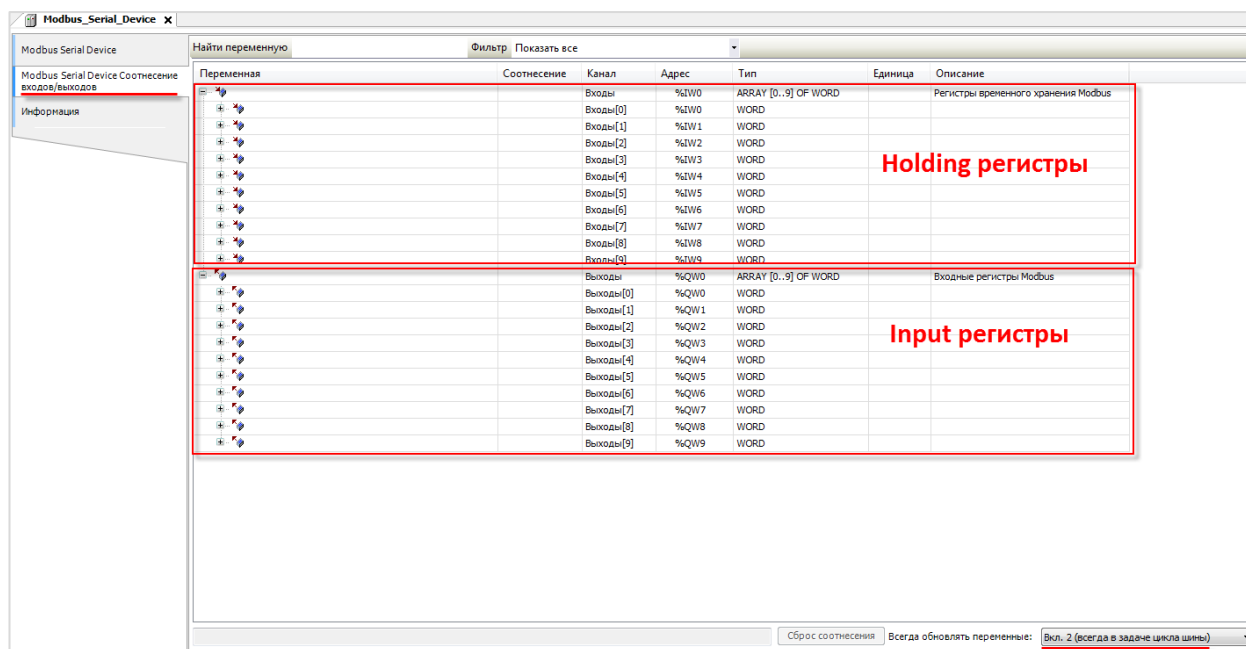


Рисунок 4.3.4 – Настройки вкладки Modbus Serial Slave Соотнесение входов/выходов

Для привязки переменных следует два раза нажать ЛКМ на ячейку столбца **Переменная**, после чего выбрать необходимую переменную проекта с помощью **Ассистента ввода** (или ввести ее имя вручную):

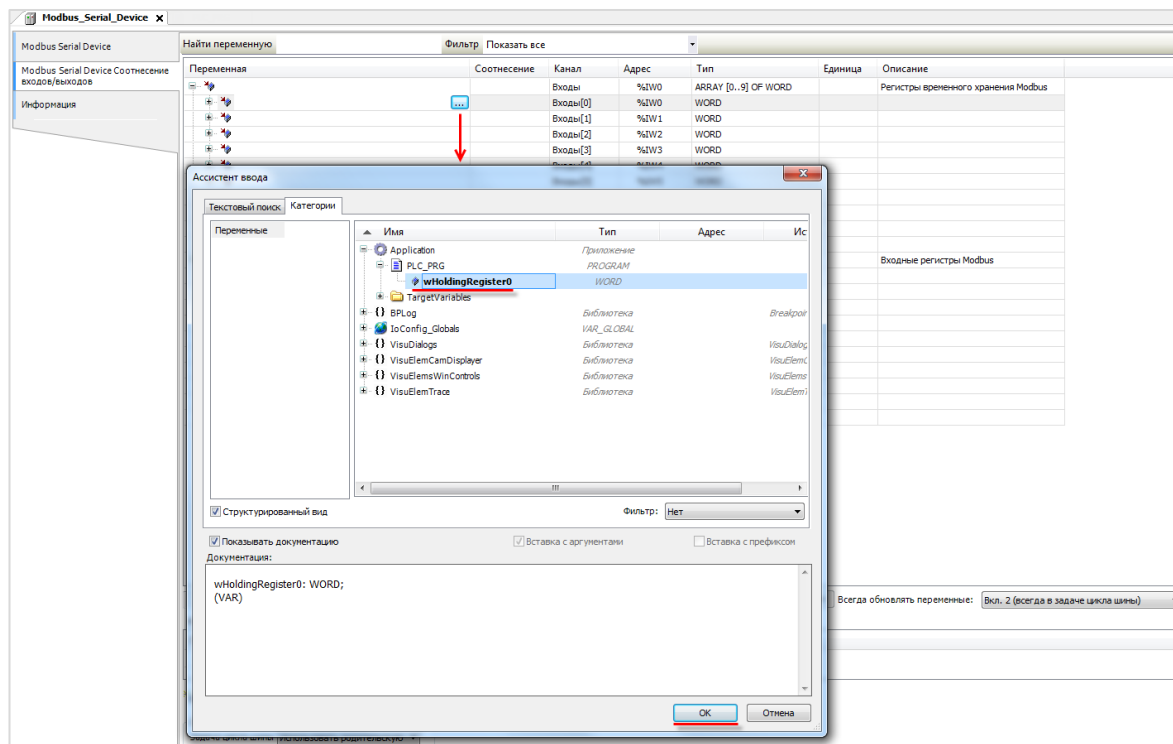


Рисунок 4.3.5 – Привязка переменных программы к регистрам Modbus RTU Slave

Пример настройки контроллера как **Modbus RTU Slave** приведен в [п. 4.10](#).

## 4.4 Настройка контроллера в режиме Modbus TCP Master

Для настройки контроллера в режиме **Modbus TCP Master** следует:

1. Нажать **ПКМ** на компонент **Device** и добавить компонент **Ethernet**, расположенный во вкладке **Промышленные сети/Ethernet-адаптер**.



### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

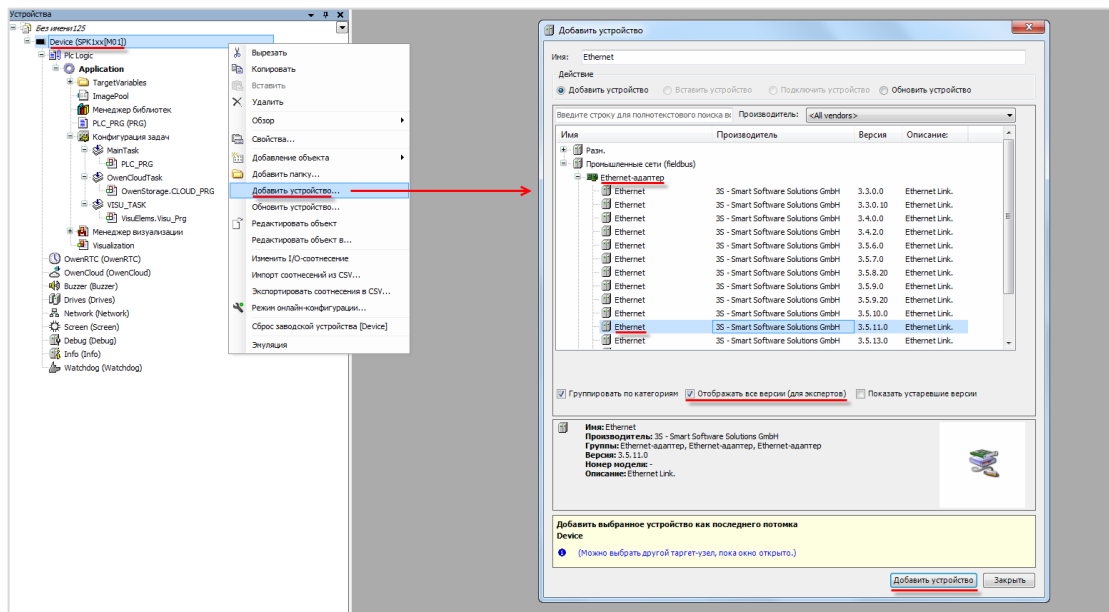


Рисунок 4.4.1 – Добавление компонента Ethernet

Затем следует установить соединение с контроллером, не загружая в него проект (**Device – Установка соединения – Сканировать сеть**) и в компоненте **Ethernet** на вкладке **Конфигурация Ethernet** выбрать нужный интерфейс.

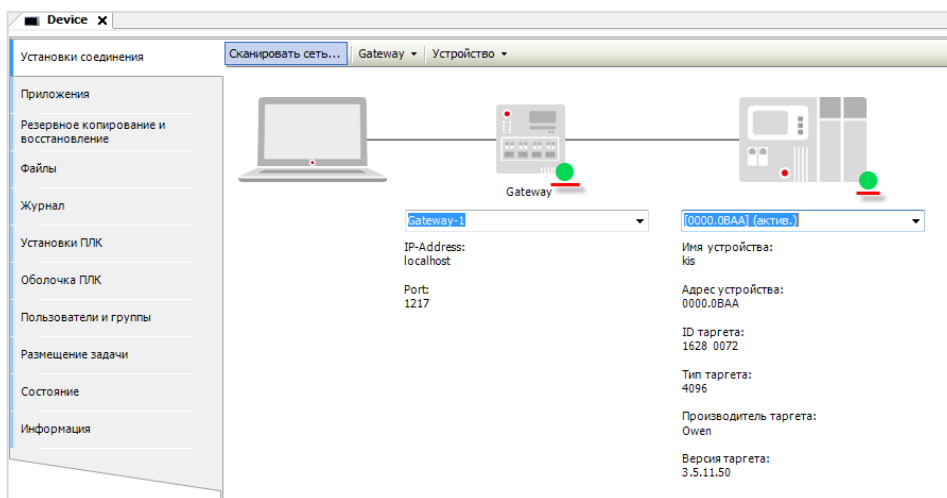


Рисунок 4.4.2 – Подключение к контроллеру

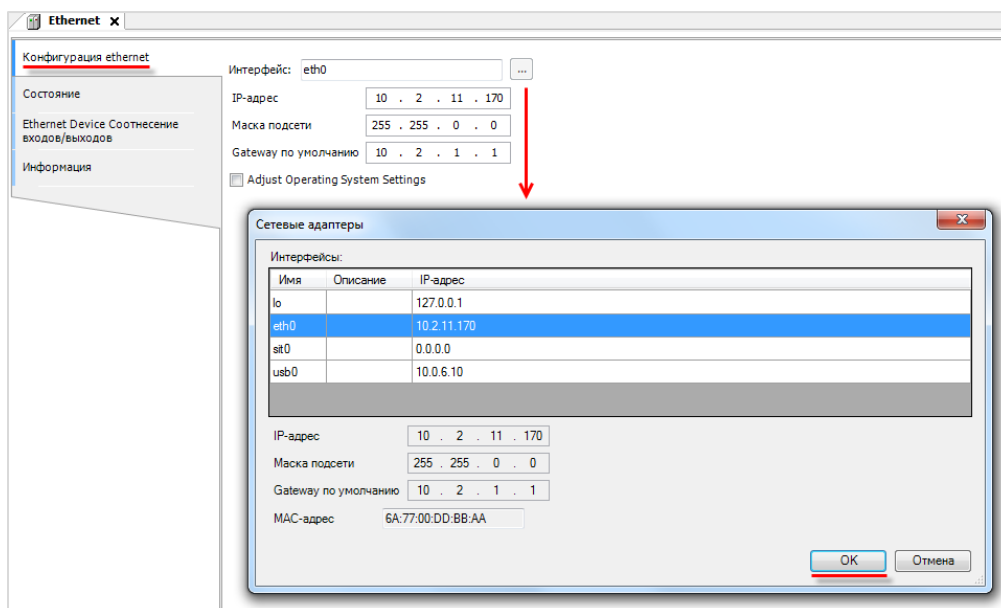


Рисунок 4.4.3 – Выбор используемого интерфейса



#### ПРИМЕЧАНИЕ

Настройки интерфейса задаются в конфигураторе контроллера (см. документ **CODESYS V3.5. FAQ**).



#### ПРИМЕЧАНИЕ

В случае установки галочки **Adjust Operating System Settings** пользователь может изменить настройки интерфейса. После загрузки проекта в контроллер эти настройки будут применены в операционной системе контроллера. Контроллеры ОВЕН **не поддерживают** данный функционал.

2. Нажать **ПКМ** на компонент **Ethernet** и добавить компонент **Modbus TCP Master**, расположенный во вкладке **Промышленные сети/Modbus/Мастер Modbus TCP**.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).



#### ПРИМЕЧАНИЕ

В компонент **Ethernet** может быть добавлено произвольное число компонентов **Modbus TCP Master**.

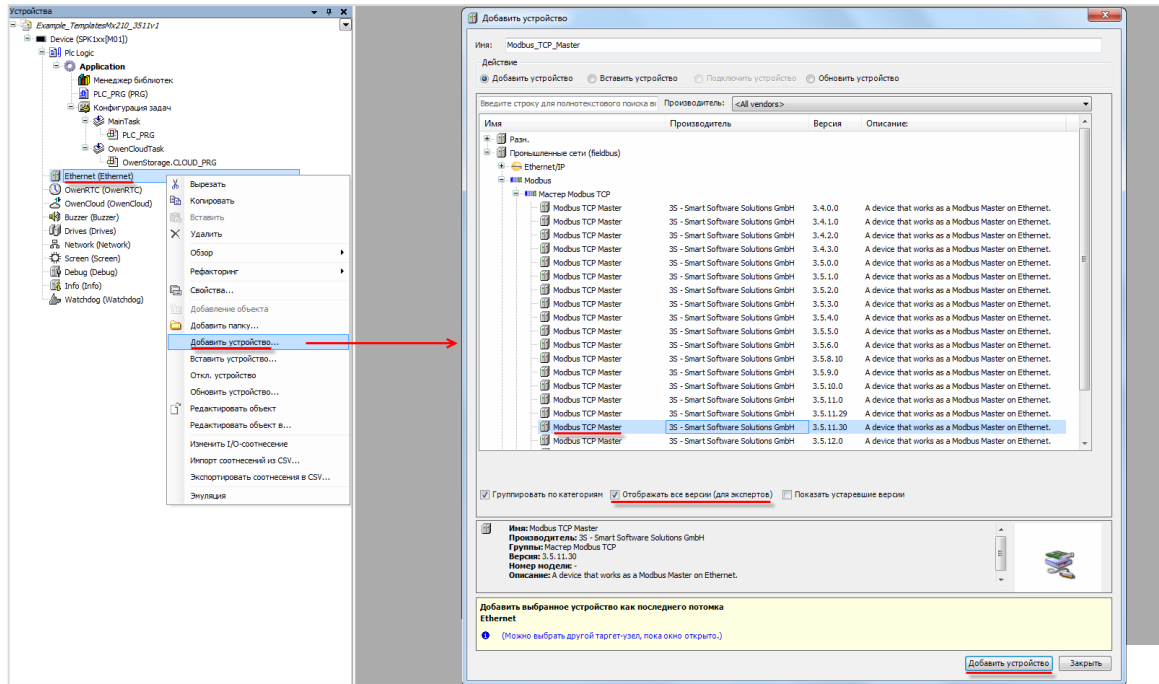


Рисунок 4.4.4 – Добавление компонента Modbus TCP Master

В настройках компонента на вкладке **Общие** следует задать настройки мастера.

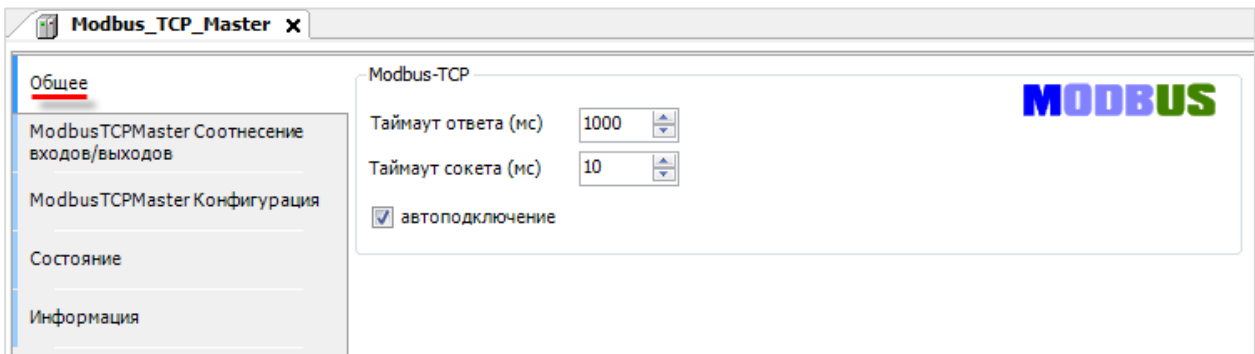


Рисунок 4.4.5 – Настройки компонента Modbus TCP Master

Таблица 4.4.1 – Настройки компонента Modbus TCP Master

Параметр	Описание
Таймаут ответа	Время (в мс), в течение которого master ожидает ответа slave-устройства. В случае отсутствия ответа по истечению этого времени master делает паузу на <b>время между фреймами</b> и переходит к опросу следующего slave-устройства. Значение, введенное здесь, будет по умолчанию использоваться для всех slave-устройств. На вкладке <b>Конфигурация Modbus Slave</b> (см. <a href="#">рисунок 4.4.7</a> ) для каждого устройства можно задать индивидуальный таймаут отклика
Таймаут сокета	Время (в мс), в течение которого master ожидает TCP/IP пакеты от slave-устройства. В случае отсутствия пакетов по истечению этого времени соединение с устройством разрывается
Автоподключение	В случае отсутствия галочки не ответившее slave-устройство исключается из дальнейшего опроса. <b>Настоятельно рекомендуется</b> всегда включать эту опцию



#### 4. Стандартные средства конфигурирования

3. Нажать ПКМ на компонент **Modbus TCP Master** и добавить компонент **Modbus TCP Slave**, расположенный во вкладке **Промышленные сети/Modbus/Слейв Modbus TCP**. Число компонентов должно соответствовать числу slave-устройств. Максимальное возможное количество slave-устройств для одного мастера – **32** (начиная с версии 3.5.13.0 – **64**).



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

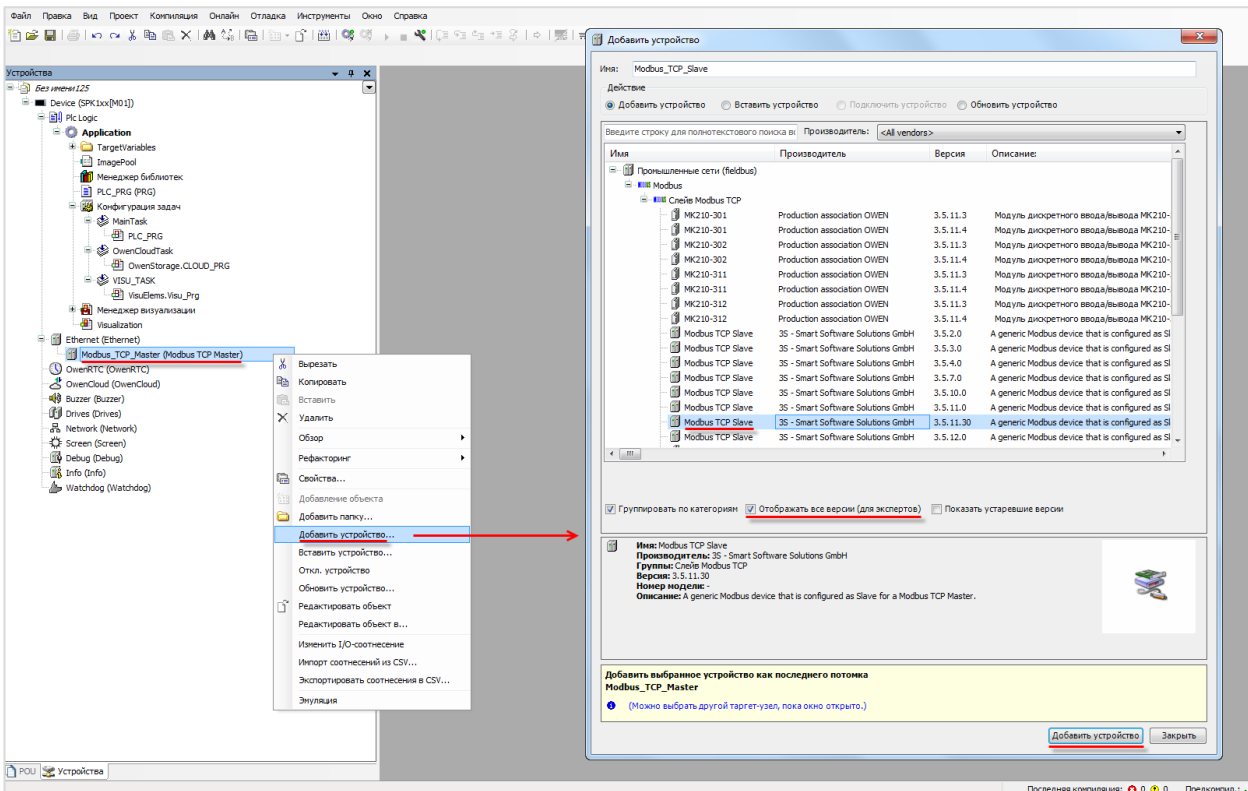


Рисунок 4.4.6 – Добавление компонента Modbus TCP Slave

В настройках компонента на вкладке **Общее** следует указать IP-адрес и порт slave-устройства. В случае необходимости можно указать индивидуальный таймаут ответа – он будет иметь приоритет по сравнению с таймаутом, установленным в настройках **Modbus TCP Master** (см. [рисунок 4.4.5](#)).

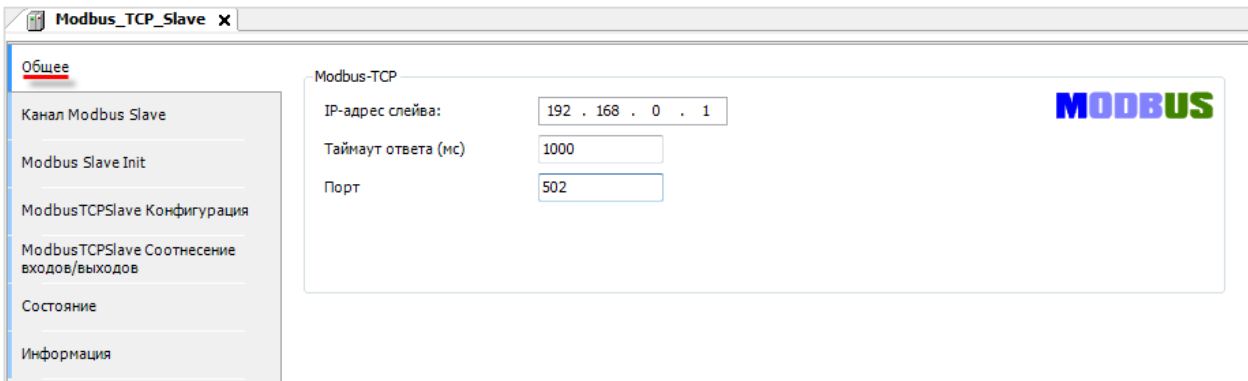
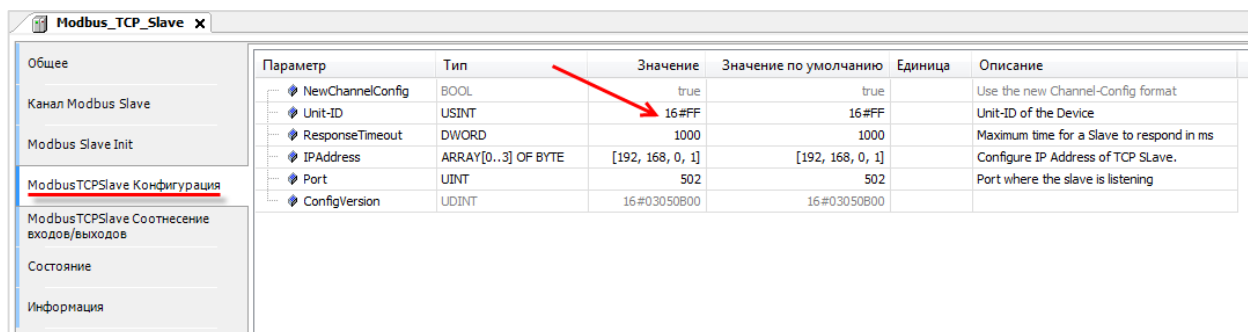


Рисунок 4.4.7 – Настройки компонента Modbus TCP Slave, вкладка Общее

Настройки вкладки **Общее** дублируются на вкладке **ModbusTCPSlave Конфигурация**. На данной вкладке также можно задать адрес (**Unit ID**) slave-устройства – это требуется в тех случаях, когда производится опрос устройства через **шлюз Modbus TCP/Modbus Serial** или если устройство не отвечает на запросы, в которых **Unit ID** имеет значение, предусмотренное спецификацией Modbus по умолчанию (**16#FF**).



Параметр	Тип	Значение	Значение по умолчанию	Единица	Описание
NewChannelConfig	BOOL	true	true		Use the new Channel-Config format
Unit-ID	USINT	16#FF	16#FF		Unit-ID of the Device
ResponseTimeout	DWORD	1000	1000		Maximum time for a Slave to respond in ms
IPAddress	ARRAY[0..3] OF BYTE	[192, 168, 0, 1]	[192, 168, 0, 1]		Configure IP Address of TCP Slave.
Port	UINT	502	502		Port where the slave is listening
ConfigVersion	UDINT	16#03050800	16#03050800		

**Рисунок 4.4.8 – Настройки компонента Modbus TCP Slave, вкладка ModbusTCPSlave Конфигурация**

Настройки вкладок **Канал Modbus Slave** и **Modbus Slave Init** идентичны настройкам одноименных вкладок компонента **Modbus Slave** и описаны в [п. 4.2 \(пп. 3\)](#).

## 4.5 Настройка контроллера в режиме Modbus TCP Slave

Для настройки контроллера в режиме **Modbus TCP Slave** следует:

1. Нажать **ПКМ** на компонент **Device** и добавить компонент **Ethernet**, расположенный во вкладке **Промышленные сети/Ethernet-адаптер**.



### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

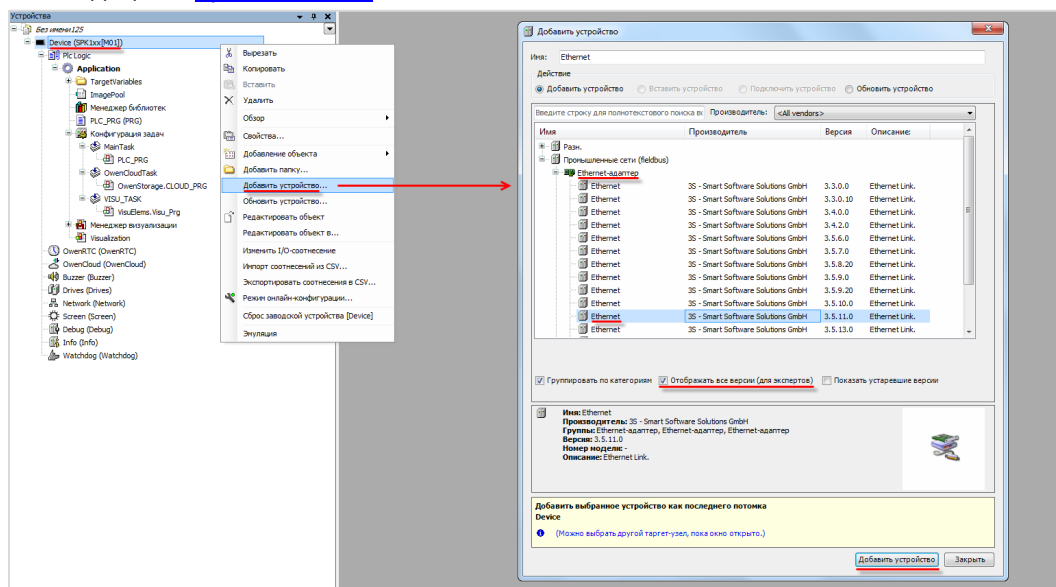


Рисунок 4.5.1 – Добавление компонента Ethernet

Настройки компонента описаны в [п. 4.4 \(пп. 1\)](#).

2. Нажать ПКМ на компонент **Ethernet** и добавить компонент **Modbus TCP Slave Device**, расположенный во вкладке **Промышленные сети/Modbus/Слейв-устройство Modbus TCP**.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).



#### ПРИМЕЧАНИЕ

В компонент **Ethernet** может быть добавлено произвольное число компонентов **Modbus TCP Slave Device**.

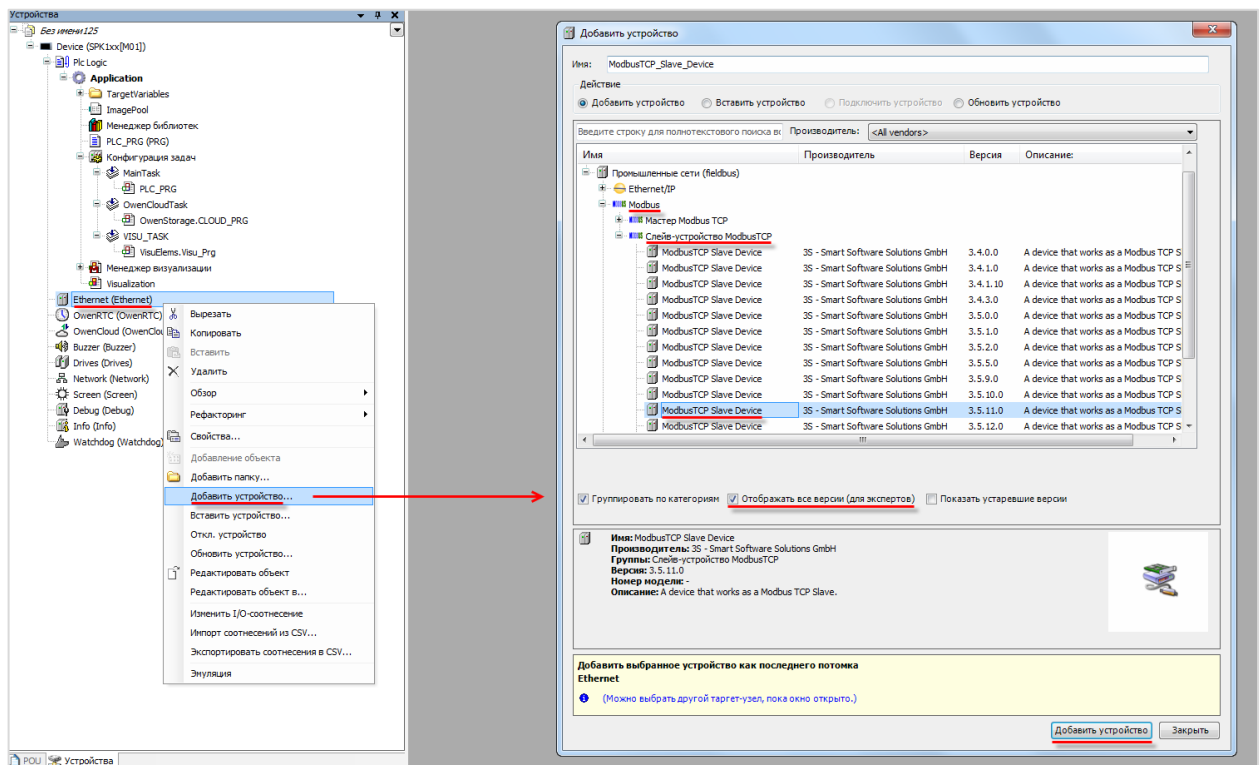


Рисунок 4.5.2 – Добавление компонента Modbus TCP Slave Device

На вкладке **Страница конфигурации** следует указать настройки slave-устройства:

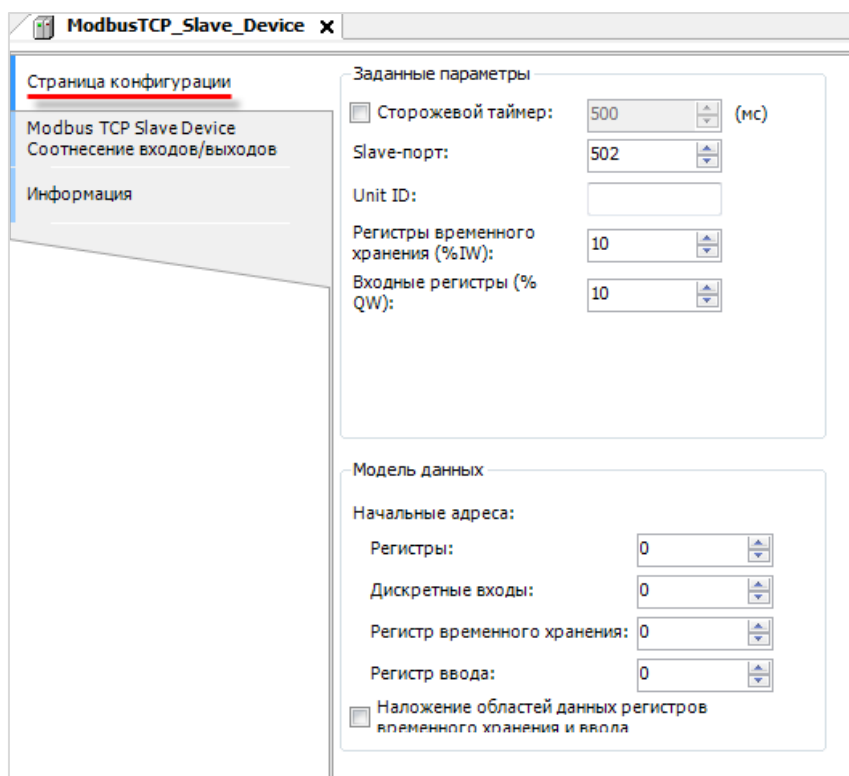


Рисунок 4.5.3 – Настройки компонента Modbus TCP Slave Device

Таблица 4.5.1 – Настройки компонента Modbus TCP Slave Device

Параметр	Описание
Сторожевой таймер	Время ожидания (в мс) запроса от master-устройства. Если за это время запроса не приходит, то данные в регистрах обнуляются. В случае отсутствия галочки обнуления данных не происходит
Slave-порт	Порт, используемый для обмена (по умолчанию – <b>502</b> )
Unit ID	Адрес контроллера в рамках выбранного интерфейса. Используется в случае интеграции контроллера с сетью <b>Modbus Serial</b> через шлюзы <b>Modbus TCP/Modbus Serial</b>
Регистр хранения	Количество <b>holding регистров</b> для данного slave-устройства ( <b>2...4096</b> )
Входные регистры	Количество <b>input регистров</b> для данного slave-устройства ( <b>2...4096</b> )
Начальные адреса	Начальный адрес для каждой <b>области памяти</b> Modbus. В случае получения запроса к регистру, адрес которого меньше, чем адрес начального регистра, контроллер вернет ошибку <b>02 (ILLEGAL_DATA_ADDRESS)</b> .  В случае установки галочки <b>Наложение областей данных регистров временного хранения и ввода</b> при считывании master-устройством holding регистров контроллера будут возвращаться значения соответствующих (совпадающих по номерам) input регистров

Настройки вкладки **Modbus TCP Slave Device** **Соотнесение входов/выходов** идентичны настройкам одноименной вкладки компонента **Modbus Serial Device** и описаны в [п. 4.3 \(пп. 2\)](#).

## 4.6 Диагностика и управление обменом

В случае необходимости контролировать процесс обмена данными можно воспользоваться системными переменными компонентов Modbus. В нужном месте программы следует ввести имя компонента из дерева проекта, поставить точку и из выпадающего списка выбрать нужную переменную:

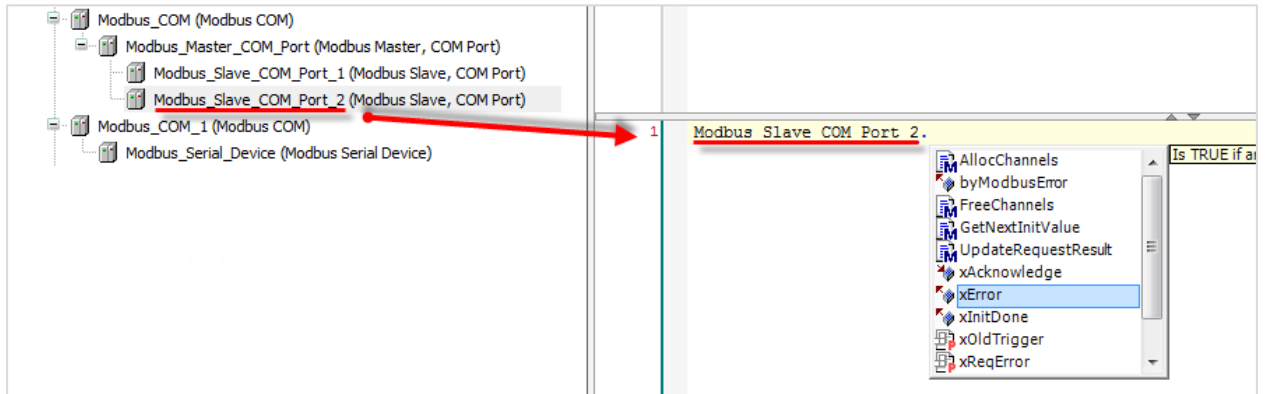


Рисунок 4.6.1 – Использование переменных диагностики в программе

Таблица 4.6.1 – Системные переменные компонентов Modbus

Переменная	Тип	Описание
<b>Компонент Modbus Master</b>		
<i><b>Входы</b></i>		
xResetComPort	BOOL	По переднему фронту выполняется переинициализация компонента <b>Modbus COM</b> , в который добавлен данный компонент
xStop	BOOL	Если вход имеет значение <b>TRUE</b> , то опрос всех slave-устройств данного компонента прекращается
<i><b>Выходы</b></i>		
uiNumberOfCommunicatingSlaves	UINT	Число slave-устройств данного компонента, от которых во время последнего сеанса опроса были получены корректные ответы (без кодов ошибок)
xAllSlavesOk	BOOL	<b>TRUE</b> – во время последнего сеанса опроса были получены корректные ответы (без кодов ошибок) от всех slave-устройств данного компонента
<b>Компонент Modbus Slave</b>		
<i><b>Входы</b></i>		
xAcknowledge	BOOL	По переднему фронту выполняется переинициализация компонента без обнуления выходов <b>xError</b> и <b>byModbusError</b>

#### 4. Стандартные средства конфигурирования

Переменная	Тип	Описание
xDoInit	BOOL	Если вход имеет значение <b>TRUE</b> , то по переднему фронту входа <b>xReset</b> также происходит повторная отправка команд инициализации, заданных на вкладке <b>Modbus Slave Init</b>
xReset	BOOL	По переднему фронту выполняется переинициализация компонента с обнулением выходов <b>xError</b> и <b>byModbusError</b>
xTrigger	BOOL	По переднему фронту выполняется опрос всех каналов компонента, у которых параметр <b>Триггер</b> имеет значение <b>Передний фронт</b>
<b>Выходы</b>		
byModbusError	<a href="#">IoDrvModbus.MB_ErrorCodes</a>	Код ошибки обмена со slave-устройством
iChannelIndex	INT	Номер текущего опрашиваемого канала (нумерация с нуля)
xBusy	BOOL	<b>TRUE</b> – выполняется опрос канала
xDone	BOOL	<b>TRUE</b> – опрос текущего канала успешно завершен
xError	BOOL	Бит ошибки обмена со slave-устройством
xInitDone	BOOL	<b>TRUE</b> – отправлены все команды инициализации, заданные на вкладке <b>Modbus Slave Init</b>
<b>Компонент Modbus Serial Device</b>		
<b>Выходы</b>		
xInternalError	BOOL	<b>TRUE</b> – внутренняя ошибка компонента (например, ошибка выделения памяти)
<b>Компонент Modbus TCP Master</b>		
<b>Входы</b>		
xStop	BOOL	Если вход имеет значение <b>TRUE</b> , то опрос всех slave-устройств данного компонента прекращается
<b>Выходы</b>		
uiNumberOfCommunicatingSlaves	UINT	Число slave-устройств данного компонента, от которых во время последнего сеанса опроса были получены корректные ответы (без кодов ошибок)
xSlaveError	BOOL	<b>TRUE</b> – во время последнего сеанса опроса произошла ошибка обмена как минимум с одним из slave-устройств данного компонента (был получен ответ с кодом ошибки или устройство не ответило)

Переменная	Тип	Описание
<b>Компонент Modbus TCP Slave</b>		
<i><b>Входы</b></i>		
xConfirmError	BOOL	По переднему фронту выполняется переинициализация компонента
xDoInit	BOOL	Если вход имеет значение <b>TRUE</b> , то по переднему фронту входа <b>xReset</b> также происходит повторная отправка команд инициализации, заданных на вкладке <b>Modbus Slave Init</b>
<i><b>Выходы</b></i>		
byModbusError	<a href="#">IoDrvModbus.MB_ErrorCodes</a>	Код ошибки обмена со slave-устройством
ComSettings	IoDrvModbusTcp. ModbusTCPComSettings	Структура сетевых настроек опрашиваемого slave-устройства (IP-адрес и порт)
ComState	<a href="#">IoDrvModbusTcp. ModbusTCPComState</a>	Статус обмена со slave-устройством
iChannelIndex	INT	Номер текущего опрашиваемого канала (нумерация с нуля)
xBusy	BOOL	<b>TRUE</b> – выполняется опрос канала
xDone	BOOL	<b>TRUE</b> – опрос текущего канала успешно завершен
xError	BOOL	Бит ошибки обмена со slave-устройством
xInitDone	BOOL	<b>TRUE</b> – отправлены все команды инициализации, заданные на вкладке <b>Modbus Slave Init</b>
<b>Компонент Modbus TCP Slave Device</b>		
<i><b>Входы</b></i>		
Enable	BOOL	<b>TRUE</b> – компонент находится в работе, <b>FALSE</b> – компонент отключен
uiClientConnections	UINT	Число master-устройств, подключенных к компоненту
xInternalError	BOOL	<b>TRUE</b> – внутренняя ошибка компонента (например, ошибка выделения памяти)



Таблица 4.6.2 – Описание элементов перечисления MB\_ErrorCodes

Название	Значение	Описание
RESPONSE_SUCCESS	16#0	Отсутствие ошибок обмена
ILLEGAL_FUNCTION	16#1	Slave-устройство не поддерживает функцию Modbus, указанную в запросе
ILLEGAL_DATA_ADDRESS	16#2	Slave-устройство не содержит одного или нескольких регистров, указанных в запросе
ILLEGAL_DATA_VALUE	16#3	Данная команда записи является некорректной с точки зрения протокола Modbus
SLAVE_DEVICE_FAILURE	16#4	Во время выполнения запроса в slave-устройстве произошла внутренняя ошибка
ACKNOWLEDGE	16#5	Slave-устройство приняло запрос и обрабатывает его, но это потребует некоторого времени. Этот ответ предохраняет master-устройство от генерации ошибки таймаута
SLAVE_DEVICE_BUSY	16#6	Slave-устройство занято обработкой другой команды. Master-устройство должно повторить запрос позже, когда slave-устройство освободится
MEMORY_PARITY_ERROR	16#8	Произошла ошибка во время использования функции Modbus 20 или 21 (см. более подробную информацию в спецификации протокола)
GATEWAY_PATH_UNAVAILABLE	16#A	Ошибка конфигурации сетевого шлюза
GATEWAY_DEVICE_FAILED_TO_RESPONSE	16#B	Отправленный запрос не был получен сетевым шлюзом
RESPONSE_TIMEOUT	16#A1	В течение времен таймаута не был получен ответ от slave-устройства
RESPONSE_CRC_FAIL	16#A2	Контрольная сумма ответа некорректна
RESPONSE_WRONG_SLAVE	16#A3	Получен ответ от другого slave-устройства (не от того, которому был отправлен запрос)
RESPONSE_WRONG_FUNCTION_CODE	16#A4	Получен ответ с неверным кодом функции
REQUEST_FAILED_TO_SEND	16#A5	Ошибка COM-порта master-устройства. Запрос не был отправлен
RESPONSE_INVALID_DATA	16#A6	Ответ содержит данные, некорректные с точки зрения протокола Modbus (например, размер полученных данных не соответствует ожидаемому)
RESPONSE_INVALID_PROTOCOL	16#A7	Protocol ID в MBAP Header отличается от 0 (т.е. используемый протокол не является Modbus TCP)
RESPONSE_INVALID_HEADER	16#A8	MBAP Header является некорректным (для протокола Modbus TCP)
UNDEFINED	16#FF	Код ошибки не может быть определен (например, еще не было отправлено ни одного запроса)

Таблица 4.6.3 – Описание элементов перечисления ModbusTCPComState

Название	Значение	Описание
OFF	16#0	Никаких действий не выполняется (например, slave-устройство исключено из опроса)
CONNECTING	16#1	Устанавливается соединение со slave-устройством
CONNECTED	16#2	Соединение установлено
DISCONNECTING	16#3	Соединение разорвано (со стороны контроллера)
SOCKET_ERROR	16#4	Ошибка операций чтения/записи через сокет (например, из-за разрыва линии связи)

Ниже приведен пример использования переменных диагностики. В случае возникновения ошибки обмена с одним из slave-устройств запускается таймер, который каждую секунду подает импульс в переменную диагностики **xResetComPort**, что приводит к переинициализации COM-порта, соответствующего данному компоненту **Modbus Master**. Это может быть полезным, когда на линию связи действуют помехи (например, от преобразователей частоты), что может привести к остановке работы COM-порта контроллера.

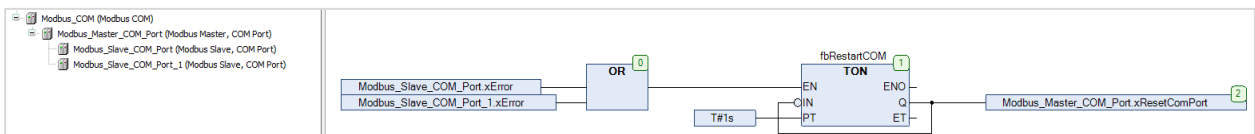


Рисунок 4.6.2 – Переинициализация COM-порта

Если контроллер работает в режиме **Modbus Serial Master** или **Modbus TCP Master**, то опросом slave-устройств можно управлять из кода программы. Для этого следует:

1. В компоненте **Device** на вкладке **Установки ПЛК** установить галочку **Включить диагностику для устройств**.

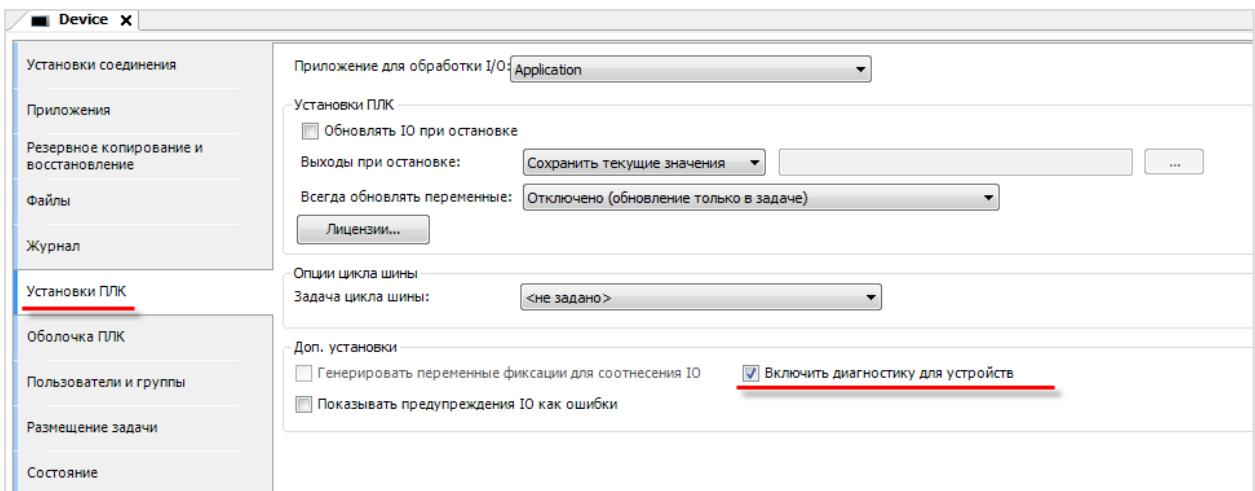


Рисунок 4.6.3 – Включение диагностики устройств

#### 4. Стандартные средства конфигурирования

2. Для отключения устройства из опроса в коде программы присвоить переменной <имя\_устройства\_из\_дерева\_проекта>.Enable значение **FALSE**.



Рисунок 4.6.4 – Исключение slave-устройств из опроса

3. Для возобновления опроса присвоить переменной <имя\_устройства\_из\_дерева\_проекта>.Enable значение **TRUE**.

Если контроллер работает в режиме **Modbus Serial Master**, то можно изменить заданный адрес опрашиваемого slave-устройства из кода программы через свойство **SlaveAddress**.



Рисунок 4.6.5 – Изменение заданного адреса опрашиваемого slave-устройства

После перезагрузки контроллера адрес опрашиваемого slave-устройства будет инициализирован значением, заданным в настройках компонента (вкладка **Общее**), поэтому процедуру потребуется провести заново.

Если контроллер работает в режиме **Modbus TCP Master**, то можно изменить заданные сетевые настройки опрашиваемого slave-устройства с помощью метода **UpdateCommunicationSettings**:

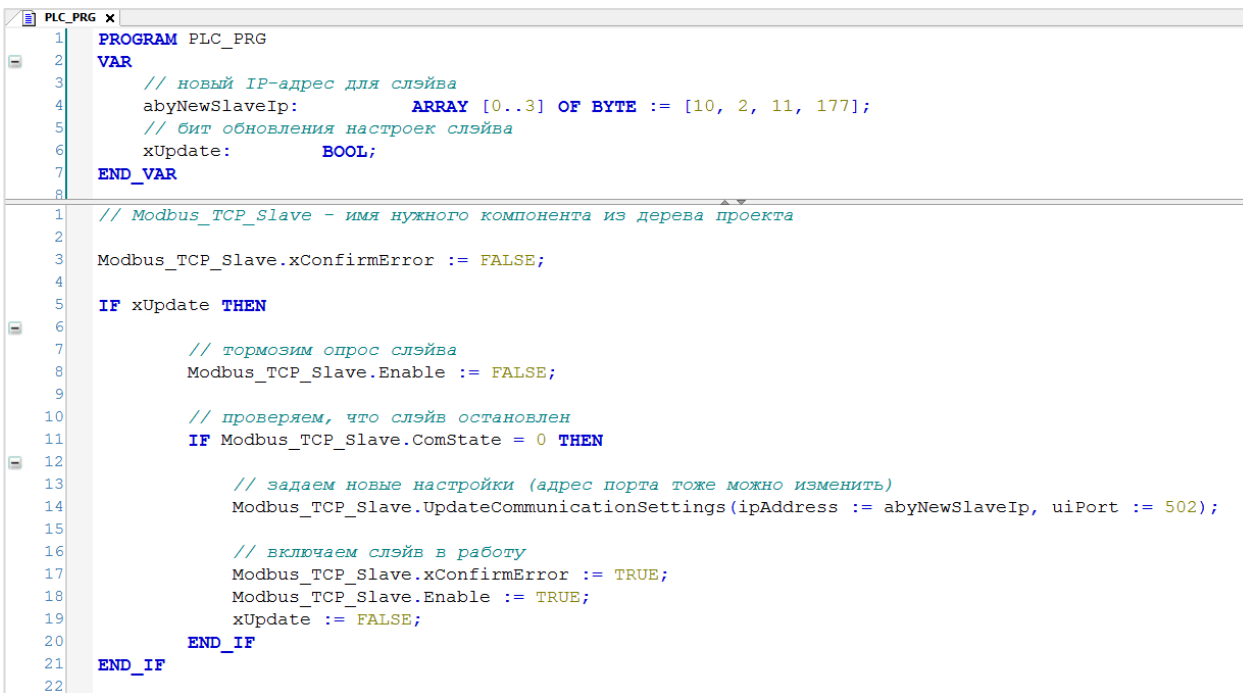


Рисунок 4.6.6 – Изменение заданных сетевых настроек опрашиваемого slave-устройства

После перезагрузки контроллера настройки slave-устройства будут инициализированы значениями, заданными в настройках компонента (вкладка **Общее**), поэтому процедуру смены настроек потребуется провести заново.

Если контроллер работает в режиме **Modbus TCP Slave**, то можно изменить номер сетевого порта, используемого компонентом, с помощью метода **UpdateCommunicationSetting**:

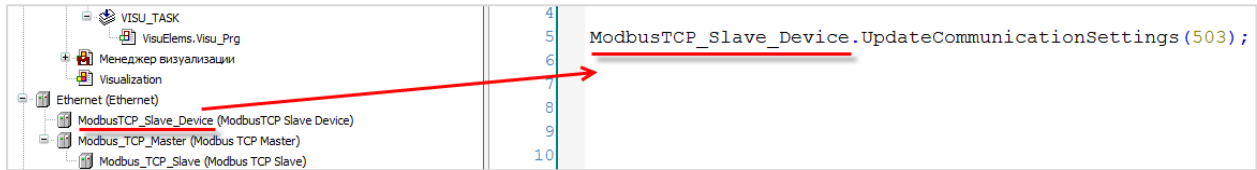


Рисунок 4.6.7 – Изменение номера порта в режиме Modbus TCP Slave

После перезагрузки контроллера номер порта будет инициализирован значением, заданным в настройках компонента (вкладка **Общие**), поэтому процедуру потребуется провести заново.

## 4.7 Компоненты Modbus и конфигурация задач

По умолчанию компоненты Modbus выполняются в задаче с наименьшим временем цикла. Для корректной работы компонентов в проекте должна присутствовать хотя бы одна задача с временем цикла **10...20 мс**. Более подробная информация по этому поводу приведена в [справке CODESYS](#).

## 4.8 Преобразование данных для передачи по Modbus

Стандарт **Modbus** описывает только два типа данных – **BOOL** и **WORD**. Достаточно часто возникает потребность передать данные других типов (например, **REAL** и **STRING**). В данном случае на устройстве, которое отправляет данные, следует преобразовать их в последовательность переменных типа **WORD**. Соответственно, на устройстве, получающем данные, должно быть выполнено обратное преобразование.

В **CODESYS V3.5** есть два базовых способа для подобных преобразований: [объединения](#) и [указатели](#). Кроме того, можно воспользоваться [функциями конвертации](#) из библиотеки **OwenCommunication**.

### 4.8.1 Использование объединений (UNION)

**Объединение (UNION)** представляет собой пользовательский тип данных, все переменные которого расположены в одной области памяти. Таким образом, переменные различных типов будут представлять различную интерпретацию одних и тех же данных. Для конвертации достаточно записать значение в одну из переменных объединения и считать его из другой.

Для конвертации значения с плавающей точкой из двух переменных типа **WORD** в переменную типа **REAL** следует:

1. Нажать **ПКМ** на приложение **Application** и добавить объект **DUT** типа **объединение** с названием **Real\_Word**:

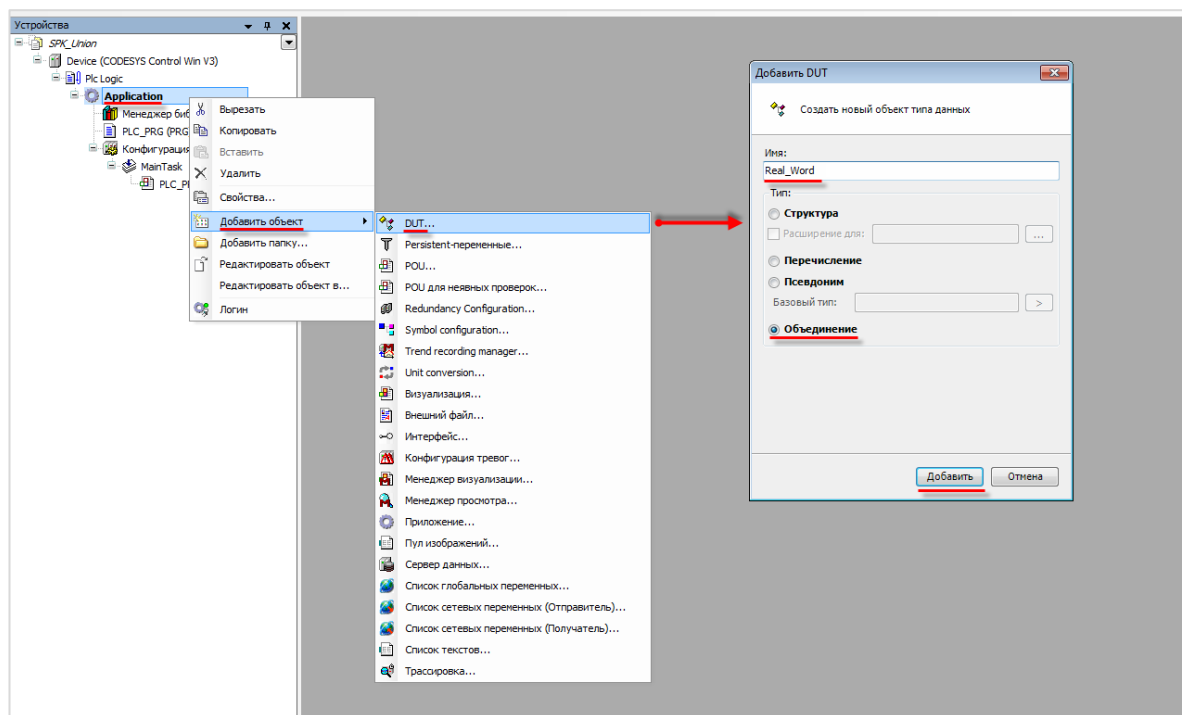


Рисунок 4.8.1 – Добавление объединения в проект CODESYS

2. В объединении объявить переменную **rRealValue** типа **REAL** и массив **awModbusReal** типа **WORD**, содержащий два элемента:

```

Real_Word x
1  TYPE Real_Word :
2  UNION
3      rRealValue      :REAL;
4      awModbusReal    :ARRAY [0..1] OF WORD;
5  END_UNION
6  END_TYPE

```

Рисунок 4.8.2 – Объявление переменных объединения

3. В программе объявить экземпляр объединения **Real\_Word** с названием **\_2WORD\_TO\_REAL**:

```

PLC_PRG x
1  PROGRAM PLC_PRG
2  VAR
3      _2WORD_TO_REAL: Real_Word;
4  END_VAR
5

```

Рисунок 4.8.3 – Объявление экземпляра объединения в программе

Для использования переменных объединения в нужном месте программы следует ввести имя экземпляра объединения и нажать точку, после чего выбрать из списка нужную переменную:

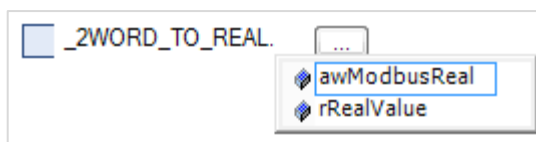


Рисунок 4.8.4 – Работа с переменными объединения в программе

4. Переменные массива **awModbusReal** будут привязаны к регистрам во время настройки **Modbus**, а переменная **rRealValue** будет использоваться в программе.

Ниже приведен пример: контроллер является мастером и считывает значение типа **REAL** из 10 и 11 holding регистров slave-устройства в переменную **rRealValue** объединения **\_2WORD\_TO\_REAL**:

Modbus_Slave_COM_Port x									
Имя	Тип доступа	Триггер	Сдвиг READ	Длина	Обработка ошибок	Сдвиг WRITE	Длина	Комментарий	
0 Channel 0	Read Holding Registers (Код функции 03)	Цикл., t#100ms	16#000A	2	Сохранить посл. значение				

Рисунок 4.8.5 – Настройка канала для считывания значения с плавающей точкой

Modbus_Slave_COM_Port x							
Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание	
Application.PLC_PRG._2WORD_TO_REAL.awModbusReal		Channel 0	%IWB	ARRAY [0..1] OF WORD		Read Holding Registers	
		Channel 0[0]	%IWB	WORD		0x000A	
		Channel 0[1]	%IWB	WORD		0x000B	

Рисунок 4.8.6 – Привязка переменной объединения к каналу

Выражение	Тип	Значение
_2WORD_TO_REAL	REAL_WORD	
rRealValue	REAL	3.3
awModbusReal	ARRAY [0..1] OF WO...	
awModbusReal[0]	WORD	16#3333
awModbusReal[1]	WORD	16#4053

Рисунок 4.8.7 – Использование REAL переменной объединения в программе

Обратное преобразование (**REAL** в два **WORD**) выполняется аналогичным способом: пользователь записывает значение в **REAL** переменную объединения, после чего работает с массивом из двух **WORD**.

Работа с **DWORD**, **STRING** и другими типами данных происходит аналогично – в приведенном выше примере достаточно изменить тип переменной объединения (вместо **rRealValue** использовать **dwDwordValue** типа **DWORD**, **sStringValue** типа **STRING** и так далее).

5. Передача **REAL** по протоколу **Modbus** не стандартизирована – значение с плавающей точкой передаются в виде двух регистров (переменных типа **WORD**), но порядок этих **WORD** переменных (или даже их байт) может отличаться. В данном случае следует привести их к нужному для конкретного устройства виду.

Порядок **WORD** можно менять на этапе привязки переменных к регистрам – например, можно сравнить рисунки 4.8.6 и 4.8.8:

Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
Application.PLC_PRG._2WORD_TO_REAL.awModbusReal[0]	↔	Channel 0	%IWO	ARRAY [0..1] OF WORD		Read Holding Registers
Application.PLC_PRG._2WORD_TO_REAL.awModbusReal[1]	↔	Channel 0[1]	%IWO	WORD		0x000A
		Channel 0[1]	%IWO	WORD		0x000B

Рисунок 4.8.8 – Привязка элементов массива объединения к регистрам канала

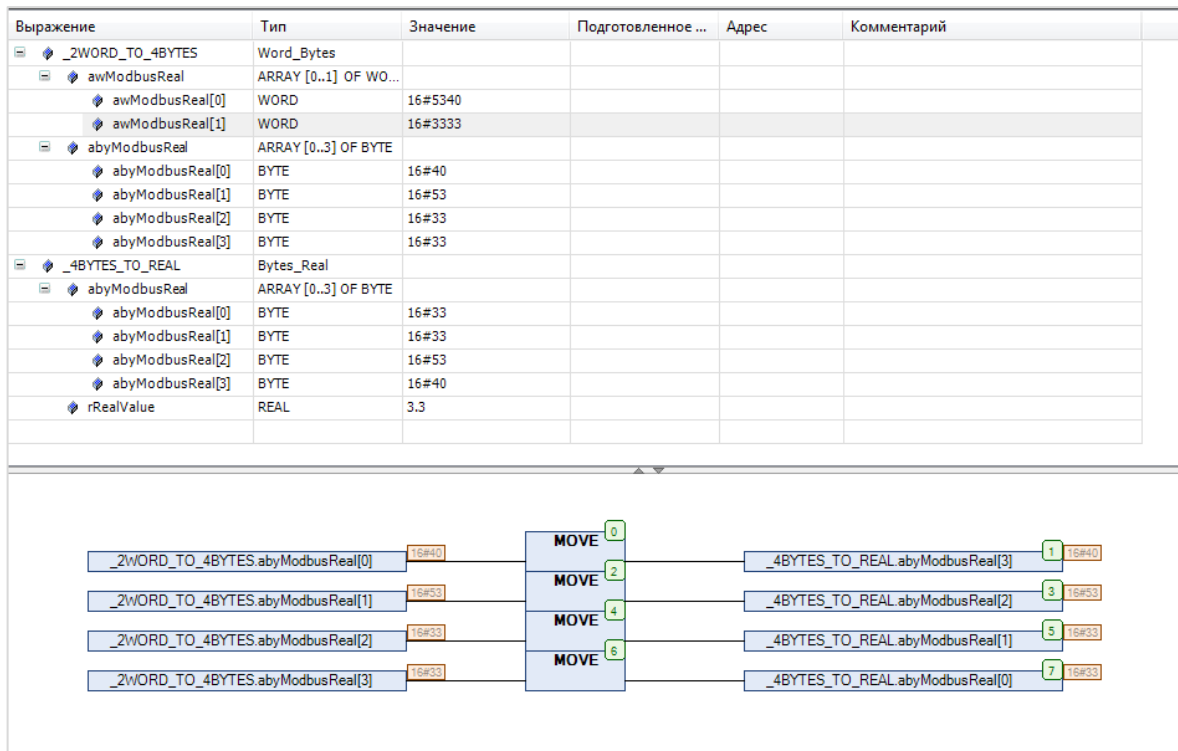
В случае необходимости изменения порядка байтов можно создать два объединения – в первом будет происходить конвертация полученных по **Modbus** значений **WORD** в массив байтов, а во втором – конвертация нового массива байтов (переставленных в нужном порядке) в переменную типа **REAL**. Ниже приведен пример конвертации 2 **WORD** в **REAL** с перестановкой байт (0–1–2–3 в 3–2–1–0):

```

1  TYPE Word_Bytes :
2  UNION
3      awModbusReal   :ARRAY [0..1] OF WORD;
4      abyModbusReal  :ARRAY [0..3] OF BYTE;
5  END_UNION
6  END_TYPE

1  TYPE Bytes_Real :
2  UNION
3      abyModbusReal   :ARRAY [0..3] OF BYTE;
4      rRealValue      :REAL;
5  END_UNION
6  END_TYPE
    
```

Рисунок 4.8.9 – Объявление двух объединений



**Рисунок 4.8.10 – Пример работы с объединениями на языке CFC. Перестановка байтов**

На основе приведенных примеров пользователь может создать свои функции и функциональные блоки для удобной конвертации данных.



### 4.8.2 Использование указателей

**Указатели** содержат адреса переменных. Обращаясь к переменной по указателю, пользователь работает непосредственно с областью памяти, в которой хранится эта переменная, что позволяет производить любую обработку находящихся в ней данных.



**ПРИМЕЧАНИЕ**

Использование указателей подразумевает соответствующую квалификацию программиста. Некорректное использование указателей может привести к «зависанию» программы и контроллера.

Для конвертации значения с плавающей точкой из двух переменных типа **WORD** в переменную типа **REAL** следует:

1. Объявить в программе нужные переменные и указатель на переменную того типа, в который производится конвертация:

```

2  VAR
3      rRealValue      :REAL;
4      awModbusReal   :ARRAY [0..1] OF WORD;
5      prModbusReal   :POINTER TO REAL;
6  END_VAR
    
```

**Рисунок 4.8.11 – Объявление указателя**

2. Переменные массива **awModbusReal** будут привязаны к регистрам при настройке **Modbus**.

Ниже приведен пример: контроллер является мастером и считывает значение типа **REAL** из 10 и 11 holding регистров slave-устройства в переменную **rRealValue**:

Имя	Тип доступа	Триггер	Сдвиг READ	Длина	Обработка ошибок	Сдвиг WRITE	Длина	Комментарий
0 Channel 0	Read Holding Registers (Код функции 03)	Цикл., t=100ms	16#000A	2	Сохранить посл. значение			

**Рисунок 4.8.12 – Настройка канала для считывания значения с плавающей точкой**

Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
Application.PLC_PRG.awModbusReal		Channel 0	%DWB	ARRAY [0.. 1] OF WORD		Read Holding Registers
		Channel 0[0]	%DWB	WORD		0x000A
		Channel 0[1]	%DWB	WORD		0x000B

**Рисунок 4.8.13 – Привязка переменной к каналу**

3. В программе с помощью оператора **ADR** записать в указатель адрес массива **awModbusReal**, после чего присвоить переменной **rRealValue** значение, хранящееся по указателю:

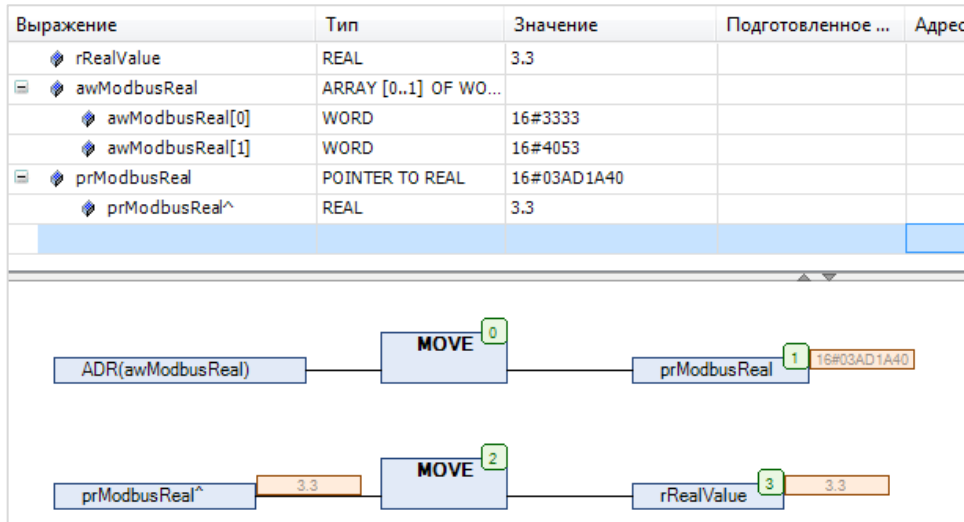


Рисунок 4.8.14 – Пример работы с указателями на языке CFC (конвертация 2 WORD в REAL)

Обратное преобразование (**REAL** в два **WORD**) выполняется аналогичным способом:

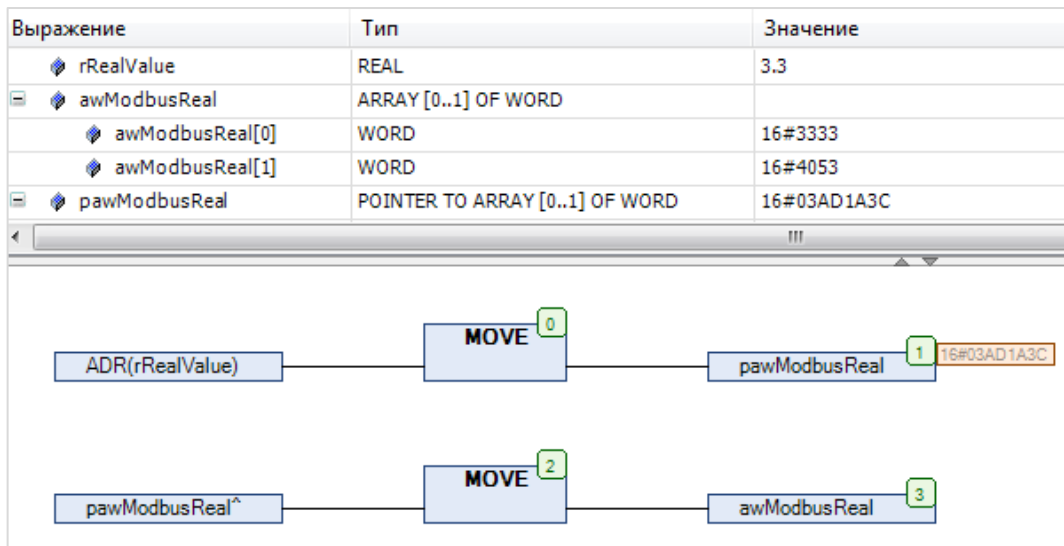


Рисунок 4.8.15 – Пример работы с указателями на языке CFC (конвертация REAL в 2 WORD)

Работа с **DWORD**, **STRING** и другими типами данных происходит аналогично – в приведенном выше примере достаточно изменить тип переменной объединения и ее указателя (например, использовать **dwDwordValue** типа **DWORD** и **pdwModbusDword** типа **POINTER TO DWORD**).

#### 4. Стандартные средства конфигурирования

4. Передача **REAL** по протоколу **Modbus** не стандартизирована – значение с плавающей точкой передаются в виде двух регистров (переменных типа **WORD**), но порядок этих **WORD** переменных (или даже их байт) может отличаться. В данном случае следует привести их к нужному для конкретного устройства виду.

Порядок **WORD** можно менять на этапе привязки переменных к регистрам – например, можно сравнить рисунки [4.8.14](#) и 4.8.16:

Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
Application.PLC_PRG.awModbusReal[1]		Channel 0	%IW0	ARRAY [0..1] OF WORD		Read Holding Registers
Application.PLC_PRG.awModbusReal[0]		Channel 0	%IWI	WORD		0x000A

Рисунок 4.8.16 – Привязка элементов массива к регистрам канала

В случае необходимости изменения порядка байтов следует вместо массива из двух **WORD** использовать массив из четырех байт и указатель на него. Ниже приведен пример конвертации 2 **WORD** в **REAL** с перестановкой байт (0–1–2–3 в 3–2–1–0):

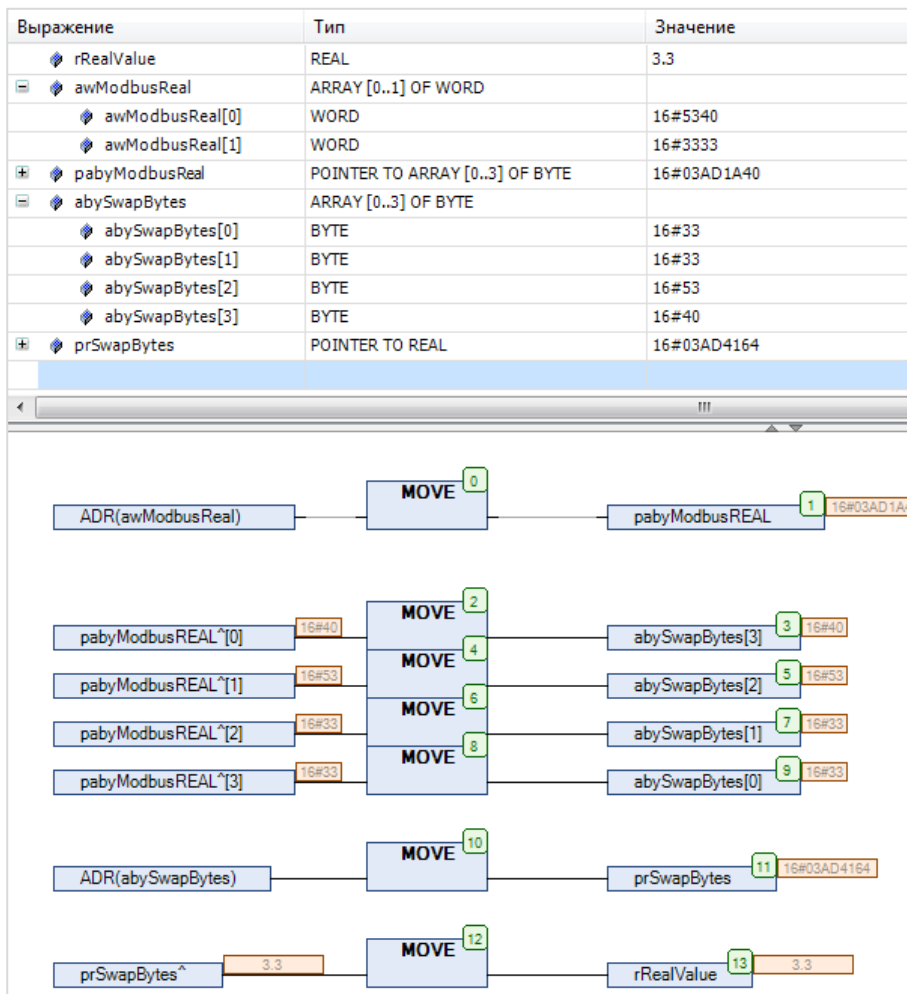


Рисунок 4.8.17 – Пример работы с указателями на языке CFC. Перестановка байт

## 4.9 Пример: СПК1хх [M01] (Modbus RTU Master) + модули Mx110

В качестве примера будет рассмотрена настройка обмена с модулями [Mx110](#) (MB110-8A, MB110-16Д, МУ110-16P) с использованием **стандартных средств конфигурации**.

**Реализуемый алгоритм:** если значение первого аналогового входа модуля **MB110-8A** превышает **30** и при этом первый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **МУ110-16P** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).

Структурная схема примера приведена на рисунке ниже:

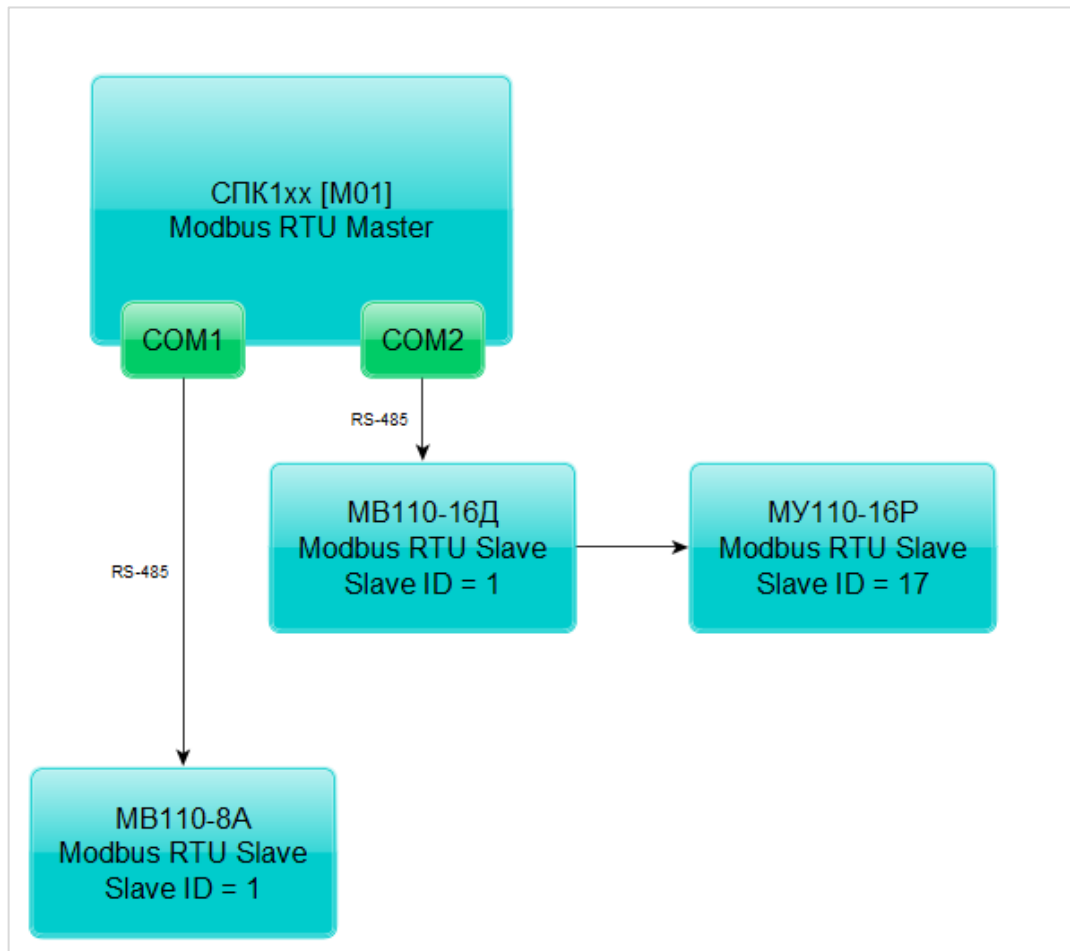


Рисунок 4.9.1 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (**ПКМ** на узел **Device** – **Обновить устройство**).

Пример доступен для скачивания: [Example.CodesysModbusRtuMasterMx110\\_3511v1.projectarchive](http://Example.CodesysModbusRtuMasterMx110_3511v1.projectarchive)

#### 4. Стандартные средства конфигурирования

Сетевые параметры модулей приведены в таблице ниже:

**Таблица 4.9.1 – Сетевые параметры модулей Mx110**

Параметр	MB110-8A	MB110-16Д	МУ110-16Р
COM-порт контроллера, к которому подключен модуль	COM1	COM2	
ID COM-порта	1	2	
Адрес модуля	1	1	17
Скорость обмена	115200		
Количество бит данных	8		
Контроль четности	Отсутствует		
Количество стоп-бит	1		

Переменные примера описаны в таблице ниже:

**Таблица 4.9.2 – Список переменных примера**

Модуль	Имя переменной	Тип	Описание
MB110-8A	awModbusReal	ARRAY [0..1] OF WORD	Значение температуры в виде двух <b>WORD</b> , считываемое с модуля
	rRealValue	REAL	Значение температуры в виде числа с плавающей точкой для использования в программе
MB110-16Д	wDI	WORD	Значение дискретных входов в виде битовой маски. При обращении к отдельным входам указывается их номер, начиная с 0: <b>wDI.0</b> – состояние первого входа (TRUE/FALSE) <b>wDI.1</b> – состояние второго входа ...
МУ110-16Р	wDO	WORD	Значение дискретных выходов в виде битовой маски. При обращении к отдельным выходам указывается их номер, начиная с 0: <b>wDO.0</b> – состояние первого выхода (TRUE/FALSE) <b>wDO.1</b> – состояние второго выхода ...
-	wPrevDO	WORD	Значение дискретных выходов в виде битовой маски из предыдущего цикла программы. Используется для отправки команды записи только в случае изменения значений выходов (иначе будет производиться циклическая запись последнего значения)
-	xTrigger	BOOL	Триггерная переменная, управляющая функцией записи дискретного выхода (запись происходит по переднему фронту переменной)

Для настройки обмена следует:

1. Настроить модули **Mx110** с помощью программы **Конфигуратор Mx110** в соответствии с [таблицей 4.9.1](#). Подключить модули к COM-портам контроллера в соответствии с [рисунком 4.9.1](#).
2. Создать новый проект **CODESYS** с программой **PLC\_PRG** на языке **CFC**:

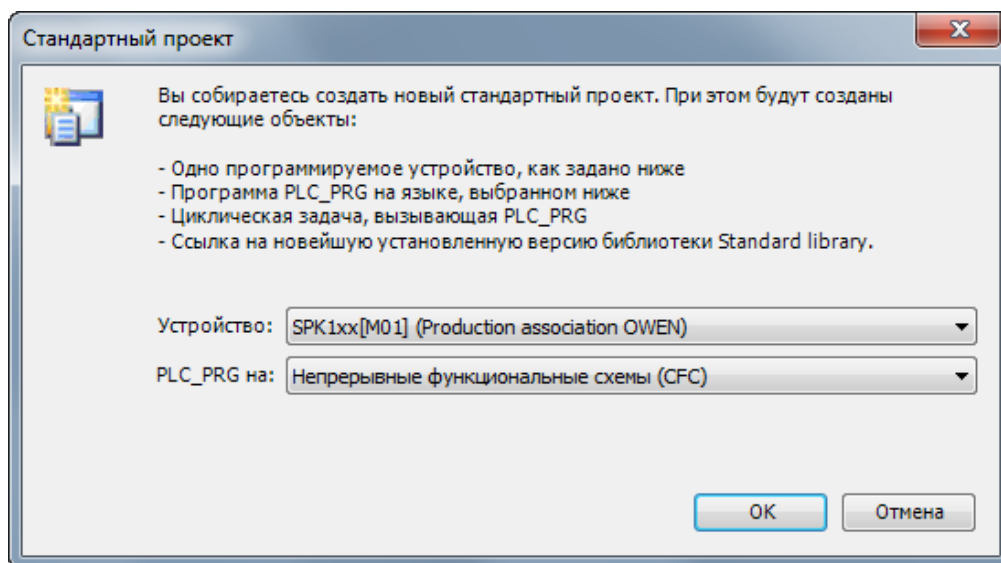


Рисунок 4.9.2 – Создание проекта CODESYS

3. Добавить в проект [объединение](#) с именем **Real\_Word**:

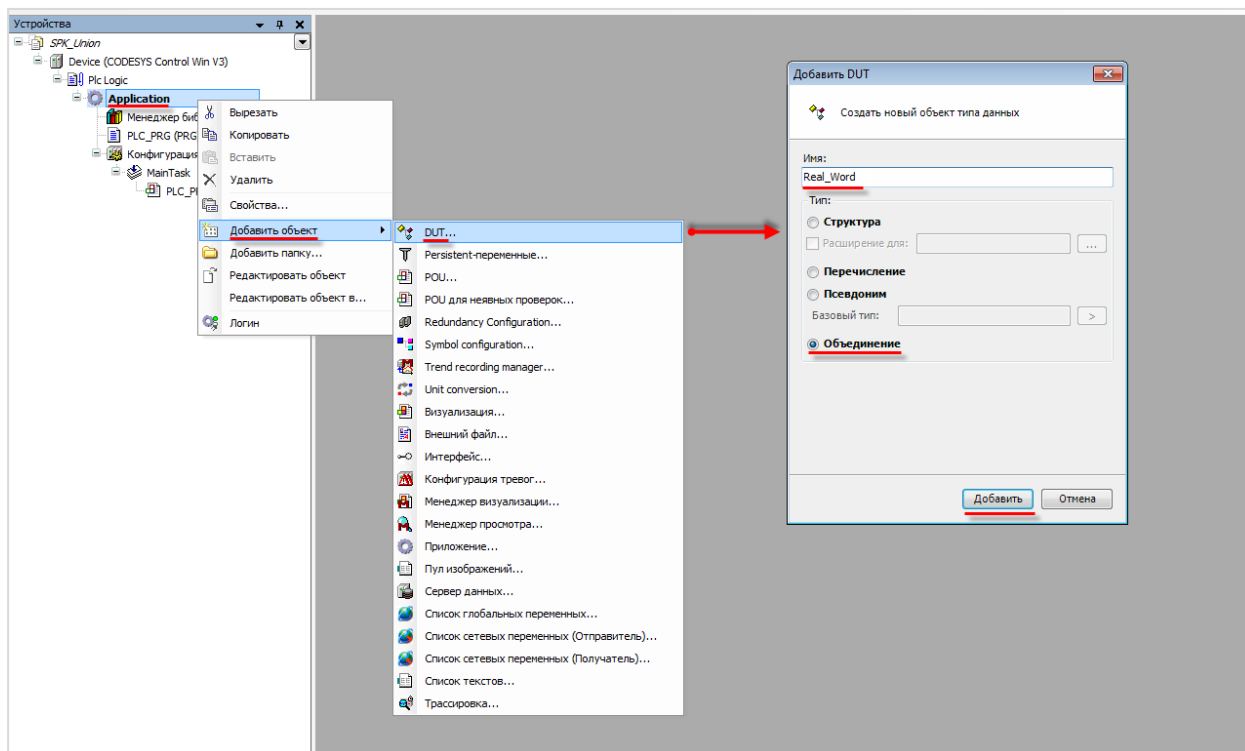


Рисунок 4.9.3 – Добавление в проект объединения

#### 4. Стандартные средства конфигурирования

4. В объединении объявить переменную **rRealValue** типа **REAL** и массив **awModbusReal** типа **WORD**, содержащий два элемента:

```
Real_Word x
1  TYPE Real_Word :
2  UNION
3      rRealValue      :REAL;
4      awModbusReal   :ARRAY [0..1] OF WORD;
5  END_UNION
6  END_TYPE
```

Рисунок 4.9.4 – Объявление переменных объединения

5. В программе **PLC\_PRG** объявить экземпляр объединения **Real\_Word** с названием **\_2WORD\_TO\_REAL**, переменные **wDI**, **wDO** и **wPrevDO** типа **WORD** и переменную **xTrigger** типа **BOOL**. Описание переменных приведено в [таблице 4.9.2](#).

```
PLC_PRG x
1  PROGRAM PLC_PRG
2  VAR
3      _2WORD_TO_REAL: Real_Word; // значение 1-го входа MB110-8A
4      wDI: WORD; // битовая маска входов MB110-16Д
5      wDO: WORD; // битовая маска выходов MV110-16P
6      wPrevDO: WORD; // битовая маска предыдущей записи выходов MV110-16P
7      xTrigger: BOOL; // триггер записи выходов
8  END_VAR
```

Рисунок 4.9.5 – Объявление переменных программы

Код программы будет выглядеть следующим образом:

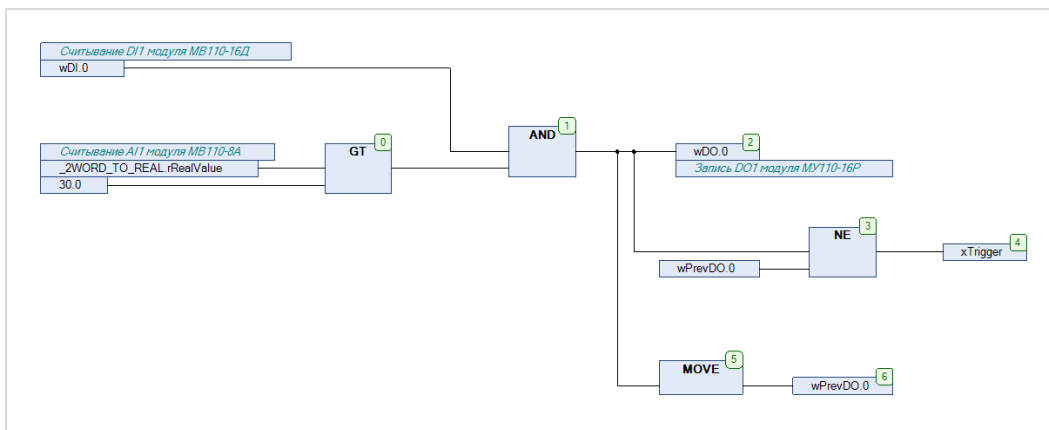


Рисунок 4.9.6 – Код программы PLC\_PRG

Программа работает следующим образом: если значение переменной **rRealValue** (связанной с первым аналоговым входом модуля **MB110-8A**) превышает 30 и при этом значение нулевого бита переменной **wDI** (связанной с первым дискретным входом модуля **MB110-16Д**) имеет значение **TRUE**, то нулевому биту переменной **wDO** присваивается значение **TRUE**. Если на предыдущем цикле значение нулевого бита **wDO** отличалось от текущего, то переменная **xTrigger** принимает значение **TRUE**, что приводит к однократной записи текущего значения бита в первый дискретный выход модуля **MV110-16P**.

6. Добавить в проект два компонента **Modbus COM** с названиями **COM1** и **COM2**.**ПРИМЕЧАНИЕ**

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

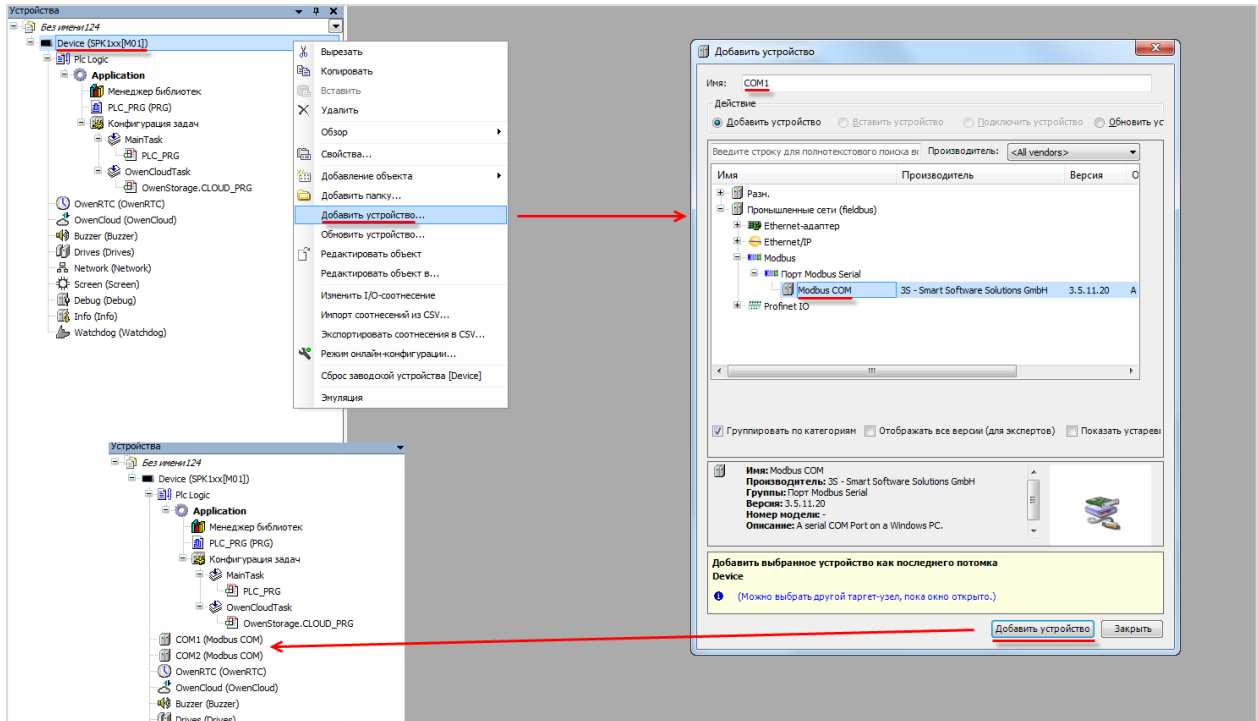


Рисунок 4.9.7 – Добавление компонента Modbus COM

В конфигурации COM-портов следует указать [номера COM-портов](#) и сетевые настройки в соответствии с [таблицей 4.9.1](#):

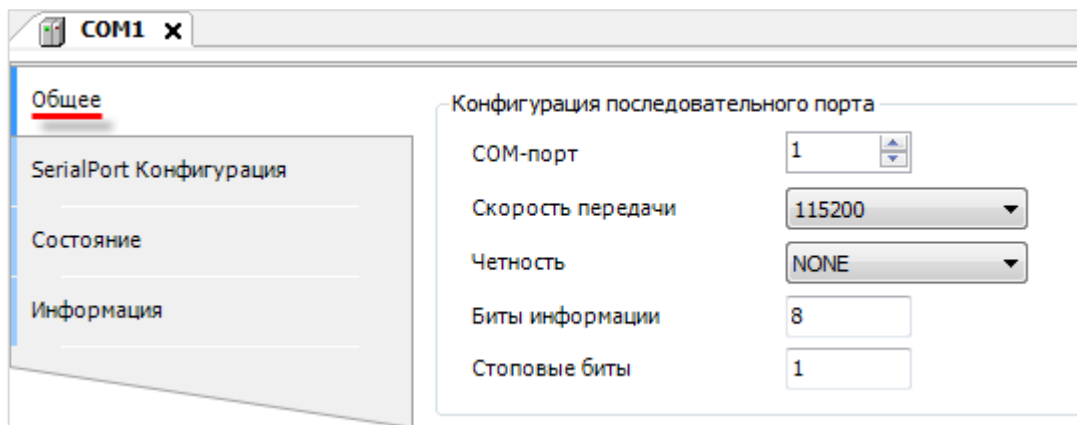


Рисунок 4.9.8 – Настройки COM-порта COM1

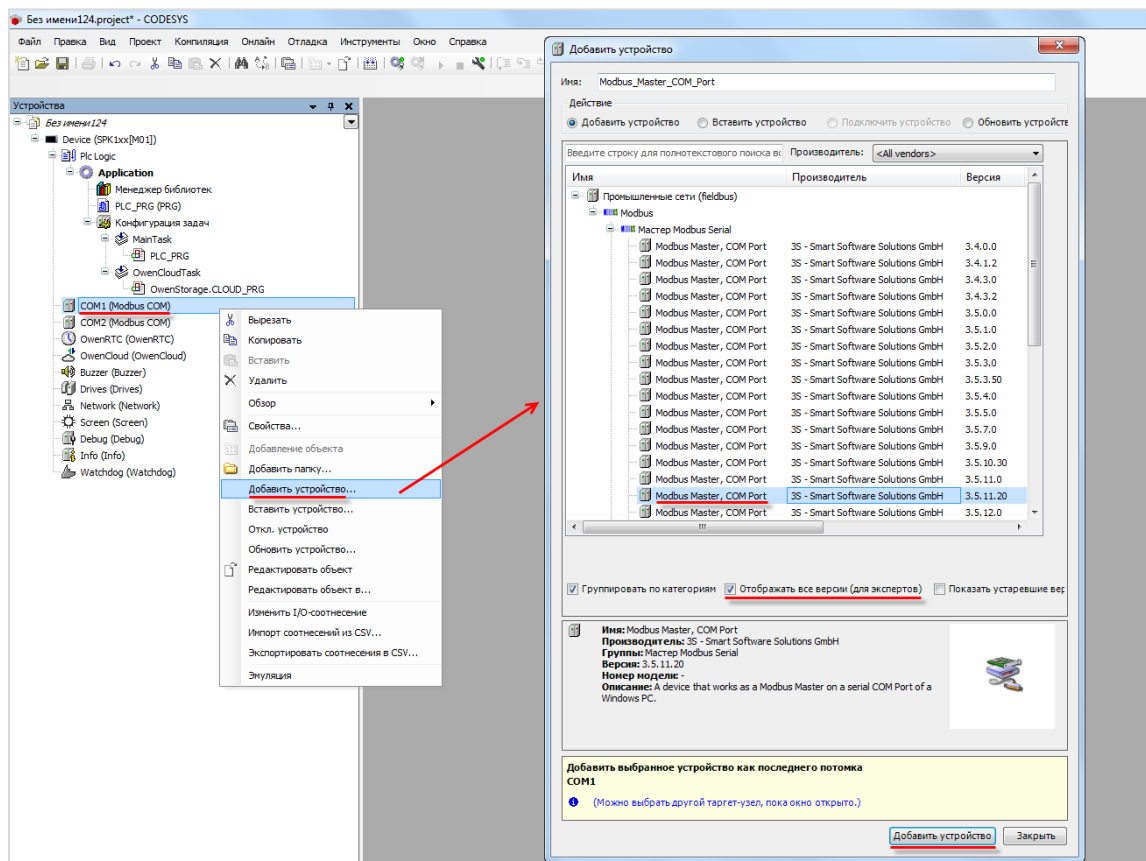


7. В каждый из COM-портов добавить компонент **Modbus Master**.

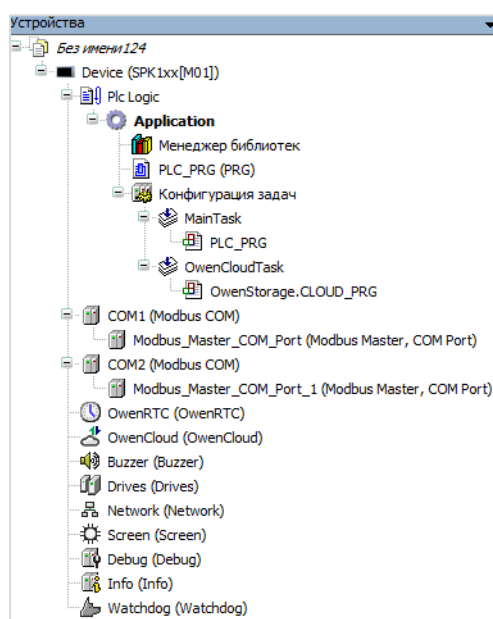


**ПРИМЕЧАНИЕ**

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).



**Рисунок 4.9.9 – Добавление компонента Modbus Master**



**Рисунок 4.9.10 – Внешний вид дерева проекта после добавления Modbus Master**

В настройках компонентов на вкладке **Общее** следует установить галочку **Автоперезапуск соединения**. В параметре **Время между фреймами** установить значение **20 мс**.

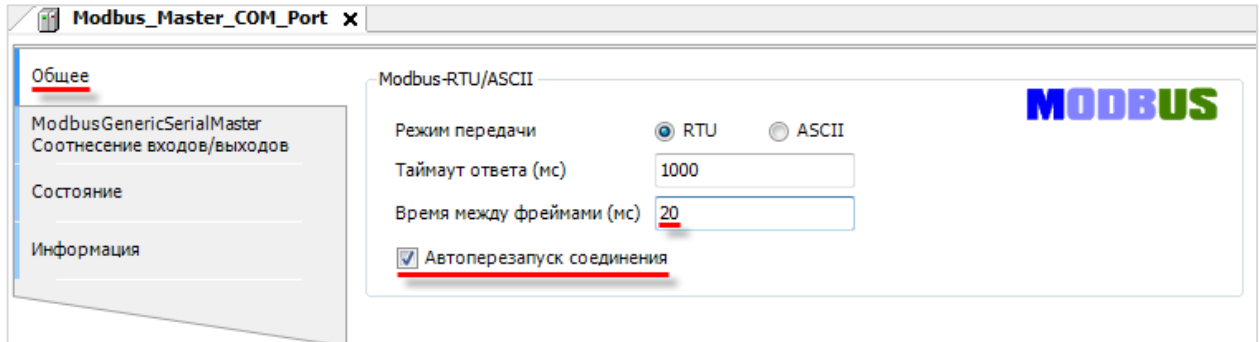


Рисунок 4.9.11 – Настройки компонентов Modbus Master

8. В компонент **Modbus Master** порта **COM1** следует компонент **Modbus Slave** с именем **MV110\_8A**, а в компонент **Modbus Master** порта **COM2** – компоненты **Modbus Slave** с именами **MV110\_16D** и **MU110\_16R**.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

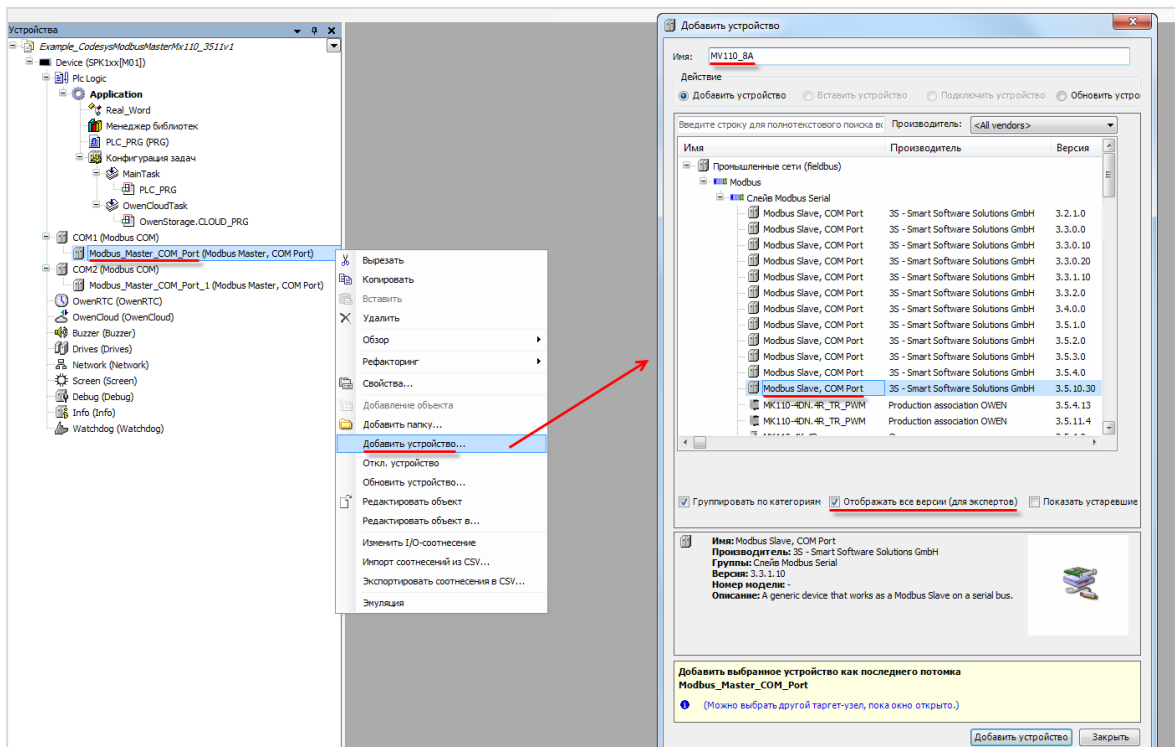


Рисунок 4.9.12 – Добавление slave-устройств в проект CODESYS

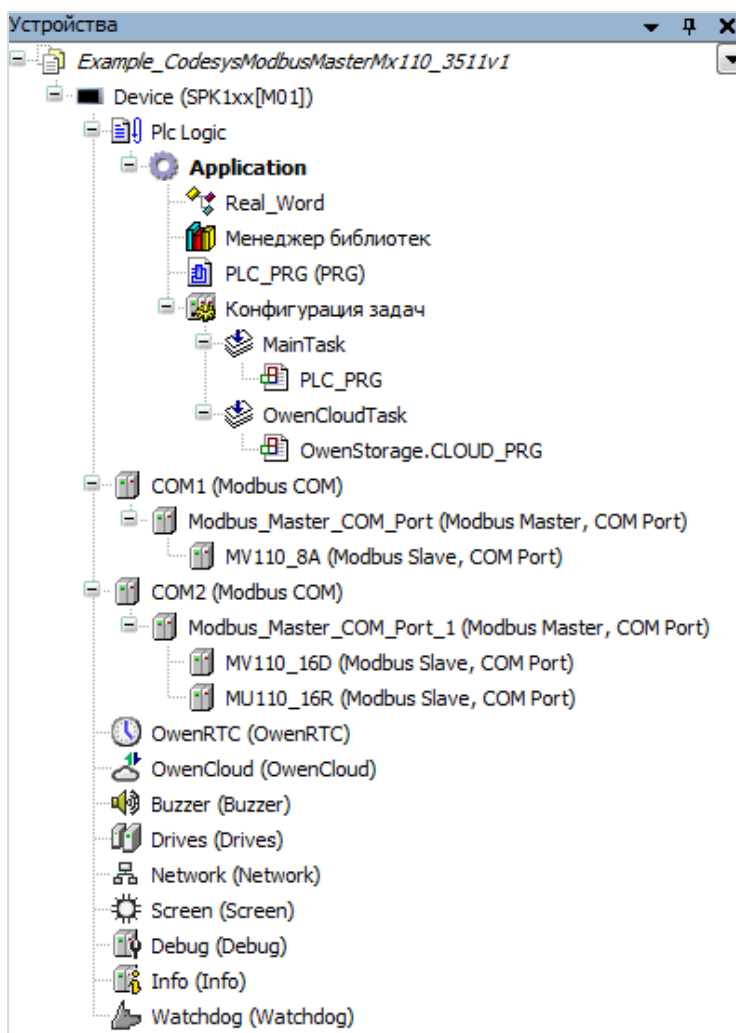


Рисунок 4.9.13 – Внешний вид дерева проекта после добавления slave-устройств

В настройках slave-устройств следует указать их адреса согласно [таблице 4.9.1](#) (MB110-8A – адрес 1, MB110-16Д – адрес 1, МУ110-16Р – адрес 17):

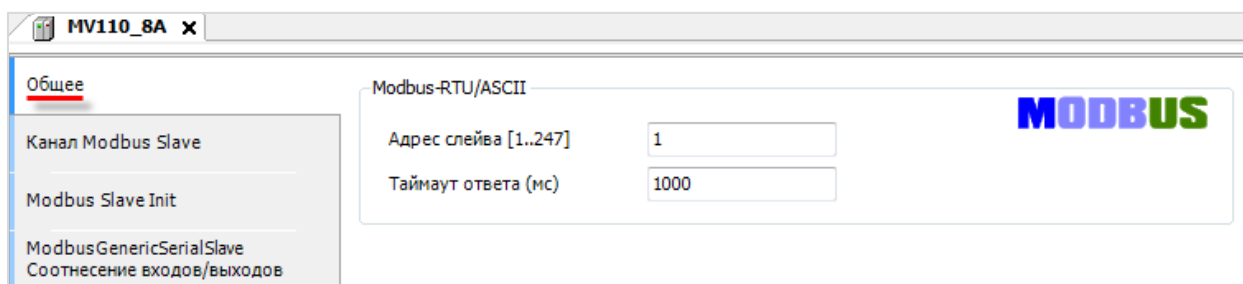


Рисунок 4.9.14 – Настройка slave-устройства MV110\_8A

9. В настройках компонента **MV110\_8A** на вкладке **Канал Modbus Slave** следует добавить канал, в котором с помощью функции **Read Holding Registers** будет считываться значение **четвертого** и **пятого** регистров модуля. В данных регистрах содержится значение входа 1 в представлении с плавающей точкой. Таблица регистров модуля и поддерживаемые функции приведены в руководстве по эксплуатации.

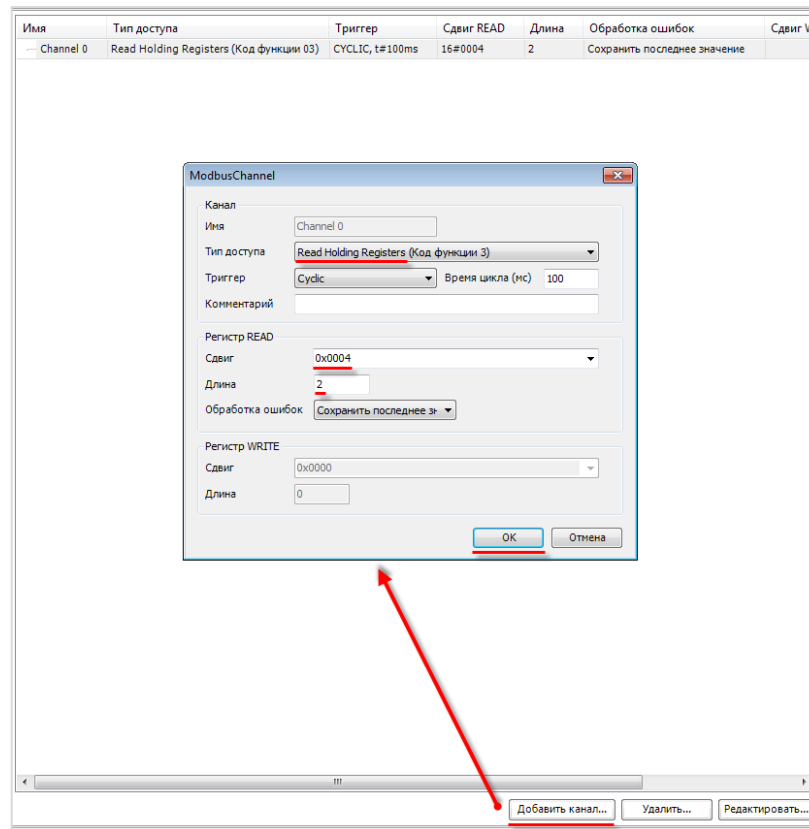


Рисунок 4.9.15 – Добавление канала в конфигурацию slave-устройства MV110\_8A

На вкладке **ModbusGenericSerialSlave** **Соотнесение входов/выходов** следует привязать к каналу элементы объединения **\_2WORD\_TO\_REAL**. Первый считываемый регистр присваивается первому элементу массива, а второй – нулевому. Это связано с тем, что порядок **WORD** в **REAL** у модуля и контроллера отличается.

Для параметра **Всегда обновлять переменные** следует установить значение **Включено 2**.

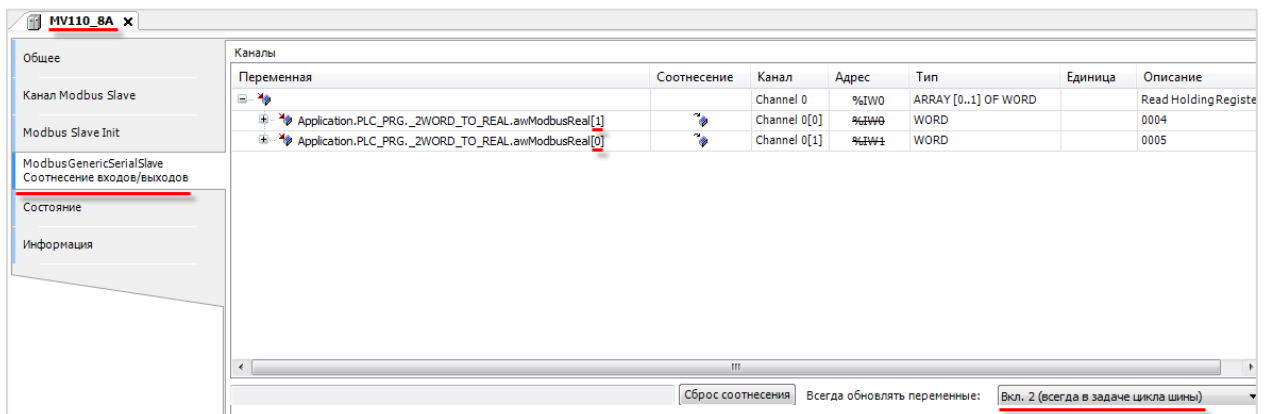


Рисунок 4.9.16 – Привязка переменных к каналу

10. В настройках компонента **MV110\_16D** на вкладке **Канал Modbus Slave** следует добавить канал, в котором с помощью функции **Read Holding Registers** будет считываться значение регистра **0x0033**. В данном регистре содержится битовая маска состояний дискретных входов.

Таблица регистров модуля и поддерживаемые функции приведены в руководстве по эксплуатации.

#### 4. Стандартные средства конфигурирования

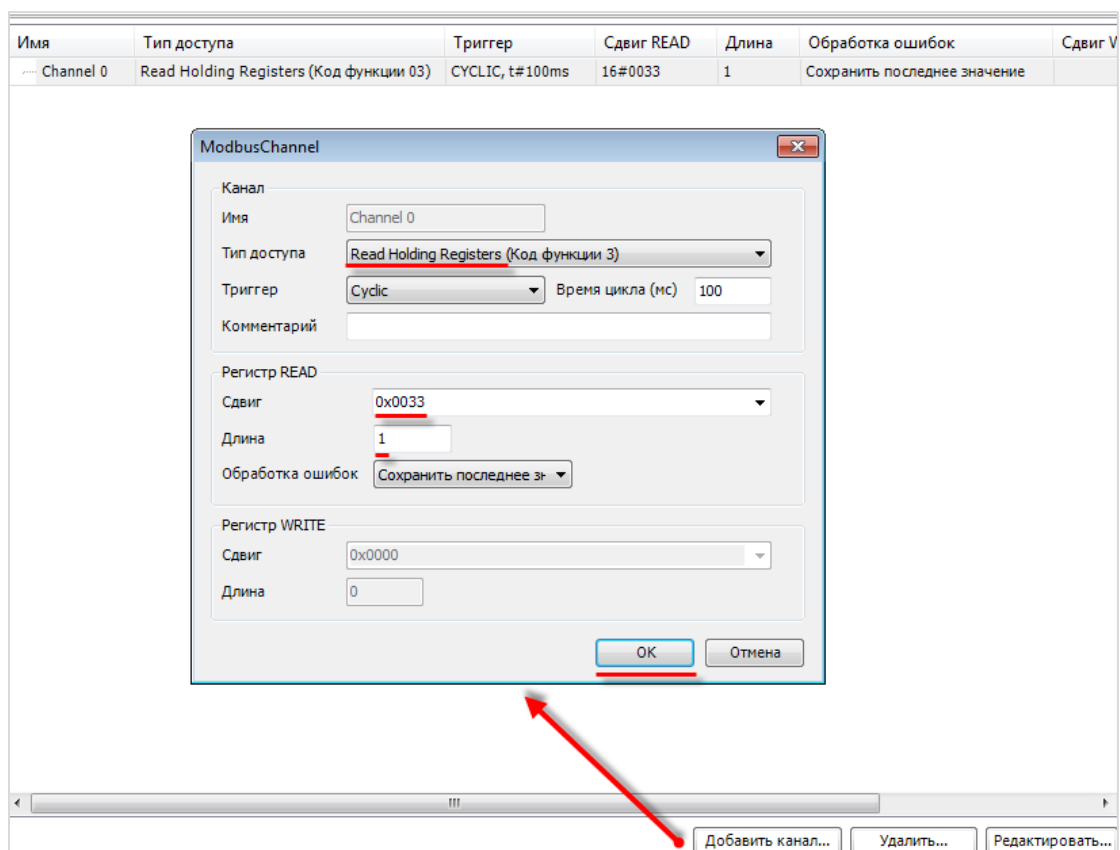


Рисунок 4.9.17 – Добавление канала в конфигурацию slave-устройства MV110\_16D

На вкладке **ModbusGenericSerialSlave** **Соотнесение входов/выходов** следует к каналу переменную **wDI**. Для параметра **Всегда обновлять переменные** следует установить значение **Включено 2**.

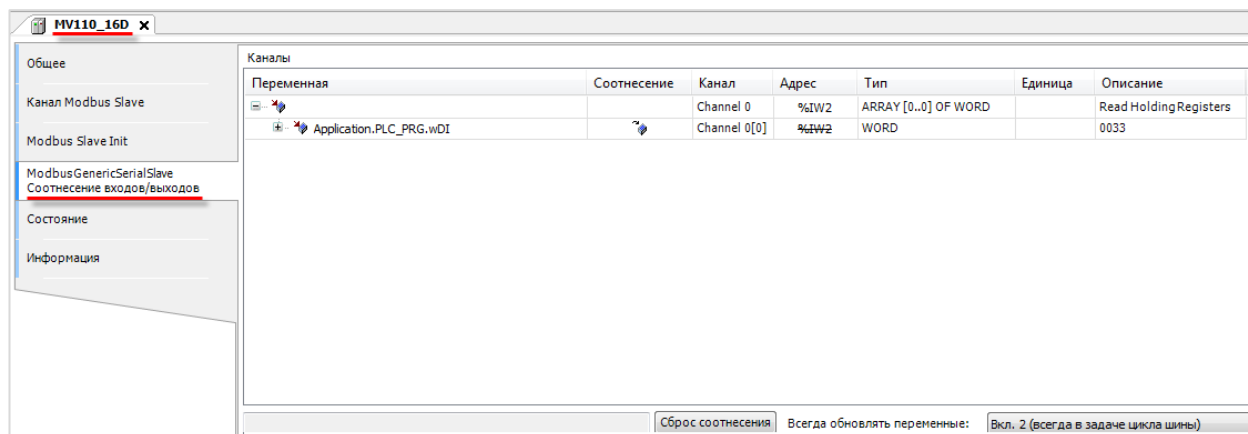


Рисунок 4.9.18 – Привязка переменных к каналу

11. В настройках компонента **MU110\_16R** на вкладке **Канал Modbus Slave** следует добавить канал, в котором с помощью функции **Write Multiple Registers** будет записываться значение в регистр **0x0032**. В данном регистре содержатся значения выходов модуля в виде битовой маски. У параметра **Триггер** следует установить значение **Передний фронт**, чтобы иметь возможность управлять записью в модуль с помощью логической переменной.

Таблица регистров модуля и поддерживаемые функции приведены в руководстве по эксплуатации.

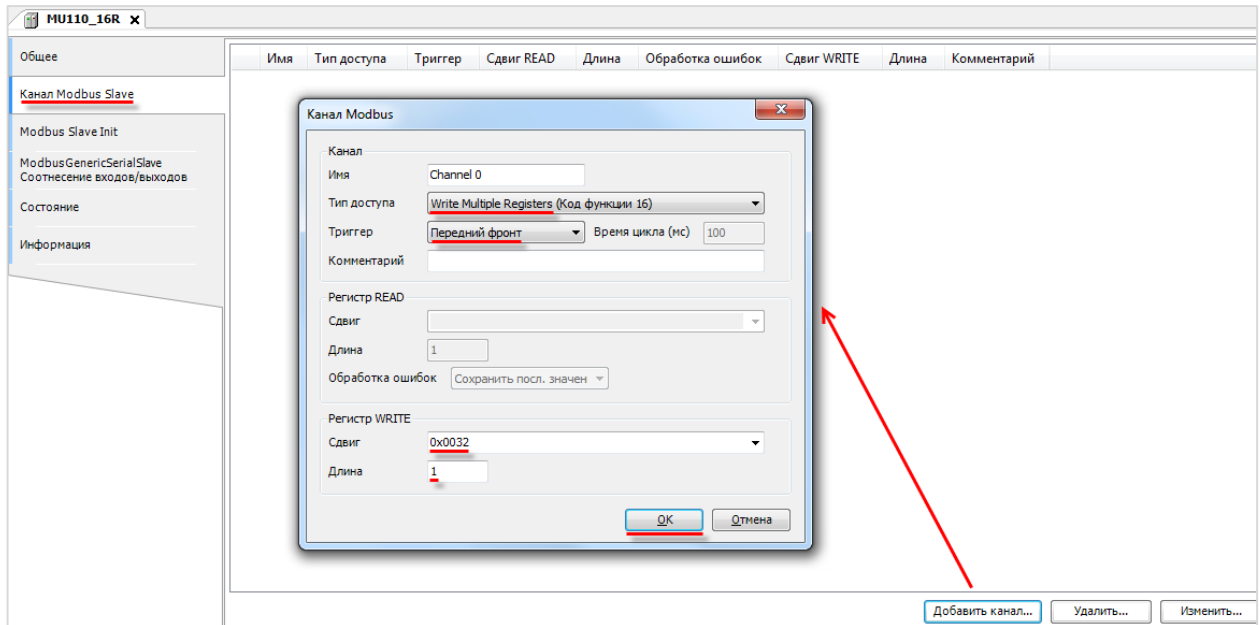


Рисунок 4.9.19 – Добавление канала в конфигурацию slave-устройства MU110\_16R

На вкладке **ModbusGenericSerialSlave** **Соотнесение входов/выходов** следует привязать к каналу переменную **wDO** и триггерную переменную **xTrigger**. Для параметра **Всегда обновлять переменные** следует установить значение **Включено 2**.

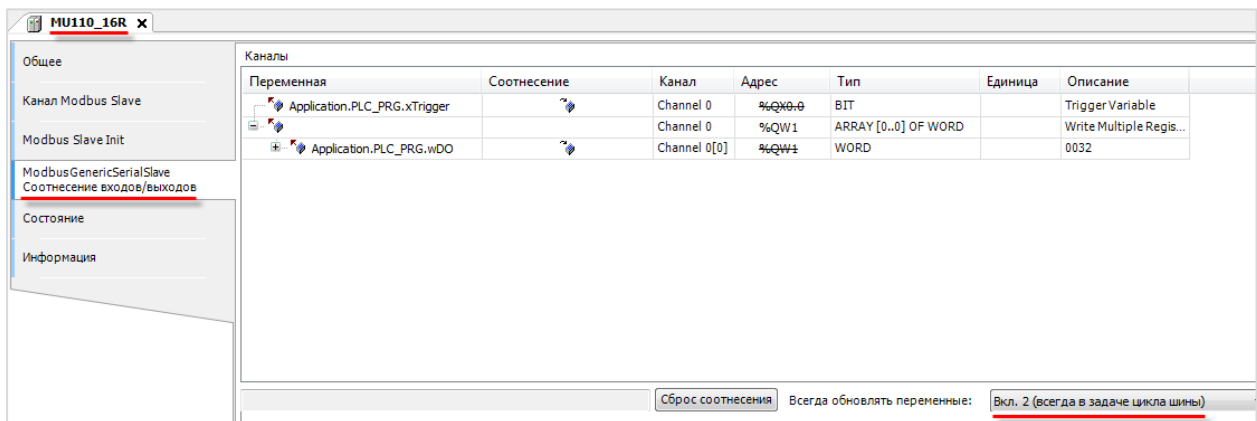


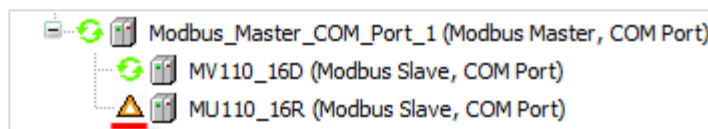
Рисунок 4.9.20 – Привязка переменных к каналу

12. Загрузить проект в контроллер и запустить его.



**ПРИМЕЧАНИЕ**

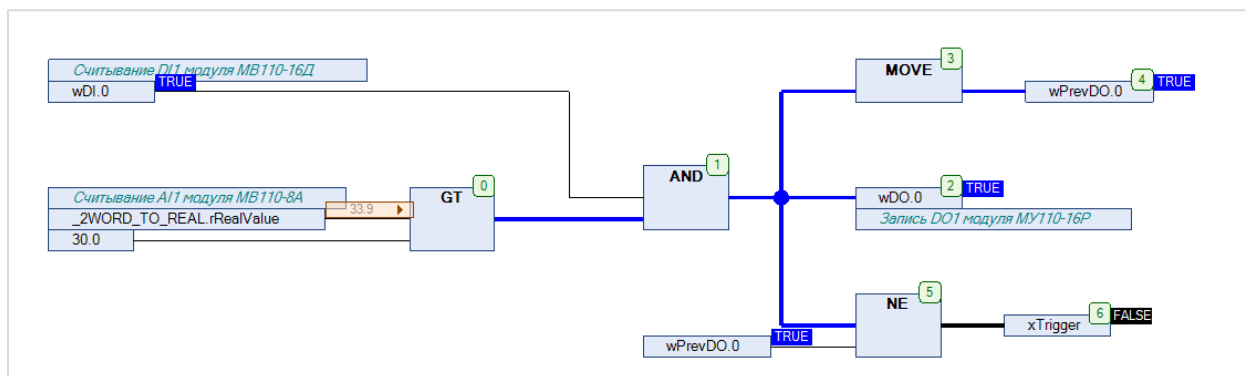
В дереве проекта рядом с модулем **MU110\_16R** будет отображаться пиктограмма «ожидание соединения». Это связано с тем, что запись в модуль производится по триггеру, и в данный момент обмен с модулем отсутствует. После первой записи в модуль статус связи изменится на «связь установлена».



**Рисунок 4.9.21 – Пиктограммы статусов связи**

В переменной **\_2WORD\_TO\_REAL.rRealValue** будет отображаться текущее значение первого аналогового входа модуля **MB110-8A**. В нулевом бите переменной **wDI (wDI.0)** будет отображаться текущее значение первого дискретного входа модуля **MB110-16Д**.

Если значение **\_2WORD\_TO\_REAL.rRealValue** превысит **30** и при этом значение **wDI.0** будет равно **TRUE**, то в нулевой бит переменной **wDO (wDO.0)** будет однократно (по триггеру) записано значение **TRUE**, что приведет к замыканию первого дискретного выхода модуля **MU110-16P**. Если одно из условий перестанет выполняться, то выход будет разомкнут.



**Рисунок 4.9.22 – Выполнение программы в режиме Online**

#### 4.10 Пример: СПК1хх [M01] (Modbus RTU Slave) + MasterOPC Universal Modbus Server

В качестве примера будет рассмотрена настройка обмена с OPC-сервером [Insat MasterOPC Universal Modbus Server](#), который будет использоваться в режиме **Modbus RTU Master**.

Структурная схема примера приведена на рисунке ниже:

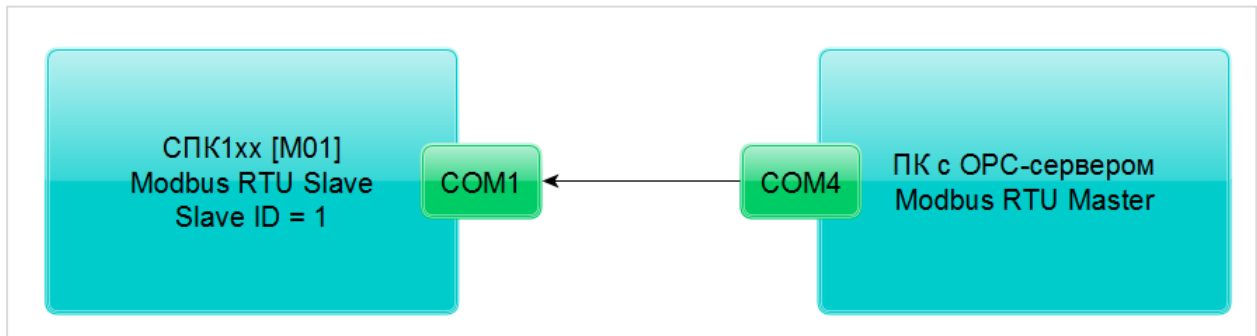


Рисунок 4.10.1 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел **Device** – **Обновить устройство**).

Пример доступен для скачивания: [Example CodesysModbusRtuSlave 3511v1.zip](#)



#### 4. Стандартные средства конфигурирования

Сетевые параметры устройств приведены в таблице ниже:

**Таблица 4.10.1 – Сетевые параметры устройств**

Параметр	СПК1хх [M01]	ПК с OPC-сервером
COM-порт контроллера, к которому подключен модуль	COM1	COM4
ID COM-порта	1	4
Режим работы	slave	master
Slave ID	1	-
Скорость обмена	115200	
Количество бит данных	8	
Контроль четности	Отсутствует	
Количество стоп-бит	1	

Переменные примера описаны в таблице ниже:

**Таблица 4.10.2 – Список переменных примера**

Имя	Тип	Область памяти Modbus	Адрес регистра/бита
<b>Переменные, считываемые OPC-сервером</b>			
xVar_OpcRead	BOOL	Coils	0/0
wVar_OpcRead	WORD	Holding регистры	1
rVar_OpcRead	REAL		2–3
sVar_OpcRead	STRING(16)		4–11
<b>Переменные, записываемые OPC-сервером</b>			
xVar_OpcWrite	BOOL	Discrete inputs	0
wVar_OpcWrite	WORD	Input регистры	1
rVar_OpcWrite	REAL		2–3
sVar_OpcWrite	STRING(16)		4–11

Для настройки обмена следует:

1. Подключить контроллер к ПК (например, с помощью конвертера [ОВЕН АС4](#)).
2. Создать новый проект **CODESYS** с программой **PLC\_PRG** на языке **CFC**:

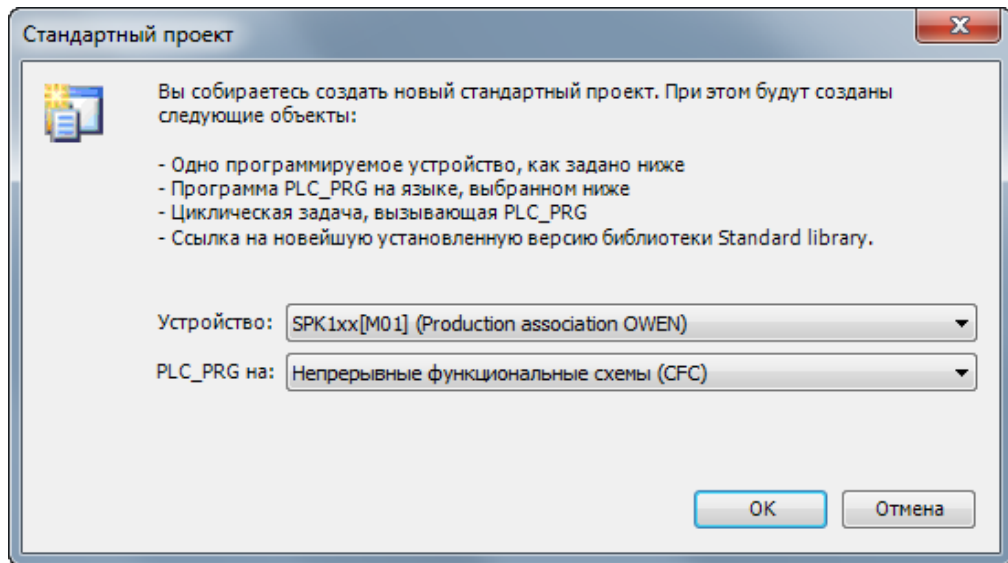


Рисунок 4.10.2 – Создание проекта CODESYS

3. Добавить в проект [объединения](#) с именами **Real\_Word** и **String\_Word**:

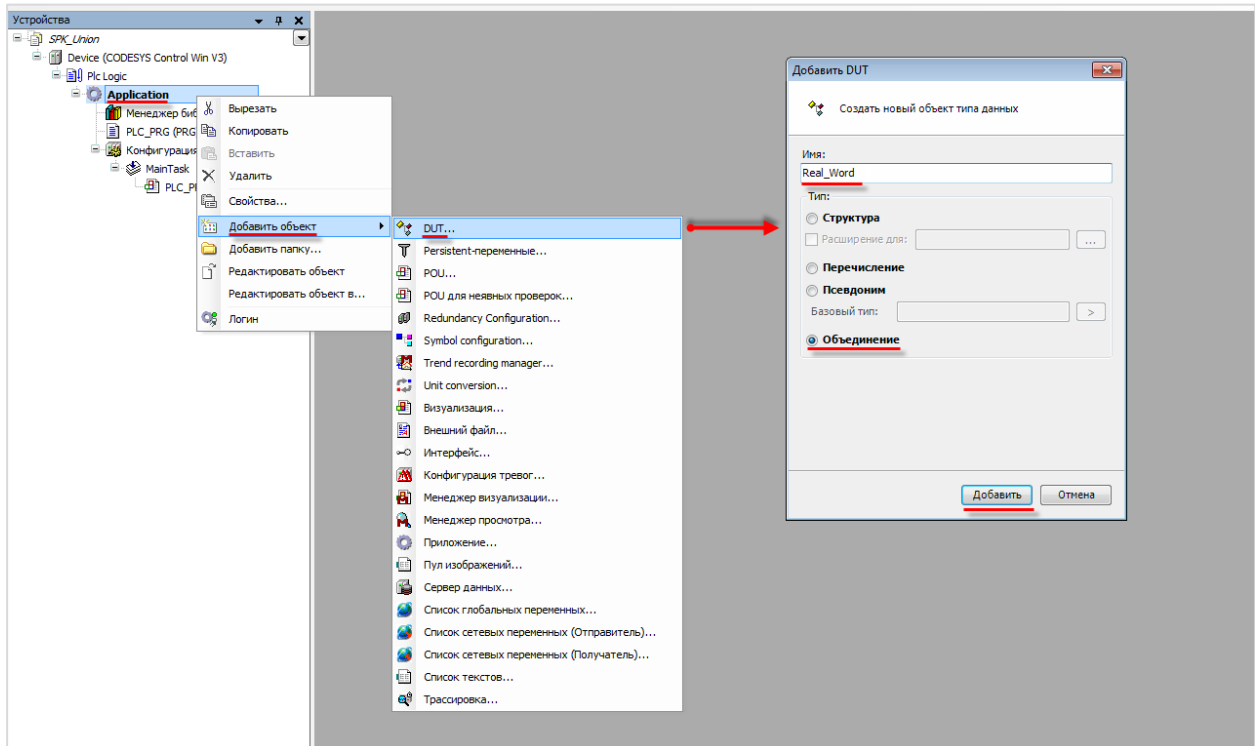
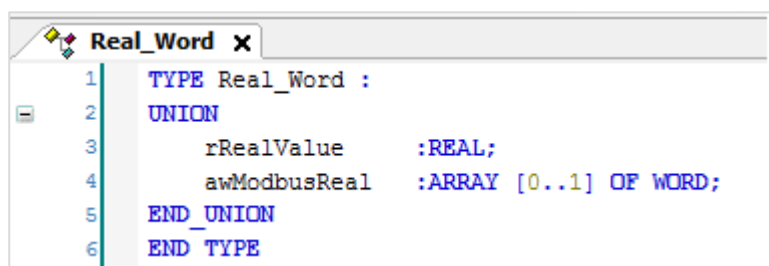


Рисунок 4.10.3 – Добавление в проект объединения Real\_Word

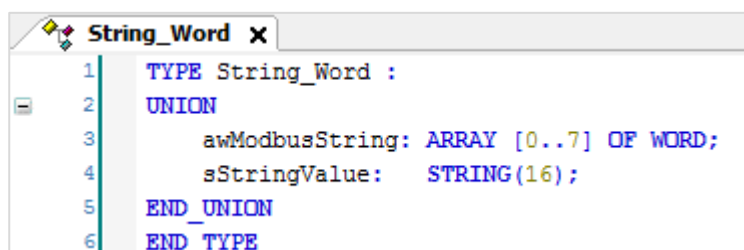
Объединения потребуются для преобразования переменных типов **REAL** и **STRING** в набор переменных типа **WORD** для привязки к компоненту **Modbus Serial Device**.

4. В объединениях объявить следующие переменные:



```
Real_Word x
1  TYPE Real_Word :
2  UNION
3      rRealValue      :REAL;
4      awModbusReal    :ARRAY [0..1] OF WORD;
5  END_UNION
6  END_TYPE
```

Рисунок 4.10.4 – Объявление переменных объединения Real\_Word

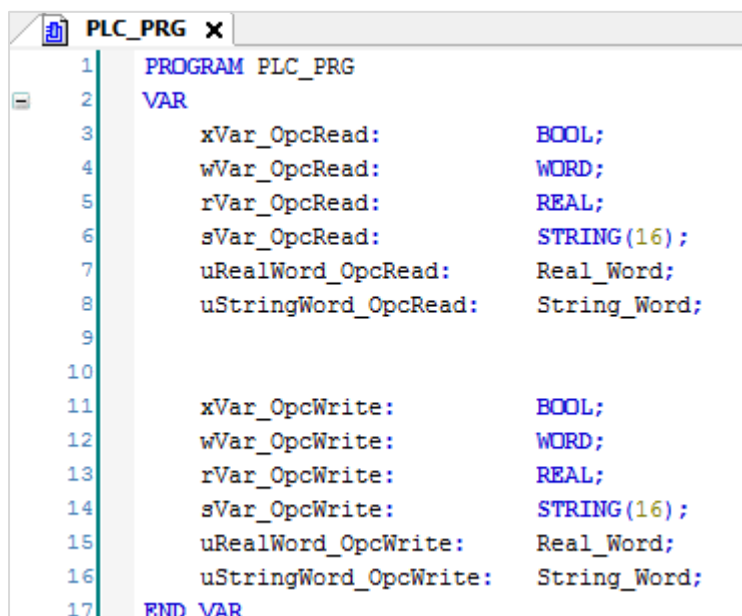


```
String_Word x
1  TYPE String_Word :
2  UNION
3      awModbusString: ARRAY [0..7] OF WORD;
4      sStringValue:  STRING(16);
5  END_UNION
6  END_TYPE
```

Рисунок 4.10.5 – Объявление переменных объединения String\_Word

5. В менеджере библиотек добавить библиотеку **CAA Memory**.

6. В программе **PLC\_PRG** объявить переменные в соответствии с [таблицей 4.10.2](#).



```
PLC_PRG x
1  PROGRAM PLC_PRG
2  VAR
3      xVar_OpcRead:      BOOL;
4      wVar_OpcRead:      WORD;
5      rVar_OpcRead:      REAL;
6      sVar_OpcRead:      STRING(16);
7      uRealWord_OpcRead: Real_Word;
8      uStringWord_OpcRead: String_Word;
9
10
11     xVar_OpcWrite:      BOOL;
12     wVar_OpcWrite:      WORD;
13     rVar_OpcWrite:      REAL;
14     sVar_OpcWrite:      STRING(16);
15     uRealWord_OpcWrite: Real_Word;
16     uStringWord_OpcWrite: String_Word;
17 END VAR
```

Рисунок 4.10.6 – Объявление переменных программы

Код программы будет выглядеть следующим образом:



Рисунок 4.10.7 – Код программы PLC\_PRG

Функция **ReverseBYTEsInWORD** из библиотеки **CAA Memory** используется для изменения порядка байтов в переменной типа **STRING** для соответствия порядку байтов в OPC-сервере (функционал перестановки байт в OPC-сервере не распространяется на тип STRING).

#### 4. Стандартные средства конфигурирования

7. Добавить компоненты **Modbus COM** и **Modbus Serial Device** в соответствии с [п. 4.3](#).  
Настроить компоненты в соответствии с [таблицей 4.10.1](#).

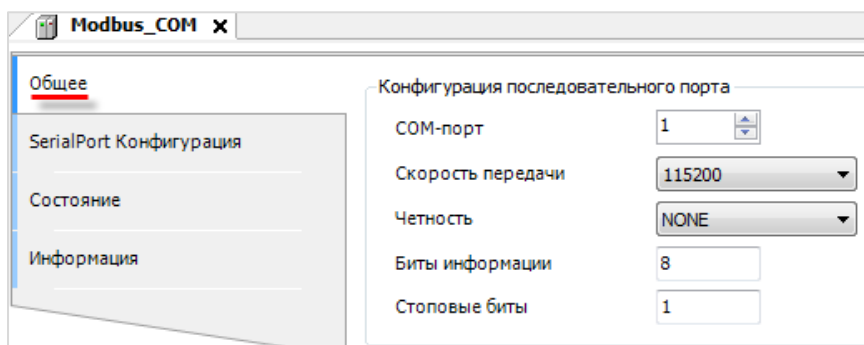


Рисунок 4.10.8 – Настройки компонента Modbus COM

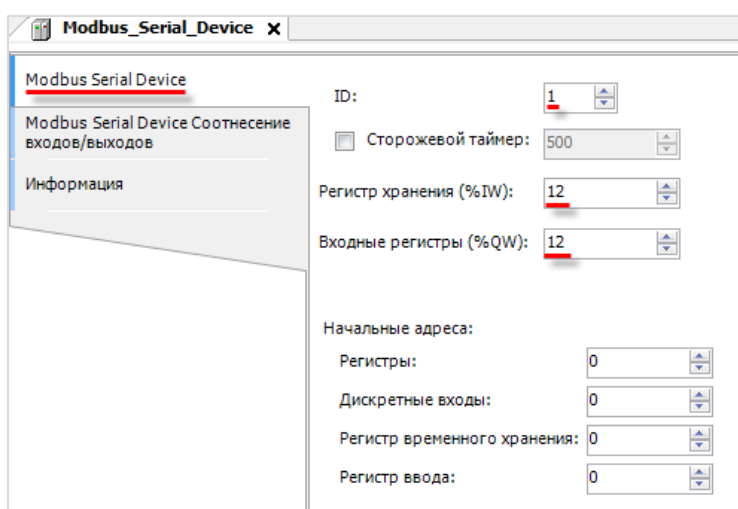


Рисунок 4.10.9 – Настройки компонента Modbus Serial Device

8. Привязать к каналам компонента **Modbus Serial Device** переменные программы в соответствии с [таблицей 4.10.2](#). Установить галочку **Вкл. 2 (Всегда в задаче цикла шины)**.

Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
		Входы	%IW0	ARRAY [0..11] OF WORD		Регистры временного хранения Modbus
		Входы[0]	%IW0	WORD		Holding регистр 0 (cols 0-15)
Application.PLC_PRG.wVar_OpcWrite		Bt0	%IX0.0	BOOL		
		Bt1	%IX0.1	BOOL		
		Bt2	%IX0.2	BOOL		
		Bt3	%IX0.3	BOOL		
		Bt4	%IX0.4	BOOL		
		Bt5	%IX0.5	BOOL		
		Bt6	%IX0.6	BOOL		
		Bt7	%IX0.7	BOOL		
		Bt8	%IX1.0	BOOL		
		Bt9	%IX1.1	BOOL		
		Bt10	%IX1.2	BOOL		
		Bt11	%IX1.3	BOOL		
		Bt12	%IX1.4	BOOL		
		Bt13	%IX1.5	BOOL		
		Bt14	%IX1.6	BOOL		
		Bt15	%IX1.7	BOOL		
Application.PLC_PRG.wVar_OpcWrite		Входы[1]	%IW1	WORD		Holding регистр 1
Application.PLC_PRG.uRealWord_OpcWrite.avModbusReal[0]		Входы[2]	%IW2	WORD		Holding регистр 2
Application.PLC_PRG.uRealWord_OpcWrite.avModbusReal[1]		Входы[3]	%IW3	WORD		Holding регистр 3
Application.PLC_PRG.uStringWord_OpcWrite.avModbusString[0]		Входы[4]	%IW4	WORD		Holding регистр 4
Application.PLC_PRG.uStringWord_OpcWrite.avModbusString[1]		Входы[5]	%IW5	WORD		Holding регистр 5
Application.PLC_PRG.uStringWord_OpcWrite.avModbusString[2]		Входы[6]	%IW6	WORD		Holding регистр 6
Application.PLC_PRG.uStringWord_OpcWrite.avModbusString[3]		Входы[7]	%IW7	WORD		Holding регистр 7
Application.PLC_PRG.uStringWord_OpcWrite.avModbusString[4]		Входы[8]	%IW8	WORD		Holding регистр 8
Application.PLC_PRG.uStringWord_OpcWrite.avModbusString[5]		Входы[9]	%IW9	WORD		Holding регистр 9
Application.PLC_PRG.uStringWord_OpcWrite.avModbusString[6]		Входы[10]	%IW10	WORD		Holding регистр 10
Application.PLC_PRG.uStringWord_OpcWrite.avModbusString[7]		Входы[11]	%IW11	WORD		Holding регистр 11
		Выходы	%QW0	ARRAY [0..11] OF WORD		Выходные регистры Modbus
		Выходы[0]	%QW0	WORD		Input регистр 0 (discrete inputs 0-15)
Application.PLC_PRG.wVar_OpcRead		Выходы[1]	%QW1	WORD		Input регистр 1

Рисунок 4.10.10 – Привязка переменных к компоненту Modbus Serial Device (holding регистры)

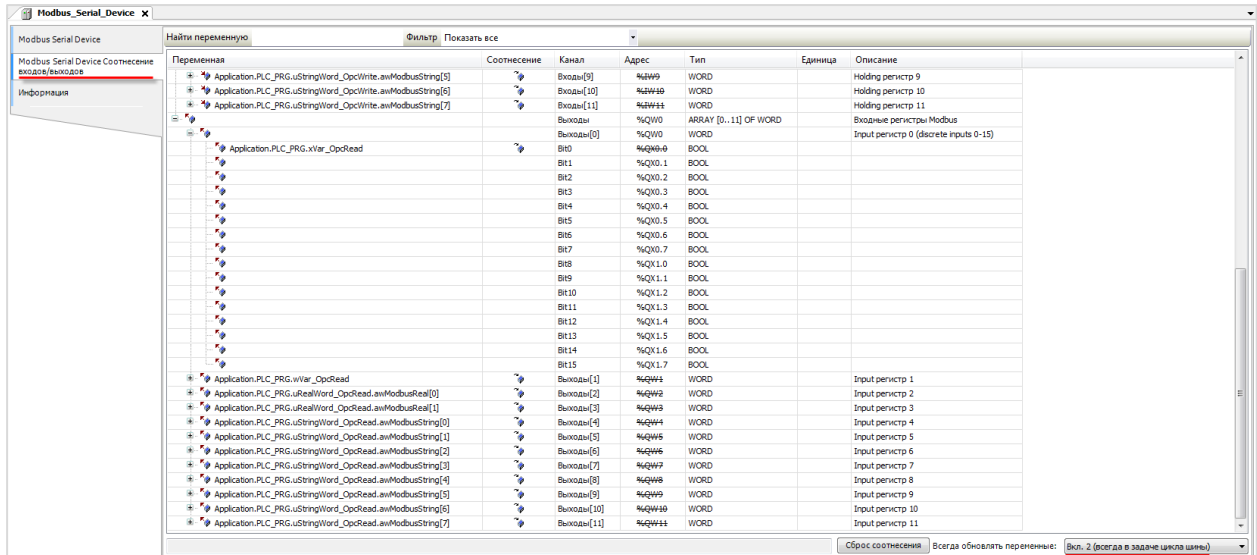


Рисунок 4.10.11 – Привязка переменных к компоненту Modbus Serial Device (input регистры)

9. Установить и запустить [MasterOPC Universal Modbus Server](#).

10. Нажать ПКМ на узел **Server** и добавить коммуникационный узел типа **COM**. В узле следует указать сетевые настройки в соответствии с [таблицей 4.10.1](#). Для работы OPC-сервера в режиме **Modbus RTU Master** параметры **Использовать режим ASCII** и **Slave подключение** должны иметь значение **FALSE**.

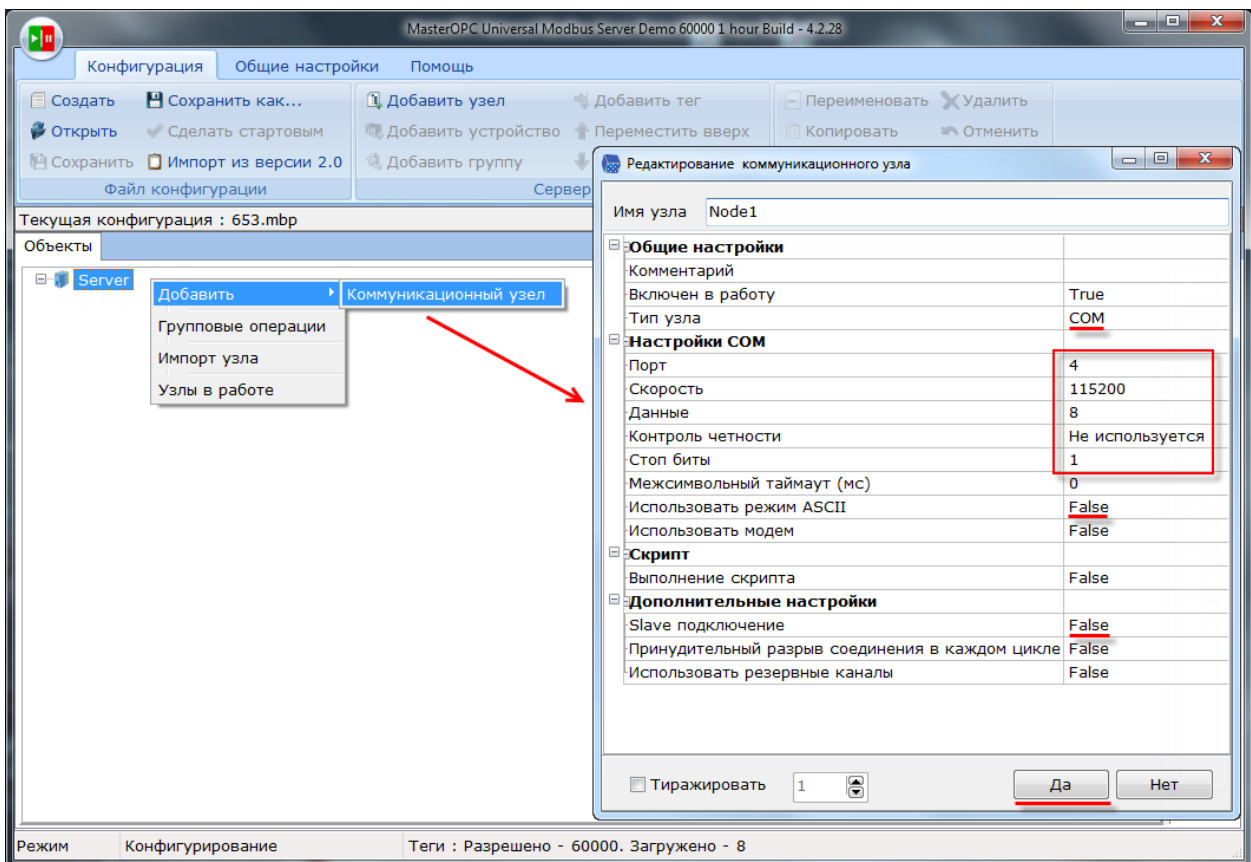


Рисунок 4.10.12 – Добавление коммуникационного узла

#### 4. Стандартные средства конфигурирования

11. Нажать ПКМ на коммуникационный узел и добавить устройство с настройками по умолчанию (Slave ID = 1 в соответствии с [таблицей 4.10.1](#)).

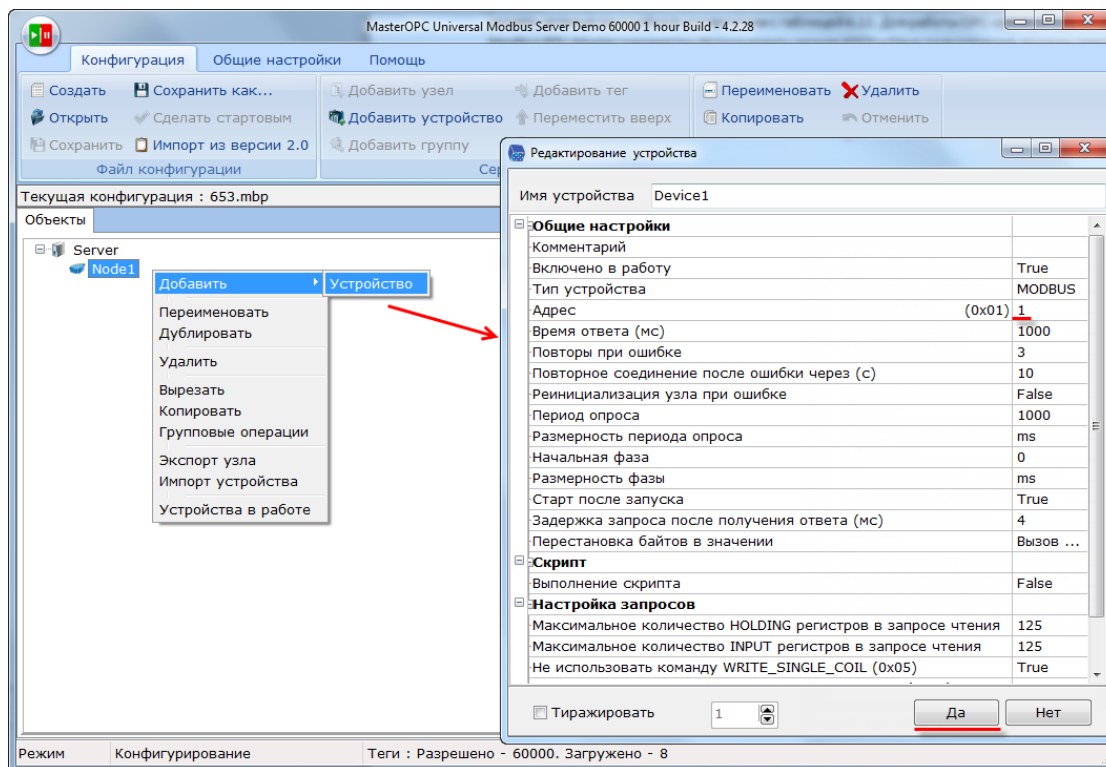


Рисунок 4.10.13 – Добавление устройства

12. Нажать ПКМ на устройство и добавить 8 тегов. Число тегов соответствует числу переменных, считываемых/записываемых OPC-сервером. Настройки тегов приведены ниже.

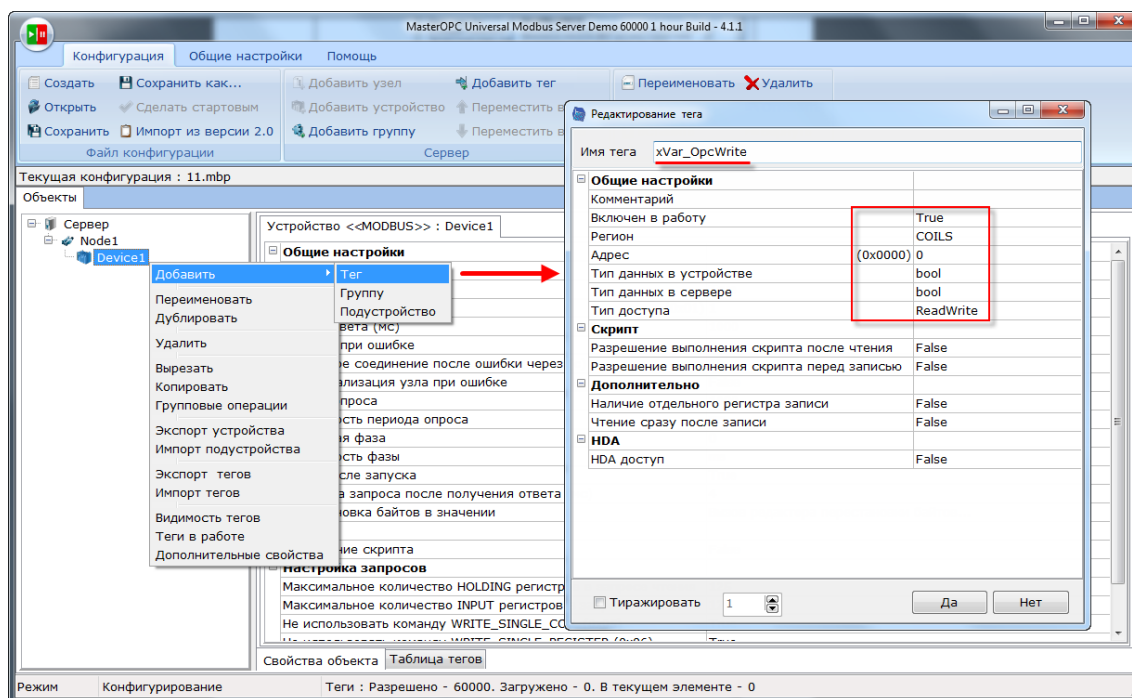


Рисунок 4.10.14 – Добавление тега xVar\_OpcWrite

Имя тега <u>wVar_OpcWrite</u>	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Регион	HOLDING_REGISTERS
Адрес (0x0001)	1
Тип данных в устройстве	uint16
Тип данных в сервере	uint32
Тип доступа	ReadWrite
Использовать перестановку байтов устройства	True
Последний тег в групповом запросе	False
Пересчет (A*X + B)	False

Рисунок 4.10.15 – Добавление тега wVar\_OpcWrite

Имя тега <u>rVar_OpcWrite</u>	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Регион	HOLDING_REGISTERS
Адрес (0x0002)	2
Тип данных в устройстве	float
Тип данных в сервере	float
Тип доступа	ReadWrite
Использовать перестановку байтов устройства	False
Перестановка байтов в значении	10325476
Последний тег в групповом запросе	False
Пересчет (A*X + B)	False

Рисунок 4.10.16 – Добавление тега rVar\_OpcWrite

Имя тега <u>sVar_OpcWrite</u>	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Регион	HOLDING_REGISTERS
Адрес (0x0004)	4
Тип данных в устройстве	string
Тип данных в сервере	string
Количество байт для строкового типа	16
Тип строки для строкового типа	ascii
Тип доступа	ReadWrite
Использовать перестановку байтов устройства	True
Последний тег в групповом запросе	False
Пересчет (A*X + B)	False

Рисунок 4.10.17 – Добавление тега sVar\_OpcWrite



Имя тега	<u>xVar_OpcRead</u>	
<b>Общие настройки</b>		
Комментарий		
Включен в работу	True	
Регион	DISCRETE_INPUTS	
Адрес	(0x0001)	0
Тип данных в устройстве	bool	
Тип данных в сервере	bool	
Тип доступа	ReadOnly	

Рисунок 4.10.18 – Добавление тега xVar\_OpcRead

Имя тега	<u>wVar_OpcRead</u>	
<b>Общие настройки</b>		
Комментарий		
Включен в работу	True	
Регион	INPUT_REGISTERS	
Адрес	(0x0001)	1
Тип данных в устройстве	uint16	
Тип данных в сервере	uint32	
Тип доступа	ReadOnly	

Рисунок 4.10.19 – Добавление тега wVar\_OpcRead

Имя тега	<u>rVar_OpcRead</u>	
<b>Общие настройки</b>		
Комментарий		
Включен в работу	True	
Регион	INPUT_REGISTERS	
Адрес	(0x0002)	2
Тип данных в устройстве	float	
Тип данных в сервере	float	
Тип доступа	ReadOnly	
Использовать перестановку байтов устройства	False	
Перестановка байтов в значении	10325476	
Последний тег в групповом запросе	False	
Пересчет (A*X + B)	False	

Рисунок 4.10.20 – Добавление тега rVar\_OpcRead

Имя тега <u>sVar_OpcRead</u>	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Регион	INPUT_REGISTERS
Адрес	(0x0004) 4
Тип данных в устройстве	string
Тип данных в сервере	string
Количество байт для строкового типа	16
Тип строки для строкового типа	ascii
Тип доступа	ReadOnly

Рисунок 4.10.21 – Добавление тега sVar\_OpcRead

13. Загрузить проект в контроллер и запустить его. Запустить OPC-сервер для контроля значений переменных.

В редакторе CODESYS следует изменить значения **OpcRead** переменных и наблюдать соответствующие изменения в OPC-сервере. В OPC-сервере следует изменить значения **OpcWrite** переменных и наблюдать соответствующие значения в CODESYS.

Device.Application.PLC_PRG		
Выражение	Тип	Значение
xVar_OpcRead	BOOL	TRUE
wVar_OpcRead	WORD	11
rVar_OpcRead	REAL	22.33
sVar_OpcRead	STRING(16)	'тест'
uRealWord_OpcRead	Real_Word	
uStringWord_OpcRead	String_Word	
xVar_OpcWrite	BOOL	TRUE
wVar_OpcWrite	WORD	44
rVar_OpcWrite	REAL	55.66
sVar_OpcWrite	STRING(16)	'test'
uRealWord_OpcWrite	Real_Word	
uStringWord_OpcWrite	String_Word	

MasterOPC Universal Modbus Server Demo 60000 1 hour Build - 4.2.28

Стартовая конфигурация : 653.mbp

Объекты

- Server
  - Node1
    - Device1
      - xVar\_OpcRead
      - wVar\_OpcRead
      - rVar\_OpcRead
      - sVar\_OpcRead
      - xVar\_OpcWrite
      - wVar\_OpcWrite
      - rVar\_OpcWrite
      - sVar\_OpcWrite

Устройство <<Device1>>

Имя	Регион	Адрес	Значение	Качество	Время (UTC)	Тип в сер...
Node1.Device1.xVa...	DISC...	(0x00...	True	GOOD	2019-07-0...	bool
Node1.Device1.wV...	INPU...	(0x00...	11	GOOD	2019-07-0...	uint32
Node1.Device1.rVa...	INPU...	(0x00...	22.330000	GOOD	2019-07-0...	float
Node1.Device1.sVa...	INPU...	(0x00...	тест	GOOD	2019-07-0...	string
Node1.Device1.xVa...	COILS	(0x00...	True	GOOD	2019-07-0...	bool
Node1.Device1.wV...	HOL...	(0x00...	44	GOOD	2019-07-0...	uint32
Node1.Device1.rVa...	HOL...	(0x00...	55.660000	GOOD	2019-07-0...	float
Node1.Device1.sVa...	HOL...	(0x00...	test	GOOD	2019-07-0...	string

Рисунок 4.10.22 – Чтение и запись данных через OPC-сервер

## 4.11 Пример: СПК1хх [M01] (Modbus TCP Master) + модули Mx210

В качестве примера будет рассмотрена настройка обмена с модулями Mx210 (MB210-101 и МК210-301) с использованием **шаблонов**. В примере используются шаблоны версии **3.5.11.4**.

**Реализуемый алгоритм:** если значение первого аналогового входа модуля **MB210-101** превышает **30** и при этом первый дискретный вход модуля **МК210-301** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **МК210-301** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).

Структурная схема примера приведена на рисунке ниже:

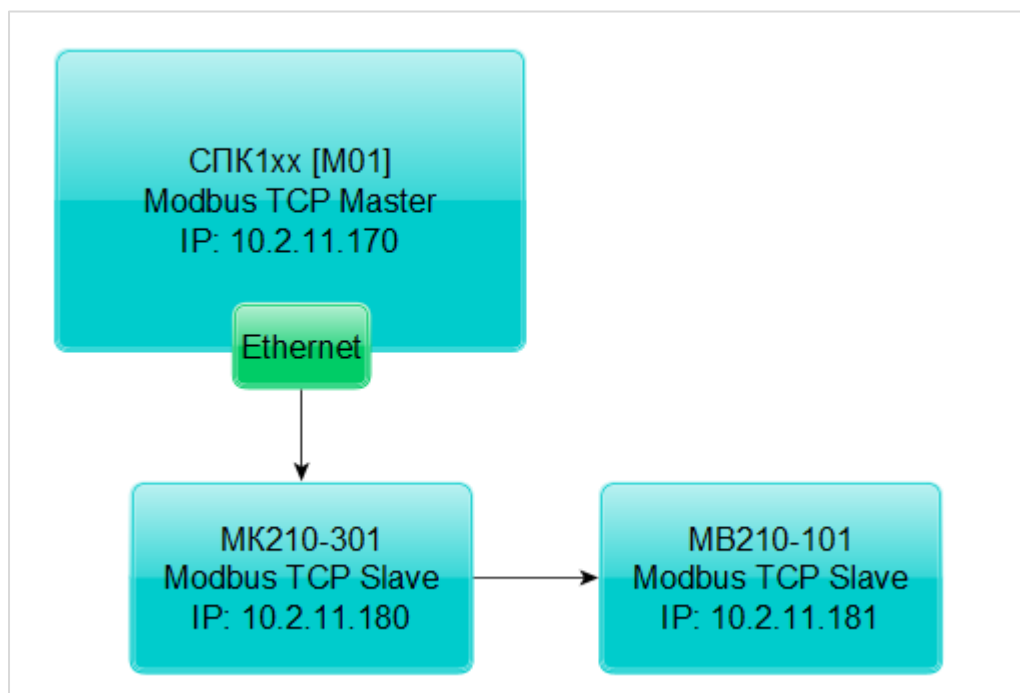


Рисунок 4.11.1 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (**ПКМ** на узел **Device** – **Обновить устройство**).

Пример доступен для скачивания: [Example\\_CodesysModbusTcpMasterMx210\\_3511v1.projectarchive](http://www.projectarchive.com/Example_CodesysModbusTcpMasterMx210_3511v1.projectarchive)

Видеоверсия примера доступна по [ссылке](#).

Сетевые параметры устройств приведены в таблице ниже:

**Таблица 4.11.1 – Сетевые параметры устройств**

Параметр	СПК1xx [M01]	МК210-301	МВ210-101
Режим работы	master	slave	slave
IP-адрес	10.2.11.170	10.2.11.180	10.2.11.181
Маска подсети	255.255.0.0		
IP-адрес шлюза	10.2.1.1		
Порт	502		
Unit ID	-	1	1

Переменные примера описаны в таблице ниже:

**Таблица 4.11.2 – Список переменных примера**

Модуль	Имя переменной	Тип	Описание
МВ210-101	awModbusReal	ARRAY [0..1] OF WORD	Значение температуры в виде двух <b>WORD</b> , считываемое с модуля
	rRealValue	REAL	Значение температуры в виде числа с плавающей точкой для использования в программе
МК210-301	wDI	WORD	Значение дискретных входов в виде битовой маски. При обращении к отдельным входам указывается их номер, начиная с 0: <b>wDI.0</b> – состояние первого входа (TRUE/FALSE) <b>wDI.1</b> – состояние второго входа ...
	wDO	WORD	Значение дискретных выходов в виде битовой маски. При обращении к отдельным выходам указывается их номер, начиная с 0: <b>wDO.0</b> – состояние первого выхода (TRUE/FALSE) <b>wDO.1</b> – состояние второго выхода ...
-	wPrevDO	WORD	Значение дискретных выходов в виде битовой маски из предыдущего цикла программы. Используется для отправки команды записи только в случае изменения значений выходов (иначе будет производиться циклическая запись последнего значения)
-	xTrigger	BOOL	Триггерная переменная, управляющая функцией записи дискретного выхода (запись происходит по переднему фронту переменной)

#### 4. Стандартные средства конфигурирования

Для настройки обмена следует:

1. Настроить модули **Mx210** с помощью программы **ОВЕН Конфигуратор** в соответствии с [таблицей 4.11.1](#) (см. руководство **Mx210. Примеры настройки обмена**). Подключить модули к контроллеру.
2. Создать новый проект **CODESYS** с программой **PLC\_PRG** на языке **CFC**:

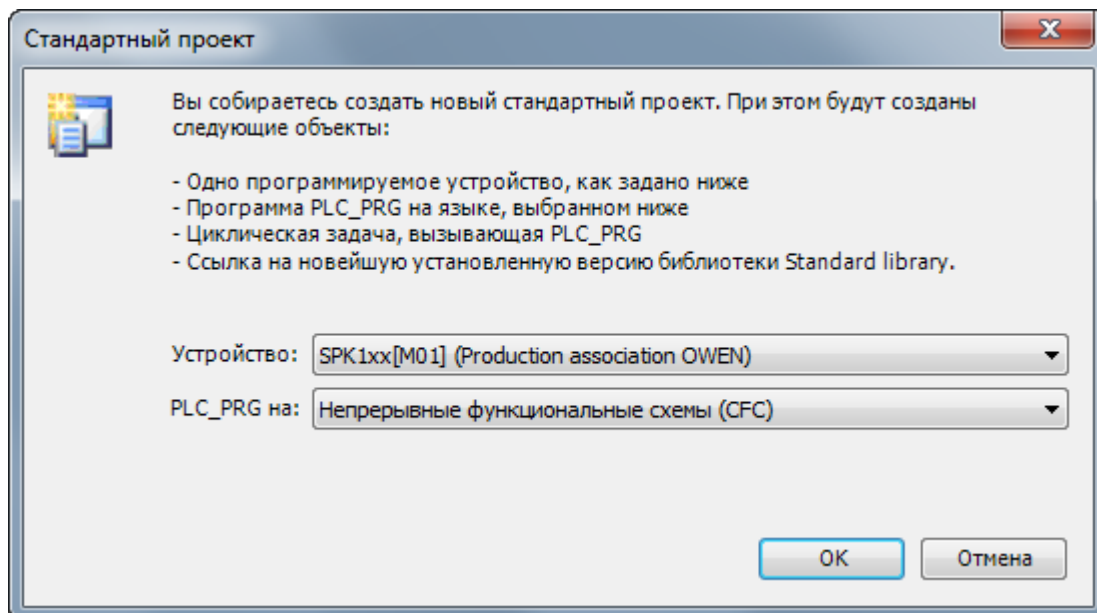


Рисунок 4.11.2 – Создание проекта CODESYS

3. Добавить в проект [объединение](#) с именем **Real\_Word**:

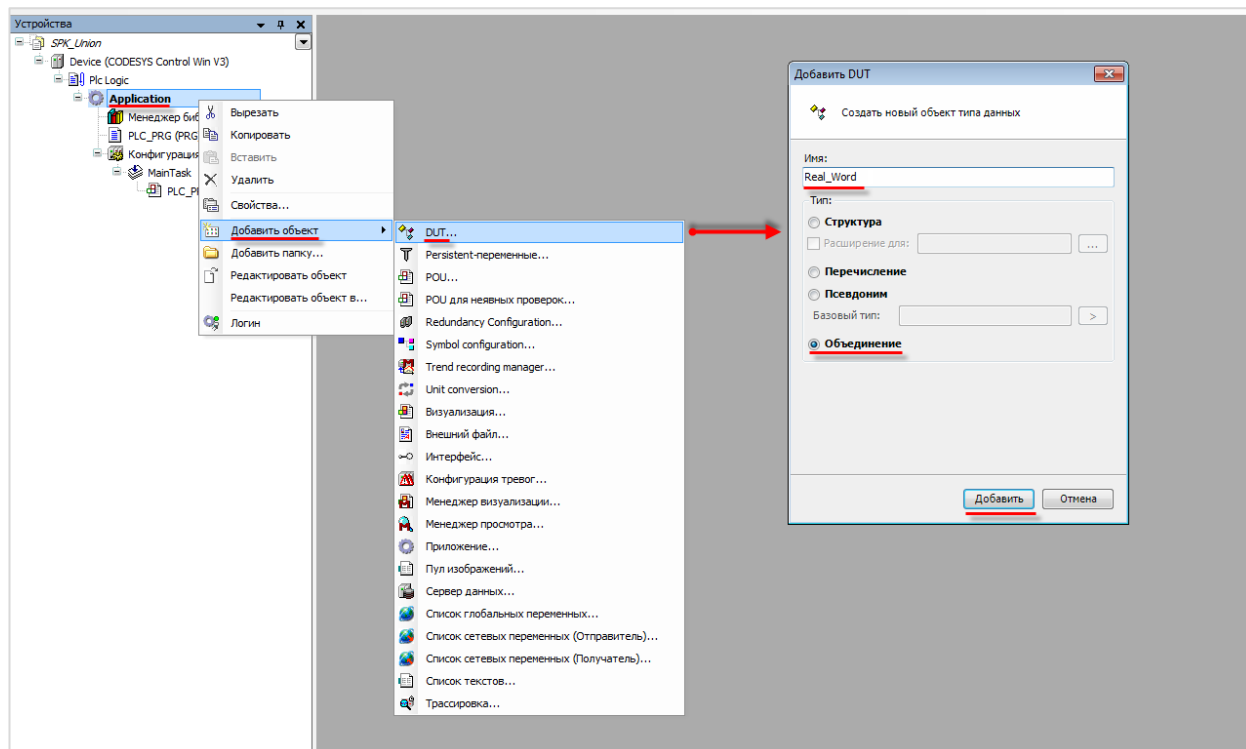


Рисунок 4.11.3 – Добавление в проект объединения

4. В объединении объявить переменную **rRealValue** типа **REAL** и массив **awModbusReal** типа **WORD**, содержащий два элемента:

```

Real_Word x
1  TYPE Real_Word :
2  UNION
3      rRealValue      :REAL;
4      awModbusReal   :ARRAY [0..1] OF WORD;
5  END_UNION
6  END_TYPE

```

Рисунок 4.11.4 – Объявление переменных объединения

5. В программе **PLC\_PRG** объявить экземпляр объединения **Real\_Word** с названием **\_2WORD\_TO\_REAL**, переменные **wDI**, **wDO** и **wPrevDO** типа **WORD** и переменную **xTrigger** типа **BOOL**. Описание переменных приведено в [таблице 4.11.2](#).

```

PLC_PRG x
1  PROGRAM PLC_PRG
2  VAR
3      _2WORD_TO_REAL: Real_Word; // значение 1-го входа MB210-101
4      wDI:           WORD;       // битовая маска входов MK210-301
5      wDO:           WORD;       // битовая маска выходов MK210-301
6      wPrevDO:       WORD;       // битовая маска предыдущей записи выходов MK210-301
7      xTrigger:      BOOL;       // триггер записи выходов
8  END_VAR

```

Рисунок 4.11.5 – Объявление переменных программы

Код программы будет выглядеть следующим образом:

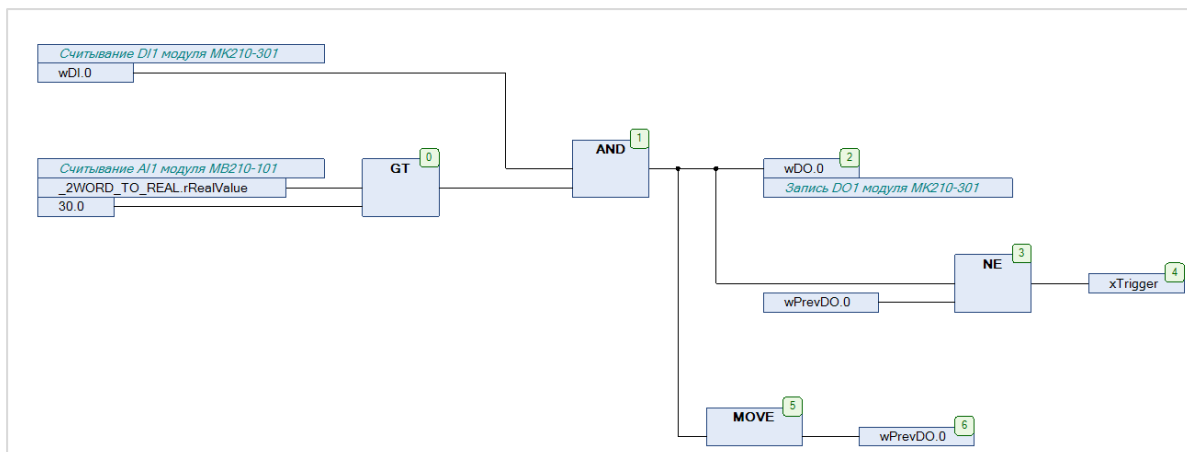


Рисунок 4.11.6 – Код программы PLC\_PRG

Программа работает следующим образом: если значение переменной **rRealValue** (связанной с первым аналоговым входом модуля **MB210-101**) превышает **30** и при этом значение нулевого бита переменной **wDI** (связанной с первым дискретным входом модуля **MK210-301**) имеет значение **TRUE**, то нулевому биту переменной **wDO** присваивается значение **TRUE**. Если на предыдущем цикле значение нулевого бита **wDO** отличалось от текущего, то переменная **xTrigger** принимает значение **TRUE**, что приводит к однократной записи текущего значения бита в первый дискретный выход модуля **MK210-301**.

#### 4. Стандартные средства конфигурирования

##### 6. Добавить в проект компонент Ethernet.



##### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

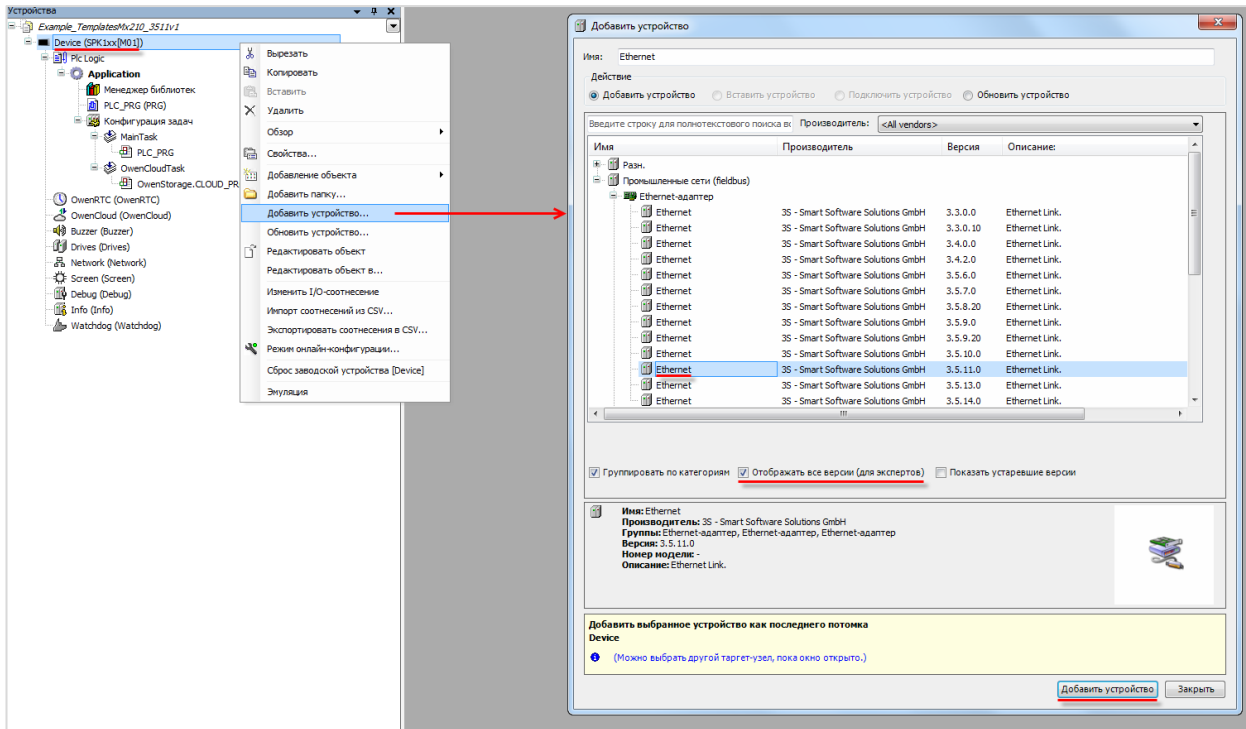


Рисунок 4.11.7 – Добавление компонента Ethernet

Затем следует установить соединение с контроллером, не загружая в него проект (**Device – Установка соединения – Сканировать сеть**) и в компоненте **Ethernet** на вкладке **Конфигурация Ethernet** выбрать нужный интерфейс.

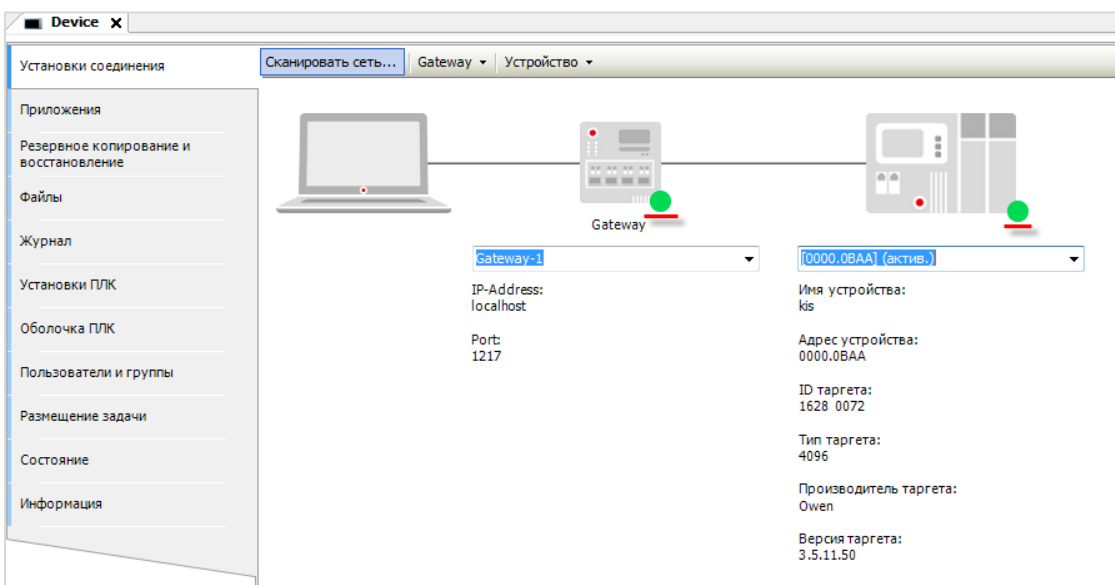


Рисунок 4.11.8 – Подключение к контроллеру

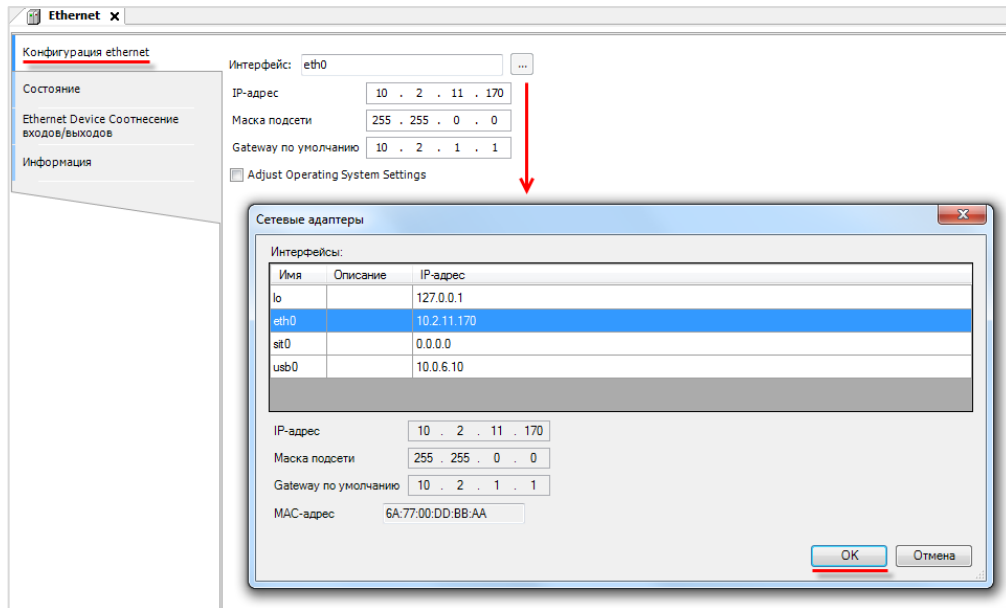


Рисунок 4.11.9 – Выбор используемого интерфейса

**ПРИМЕЧАНИЕ**

Настройки интерфейса задаются в конфигураторе контроллера (см. документ **CODESYS V3.5. FAQ**).

7. В компонент **Ethernet** добавить компонент **Modbus TCP Master**.**ПРИМЕЧАНИЕ**

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

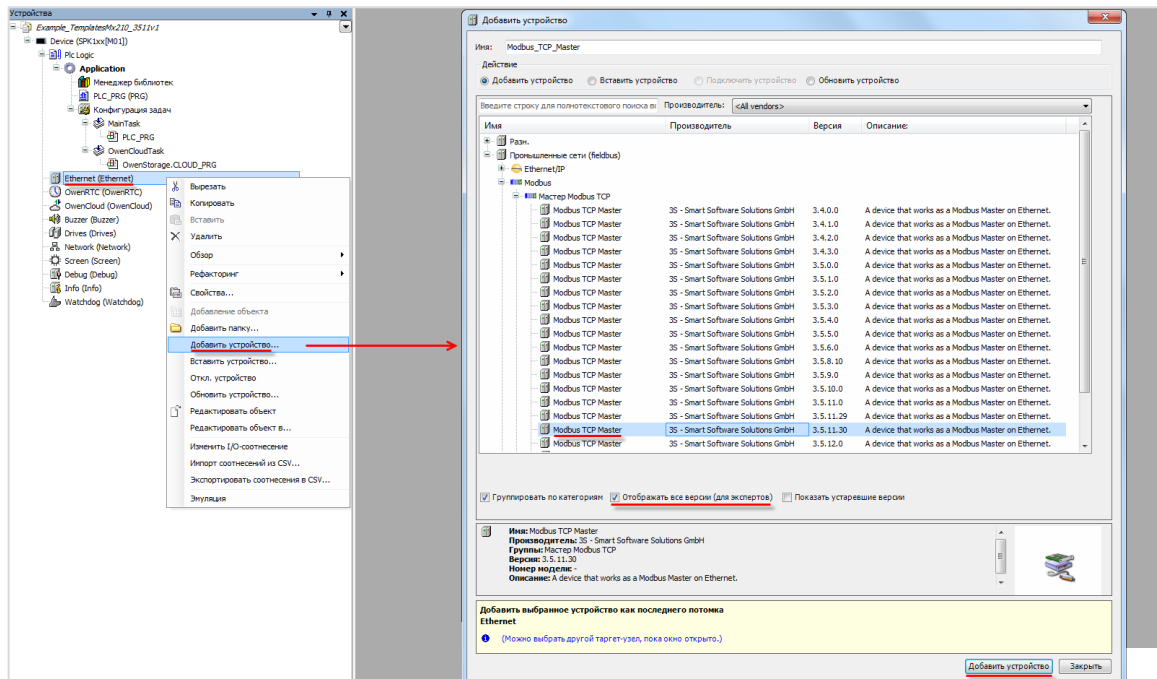


Рисунок 4.11.10 – Добавление компонента Modbus TCP Master



#### 4. Стандартные средства конфигурирования

В настройках компонента вкладке **Общее** следует установить галочку **Автоподключение**.

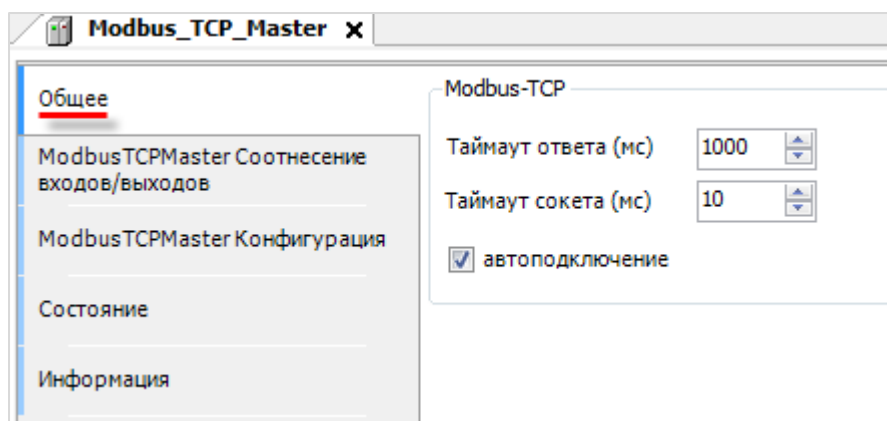


Рисунок 4.11.11 – Настройки компонента Modbus TCP Master

8. В компонент **Modbus TCP Master** следует добавить компоненты **Modbus TCP Slave** с именами **MV210\_101** и **MK210\_301**.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в [приложении А](#).

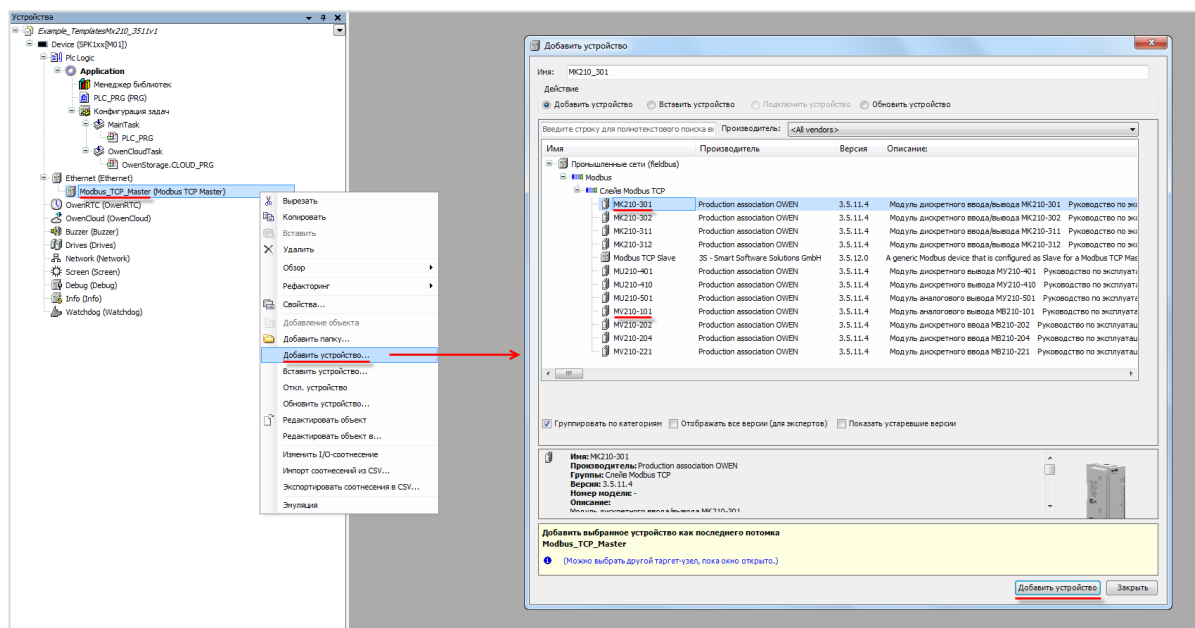


Рисунок 4.11.12 – Добавление slave-устройств в проект CODESYS

В настройках компонентов следует указать IP-адреса согласно [таблице 4.11.1](#) (**MK210-301** – **10.2.11.180**, **MB210-101** – **10.2.11.181**). На вкладке **Modbus TCP Slave Конфигурация** следует установить для параметра **Unit ID** значение **1**.

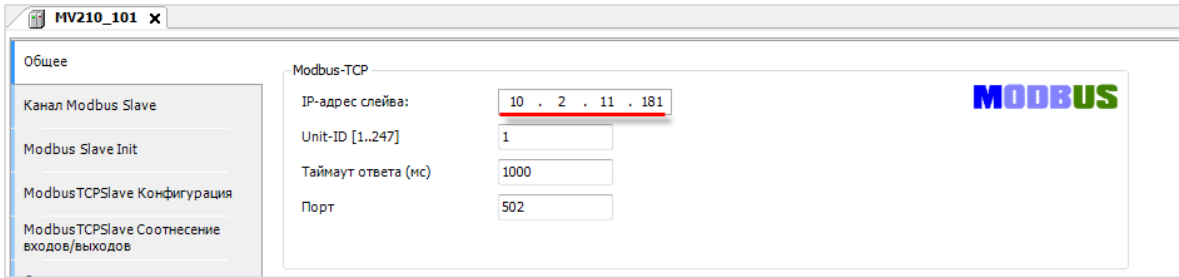


Рисунок 4.11.13 – Настройки компонента Modbus TCP Slave, вкладка Общее

Параметр	Тип	Значение	Значение по умолчанию	Единица	Описание
NewChannelConfig	BOOL	true	true		Use the new Channel-Config format
Unit-ID	USINT	1	16#FF		Unit-ID of the Device
ResponseTimeout	DWORD	1000	1000		Maximum time for a Slave to respond in ms
IPAddress	ARRAY[0..3] OF BYTE	[10, 2, 11, 180]	[192, 168, 0, 1]		Configure IP Address of TCP Slave.
Port	UINT	502	502		Port where the slave is listening
ConfigVersion	UDINT	16#03050800	16#03050800		

Рисунок 4.11.14 – Настройки компонента Modbus TCP Slave, вкладка Modbus TCP Slave Конфигурация

9. В настройках компонента **MV210\_101** на вкладке **Канал Modbus Slave** следует добавить канал, в котором с помощью функции **Read Holding Registers** будет считываться значение **4000** и **4001** регистров модуля. В данных регистрах содержится значение входа 1 в представлении с плавающей точкой. Таблица регистров модуля и поддерживаемые функции приведены в руководстве по эксплуатации.

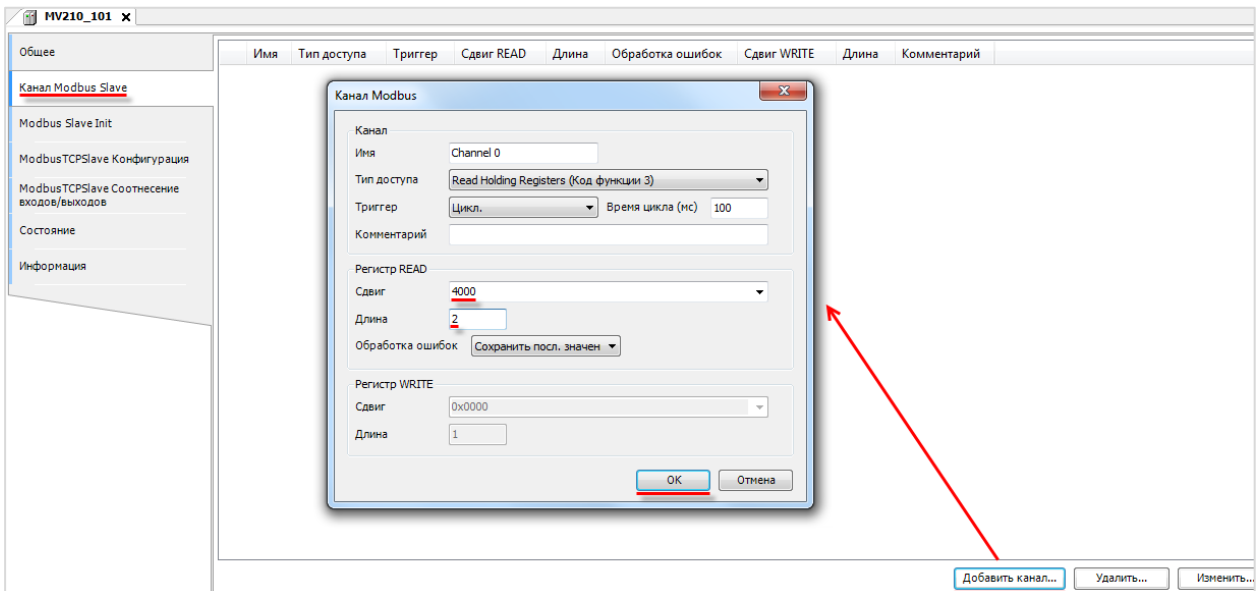


Рисунок 4.11.15 – Добавление канала в конфигурацию slave-устройства MV210\_101

На вкладке **ModbusGenericSerialSlave** **Соотнесение входов/выходов** следует привязать к каналу элементы объединения **\_2WORD\_TO\_REAL**.

Для параметра **Всегда обновлять переменные** следует установить значение **Включено 2**.

#### 4. Стандартные средства конфигурирования

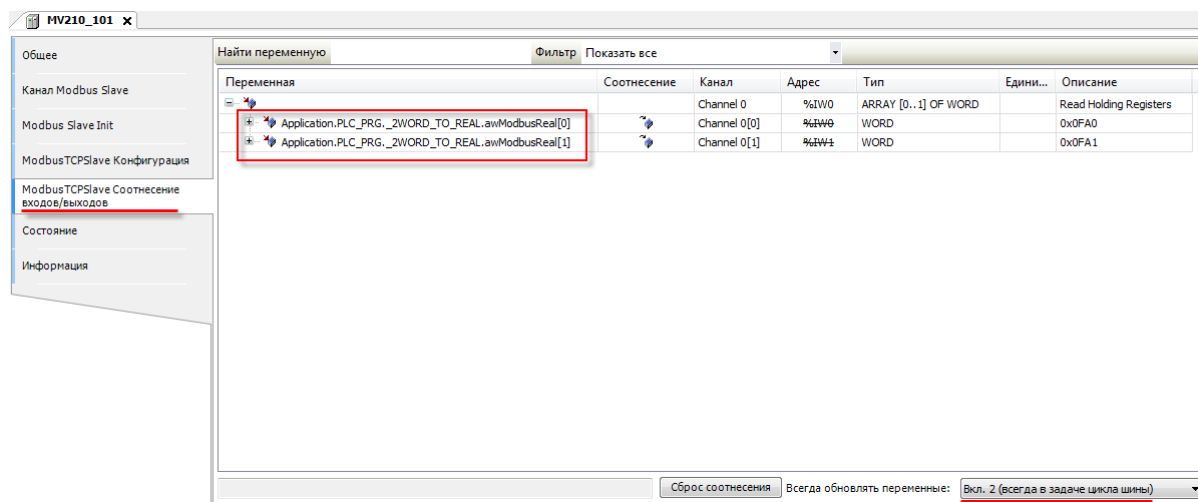
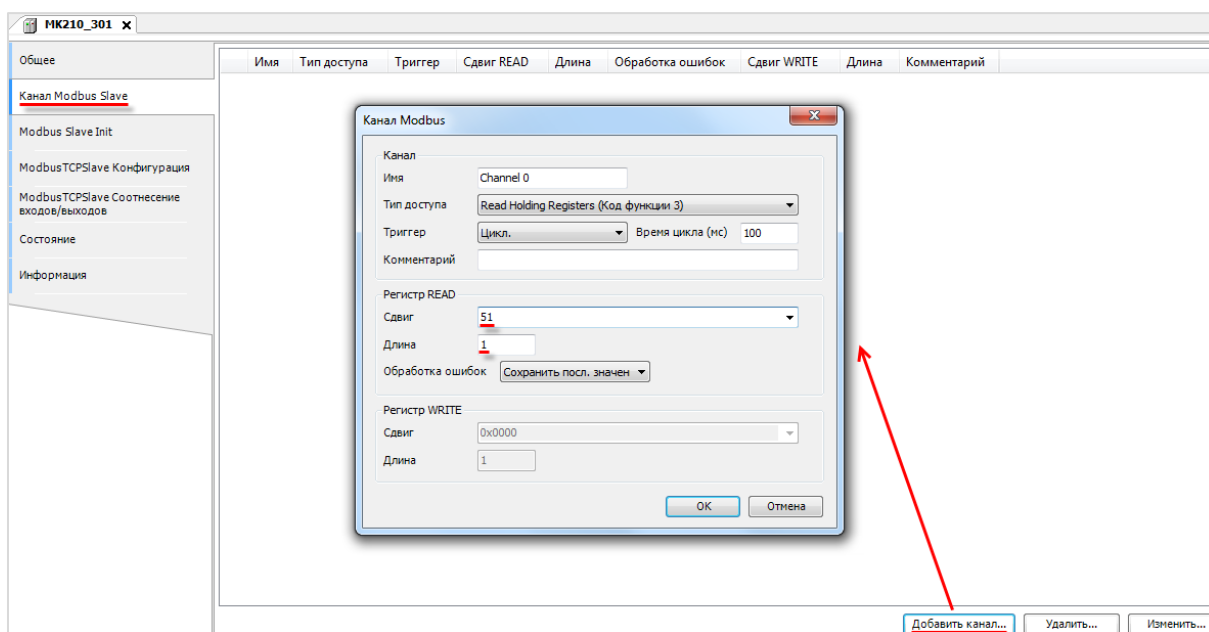


Рисунок 4.11.16 – Привязка переменных к каналу

10. В настройках компонента **MK210\_301** на вкладке **Канал Modbus Slave** следует добавить канал, в котором с помощью функции **Read Holding Registers** будет считываться значение регистра **51**. В данном регистре содержится битовая маска состояний дискретных входов. Также следует добавить канал, в котором с помощью функции **Write Multiple Registers** будет записываться значение в регистр **470**. В данном регистре содержатся значения выходов модуля в виде битовой маски. У параметра **Триггер** следует установить значение **Передний фронт**, чтобы иметь возможность управлять записью в модуль с помощью логической переменной.

Таблица регистров модуля и поддерживаемые функции приведены в руководстве по эксплуатации.



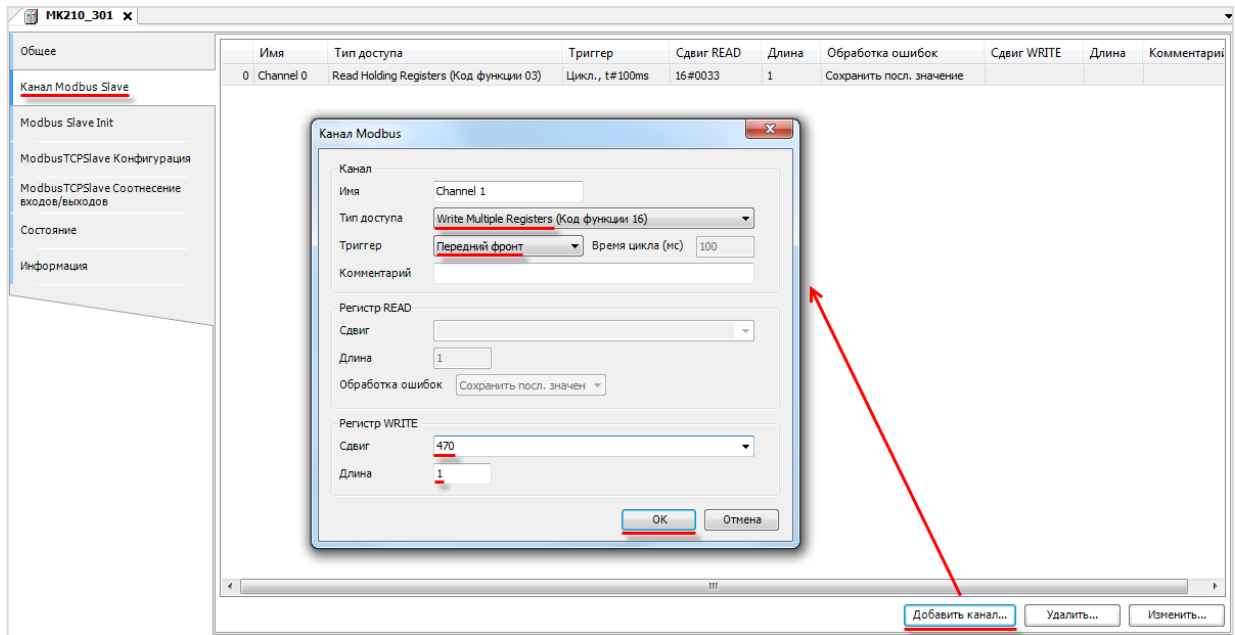


Рисунок 4.11.17 – Добавление каналов в конфигурацию slave-устройства MK210\_301

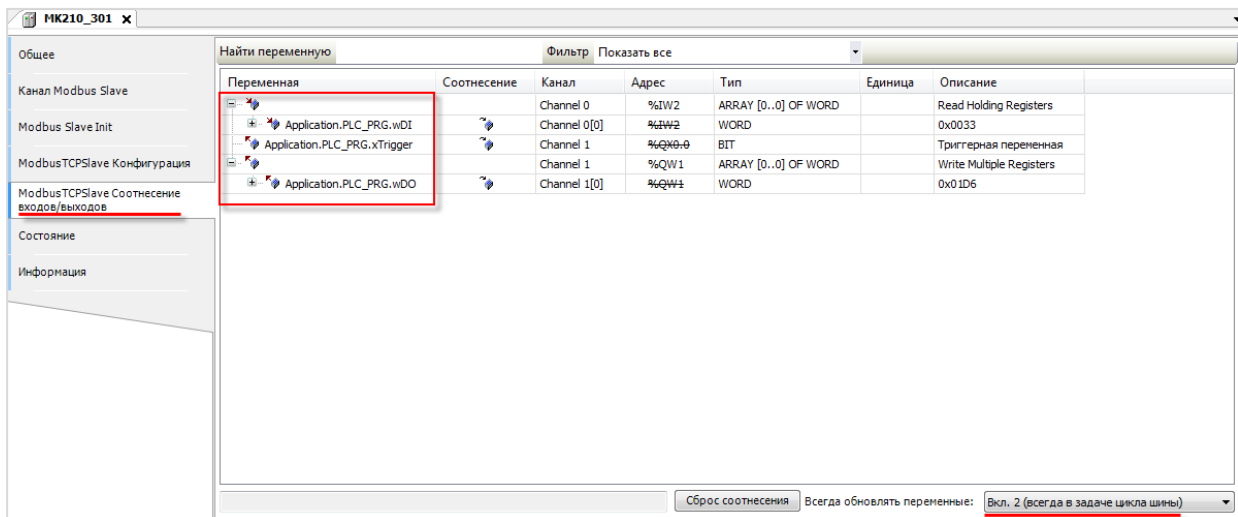


Рисунок 4.11.18 – Привязка переменных к каналам

12. Загрузить проект в контроллер и запустить его.

В переменной **\_2WORD\_TO\_REAL.rRealValue** будет отображаться текущее значение первого аналогового входа модуля **MV210\_101**. В нулевом бите переменной **wDI (wDI.0)** будет отображаться текущее значение первого дискретного входа модуля **MK210\_301**.

Если значение **\_2WORD\_TO\_REAL.rRealValue** превысит **30** и при этом значение **wDI.0** будет равно **TRUE**, то в нулевой бит переменной **wDO (wDO.0)** будет однократно (по триггеру) записано значение **TRUE**, что приведет к замыканию первого дискретного выхода модуля **MK210\_301**. Если одно из условий перестанет выполняться, то выход будет разомкнут.

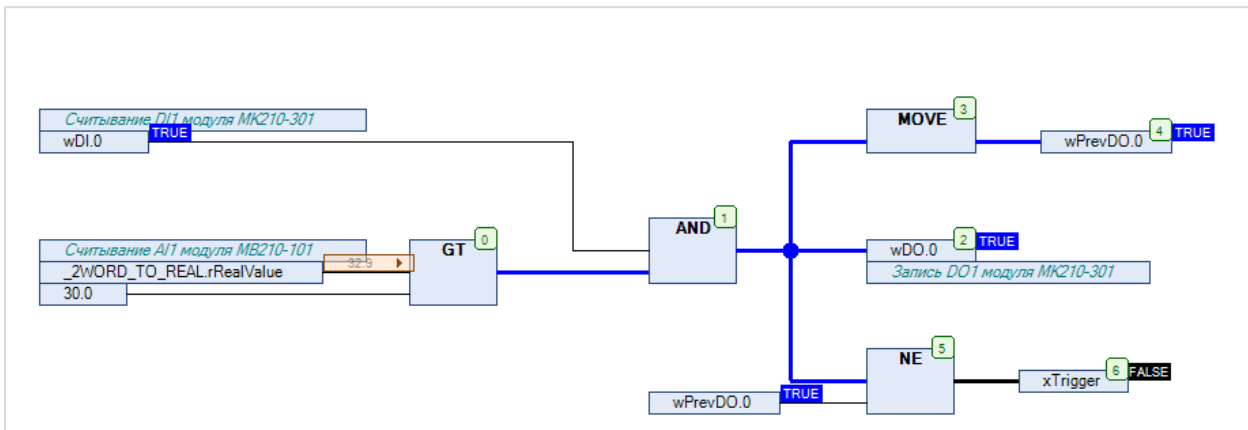


Рисунок 4.11.19 – Выполнение программы в режиме Online

## 4.12 Пример: СПК1хх [M01] (Modbus TCP Slave) + MasterOPC Universal Modbus Server

В качестве примера будет рассмотрена настройка обмена с OPC-сервером [Insat MasterOPC Universal Modbus Server](#), который будет использоваться в режиме **Modbus TCP Master**.

Структурная схема примера приведена на рисунке ниже:

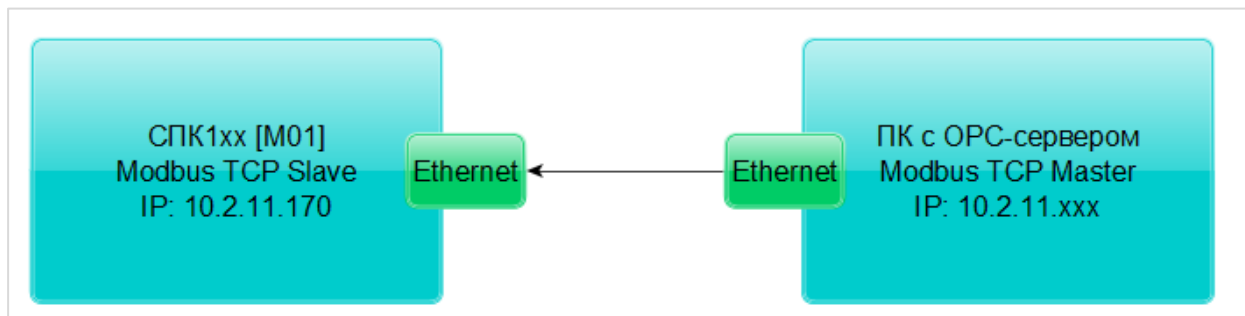


Рисунок 4.12.1 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел **Device** – **Обновить устройство**).

Пример доступен для скачивания: [Example CodesysModbusTcpSlave 3511v1.zip](#)

Сетевые параметры устройств приведены в таблице ниже:

**Таблица 4.12.1 – Сетевые параметры устройств**

Параметр	СПК1хх [M01]	ПК с OPC-сервером
IP-адрес	10.2.11.170	<i>любой адрес из сети, к которой подключен контроллер</i>
Порт		502
Режим работы	slave	master

Переменные примера описаны в таблице ниже:

**Таблица 4.12.2 – Список переменных примера**

Имя	Тип	Область памяти Modbus	Адрес регистра/бита
<b>Переменные, считываемые OPC-сервером</b>			
xVar_OpcRead	BOOL	Coils	0/0
wVar_OpcRead	WORD	Holding регистры	1
rVar_OpcRead	REAL		2–3
sVar_OpcRead	STRING(16)		4–11
<b>Переменные, записываемые OPC-сервером</b>			
xVar_OpcWrite	BOOL	Discrete inputs	0
wVar_OpcWrite	WORD	Input регистры	1
rVar_OpcWrite	REAL		2–3
sVar_OpcWrite	STRING(16)		4–11

Для настройки обмена следует:

1. Подключить контроллер и ПК с общей локальной сети.
2. Создать новый проект **CODESYS** с программой **PLC\_PRG** на языке **CFC**:

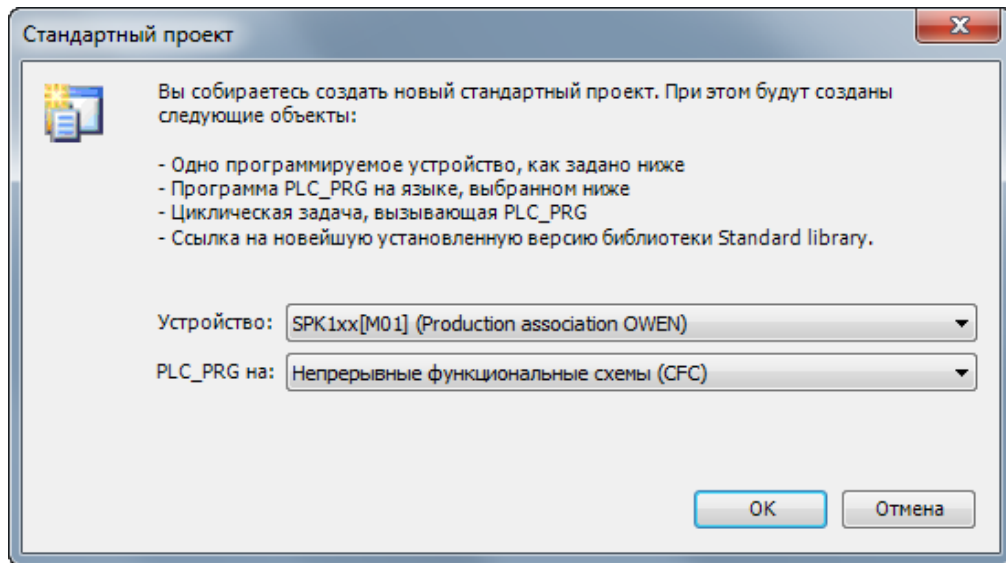


Рисунок 4.12.2 – Создание проекта CODESYS

3. Добавить в проект объединения с именами **Real\_Word** и **String\_Word**:

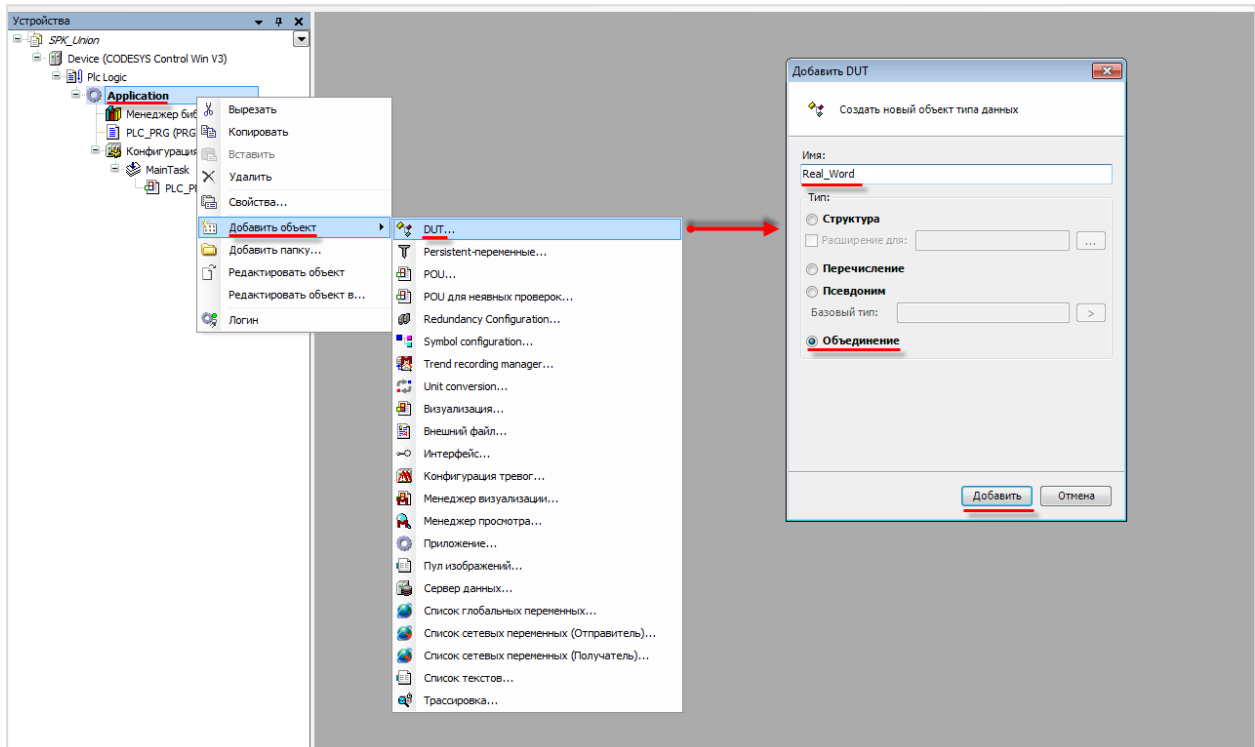
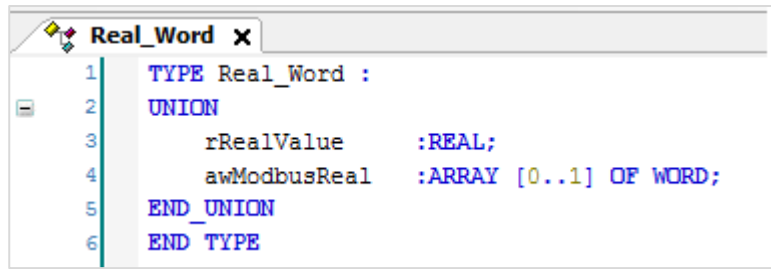


Рисунок 4.12.3 – Добавление в проект объединения Real\_Word

Объединения потребуются для преобразования переменных типов **REAL** и **STRING** в набор переменных типа **WORD** для привязки к компоненту **Modbus TCP Slave Device**.

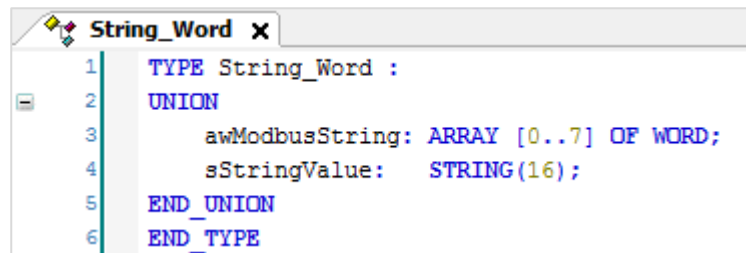


4. В объединениях объявить следующие переменные:



```
Real_Word x
1  TYPE Real_Word :
2  UNION
3      rRealValue      :REAL;
4      awModbusReal    :ARRAY [0..1] OF WORD;
5  END_UNION
6  END_TYPE
```

Рисунок 4.12.4 – Объявление переменных объединения Real\_Word

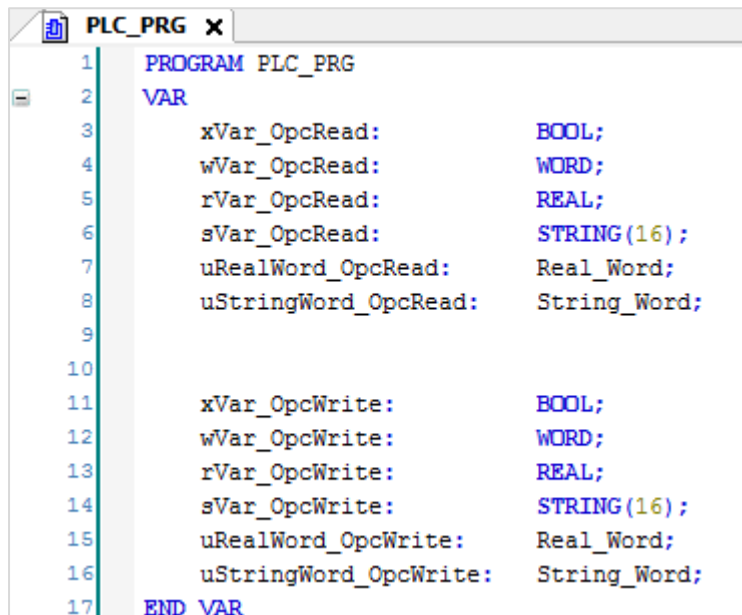


```
String_Word x
1  TYPE String_Word :
2  UNION
3      awModbusString: ARRAY [0..7] OF WORD;
4      sStringValue:  STRING(16);
5  END_UNION
6  END_TYPE
```

Рисунок 4.12.5 – Объявление переменных объединения String\_Word

5. В менеджере библиотек добавить библиотеку **CAA Memory**.

6. В программе **PLC\_PRG** объявить переменные в соответствии с таблицей 4.13.



```
PLC_PRG x
1  PROGRAM PLC_PRG
2  VAR
3      xVar_OpcRead:      BOOL;
4      wVar_OpcRead:      WORD;
5      rVar_OpcRead:      REAL;
6      sVar_OpcRead:      STRING(16);
7      uRealWord_OpcRead: Real_Word;
8      uStringWord_OpcRead: String_Word;
9
10
11     xVar_OpcWrite:      BOOL;
12     wVar_OpcWrite:      WORD;
13     rVar_OpcWrite:      REAL;
14     sVar_OpcWrite:      STRING(16);
15     uRealWord_OpcWrite: Real_Word;
16     uStringWord_OpcWrite: String_Word;
17 END VAR
```

Рисунок 4.12.6 – Объявление переменных программы

Код программы будет выглядеть следующим образом:



Рисунок 4.12.7 – Код программы PLC\_PRG

Функция **ReverseBYTEsInWORD** из библиотеки **CAA Memory** используется для изменения порядка байтов в переменной типа **STRING** для соответствия порядку байтов в OPC-сервере (функционал перестановки байт в OPC-сервере не распространяется на тип STRING).

7. Добавить компоненты **Ethernet** и **Modbus TCP Slave Device** в соответствии с [п. 4.4](#).  
 Настроить компонент **Modbus TCP Slave Device** в соответствии с рисунком ниже:

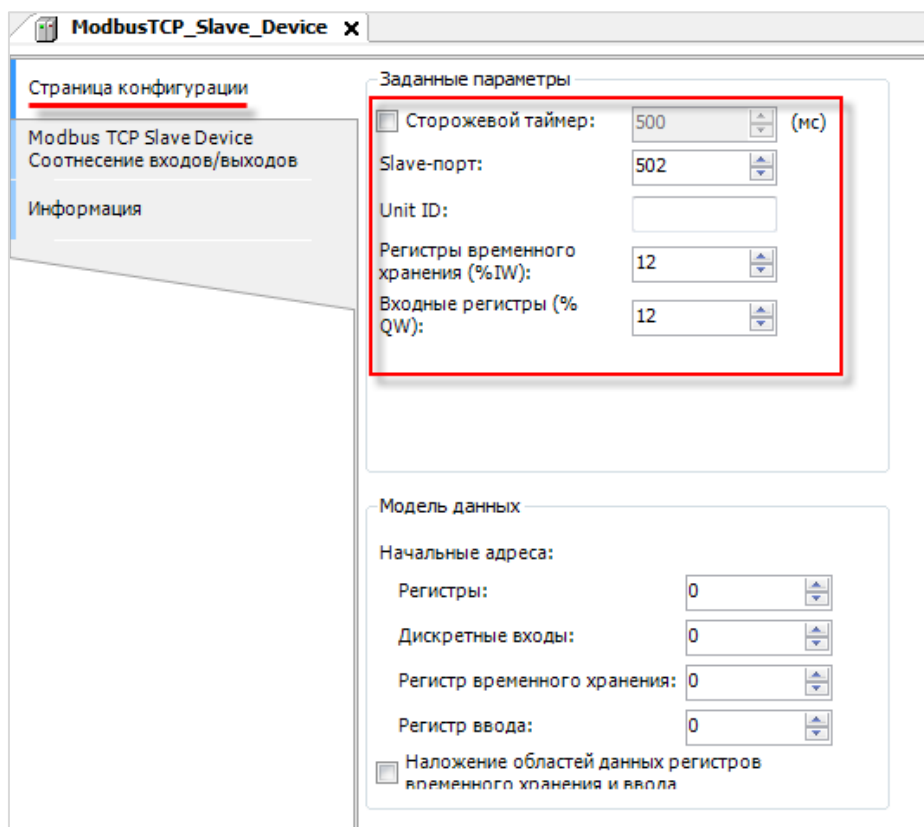


Рисунок 4.12.8 – Настройки компонента Modbus TCP Slave Device

8. Привязать к каналам компонента **Modbus TCP Slave Device** переменные программы в соответствии с [таблицей 4.12.2](#). Установить галочку **Вкл. 2 (Всегда в задаче цикла шины)**.

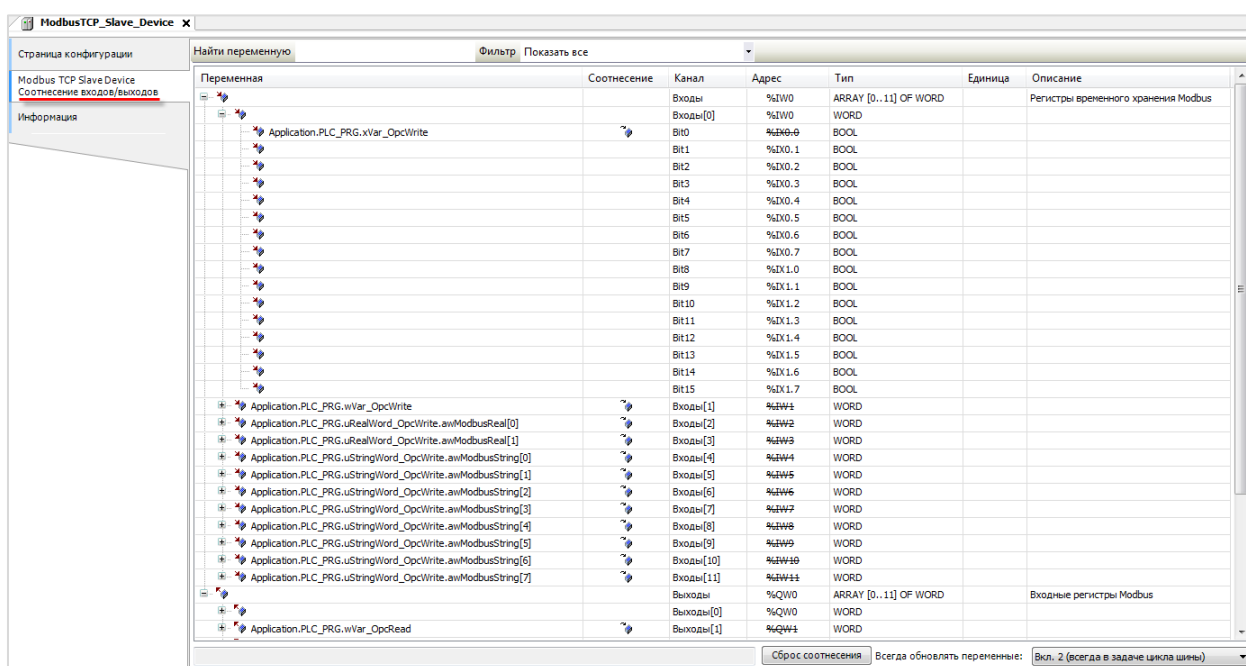


Рисунок 4.12.9 – Привязка переменных к компоненту Modbus TCP Slave Device (holding регистры)

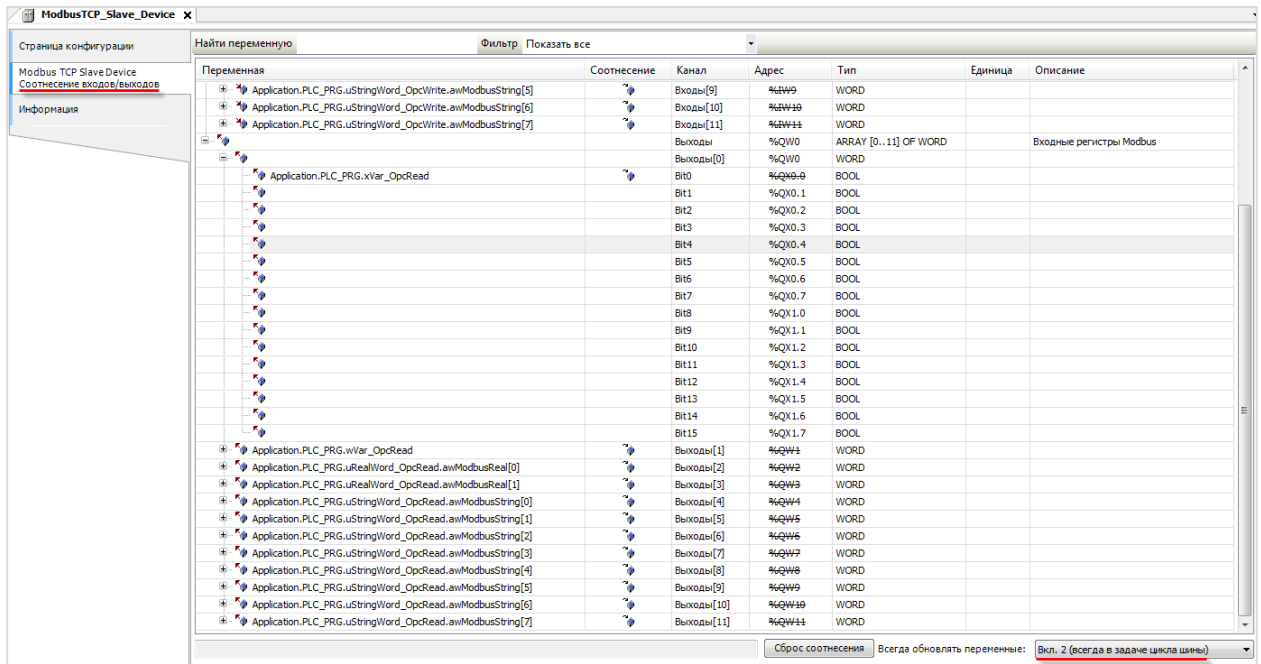


Рисунок 4.12.10 – Привязка переменных к компоненту Modbus TCP Slave Device (input регистры)

9. Установить и запустить [MasterOPC Universal Modbus Server](#).

10. Нажать ПКМ на узел **Server** и добавить коммуникационный узел типа **TCP/IP**. В узле указать сетевые настройки в соответствии с [таблицей 4.11.2](#). Для работы OPC-сервера в режиме **Modbus TCP Master** параметр **Slave подключение** должны иметь значение **FALSE**.

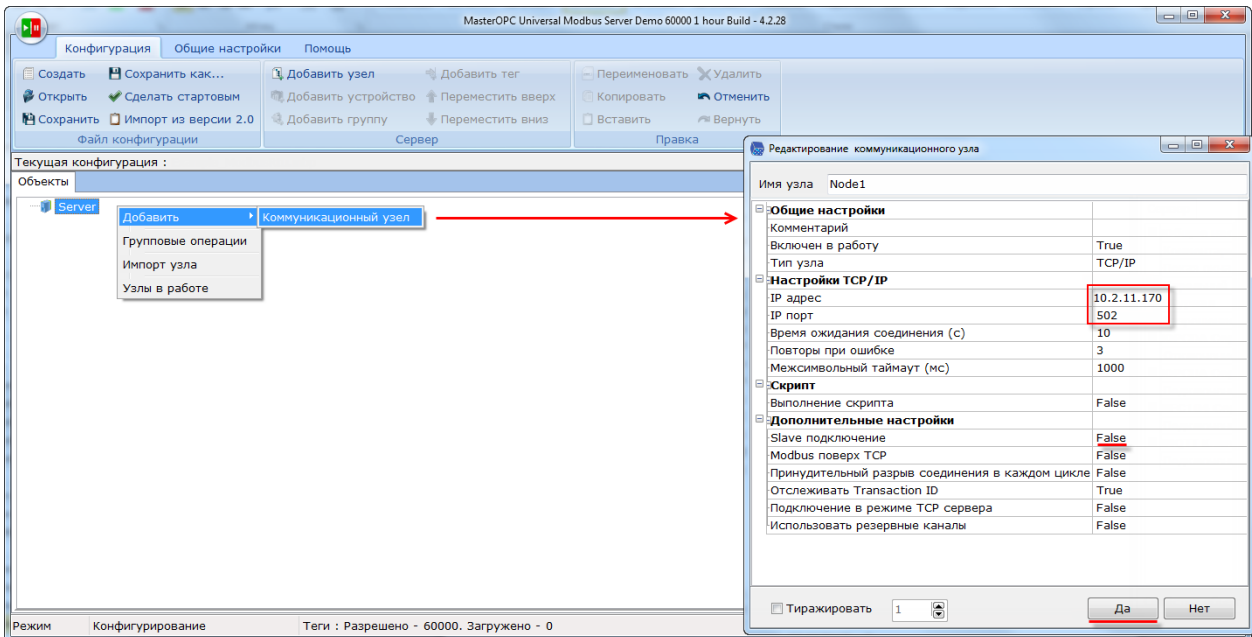


Рисунок 4.12.11 – Добавление коммуникационного узла

#### 4. Стандартные средства конфигурирования

11. Нажать ПКМ на коммуникационный узел и добавить устройство с настройками по умолчанию.

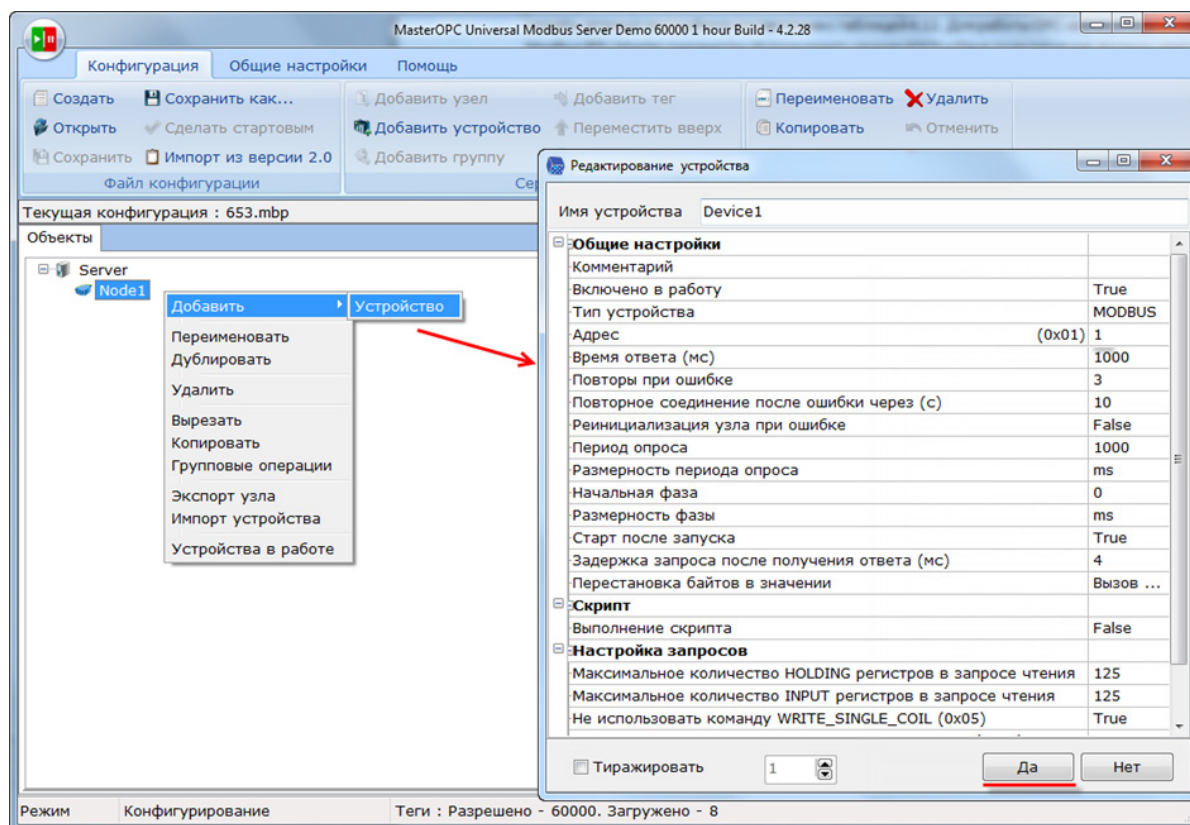


Рисунок 4.12.12 – Добавление устройства

12. Нажать ПКМ на устройство и добавить 8 тегов. Число тегов соответствует числу переменных, считываемых/записываемых OPC-сервером. Настройки тегов приведены ниже.

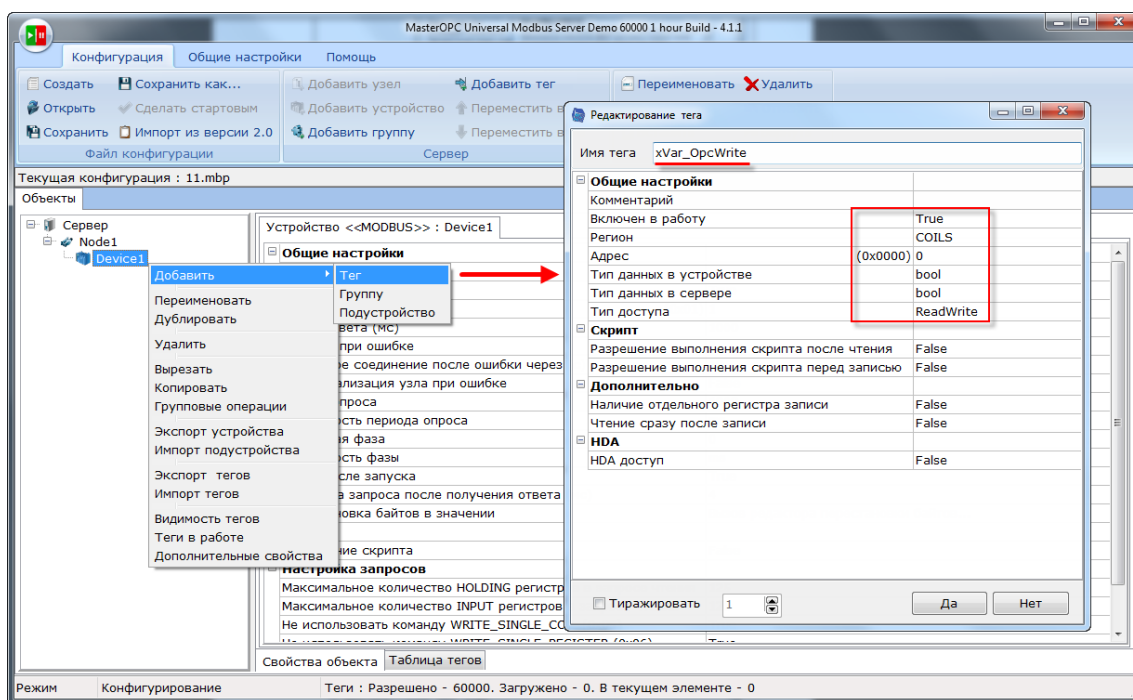


Рисунок 4.12.13 – Добавление тега xVar\_OpcWrite

Имя тега	<u>wVar_OpcWrite</u>	
<b>Общие настройки</b>		
Комментарий		
Включен в работу	True	
Регион	HOLDING_REGISTERS	
Адрес	(0x0001)	1
Тип данных в устройстве	uint16	
Тип данных в сервере	uint32	
Тип доступа	ReadWrite	
Использовать перестановку байтов устройства	True	
Последний тег в групповом запросе	False	
Пересчет (A*X + B)	False	

Рисунок 4.12.14 – Добавление тега wVar\_OpcWrite

Имя тега	<u>rVar_OpcWrite</u>	
<b>Общие настройки</b>		
Комментарий		
Включен в работу	True	
Регион	HOLDING_REGISTERS	
Адрес	(0x0002)	2
Тип данных в устройстве	float	
Тип данных в сервере	float	
Тип доступа	ReadWrite	
Использовать перестановку байтов устройства	False	
Перестановка байтов в значении	10325476	
Последний тег в групповом запросе	False	
Пересчет (A*X + B)	False	

Рисунок 4.12.15 – Добавление тега rVar\_OpcWrite

Имя тега	<u>sVar_OpcWrite</u>	
<b>Общие настройки</b>		
Комментарий		
Включен в работу	True	
Регион	HOLDING_REGISTERS	
Адрес	(0x0004)	4
Тип данных в устройстве	string	
Тип данных в сервере	string	
Количество байт для строкового типа	16	
Тип строки для строкового типа	ascii	
Тип доступа	ReadWrite	
Использовать перестановку байтов устройства	True	
Последний тег в групповом запросе	False	
Пересчет (A*X + B)	False	

Рисунок 4.12.16 – Добавление тега sVar\_OpcWrite

Имя тега	<u>xVar_OpcRead</u>	
<b>Общие настройки</b>		
Комментарий		
Включен в работу	True	
Регион	DISCRETE_INPUTS	
Адрес	(0x0001)	0
Тип данных в устройстве	bool	
Тип данных в сервере	bool	
Тип доступа	ReadOnly	

Рисунок 4.12.17 – Добавление тега xVar\_OpcRead

Имя тега	<u>wVar_OpcRead</u>	
<b>Общие настройки</b>		
Комментарий		
Включен в работу	True	
Регион	INPUT_REGISTERS	
Адрес	(0x0001)	1
Тип данных в устройстве	uint16	
Тип данных в сервере	uint32	
Тип доступа	ReadOnly	

Рисунок 4.12.18 – Добавление тега wVar\_OpcRead

Имя тега	<u>rVar_OpcRead</u>	
<b>Общие настройки</b>		
Комментарий		
Включен в работу	True	
Регион	INPUT_REGISTERS	
Адрес	(0x0002)	2
Тип данных в устройстве	float	
Тип данных в сервере	float	
Тип доступа	ReadOnly	
Использовать перестановку байтов устройства	False	
Перестановка байтов в значении	10325476	
Последний тег в групповом запросе	False	
Пересчет (A*X + B)	False	

Рисунок 4.12.19 – Добавление тега rVar\_OpcRead

Имя тега <u>sVar_OpcRead</u>	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Регион	INPUT_REGISTERS
Адрес	(0x0004) 4
Тип данных в устройстве	string
Тип данных в сервере	string
Количество байт для строкового типа	16
Тип строки для строкового типа	ascii
Тип доступа	ReadOnly

Рисунок 4.12.20 – Добавление тега sVar\_OpcRead

13. Загрузить проект в контроллер и запустить его. Запустить OPC-сервер для контроля значений переменных.

В редакторе CODESYS следует изменить значения **OpcRead** переменных и наблюдать соответствующие изменения в OPC-сервере. В OPC-сервере следует изменить значения **OpcWrite** переменных и наблюдать соответствующие значения в CODESYS.

Device.Application.PLC_PRG		
Выражение	Тип	Значение
xVar_OpcRead	BOOL	TRUE
wVar_OpcRead	WORD	11
rVar_OpcRead	REAL	22.33
sVar_OpcRead	STRING(16)	'тест'
uRealWord_OpcRead	Real_Word	
uStringWord_OpcRead	String_Word	
xVar_OpcWrite	BOOL	TRUE
wVar_OpcWrite	WORD	44
rVar_OpcWrite	REAL	55.66
sVar_OpcWrite	STRING(16)	'test'
uRealWord_OpcWrite	Real_Word	
uStringWord_OpcWrite	String_Word	

MasterOPC Universal Modbus Server Demo 60000 1 hour Build - 4.2.28						
Стартовая конфигурация : 653.mbp						
Объекты						
Server	Устройство <<Device1>>					
Node1	Теги					
Device1	Имя	Регион	Адрес	Значение	Качество	Время (UTC)
xVar_OpcRead	Node1.Device1.xVa...	DISC...	(0x00...	True	GOOD	2019-07-0...
wVar_OpcRead	Node1.Device1.wV...	INPU...	(0x00...	11	GOOD	2019-07-0...
rVar_OpcRead	Node1.Device1.rVa...	INPU...	(0x00...	22.330000	GOOD	2019-07-0...
sVar_OpcRead	Node1.Device1.sVa...	INPU...	(0x00...	тест	GOOD	2019-07-0...
xVar_OpcWrite	Node1.Device1.xVa...	COILS	(0x00...	True	GOOD	2019-07-0...
wVar_OpcWrite	Node1.Device1.wV...	HOL...	(0x00...	44	GOOD	2019-07-0...
rVar_OpcWrite	Node1.Device1.rVa...	HOL...	(0x00...	55.660000	GOOD	2019-07-0...
sVar_OpcWrite	Node1.Device1.sVa...	HOL...	(0x00...	test	GOOD	2019-07-0...

Рисунок 4.12.21 – Считывание и запись данных через OPC-сервер



## 5 Библиотека OwenCommunication

### 5.1 Основная информация

Библиотека **OwenCommunication** используется для реализации обмена в программе пользователя. Библиотека лишена ограничений стандартных средств конфигурирования и включает дополнительный функционал. Библиотека содержит:

- [ФБ для настройки интерфейсов](#);
- [ФБ обмена по Modbus](#);
- [ФБ для реализации нестандартных протоколов](#);
- [функции и ФБ преобразования данных](#).



#### ПРИМЕЧАНИЕ

Работа библиотеки поддерживается только на контроллерах ОВЕН и виртуальном контроллере **CODESYS Control Win V3**.



#### ПРИМЕЧАНИЕ

Библиотека соответствует PLCopen Behavior Model.

### 5.2 Установка библиотеки

Библиотека **OwenCommunication** доступна на сайте компании [ОВЕН](#) в разделе [CODESYS V3/Библиотеки и компоненты](#). Для установки библиотеки в **CODESYS** в меню **Инструменты** следует выбрать пункт **Репозиторий библиотек**, после чего нажать **Установить** и указать путь к файлу библиотеки:

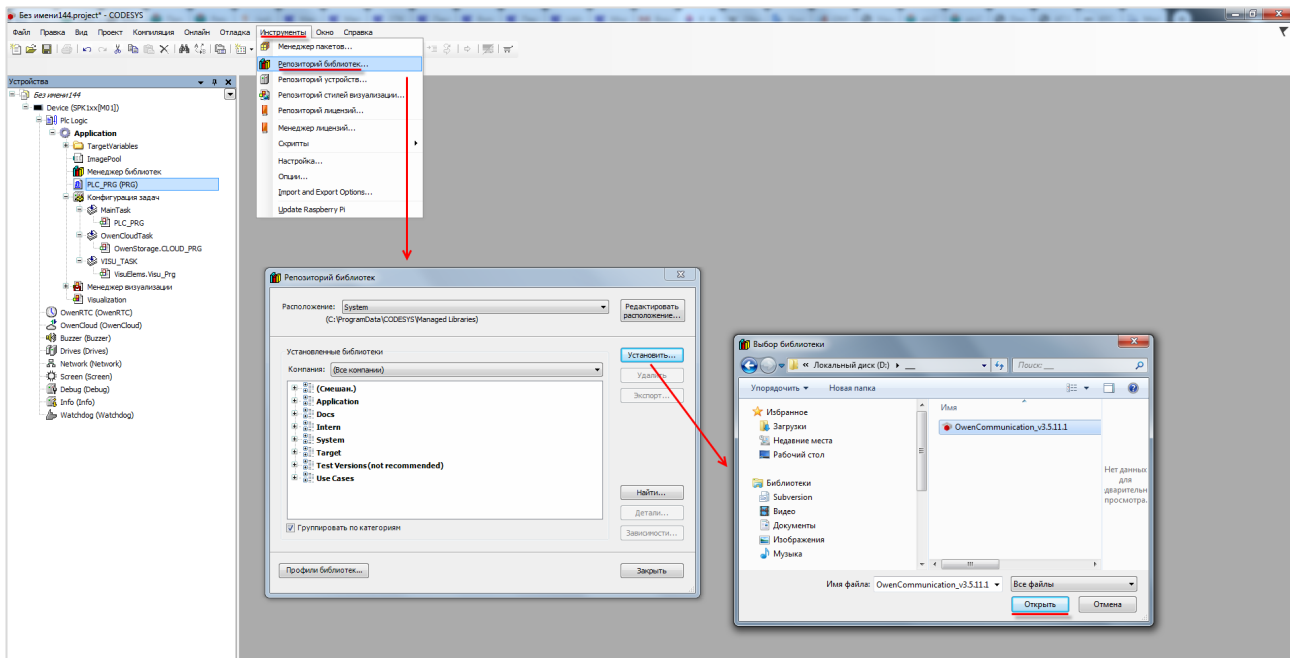


Рисунок 5.2.1 – Установка библиотеки

### 5.3 Добавление библиотеки в проект CODESYS

Для добавления библиотеки **OwenCommunication** в проект **CODESYS** в **Менеджере библиотек** следует нажать кнопку **Добавить библиотеку**, в появившемся списке выбрать библиотеку **OwenCommunication** и нажать **ОК**.

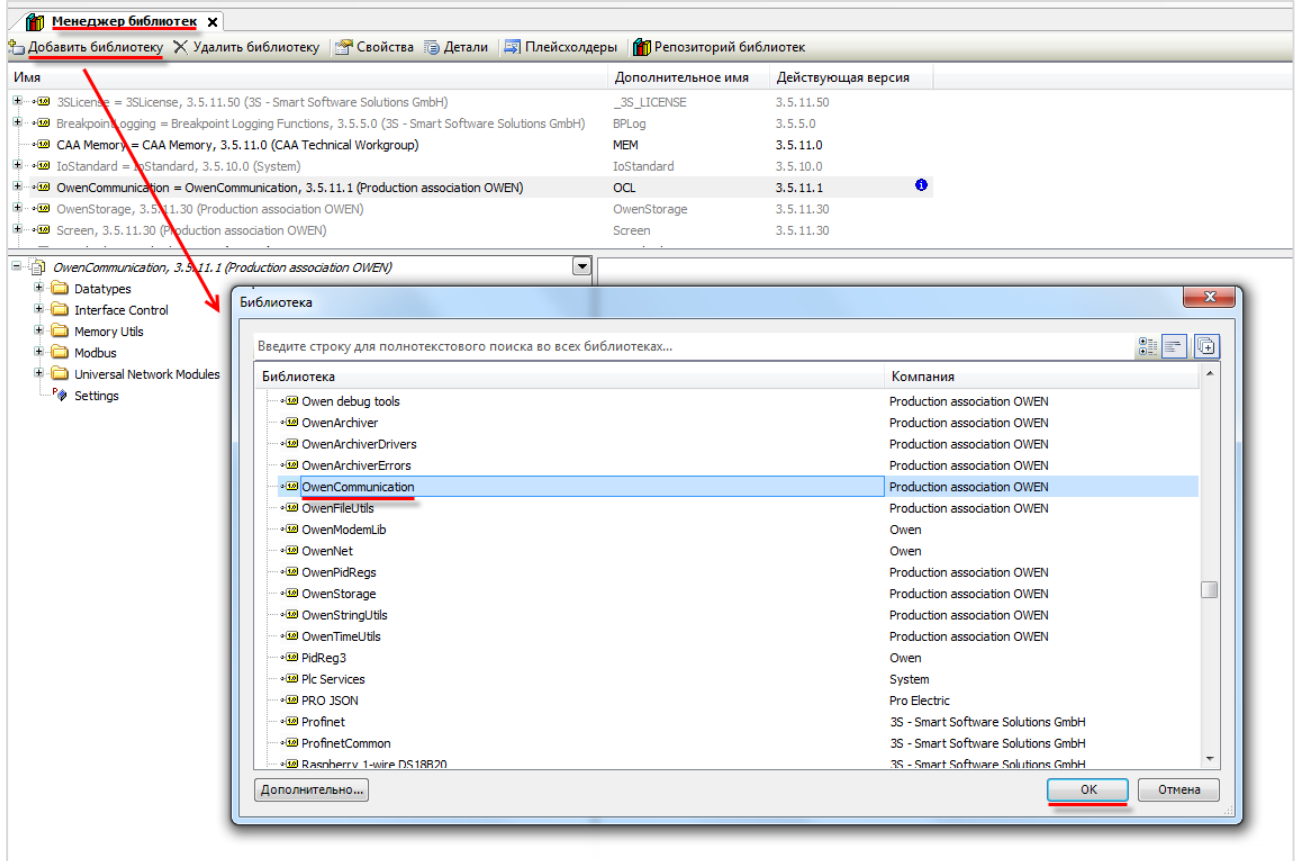


Рисунок 5.3.1 – Добавление библиотеки OwenCommunication

После добавления библиотека появится в списке **Менеджера библиотек**:

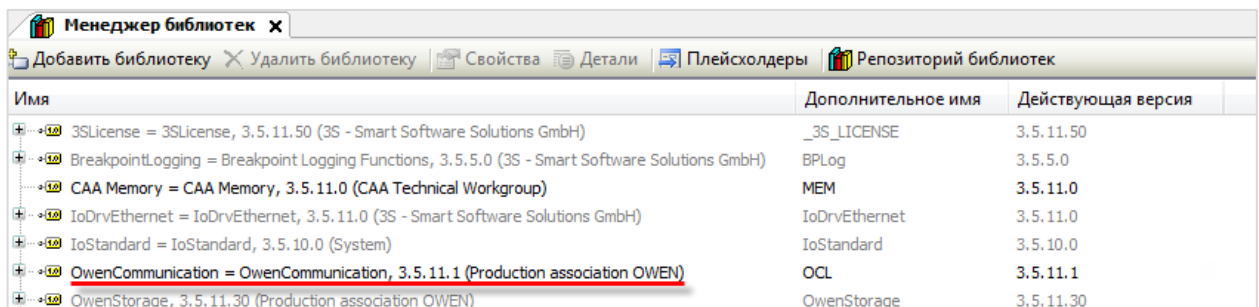


Рисунок 5.3.2 – Список библиотек проекта



#### ПРИМЕЧАНИЕ

При обращении к функциям и ФБ библиотеки следует перед их названием указывать префикс **OCL** (пример: **OCL.COM\_Control**).

## 5.4 Структуры и перечисления

### 5.4.1 Перечисление ERROR

Перечисление **ERROR** содержит ошибки, которые могут возникнуть при вызове ФБ библиотеки.

Таблица 5.4.1 – Описание элементов перечисления ERROR

Название	Значение	Описание
<b>Общие ошибки</b>		
NO_ERROR	0	Нет ошибок
TIME_OUT	1	Ошибка таймаута
HANDLE_INVALID	10	Некорректное значение дескриптора интерфейса
ERROR_UNKNOWN	11	Неизвестная ошибка (зарезервировано для будущих версий)
WRONG_PARAMETER	12	Как минимум один из аргументов ФБ имеет некорректное значение
WRITE_INCOMPLETE	13	Отправка сообщения не была завершена
INVALID_DATAPOINTER	20	Некорректный указатель на буфер данных
INVALID_DATASIZE	21	Некорректный размер буфера данных
INVALID_ADDR	22	Некорректное значение в поле IP-адреса
<b>Ошибки UDP/TCP</b>		
UDP_RECEIVE_ERROR	30	Ошибка получения UDP-запроса
UDP_SEND_ERROR	31	Ошибка отправки UDP-запроса
UDP_SEND_NOT_COMPLETE	32	Отправка UDP-запроса не была завершена (зарезервировано для будущих версий)
UDP_OPEN_ERROR	33	Ошибка создания UDP-сокета
UDP_CLOSE_ERROR	34	Ошибка закрытия UDP-сокета
TCP_SEND_ERROR	40	Ошибка отправки TCP-запроса
TCP_RECEIVE_ERROR	41	Ошибка получения TCP-запроса
TCP_OPEN_ERROR	42	Ошибка создания TCP-сокета
TCP_CONNECT_ERROR	43	Ошибка при установке TCP-соединения
TCP_CLOSE_ERROR	44	Ошибка при закрытия TCP-соединения
TCP_SERVER_ERROR	45	Ошибка TCP-сервера (зарезервировано для будущих версий)
TCP_NO_CONNECTION	46	TCP-соединение отсутствует
IOCTL_ERROR	47	Внутренняя ошибка при использовании системных вызовов
<b>Ошибки Modbus</b>		
ILLEGAL_FUNCTION	50	Данная функция Modbus не поддерживается slave-устройством
ILLEGAL_DATA_ADDRESS	51	Как минимум один из регистров, указанных в запросе, отсутствует в slave-устройстве
ILLEGAL_DATA_VALUE	52	Некорректное значение в поле данных
SLAVE_DEVICE_FAILURE	53	Slave-устройство не может обработать данный запрос
<b>Специфические ошибки</b>		
RESPONSE_CRC_FAIL	60	Рассчитанная CRC не соответствует CRC посылки
NOT_OWEN_DEVICE	61	Данное устройство не является контроллером OWEN

#### 5.4.2 Перечисление COM\_PARITY

Перечисление **COM\_PARITY** описывает режим контроля четности COM-порта.

**Таблица 5.4.2 – Описание элементов перечисления COM\_PARITY**

Название	Значение	Описание
EVEN	0	Проверка на четность
ODD	1	Проверка на нечетность
NONE	2	Проверка отсутствует

#### 5.4.3 Перечисление COM\_STOPBIT

Перечисление **COM\_STOPBIT** описывает возможное число стоп-битов при обмене через COM-порт.

**Таблица 5.4.3 – Описание элементов перечисления COM\_STOPBIT**

Название	Значение	Описание
ONE	0	Один стоп-бит
ONE_HALF	1	Полтора стоп-бита
TWO	2	Два стоп-бита

#### 5.4.4 Перечисление MB\_FC

Перечисление **MB\_FC** описывает используемую [функцию Modbus](#).

**Таблица 5.4.4 – Описание элементов перечисления MB\_FC**

Название	Значение	Описание
READ_COILS	16#01	Чтение значений из регистров флагов
READ_DISCRETE_INPUTS	16#02	Чтение значений из дискретных входов
READ_HOLDING_REGISTERS	16#03	Чтение значений из регистров хранения
READ_INPUT_REGISTERS	16#04	Чтение значений из регистров ввода
WRITE_SINGLE_COIL	16#05	Запись значения в один регистр флага
WRITE_SINGLE_REGISTER	16#06	Запись значения в один регистр хранения
WRITE_MULTIPLE_COILS	16#0F	Запись значений в несколько регистров флагов
WRITE_MULTIPLE_REGISTERS	16#10	Запись значений в несколько регистров хранения

#### 5.4.5 Структура MB\_REQ\_INFO

Структура **MB\_REQ\_INFO** описывает запрос Modbus, полученный slave-устройством. Структура используется в ФБ [MB\\_SerialSlave](#) и [MB\\_TcpSlave](#) для предоставления пользователю информации о запросах, поступающих от master-устройства, а также для запрета обработки определенных запросов.

**Таблица 5.4.5 – Описание элементов структуры MB\_REQ\_INFO**

Название	Тип	Описание
eFuncCode	<a href="#">MB_FC</a>	Код функции Modbus, указанный в запросе
uiDataAddr	UINT	Начальный адрес регистра, указанный в запросе
uiDataCount	UINT	Количество считываемых или записываемых битов/регистров, указанное в запросе
xForbidden	BOOL	<b>TRUE</b> – запрос считается запрещенным для данного экземпляра ФБ, <b>FALSE</b> – запрос считается разрешенным

Запрет запросов работает следующим образом: если блок получает запрос с кодом функции **eFuncCode**, в котором хотя бы один из считываемых или записываемых битов/регистров попадает в диапазон **[uiDataAddr...uiDataAddr+uiDataCount-1]**, то этот запрос игнорируется, а master-устройству в ответ отправляется сообщение с кодом ошибки **02 (ILLEGAL\_DATA\_ADDRESS)**.

Если **uiDataAddr = 16#FFFF**, то блок игнорирует все запросы с кодом функции **eFuncCode**. В ответ master-устройству будет отправлено сообщение с кодом ошибки **01 (ILLEGAL\_FUNCTION)**. Это, например, может использоваться для создания slave-устройства, все регистры которого доступны только для чтения.

Изменение элементов структуры сразу влияет на работу блока (то есть воздействие на вход **xEnable** для применения новых значений не требуется).

## 5.5 ФБ настройки интерфейсов

### 5.5.1 ФБ COM\_Control

Функциональный блок **COM\_Control** используется для открытия COM-порта с заданными настройками, а также его закрытия.



#### ПРИМЕЧАНИЕ

Не допускается открытие уже используемого COM-порта (например, добавленного в проект с помощью [стандартных средств конфигурирования](#)).

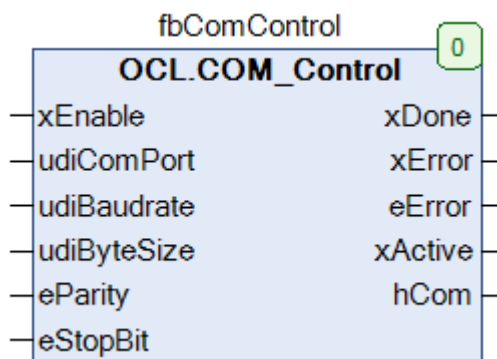


Рисунок 5.5.1 – Внешний вид ФБ COM\_Control на языке CFC

Таблица 5.5.1 – Описание входов и выходов ФБ COM\_Control

Название	Тип	Описание
<b>Входы</b>		
xEnable	BOOL	По переднему фронту происходит открытие COM-порта, по заднему – закрытие
udiComPort	UDINT	<a href="#">Номер COM-порта</a>
udiBaudrate	UDINT	Скорость обмена в бодах. Стандартные возможные значения: <b>1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200</b>
udiByteSize	UDINT(7..8)	Число бит данных ( <b>7</b> или <b>8</b> )
eParity	<a href="#">COM_PARITY</a>	Режим контроля четности
eStopBit	<a href="#">COM_STOPBIT</a>	Число стоп-бит
<b>Выходы</b>		
xDone	BOOL	Принимает <b>TRUE</b> на один цикл ПЛК при успешном открытии порта
xError	BOOL	Принимает значение <b>TRUE</b> в случае возникновения ошибки
eError	<a href="#">ERROR</a>	Статус работы ФБ (или код ошибки)
xActive	BOOL	Пока порт открыт, данный выход имеет значение <b>TRUE</b>
hCom	CAA.HANDLE	Дескриптор COM-порта

## 5.5.2 ФБ TCP\_Client

Функциональный блок **TCP\_Client** используется для открытия и закрытия TCP-соединения.

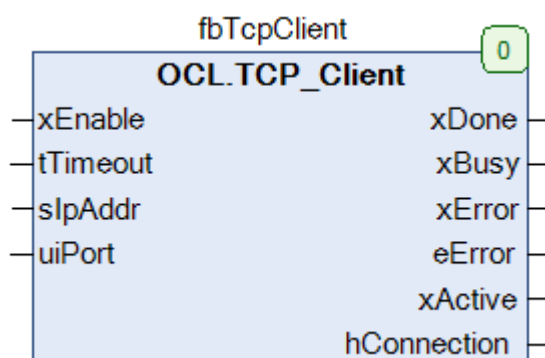


Рисунок 5.5.2 – Внешний вид ФБ TCP\_Client на языке CFC

Таблица 5.5.2 – Описание входов и выходов ФБ TCP\_Client

Название	Тип	Описание
<b>Входы</b>		
xEnable	BOOL	По переднему фронту происходит открытие TCP-соединения, по заднему – закрытие
tTimeOut	TIME	Таймаут установки соединения (0 – время ожидания не ограничено)
slpAddr	STRING	IP-адрес сервера в формате <a href="#">IPv4</a> ('xxx.xxx.xxx.xxx')
uiPort	UINT	Порт сервера
<b>Выходы</b>		
xDone	BOOL	<b>TRUE</b> – сервер закрыл соединение. Для повторного соединения требуется создать передний фронт на входе <b>xEnable</b>
xBusy	BOOL	<b>TRUE</b> – выполняется установка соединения
xError	BOOL	Принимает значение <b>TRUE</b> в случае возникновения ошибки
eError	<a href="#">ERROR</a>	Статус работы ФБ (или код ошибки)
xActive	BOOL	Пока соединение активно, данный выход имеет значение <b>TRUE</b>
hConnection	CAA.HANDLE	Дескриптор соединения

## 5.6 ФБ протокола Modbus

### 5.6.1 ФБ MB\_SerialRequest

Функциональный блок **MB\_SerialRequest** используется для работы в режиме **Modbus Serial Master**. По переднему фронту на входе **xExecute** происходит отправка запроса, определяемого параметрами **usiSlaveld**, **eFuncCode**, **uiDataAddr** и **uiDataCount** по протоколу **Modbus RTU** (если **xIsAsciiMode = FALSE**) или **Modbus ASCII** (если **xIsAsciiMode = TRUE**) через COM-порт, определяемый дескриптором **hCom**, полученным от ФБ [COM\\_Control](#). Считываемые или записываемые данные размещаются в буфере, расположенном по указателю **pData** размером **szSize** байт.

Ответ от slave-устройства ожидается в течение времени **tTimeout**. В случае отсутствия ответа ФБ повторяет запрос. Число повторений определяется входом **usiRetry** (значение **0** соответствует отсутствию повторений). Если ни на один из запросов не был получен ответ, то выход **xError** принимает значение **TRUE**, а выход **eError = TIME\_OUT**. В случае получения корректного ответа выход **xDone** принимает значение **TRUE**, а выход **eError = NO\_ERROR**. В случае получения ответа с кодом ошибки Modbus **xError** принимает значение **TRUE**, а выход **eError** содержит код ошибки (при этом выход **xDone** не принимает значение **TRUE**). Для отправки следующего запроса следует создать передний фронт на входе **xExecute**.



#### ПРИМЕЧАНИЕ

В случае использования широковещательной рассылки (на адрес **0**) рекомендуется для параметра **tTimeout** установить значение **T#1ms**, так как получение ответа в данном случае не подразумевается.

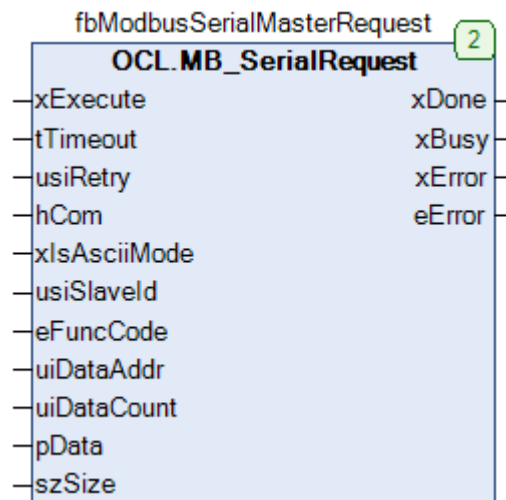


Рисунок 5.6.1 – Внешний вид ФБ MB\_SerialRequest на языке CFC



Таблица 5.6.1 – Описание входов и выходов ФБ MB\_SerialRequest

Название	Тип	Описание
<b>Входы</b>		
xExecute	BOOL	По переднему фронту происходит однократная (с возможностью переповторов в случае отсутствия ответа) отправка запроса
tTimeout	TIME	Таймаут ожидания ответа от slave-устройства ( <b>T#0ms</b> – время ожидания не ограничено)
usiRetry	USINT	Число переповторов в случае отсутствия ответа
hCom	CAA.HANDLE	Дескриптор COM-порта, полученный от ФБ <a href="#">COM_Control</a>
xIsAsciiMode	BOOL	Используемый протокол: <b>FALSE</b> – Modbus RTU, <b>TRUE</b> – Modbus ASCII
usiSlaveId	USINT	Адрес slave-устройства ( <b>0</b> – широковещательная рассылка)
eFuncCode	<a href="#">MB_FC</a>	Используемая функция Modbus
uiDataAddr	UINT	Начальный адрес бита/регистра в запросе
uiDataCount	UINT	Число битов/регистров в запросе
pData	CAA.PVOID	Указатель на буфер записываемых или считываемых данных
szSize	CAA.SIZE	Размер буфера в байтах
<b>Выходы</b>		
xDone	BOOL	<b>TRUE</b> – получен корректный ответ от slave-устройства
xBusy	BOOL	<b>TRUE</b> – ФБ находится в работе
xError	BOOL	Принимает значение <b>TRUE</b> в случае возникновения ошибки
eError	<a href="#">ERROR</a>	Статус работы ФБ (или код ошибки)

### 5.6.2 ФБ MB\_SerialSlave

Функциональный блок **MB\_SerialSlave** используется для работы в режиме **Modbus Serial Slave**. Пока вход **xEnable** имеет значение **TRUE**, блок находится в работе. На вход **hCom** следует подать дескриптор используемого COM-порта, полученный с помощью ФБ [COM\\_Control](#). Вход **usiSlaveId** определяет адрес slave'a. Под регистры slave'a выделяется область памяти по указателю **pData** размером **szSize** байт.

Блок поддерживает протоколы Modbus RTU и Modbus ASCII. Протокол запроса определяется автоматически, и ответ отправляется в том же формате.

Блок поддерживает все стандартные функции Modbus, приведенные в [таблице 2.2](#).

В реализации блока все области памяти Modbus наложены друг на друга и имеют общую адресацию (**MODBUS Data Model with only 1 block** согласно спецификации Modbus). Максимальный размер буфера slave'a – **65536** регистров.

Блок поддерживает получение широковещательных запросов (отправленных на адрес **0**).

При получении запроса от master-устройства выход **xNewRequest** на один цикл контроллера принимает значение **TRUE**, при этом выход **stRequestInfo** содержит информацию о полученном запросе.

Блок позволяет запретить обработку определенных запросов, полученных от master-устройства. Для этого на вход **pastForbiddenRequest** передается указатель на структуру (или массив структур) типа [MB\\_REQ\\_INFO](#), а на входе **szForbiddenRequests** указывается размер этой структуры или массива в байтах. Каждый экземпляр структуры описывает один (или несколько – при использовании специальных заполнителей, см. [п. 5.4.5](#)) запрещенный запрос. В случае запрещенного запроса slave отправляет master-устройству ответ с кодом ошибки (см. подробнее в [п. 5.4.5](#)). В случае получения запрещенного запроса с функцией записи полученные значения игнорируются.

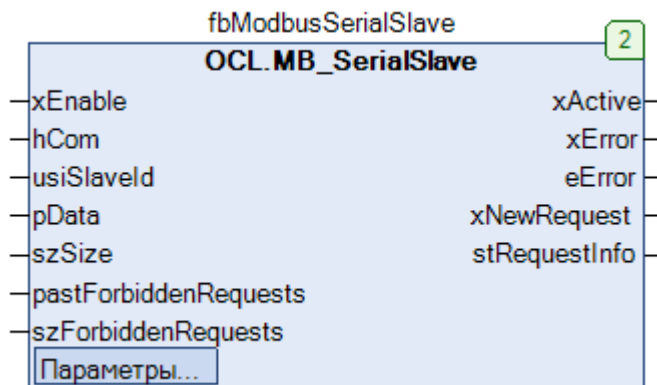


Рисунок 5.6.2 – Внешний вид ФБ MB\_SerialSlave на языке CFC

Таблица 5.6.2 – Описание входов и выходов ФБ MB\_SerialSlave

Название	Тип	Описание
<b>Входы</b>		
xEnable	BOOL	По переднему фронту происходит включение slave'a, по заднему – отключение
hCom	CAA.HANDLE	Дескриптор COM-порта, полученный от ФБ <a href="#">COM_Control</a>
usiSlaveId	USINT	Адрес slave-устройства. Если указан адрес <b>255</b> , то slave отвечает на запросы с любым адресом (это может быть удобным при отладке для имитации нескольких slave-устройств)
pData	CAA.PVOID	Указатель на буфер данных slave'a
szSize	CAA.SIZE	Размер буфера в байтах
pastForbiddenRequest	CAA.PVOID	Указатель на массив структур запрещенных запросов
szForbiddenRequests	CAA.SIZE	Размер массива структур запрещенных запросов в байтах
<b>Выходы</b>		
xActive	BOOL	<b>TRUE</b> – ФБ находится в работе
xError	BOOL	Принимает значение <b>TRUE</b> в случае возникновения ошибки
eError	<a href="#">ERROR</a>	Статус работы ФБ (или код ошибки)
xNewRequest	BOOL	Принимает <b>TRUE</b> на один цикл контроллера при получении запроса от master-устройства
stRequestInfo	<a href="#">MB_REQ_INFO</a>	Информация о полученном запросе (актуальна, пока <b>xNewRequest = TRUE</b> )
<b>Параметры</b>		
c_xReverseByteOrder	BOOL	<b>TRUE</b> – изменить порядок байт в буфере данных slave'a на противоположный
c_uiStartAddr	UINT	Начальный адрес slave'a. При получении запроса, в котором присутствует регистр с адресом < <b>c_uiStartAddr</b> , slave отправит master-устройству ошибку <b>ILLEGAL_DATA_ADDRESS</b>

### 5.6.3 ФБ MB\_TcpRequest

Функциональный блок **MB\_TcpRequest** используется для работы в режиме **Modbus TCP Master**. По переднему фронту на входе **xExecute** происходит отправка запроса, определяемого параметрами **usiUnitId**, **eFuncCode**, **uiDataAddr** и **uiDataCount** по протоколу **Modbus TCP** (если **xIsRtuOverTcp = FALSE**) или **Modbus RTU over TCP** (если **xIsRtuOverTcp = TRUE**) через TCP-соединение, определяемое дескриптором **hConnection**, полученным от ФБ [TCP Client](#). Считываемые или записываемые данные размещаются в буфере, расположенном по указателю **pData** размером **szSize** байт.

Ответ от slave-устройства ожидается в течение времени **tTimeout**. В случае отсутствия ответа ФБ повторяет запрос. Число повторений определяется входом **usiRetry** (значение **0** соответствует отсутствию повторений). Если ни на один из запросов не был получен ответ, то выход **xError** принимает значение **TRUE**, а выход **eError = TIME\_OUT**. В случае получения корректного ответа выход **xDone** принимает значение **TRUE**, а выход **eError = NO\_ERROR**. В случае получения ответа с кодом ошибки Modbus **xError** принимает значение **TRUE**, а выход **eError** содержит код ошибки (при этом выход **xDone** не принимает значение **TRUE**). Для отправки следующего запроса следует создать передний фронт на входе **xExecute**.

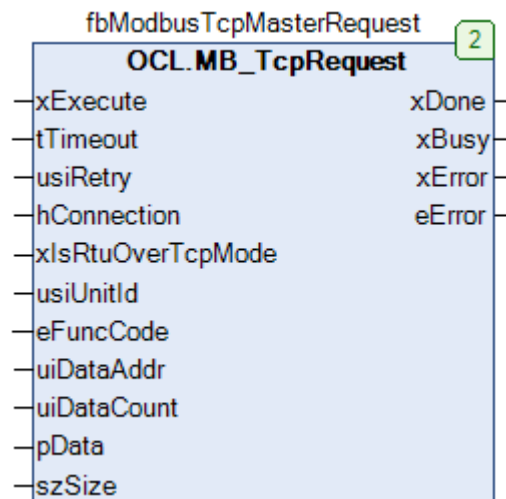


Рисунок 5.6.3 – Внешний вид ФБ MB\_TcpRequest на языке CFC

Таблица 5.6.3 – Описание входов и выходов ФБ MB\_TcpRequest

Название	Тип	Описание
<b>Входы</b>		
xExecute	BOOL	По переднему фронту происходит однократная (с возможностью переповторов в случае отсутствия ответа) отправка запроса
tTimeout	TIME	Таймаут ожидания ответа от slave-устройства ( <b>T#0ms</b> – время ожидания не ограничено)
usiRetry	USINT	Число переповторов в случае отсутствия ответа
hConnection	CAA.HANDLE	Дескриптор соединения, полученный от ФБ <a href="#">TCP_Client</a>
xIsRtuOverTcp	BOOL	Используемый протокол: <b>FALSE</b> – Modbus TCP, <b>TRUE</b> – Modbus RTU over TCP
usiUnitId	USINT	Адрес slave-устройства (значение по умолчанию – <b>16#FF</b> , другие значения требуются только при работе со шлюзами Modbus TCP/Modbus Serial и специфичными slave-устройствами)
eFuncCode	<a href="#">MB_FC</a>	Используемая функция Modbus
uiDataAddr	UINT	Начальный адрес бита/регистра в запросе
uiDataCount	UINT	Число битов/регистров в запросе
pData	CAA.PVOID	Указатель на буфер записываемых или считываемых данных
szSize	CAA.SIZE	Размер буфера в байтах
<b>Выходы</b>		
xDone	BOOL	<b>TRUE</b> – получен корректный ответ от slave-устройства
xBusy	BOOL	<b>TRUE</b> – ФБ находится в работе
xError	BOOL	Принимает значение <b>TRUE</b> в случае возникновения ошибки
eError	<a href="#">ERROR</a>	Статус работы ФБ (или код ошибки)

#### 5.6.4 ФБ MB\_TcpSlave

Функциональный блок **MB\_TcpSlave** используется для работы в режиме **Modbus TCP Slave**. Пока вход **xEnable** имеет значение **TRUE**, блок находится в работе. На входе **slpAddr** следует указать IP-адрес используемого сетевого интерфейса контроллера, на входе **uiPort** – используемый порт.

Вход **usiUnitId** определяет адрес slave'a (slave также отвечает на запросы с Unit ID = 255). Под регистры slave'a выделяется область памяти по указателю **pData** размером **szSize** байт.

Блок поддерживает все стандартные функции Modbus, приведенные в [таблице 2.2](#), а также функцию **20 (Read File Record)**. Для функции **20** поддерживается доступ к 8 файлам, нумерация файлов ведется с 1.

Блок поддерживает до 16 одновременно подключенных клиентов. Максимально допустимое число клиентов определяется глобальным параметром библиотеки **g\_c\_usiMaxCountClients** (вкладка **Settings** в дереве библиотеки в **Менеджере библиотек**). Значение по умолчанию – 1.

В реализации блока все области памяти Modbus наложены друг на друга и имеют общую адресацию (**MODBUS Data Model with only 1 block** согласно спецификации Modbus). Максимальный размер буфера slave'a – **65536** регистров.

При получении запроса от master-устройства выход **xNewRequest** на один цикл контроллера принимает значение **TRUE**, при этом выход **stRequestInfo** содержит информацию о полученном запросе. Если контроллер одновременно опрашивается несколькими master-устройствами, то будет отображена информация о последнем полученном в цикле запросе.

Блок позволяет запретить обработку определенных запросов, полученных от master-устройства. Для этого на вход **pastForbiddenRequest** передается указатель на структуру (или массив структур) типа [MB\\_REQ\\_INFO](#), а на входе **szForbiddenRequests** указывается размер этой структуры или массива в байтах. Каждый экземпляр структуры описывает один (или несколько – при использовании специальных заполнителей, см. [п. 5.4.5](#)) запрещенный запрос. В случае запрещенного запроса slave отправляет master-устройству ответ с кодом ошибки (см. подробнее в [п. 5.4.5](#)). В случае получения запрещенного запроса с функцией записи полученные значения игнорируются.

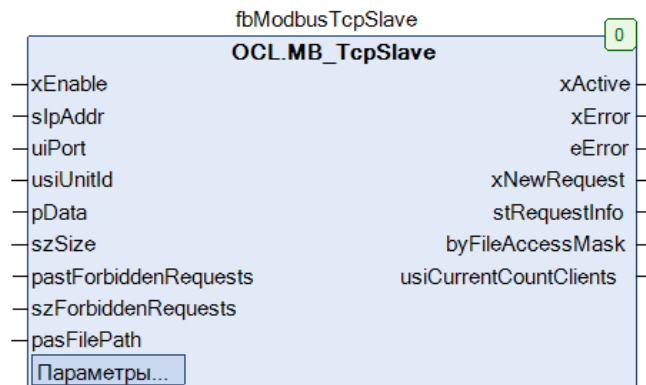


Рисунок 5.6.4 – Внешний вид ФБ MB\_TcpSlave на языке CFC

Таблица 5.6.4 – Описание входов и выходов ФБ MB\_TcpSlave

Название	Тип	Описание
<b>Входы</b>		
xEnable	BOOL	По переднему фронту происходит включение slave'a, по заднему – отключение
slpAddr	STRING	IP-адрес используемого сетевого интерфейса в формате <a href="#">IPv4</a> ('xxx.xxx.xxx.xxx').
uiPort	UINT	Используемый порт
usiUnitId	USINT	Адрес slave-устройства
pData	CAA.PVOID	Указатель на буфер данных slave'a
szSize	CAA.SIZE	Размер буфера в байтах
pastForbiddenRequest	CAA.PVOID	Указатель на массив структур запрещенных запросов
szForbiddenRequests	CAA.SIZE	Размер массива структур запрещенных запросов в байтах
pasFilePath	POINTER TO ARRAY [1..8] OF STRING	Указатель на массив путей к файлам архива (для функции <b>20</b> )
<b>Выходы</b>		
xActive	BOOL	<b>TRUE</b> – ФБ находится в работе
xError	BOOL	Принимает значение <b>TRUE</b> в случае возникновения ошибки
eError	<a href="#">ERROR</a>	Статус работы ФБ (или код ошибки)
xNewRequest	BOOL	Принимает <b>TRUE</b> на один цикл контроллера при получении запроса от master-устройства
stRequestInfo	<a href="#">MB_REQ_INFO</a>	Информация о полученном запросе (актуальна, пока <b>xNewRequest = TRUE</b> )
byFileAccessMask	BYTE	Битовая маска открытых файлов (для функции <b>20</b> )
usiCurrentCountClients	USINT	Число клиентов, подключенных к slave'у. Максимальное число клиентов определяется глобальным параметром библиотеки <b>g_c_usiMaxCountClients</b> (вкладка <b>Settings</b> в дереве библиотеки в <b>Менеджере библиотек</b> ). Максимальное число клиентов – <b>16</b>
<b>Параметры</b>		
c_xReverseByteOrder	BOOL	<b>TRUE</b> – изменить порядок байт в буфере данных slave'a на противоположный
c_xReverseByteOrderFiles	BOOL	<b>TRUE</b> – изменить порядок байт при передаче файлов для функции <b>20</b> на противоположный. Для работы с <b>MasterOPC Universal Modbus Server</b> следует установить значение <b>TRUE</b>
c_uiStartAddr	UINT	Начальный адрес slave'a. При получении запроса, в котором присутствует регистр с адресом < <b>c_uiStartAddr</b> , slave отправит master-устройству ошибку <b>ILLEGAL_DATA_ADDRESS</b>
c_usiAmountBytes	USINT	Размер записи для функции <b>20</b> . По спецификации Modbus этот параметр должен иметь значение <b>2</b> . Для работы с <b>MasterOPC Universal Modbus Server</b> следует установить значение <b>10</b>

## 5.7 ФБ нестандартных протоколов

### 5.7.1 ФБ UNM\_SerialRequest

Функциональный блок **UNM\_SerialRequest** используется для реализации нестандартного протокола при обмене через COM-порт. По переднему фронту на входе **xExecute** происходит отправка содержимого буфера запроса, расположенного по указателю **pRequest**, размером **szRequest** байт через COM-порт, определяемый дескриптором **hCom**, полученным от ФБ [COM\\_Control](#). Ответ от slave-устройства ожидается в течение времени **tTimeout**. При получении ответа происходит его проверка на основании значений входов **szExpectedSize** и **wStopChar**:

- если **szExpectedSize <> 0**, то ответ считается корректным, если его размер в байтах = **szExpectedSize**;
- если **szExpectedSize = 0** и **wStopChar <> 16#0000**, то последние один (при **wStopChar = 16#00xx**) или два (при **wStopChar = 16#xxxx**) байта ответа (где **x** – произвольное значение) проверяются на равенство младшему или обоим байтам **wStopChar**. Это может использоваться при реализации строковых протоколов, в которых заранее известен стоп-символ;
- если **szExpectedSize = 0** и **wStopChar = 16#0000**, то любой полученный ответ считается корректным.

В случае получения корректного ответа выход **xDone** принимает значение **TRUE**, выход **eError = NO\_ERROR**, а на выходе **uiResponseSize** отображается размер ответа в байтах. Полученные данные помещаются в буфер, расположенный по указателю **pResponse** и имеющий размер **szResponse** байт.

В случае отсутствия ответа ФБ повторяет запрос. Число повторений определяется входом **usiRetry** (значение **0** соответствует отсутствию повторений). Если ни на один из запросов не был получен ответ, то выход **xError** принимает значение **TRUE**, а выход **eError = TIME\_OUT**.

Для отправки нового запроса следует создать передний фронт на входе **xExecute**.



#### ПРИМЕЧАНИЕ

В случае отправки запросов, для которых не подразумевается получение ответа, рекомендуется для входа **tTimeout** установить значение **T#1ms**.

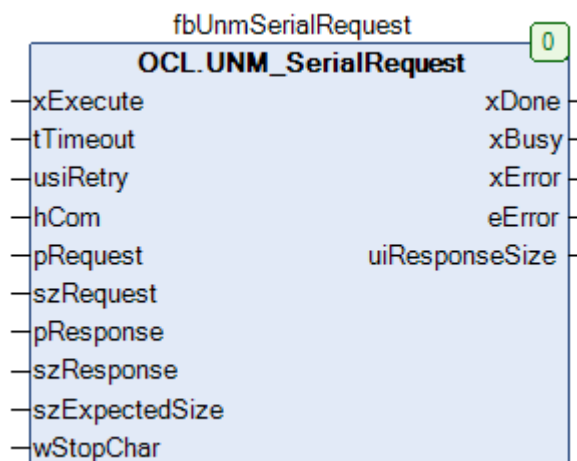


Рисунок 5.7.1 – Внешний вид ФБ UNM\_SerialRequest на языке CFC



Таблица 5.7.1 – Описание входов и выходов ФБ UNM\_SerialRequest

Название	Тип	Описание
<b>Входы</b>		
xExecute	BOOL	По переднему фронту происходит однократная (с возможностью повторений в случае отсутствия ответа) отправка запроса
tTimeout	TIME	Таймаут ожидания ответа от slave-устройства ( <b>T#0ms</b> – время ожидания не ограничено)
usiRetry	USINT	Число повторений в случае отсутствия ответа
hCom	CAA.HANDLE	Дескриптор COM-порта, полученный от ФБ <a href="#">COM_Control</a>
pRequest	CAA.PVOID	Указатель на буфер запроса
szRequest	CAA.SIZE	Размер буфера запроса в байтах
pResponse	CAA.PVOID	Указатель на буфер ответа
szResponse	CAA.SIZE	Размер буфера ответа в байтах
szExpectedSize	CAA.SIZE	Ожидаемый размер ответа в байтах ( <b>0</b> – размер неизвестен)
wStopChar	WORD	Стоп-символы протокола. Для протокола с двумя стоп-символами оба байта переменной должны быть отличны от нуля. Для протокола с одним стоп-символом старший байт должен быть равен нулю, а младший быть отличным от нуля. Если в протоколе отсутствуют стоп-символы, то следует установить значение <b>0</b>
<b>Выходы</b>		
xDone	BOOL	<b>TRUE</b> – получен корректный ответ от slave-устройства
xBusy	BOOL	<b>TRUE</b> – ФБ находится в работе
xError	BOOL	Принимает значение <b>TRUE</b> в случае возникновения ошибки
eError	<a href="#">ERROR</a>	Статус работы ФБ (или код ошибки)
uiResponseSize	UINT	Размер полученного ответа в байтах

### 5.7.2 ФБ UNM\_TcpRequest

Функциональный блок **UNM\_TcpRequest** используется для реализации нестандартного протокола поверх протокола **TCP**. По переднему фронту на входе **xExecute** происходит отправка содержимого буфера запроса, расположенного по указателю **pRequest**, размером **szRequest** байт через соединение, определяемое дескриптором **hConnection**, полученным от ФБ [TCP\\_Client](#). Ответ от slave-устройства ожидается в течение времени **tTimeout**. При получении ответа происходит его проверка на основании значений входов **szExpectedSize** и **wStopChar**:

- если **szExpectedSize**  $\neq 0$ , то ответ считается корректным, если его размер в байтах = **szExpectedSize**;
- если **szExpectedSize** = 0 и **wStopChar**  $\neq 16\#0000$ , то последние один (при **wStopChar** = **16\#00xx**) или два (при **wStopChar** = **16\#xxxx**) байта ответа (где x – произвольное значение) проверяются на равенство младшему или обоим байтам **wStopChar**. Это может использоваться при реализации строковых протоколов, в которых заранее известен стоп-символ;
- если **szExpectedSize** = 0 и **wStopChar** = **16\#0000**, то любой полученный ответ считается корректным.

В случае получения корректного ответа выход **xDone** принимает значение **TRUE**, выход **eError** = **NO\_ERROR**, а на выходе **uiResponseSize** отображается размер ответа в байтах. Полученные данные помещаются в буфер, расположенный по указателю **pResponse** и имеющий размер **szResponse** байт.

В случае отсутствия ответа ФБ повторяет запрос. Число повторений определяется входом **usiRetry** (значение 0 соответствует отсутствию повторений). Если ни на один из запросов не был получен ответ, то выход **xError** принимает значение **TRUE**, а выход **eError** = **TIME\_OUT**.

Для отправки нового запроса следует создать передний фронт на входе **xExecute**.



#### ПРИМЕЧАНИЕ

В случае отправки запросов, для которых не подразумевается получение ответа, рекомендуется для входа **tTimeout** установить значение **T#1ms**.

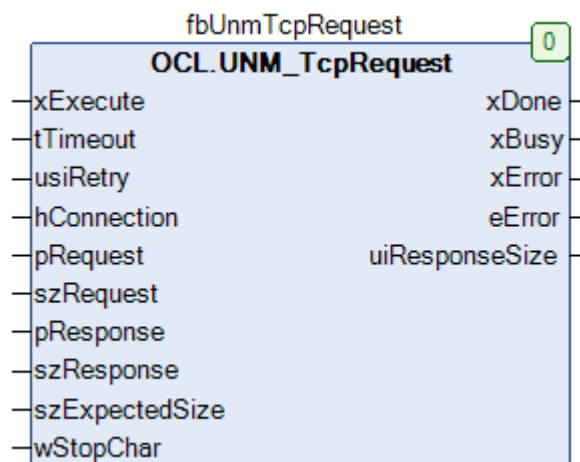


Рисунок 5.7.2 – Внешний вид ФБ UNM\_TcpRequest на языке CFC

Таблица 5.7.2 – Описание входов и выходов ФБ UNM\_TcpRequest

Название	Тип	Описание
<b>Входы</b>		
xExecute	BOOL	По переднему фронту происходит однократная (с возможностью повторений в случае отсутствия ответа) отправка запроса
tTimeout	TIME	Таймаут ожидания ответа от slave-устройства ( <b>T#0ms</b> – время ожидания не ограничено)
usiRetry	USINT	Число повторений в случае отсутствия ответа
hConnection	CAA.HANDLE	Дескриптор TCP-соединения, полученный от ФБ <a href="#">TCP_Client</a>
pRequest	CAA.PVOID	Указатель на буфер запроса
szRequest	CAA.SIZE	Размер буфера запроса в байтах
pResponse	CAA.PVOID	Указатель на буфер ответа
szResponse	CAA.SIZE	Размер буфера ответа в байтах
szExpectedSize	CAA.SIZE	Ожидаемый размер ответа в байтах ( <b>0</b> – размер неизвестен)
wStopChar	WORD	Стоп-символы протокола. Для протокола с двумя стоп-символами оба байта переменной должны быть отличны от нуля. Для протокола с одним стоп-символом старший байт должен быть равен нулю, а младший отличный от нуля. Если в протоколе отсутствуют стоп-символы, то следует установить значение 0
<b>Выходы</b>		
xDone	BOOL	<b>TRUE</b> – получен корректный ответ от slave-устройства
xBusy	BOOL	<b>TRUE</b> – ФБ находится в работе
xError	BOOL	Принимает значение <b>TRUE</b> в случае возникновения ошибки
eError	<a href="#">ERROR</a>	Статус работы ФБ (или код ошибки)
uiResponseSize	UINT	Размер полученного ответа в байтах

### 5.7.3 ФБ UNM\_UdpRequest

Функциональный блок **UNM\_UdpRequest** используется для реализации нестандартного протокола поверх протокола **UDP**. По переднему фронту на входе **xExecute** происходит отправка содержимого буфера запроса, расположенного по указателю **pRequest**, размером **szRequest** байт на IP-адрес **sServerIpAddr** и порт **uiServerPort**. На стороне контроллера для отправки используется порт **uiLocalPort** и IP-адрес **0.0.0.0**. (т. е. отправка запроса осуществляется по всем доступным интерфейсам).

Ответ от slave-устройства ожидается в течение времени **tTimeout**. При получении ответа происходит его проверка на основании значений входов **szExpectedSize** и **wStopChar**:

- если **szExpectedSize**  $\neq$  0, то ответ считается корректным, если его размер в байтах = **szExpectedSize**;
- если **szExpectedSize** = 0 и **wStopChar**  $\neq$  16#0000, то последние один (при **wStopChar** = 16#00xx) или два (при **wStopChar** = 16#xxxx) байта ответа (где x – произвольное значение) проверяются на равенство младшему или обоим байтам **wStopChar**. Это может использоваться при реализации строковых протоколов, в которых заранее известен стоп-символ;
- если **szExpectedSize** = 0 и **wStopChar** = 16#0000, то любой полученный ответ считается корректным.

В случае получения корректного ответа выход **xDone** принимает значение **TRUE**, выход **eError** = **NO\_ERROR**, а на выходе **uiResponseSize** отображается размер ответа в байтах. Полученные данные помещаются в буфер, расположенный по указателю **pResponse** и имеющий размер **szResponse** байт.

В случае отсутствия ответа ФБ повторяет запрос. Число повторений определяется входом **usiRetry** (значение 0 соответствует отсутствию повторений). Если ни на один из запросов не был получен ответ, то выход **xError** принимает значение **TRUE**, а выход **eError** = **TIME\_OUT**.

Для отправки нового запроса следует создать передний фронт на входе **xExecute**.



#### ПРИМЕЧАНИЕ

В случае отправки запросов, для которых не подразумевается получение ответа, рекомендуется для входа **tTimeout** установить значение **T#1ms**.

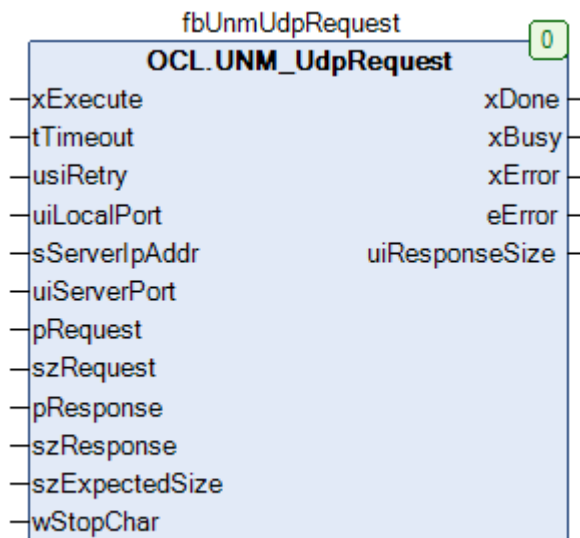


Рисунок 5.7.3 – Внешний вид ФБ UNM\_UdpRequest на языке CFC

Таблица 5.7.3 – Описание входов и выходов ФБ UNM\_UdpRequest

Название	Тип	Описание
<b>Входы</b>		
xExecute	BOOL	По переднему фронту происходит однократная (с возможностью переповторов в случае отсутствия ответа) отправка запроса
tTimeout	TIME	Таймаут ожидания ответа от slave-устройства ( <b>T#0ms</b> – время ожидания не ограничено)
usiRetry	USINT	Число переповторов в случае отсутствия ответа
uiLocalPort	UINT	Порт контроллера, через который отправляется запрос
sServerIpAddr	STRING	IP-адрес slave-устройства в формате <a href="#">IPv4</a> ('xxx.xxx.xxx.xxx')
uiServerPort	UINT	Порт slave-устройства
pRequest	CAA.PVOID	Указатель на буфер запроса
szRequest	CAA.SIZE	Размер буфера запроса в байтах
pResponse	CAA.PVOID	Указатель на буфер ответа
szResponse	CAA.SIZE	Размер буфера ответа в байтах
szExpectedSize	CAA.SIZE	Ожидаемый размер ответа в байтах ( <b>0</b> – размер неизвестен)
wStopChar	WORD	Стоп-символы протокола. Для протокола с двумя стоп-символами оба байта переменной должны быть отличны от нуля. Для протокола с одним стоп-символом старший байт должен быть равен нулю, а младший отличный от нуля. Если в протоколе отсутствуют стоп-символы, то следует установить значение 0
<b>Выходы</b>		
xDone	BOOL	<b>TRUE</b> – получен корректный ответ от slave-устройства
xBusy	BOOL	<b>TRUE</b> – ФБ находится в работе
xError	BOOL	Принимает значение <b>TRUE</b> в случае возникновения ошибки
eError	<a href="#">ERROR</a>	Статус работы ФБ (или код ошибки)
uiResponseSize	UINT	Размер полученного ответа в байтах

## 5.8 Функции и ФБ преобразования данных

### 5.8.1 ФБ DWORD\_TO\_WORD2

Функциональный блок **DWORD\_TO\_WORD2** используется для преобразования переменной типа **DWORD** в две переменные типа **WORD**.

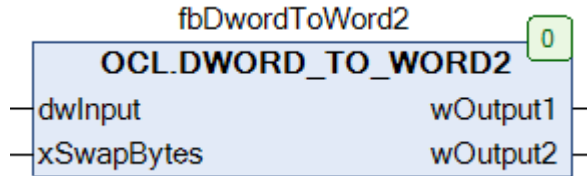


Рисунок 5.8.1 – Внешний вид ФБ DWORD\_TO\_WORD2 на языке CFC

Таблица 5.8.1 – Описание входов и выходов ФБ DWORD\_TO\_WORD2

Название	Тип	Описание
<b>Входы</b>		
dwInput	DWORD	Исходная переменная
xSwapBytes	BOOL	<b>TRUE</b> – выполнить перестановку байт (A1 B2 C3 D4 ----> B2 A1 D4 C3)
<b>Выходы</b>		
wOutput1	WORD	Старшее слово исходной переменной
wOutput2	WORD	Младшее слово исходной переменной

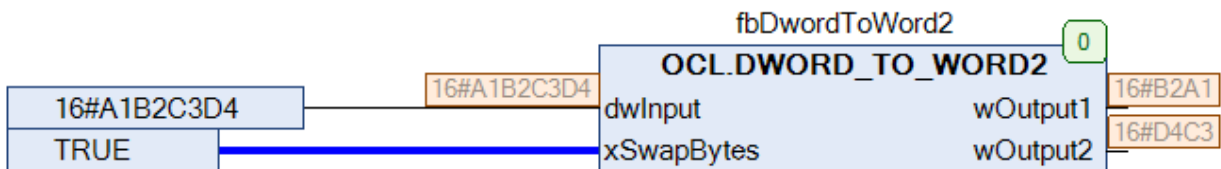


Рисунок 5.8.2 – Пример использования ФБ DWORD\_TO\_WORD2 на языке CFC

## 5.8.2 ФБ REAL\_TO\_WORD2

Функциональный блок **REAL\_TO\_WORD2** используется для преобразования переменной типа **REAL** в две переменные типа **WORD**.

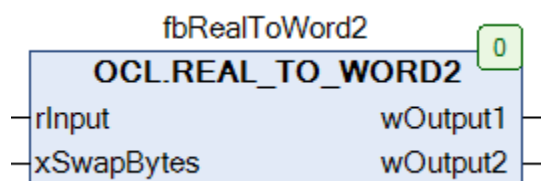


Рисунок 5.8.3 – Внешний вид ФБ REAL\_TO\_WORD2 на языке CFC

Таблица 5.8.2 – Описание входов и выходов ФБ REAL\_TO\_WORD2

Название	Тип	Описание
<b>Входы</b>		
rInput	REAL	Исходная переменная
xSwapBytes	BOOL	<b>TRUE</b> – выполнить перестановку байт (A1 B2 C3 D4 ---> B2 A1 D4 C3)
<b>Выходы</b>		
wOutput1	WORD	Старшее слово исходной переменной
wOutput2	WORD	Младшее слово исходной переменной

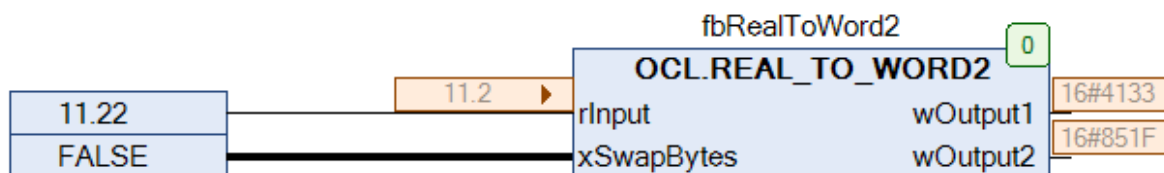


Рисунок 5.8.4 – Пример использования ФБ REAL\_TO\_WORD2 на языке CFC (см. [онлайн-конвертер](#) для проверки)

### 5.8.3 Функция WORD2\_TO\_DWORD

Функция **WORD2\_TO\_DWORD** используется для преобразования двух переменных типа **WORD** в переменную типа **DWORD**.

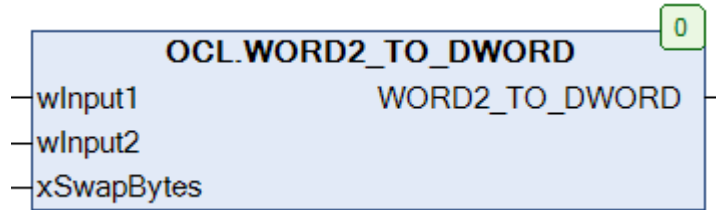


Рисунок 5.8.5 – Внешний вид функции WORD2\_TO\_DWORD на языке CFC

Таблица 5.8.3 – Описание входов и выходов функции WORD2\_TO\_DWORD

Название	Тип	Описание
<b>Входы</b>		
wInput1	WORD	Исходная переменная 1
wInput2	WORD	Исходная переменная 2
xSwapBytes	BOOL	<b>TRUE</b> – выполнить перестановку байт (A1 B2 C3 D4 ---> B2 A1 D4 C3)
<b>Выходы</b>		
WORD2_TO_DWORD	DWORD	Новая переменная типа <b>DWORD</b>

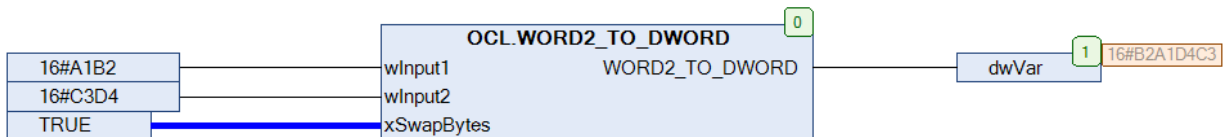


Рисунок 5.8.6 – Пример использования функции WORD2\_TO\_DWORD на языке CFC



#### 5.8.4 Функция WORD2\_TO\_REAL

Функция **WORD2\_TO\_REAL** используется для преобразования двух переменных типа **WORD** в переменную типа **REAL**.

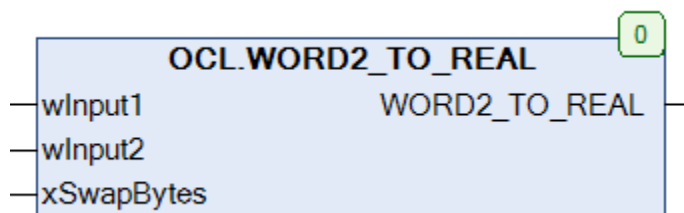


Рисунок 5.8.7 – Внешний вид функции WORD2\_TO\_REAL на языке CFC

Таблица 5.8.4 – Описание входов и выходов функции WORD2\_TO\_REAL

Название	Тип	Описание
<b>Входы</b>		
wInput1	WORD	Исходная переменная 1
wInput2	WORD	Исходная переменная 2
xSwapBytes	BOOL	<b>TRUE</b> – выполнить перестановку байт (A1 B2 C3 D4 ---> B2 A1 D4 C3)
<b>Выходы</b>		
WORD2_TO_REAL	REAL	Новая переменная типа <b>REAL</b>

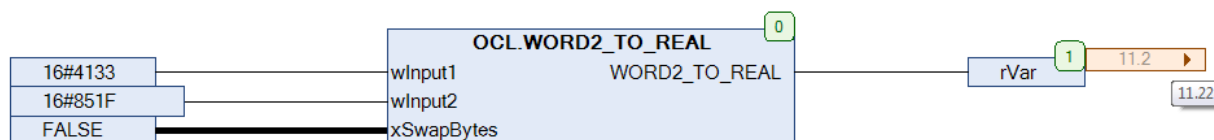


Рисунок 5.8.8 – Пример использования функции WORD2\_TO\_REAL на языке CFC (см. [онлайн-конвертер](#) для проверки)

### 5.8.5 Функция SWAP\_DATA

Функция **SWAP\_DATA** используется для копирования данных из одного буфера в другой с перестановкой байт и регистров.

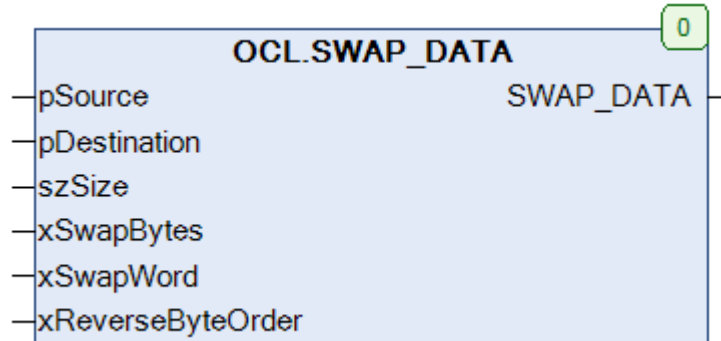


Рисунок 5.8.9 – Внешний вид функции SWAP\_DATA на языке CFC

Таблица 5.8.5 – Описание входов и выходов функции SWAP\_DATA

Название	Тип	Описание
<b>Входы</b>		
pSource	CAA.PVOID	Указатель на буфер исходных данных
pDestination	CAA.PVOID	Указатель на буфер, в который будут скопированы данные. Может совпадать с <b>pSource</b> – тогда после изменения данные будут помещены в тот же буфер
szSize	CAA.SIZE	Размер копируемых данных в байтах (должен не превышать размеры буферов)
xSwapBytes	BOOL	<b>TRUE</b> – выполнить перестановку байт (A1 B2 C3 D4 ---> B2 A1 D4 C3)
xSwapWord	BOOL	<b>TRUE</b> – выполнить перестановку регистров (A1 B2 C3 D4 ---> C3 D4 A1 B2)
xReverseByteOrder	BOOL	<b>TRUE</b> – изменить порядок байт на противоположный (A1 B2 C3 D4 ---> D4 C3 B2 A1). Если данный вход имеет значение <b>TRUE</b> , то входы <b>xSwapBytes</b> и <b>xSwapWord</b> не обрабатываются
<b>Выходы</b>		
SWAP_DATA	BOOL	<b>TRUE</b> – операция выполнена

## 5.9 Примеры

### 5.9.1 СПК1хх [M01] (Modbus RTU Master) + модули Mx110

В качестве примера будет рассмотрена настройка обмена с модулями [Mx110](#) (MB110-8A, MB110-16Д, МУ110-16P) с использованием библиотеки **OwenCommunication**. В примере используется библиотека версии **3.5.11.1**.

**Реализуемый алгоритм:** если значение первого аналогового входа модуля **MB110-8A** превышает **30** и при этом первый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **МУ110-16P** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).

Структурная схема примера приведена на рисунке ниже:

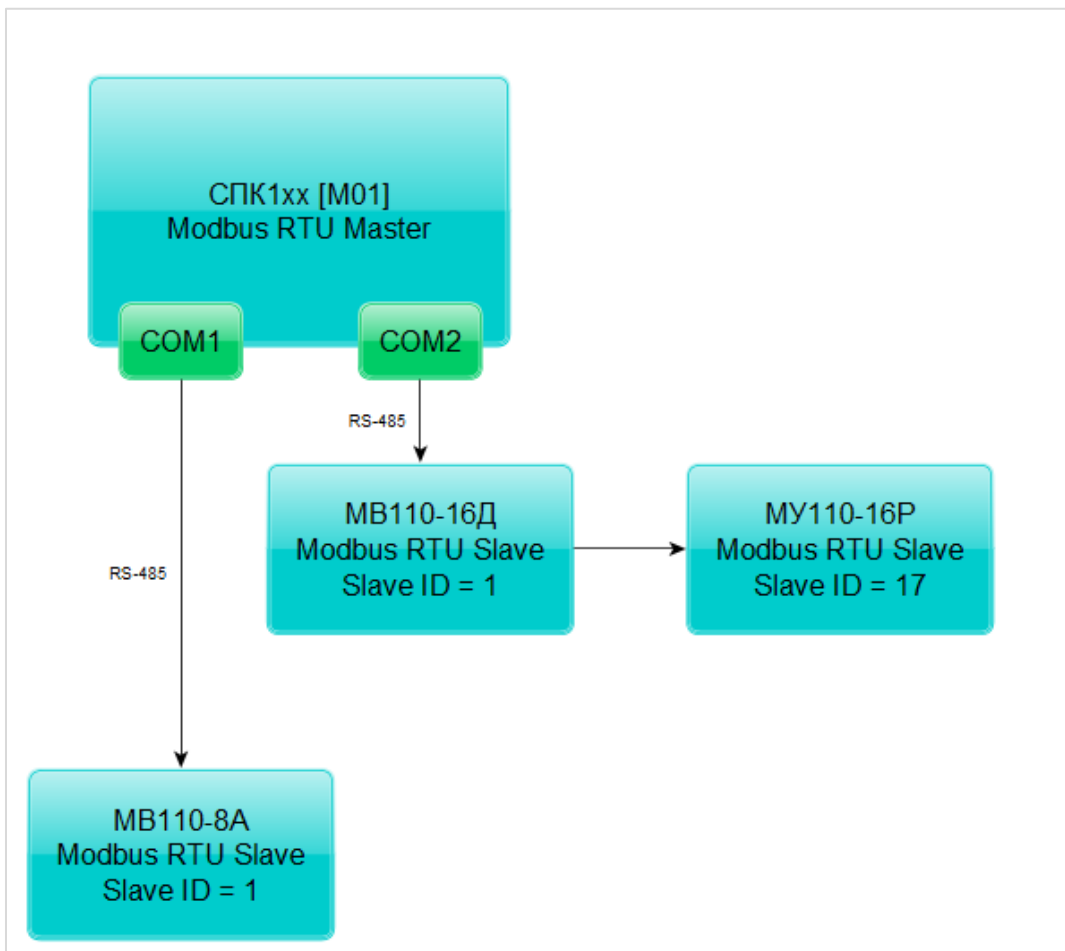


Рисунок 5.9.1 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (**ПКМ** на узел **Device – Обновить устройство**).

Пример доступен для скачивания:

[Example\\_OwenCommunicationModbusRtuMaster\\_3511v1.projectarchive](#)

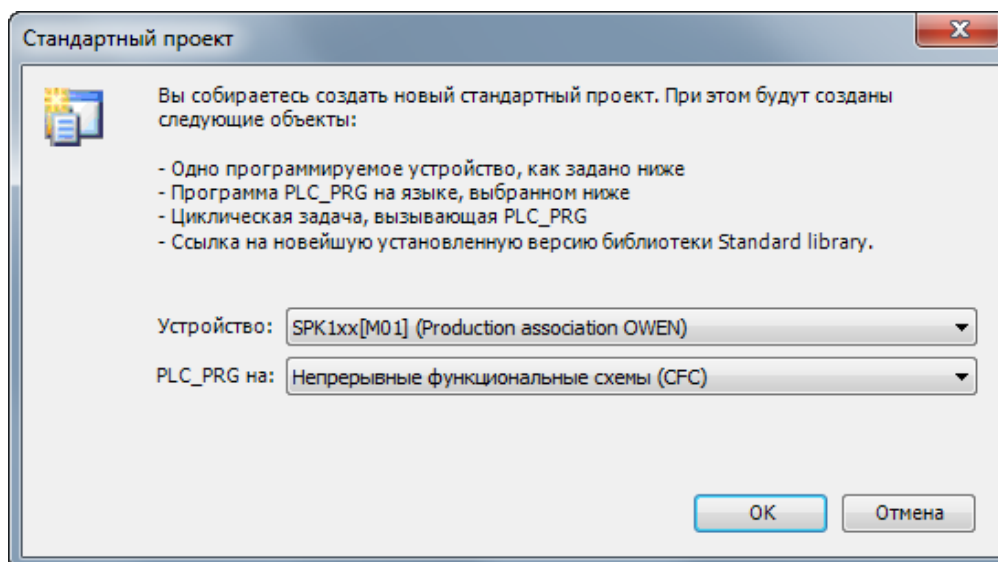
Сетевые параметры модулей приведены в таблице ниже:

**Таблица 5.9.1 – Сетевые параметры модулей Mx110**

Параметр	MB110-8A	MB110-16Д	МУ110-16Р
COM-порт контроллера, к которому подключен модуль	COM1	COM2	
ID COM-порта	1	2	
Адрес модуля	1	1	17
Скорость обмена	115200		
Количество бит данных	8		
Контроль четности	Отсутствует		
Количество стоп-бит	1		

Для настройки обмена следует:

1. Настроить модули **Mx110** с помощью программы **Конфигуратор Mx110** в соответствии с таблицей 5.9.1. Подключить модули к COM-портам контроллера в соответствии с [рисунком 5.9.1](#).
2. Установить в CODESYS библиотеку **OwenCommunication** (см. [п. 5.2](#)).
3. Создать новый проект **CODESYS** с программой на языке **ST** или **CFC**:



**Рисунок 5.9.2 – Создание проекта CODESYS**



**ПРИМЕЧАНИЕ**

Проект примера содержит программы на языках CFC и ST. По умолчанию используется программа на CFC (**PLC\_PRG\_CFC**). Для работы с программой на ST следует в конфигурации задач для задачи **MainTask** удалить вызов **PLC\_PRG\_CFC** и добавить вызов **PLC\_PRG\_ST**.

4. Добавить в проект библиотеки **OwenCommunication** и **Util** (**Менеджер библиотек – Добавить библиотеку**).



**ПРИМЕЧАНИЕ**

Библиотека **Util** используется только в программе на языке CFC.

5. Объявить в программе следующие переменные:

```

1  PROGRAM PLC_PRG_CFC
2  VAR
3      fbComControl1: OCL.COM_Control;      // ФБ управления портом COM1
4      fbComControl2: OCL.COM_Control;      // ФБ управления портом COM2
5      fbMV110_8A_AI1: OCL.MB_SerialRequest; // ФБ опроса модуля MB110-8A
6      fbMV110_16D_DI: OCL.MB_SerialRequest; // ФБ опроса модуля MB110-16Д
7      fbMU110_16R_DO: OCL.MB_SerialRequest; // ФБ опроса модуля MV110-16P
8
9      rAI1: REAL;                          // значение 1-го входа модуля MB110-8A
10     wDiMask: WORD;                        // битовая маска входов модуля MB110-16Д
11     wDoMask: WORD;                        // битовая маска выходов модуля MV110-16P
12     xDi0: BOOL;                          // значение 1-го входа модуля MB110-16Д
13     xDo0: BOOL;                          // значение 1-го выхода модуля MV110-16P
14
15     awAI1: ARRAY [0..1] OF WORD;         // регистры, считанные с модуля MB110-8A
16
17     fbUnpackWord: UTIL.WORD_AS_BIT;       // ФБ распаковки битовой маски
18     fbPackWord: UTIL.BIT_AS_WORD;        // ФБ упаковки битовой маски
19
20     iStateCom1: INT;                      // шаг опроса по порту COM1
21     iStateCom2: INT;                      // шаг опроса по порту COM2
22 END_VAR
    
```

Рисунок 5.9.3 – Объявление переменных в программе



**ПРИМЕЧАНИЕ**

Переменные **fbUnpackWord** и **fbPackWord** используются только в программе на языке CFC. Переменные **iStateCom1** и **iStateCom2** используется только в программе на языке ST.

6. Нажать ПКМ на программу, выбрать команду **Добавление объекта – Действие** и добавить действия с названиями **COM1** и **COM2** (язык реализации действий совпадает с языком программы). В рамках примера действия используются для повышения читабельности кода.

7. Код действий и программы на языке CFC будет выглядеть следующим образом:

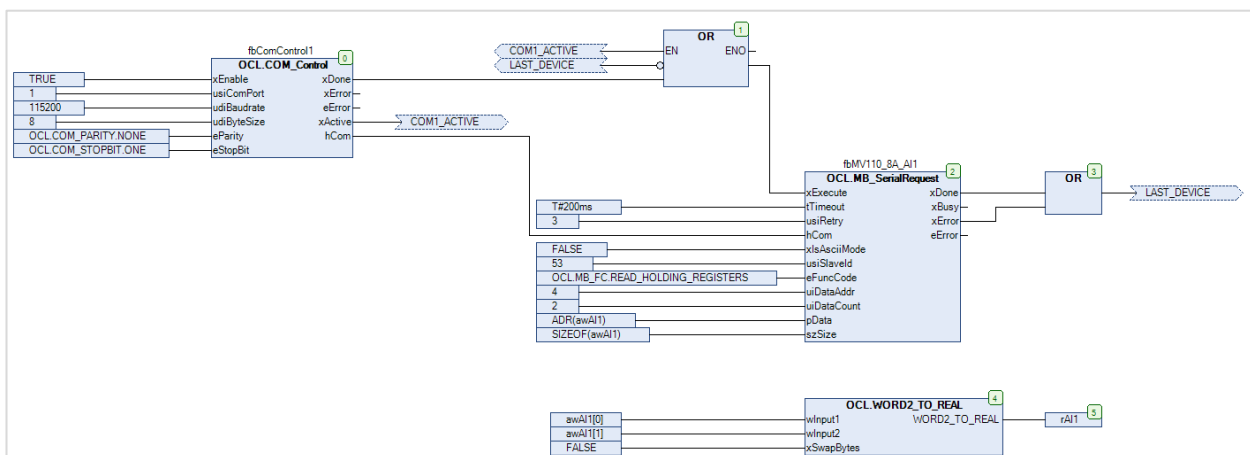


Рисунок 5.9.4 – Код действия COM1

При первом вызове действия происходит открытие COM-порта с заданными настройками с помощью экземпляра ФБ [COM Control](#) (блок 0). После успешного открытия порта на выходе **xDone** генерируется единственный импульс, что приводит к вызову экземпляра ФБ [MB SerialRequest](#) (блок 2), который производит опрос первого аналогового входа модуля **MB110-8A** с адресом **1 (usiSlaveId)**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Полученные данные помещаются в переменную **awAI1** типа **ARRAY [0..1] OF WORD**. С помощью функции [WORD2 TO REAL](#) (блоки 4–5) этот массив преобразуется в переменную типа **rAI1** типа **REAL**.

После завершения работы экземпляра ФБ [MB SerialRequest](#) один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**. В любой из этих ситуаций с помощью блока 1 происходит вызов экземпляра ФБ [MB SerialRequest](#) со значением **FALSE** на входе **xExecute** (сброс блока), а потом – его следующий вызов, что приводит к началу следующего сеанса опроса.

Если порт закрыт или находится в состоянии ошибки (выход **xActive** у экземпляра ФБ [COM CONTROL](#) имеет значение **FALSE**), то опрос прекращается.



#### ПРИМЕЧАНИЕ

В рамках примера для повышения читабельности схемы вместо некоторых линий связи использованы **метки соединения**.

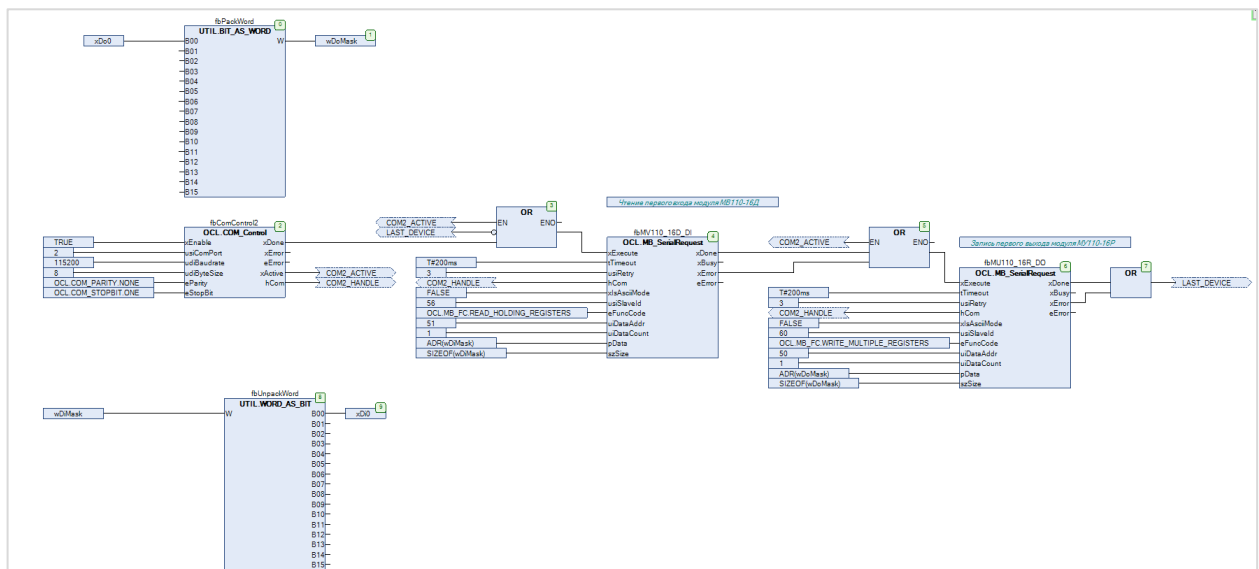


Рисунок 5.9.5 – Код действия COM2

При первом вызове действия происходит открытие COM-порта с заданными настройками с помощью ФБ [COM CONTROL](#) (блок 2). После успешного открытия порта на выходе **xDone** генерируется единственный импульс, что приводит к вызову ФБ [MB SerialRequest](#) (блок 4), который производит опрос битовой маски дискретных входов модуля **MB110-16Д** с адресом **1 (usiSlaveId)**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Полученные данные помещаются в переменную **wDIMask** типа **WORD** и разделяются на отдельные переменные типа **BOOL** с помощью экземпляра ФБ [WORD\\_AS\\_BIT](#) из библиотеки **Util**.

После завершения работы ФБ [MB SerialRequest](#) один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**. В любой из этих ситуаций начинается работа следующего блока [MB SerialRequest](#) (блоки 5–6), который производит запись переменной **wDoMask** в качестве битовой маски дискретных выходов модуля

**МУ110-16P** с адресом **17**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Экземпляр блока **BIT\_AS\_WORD** (блоки **0–1**) позволяет упаковать отдельные переменные типа **BOOL** в битовую маску типа **WORD**.

После окончания работы экземпляра последнего блока **MB SerialRequest** с помощью блока **3** происходит вызов экземпляра первого ФБ **MB SerialRequest** со значением **FALSE** на входе **xExecute** (сброс блока), а потом – его следующий вызов, что приводит к началу следующего сеанса опроса.

Если порт закрыт или находится в состоянии ошибки (выход **xActive** у ФБ **COM\_CONTROL** имеет значение **FALSE**), то опрос прекращается.



**ПРИМЕЧАНИЕ**

В рамках примера для повышения читабельности схемы вместо некоторых линий связи использованы **метки соединения**.

В программе **PLC\_PRG\_CFC** производится вызов действий **COM1** и **COM2**, а также выполнение алгоритма, описанного в условии примера.

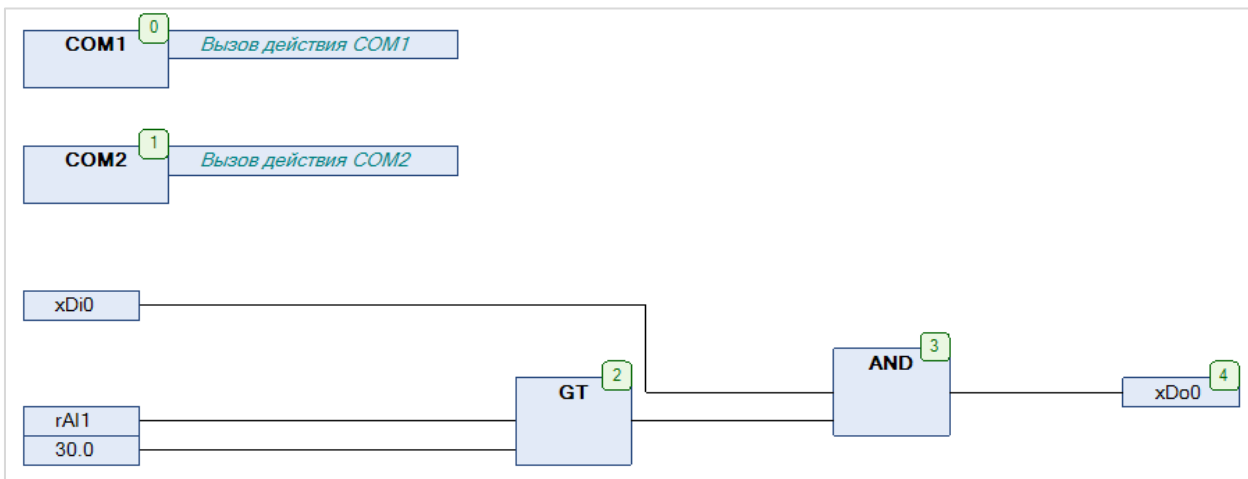


Рисунок 5.9.6 – Код программы PLC\_PRG\_CFC

## 8. Код действий и программы на языке ST будет выглядеть следующим образом:

```

1  CASE iStateCom1 OF
2
3      0: // открытие COM-порта COM1
4
5          fbComControl1
6          (
7              xEnable      := TRUE,
8              usiComPort   := 1,
9              udiBaudrate  := 115200,
10             udiByteSize  := 8,
11             eParity       := OCL.COM_PARITY.NONE,
12             eStopBit      := OCL.COM_STOPBIT.ONE
13         );
14
15         IF fbComControl1.xDone THEN
16             iStateCom1 := 1;
17         END_IF
18
19
20     1: // опрос MB110-8A
21
22         fbMV110_8A_AI1
23         (
24             xExecute      := fbComControl1.xActive,
25             tTimeout      := T#200MS,
26             usiRetry       := 3,
27             hCom          := fbComControl1.hCom,
28             xIsAsciiMode  := FALSE ,
29             usiSlaveId    := 1,
30             eFuncCode     := OCL.MB_FC.READ_HOLDING_REGISTERS,
31             uiDataAddr    := 4,
32             uiDataCount   := 2,
33             pData         := ADR(awAI1),
34             szSize        := SIZEOF(awAI1)
35         );
36
37         IF fbMV110_8A_AI1.xDone OR fbMV110_8A_AI1.xError THEN
38             // после выполнения блока его надо сбросить
39             fbMV110_8A_AI1(xExecute := FALSE);
40
41             rAI1 := OCL.WORD2_TO_REAL(awAI1[0], awAI1[1], FALSE);
42
43             iStateCom1 := 2;
44         END_IF
45
46
47     2: // здесь можно добавить опрос следующего устройства
48         // после опроса последнего опроса возвращаемся к опросу первого
49         iStateCom1 := 1;
50
51 END_CASE

```

Рисунок 5.9.7 – Код действия COM1

При первом вызове действия в **шаге 0** происходит открытие COM-порта с заданными настройками с помощью экземпляра ФБ [COM Control](#). После успешного открытия порта на выходе **xDone** генерируется единичный импульс, что приводит к переходу на **шаг 1**.

На **шаге 1** выполняется вызов экземпляра ФБ [MB SerialRequest](#), который производит опрос первого аналогового входа модуля **MB110-8A** с адресом **1 (usiSlaveId)**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Полученные данные помещаются в переменную **awAI1** типа **ARRAY [0..1] OF WORD**. С помощью функции [WORD2 TO REAL](#) этот массив преобразуется в переменную типа **rAI1** типа **REAL**.



После завершения работы экземпляра ФБ [MB\\_SerialRequest](#) один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**. В любой из этих ситуаций происходит вызов экземпляра ФБ [MB\\_SerialRequest](#) со значением **FALSE** на входе **xExecute** (сброс блока) и переход на **шаг 2**.

В рамках примера на **шаге 2** происходит переход на **шаг 1**, что приводит к началу следующего сеанса опроса. В случае опроса нескольких устройств (или одного устройства с помощью нескольких запросов) число шагов можно увеличить. После выполнения последнего шага должен происходить переход на шаг опроса первого устройства.

Если порт закрыт или находится в состоянии ошибки (выход **xActive** у экземпляра ФБ [COM\\_Control](#) имеет значение **FALSE**), то опрос прекращается.

```

1 CASE iStateCom2 OF
2
3     0: // открытие COM-порта COM2
4
5         fbComControl2
6         (
7             xEnable      := TRUE,
8             usiComPort   := 2,
9             udiBaudrate  := 115200,
10            udiByteSize  := 8,
11            eParity      := OCL.COM_PARITY.NONE,
12            eStopBit     := OCL.COM_STOPBIT.ONE
13        );
14
15        IF fbComControl2.xDone THEN
16            iStateCom2 := 1;
17        END_IF
18
19     1: // опрос модуля MB110-16Д
20
21        fbMV110_16D_DI
22        (
23            xExecute      := fbComControl2.xActive,
24            tTimeout      := T#200MS,
25            usiRetry      := 3,
26            hCom          := fbComControl2.hCom,
27            xIsAsciiMode  := FALSE ,
28            usiSlaveId    := 1,
29            eFuncCode     := OCL.MB_FC.READ_HOLDING_REGISTERS,
30            uiDataAddr    := 51,
31            uiDataCount   := 1,
32            pData         := ADR(wDiMask),
33            szSize        := SIZEOF(wDiMask)
34        );
35
36        IF fbMV110_16D_DI.xDone OR fbMV110_16D_DI.xError THEN
37            // после выполнения блока его надо сбросить
38            fbMV110_16D_DI(xExecute := FALSE);
39
40            xDi0 := wDiMask.0;
41
42            iStateCom2 := 2;
43        END_IF
44
45     2: // опрос модуля MV110-16P
46
47        wDoMask.0 := xDo0;
48
49
50        fbMU110_16R_DO
51        (
52            xExecute      := fbComControl2.xActive,
53            tTimeout      := T#200MS,
54            usiRetry      := 17,
55            hCom          := fbComControl2.hCom,
56            xIsAsciiMode  := FALSE ,
57            usiSlaveId    := 60,
58            eFuncCode     := OCL.MB_FC.WRITE_MULTIPLE_REGISTERS,
59            uiDataAddr    := 50,
60            uiDataCount   := 1,
61            pData         := ADR(wDoMask),
62            szSize        := SIZEOF(wDoMask)
63        );
64
65        IF fbMU110_16R_DO.xDone OR fbMU110_16R_DO.xError THEN
66            // после выполнения блока его надо сбросить
67            fbMU110_16R_DO(xExecute := FALSE);
68
69            // возвращаемся к опросу первого модуля
70            iStateCom2 := 1;
71        END_IF
72

```

Рисунок 5.9.8 – Код действия COM2

При первом вызове действия в **шаре 0** происходит открытие COM-порта с заданными настройками с помощью экземпляра ФБ [COM Control](#). После успешного открытия порта на выходе **xDone** генерируется единичный импульс, что приводит к переходу на **шаг 1**.

На **шаре 1** выполняется вызов экземпляра ФБ [MB SerialRequest](#), который производит опрос битовой маски дискретных входов модуля **MB110-16Д** с адресом **1 (usiSlaveId)**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Полученные данные помещаются в переменную **wDiMask** типа **WORD**, из которой происходит копирование данных в нужные переменные типа **BOOL** с помощью побитового доступа (*переменная.номер\_бита*).

После завершения работы экземпляра ФБ [MB SerialRequest](#) один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**. В любой из этих ситуаций происходит вызов экземпляра ФБ [MB SerialRequest](#) со значением **FALSE** на входе **xExecute** (сброс блока) и переход на **шаг 2**.

На **шаре 2** выполняется вызов экземпляра ФБ [MB SerialRequest](#), который производит запись переменной **wDoMask** в качестве битовой маски дискретных выходов модуля **MY110-16P** с адресом **17**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Битовая маска может формироваться из отдельных переменных типа **BOOL** с помощью побитового доступа.

В рамках примера на **шаре 2** происходит переход на **шаг 1**, что приводит к началу следующего сеанса опроса. В случае опроса нескольких устройств (или одного устройства с помощью нескольких запросов) число шагов можно увеличить. После выполнения последнего шага должен происходить переход на шаг опроса первого устройства.

Если порт закрыт или находится в состоянии ошибки (выход **xActive** у экземпляра ФБ [COM Control](#) имеет значение **FALSE**), то опрос прекращается.

В программе **PLC\_PRG\_ST** производится вызов действий **COM1** и **COM2**, а также выполнение алгоритма, описанного в условии примера.

```
1 // чтобы запустить пример на ST на контроллере требуется:
2 // 1. Удалить из задачи MainTask вызов программы PLC_PRG_CFC
3 // 2. Привязать к задаче MainTask вызов программа PLC_PRG_ST
4
5 COM1 ();
6 COM2 ();
7
8 xDo0 := xDi0 AND (rAI1 > 30.0);
```

Рисунок 5.9.9 – Код программы PLC\_PRG\_ST

9. Загрузить проект в контроллер и запустить его.

В переменной **rAI1** будет отображаться текущее значение первого аналогового входа модуля **MB110-8A**. В переменной **xDi0** будет отображаться текущее значение первого дискретного входа модуля **MB110-16Д**.

Если значение **rAI1** превысит **30** и при этом значение **xDi0** будет равно **TRUE**, то в переменную **xDo0** будет записано значение **TRUE**, что приведет к замыканию первого дискретного выхода модуля **МУ110-16Р**. Если одно из условий перестанет выполняться, то выход будет разомкнут.

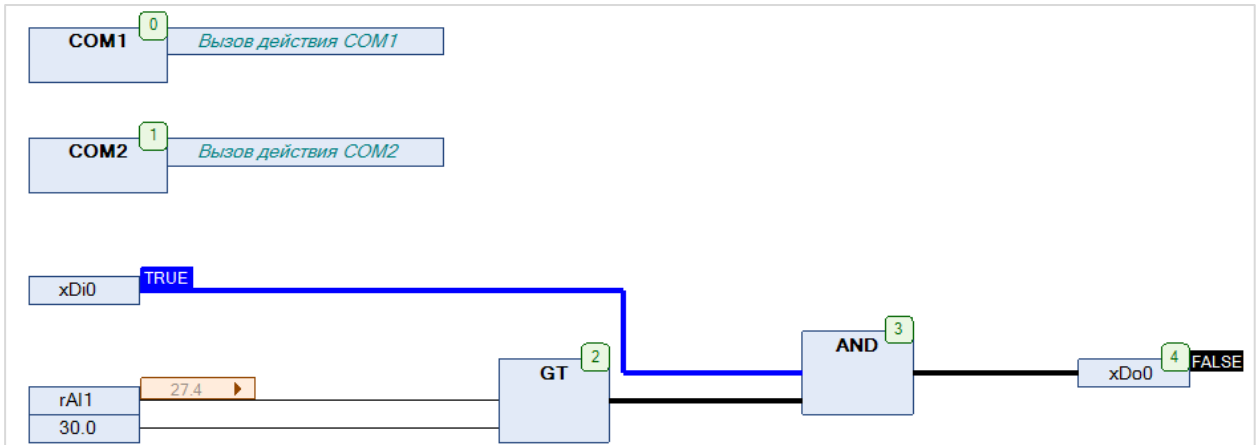


Рисунок 5.9.10 – Выполнение программы в режиме Online

### 5.9.2 СПК1хх [M01] (Modbus RTU Slave) + MasterOPC Universal Modbus Server

В качестве примера будет рассмотрена настройка обмена с OPC-сервером [Insat MasterOPC Universal Modbus Server](#), который будет использоваться в режиме **Modbus RTU Master**, с помощью библиотеки **OwenCommunication**. В примере используется библиотека версии **3.5.11.1**.

Структурная схема примера приведена на рисунке ниже:

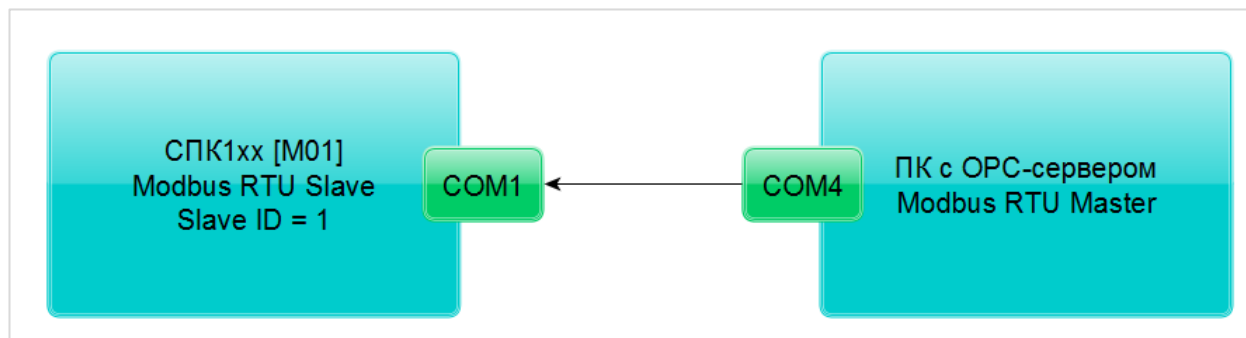


Рисунок 5.9.11 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (**ПКМ** на узел **Device** – **Обновить устройство**).

Пример доступен для скачивания: [Example OwenCommunicationModbusRtuSlave\\_3511v1.zip](#)

Сетевые параметры модулей приведены в таблице ниже:

Таблица 5.9.2 – Сетевые параметры устройств

Параметр	СПК1хх [M01]	ПК с OPC-сервером
COM-порт контроллера, к которому подключен модуль	COM1	COM4
ID COM-порта	1	4
Режим работы	slave	master
Slave ID	1	-
Скорость обмена	115200	
Количество бит данных	8	
Контроль четности	Отсутствует	
Количество стоп-бит	1	

Переменные примера описаны в таблице ниже:

Таблица 5.9.3 – Список переменных примера

Имя	Тип	Область памяти Modbus	Адрес регистра/бита
xVar_Opc	BOOL	Coils	0/0
wVar_Opc	WORD	Holding регистры	1
rVar_Opc	REAL		2–3
sVar_Opc	STRING(15)		4–11



#### ПРИМЕЧАНИЕ

В рамках примера значения переменных slave'a могут быть изменены как из OPC, так и из программы контроллера (с помощью переменных с постфиксом **\_Pic**).

Для настройки обмена следует:

1. Подключить контроллер к ПК (например, с помощью конвертера [ОВЕН АС4](#)).
2. Создать новый проект **CODESYS** с программой на языке **ST** или **CFC**:

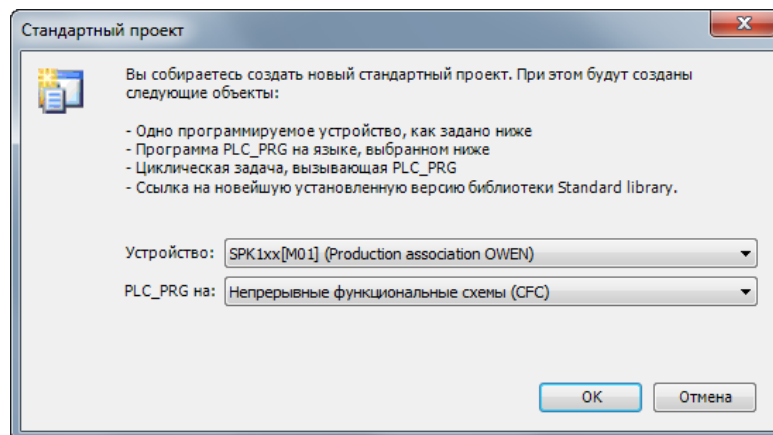


Рисунок 5.9.12 – Создание проекта CODESYS



#### ПРИМЕЧАНИЕ

Проект примера содержит программы на языках CFC и ST. По умолчанию используется программа на CFC (**PLC\_PRG\_CFC**). Для работы с программой на ST следует в конфигурации задач для задачи **MainTask** удалить вызов **PLC\_PRG\_CFC** и добавить вызов **PLC\_PRG\_ST**.

3. Добавить в проект библиотеки **OwenCommunication** и **Util** (**Менеджер библиотек – Добавить библиотеку**).



#### ПРИМЕЧАНИЕ

Библиотека **Util** используется только в программе на языке CFC.

4. Объявить в программе следующие переменные:

```

1  PROGRAM PLC_PRG_CFC
2  VAR
3      fbComControl1:      OCL.COM_Control;      // ФБ управления портом COM1
4
5      fbUnpackWord:       UTIL.WORD_AS_BIT;      // ФБ распаковки битовой маски
6      fbPackWord:         UTIL.BIT_AS_WORD;      // ФБ упаковки битовой маски
7      fbRealToWorld2:     OCL.REAL_TO_WORD2;    // ФБ преобразования REAL в две переменные типа WORD
8
9
10     fbModbusSerialSlave: OCL.MB_SerialSlave;   // ФБ для реализации Modbus Slave
11
12     awSlaveData:         ARRAY [0..15] OF WORD; // буфер данных Modbus Slave
13
14     (* значения, полученные от OPC *)
15     xVar_Opc:            BOOL;
16     wVar_Opc:            WORD;
17     rVar_Opc:            REAL;
18     sVar_Opc:            STRING(15);
19
20     (* значения для передачи в OPC *)
21     xVar_Plc:            BOOL;
22     wVar_Plc:            WORD;
23     rVar_Plc:            REAL;
24     sVar_Plc:            STRING(15);
25
26     xWrite:              BOOL;                  // команда записи данных из программы в регистры Modbus Slave
27     fbWriteEdge:         R_TRIG;              // триггер для однократной записи
28 END VAR

```

Рисунок 5.9.13 – Объявление переменных программы



#### ПРИМЕЧАНИЕ

Переменные **fbUnpackWord** и **fbPackWord** используются только в программе на языке CFC.

5. Код программы на языке CFC будет выглядеть следующим образом:

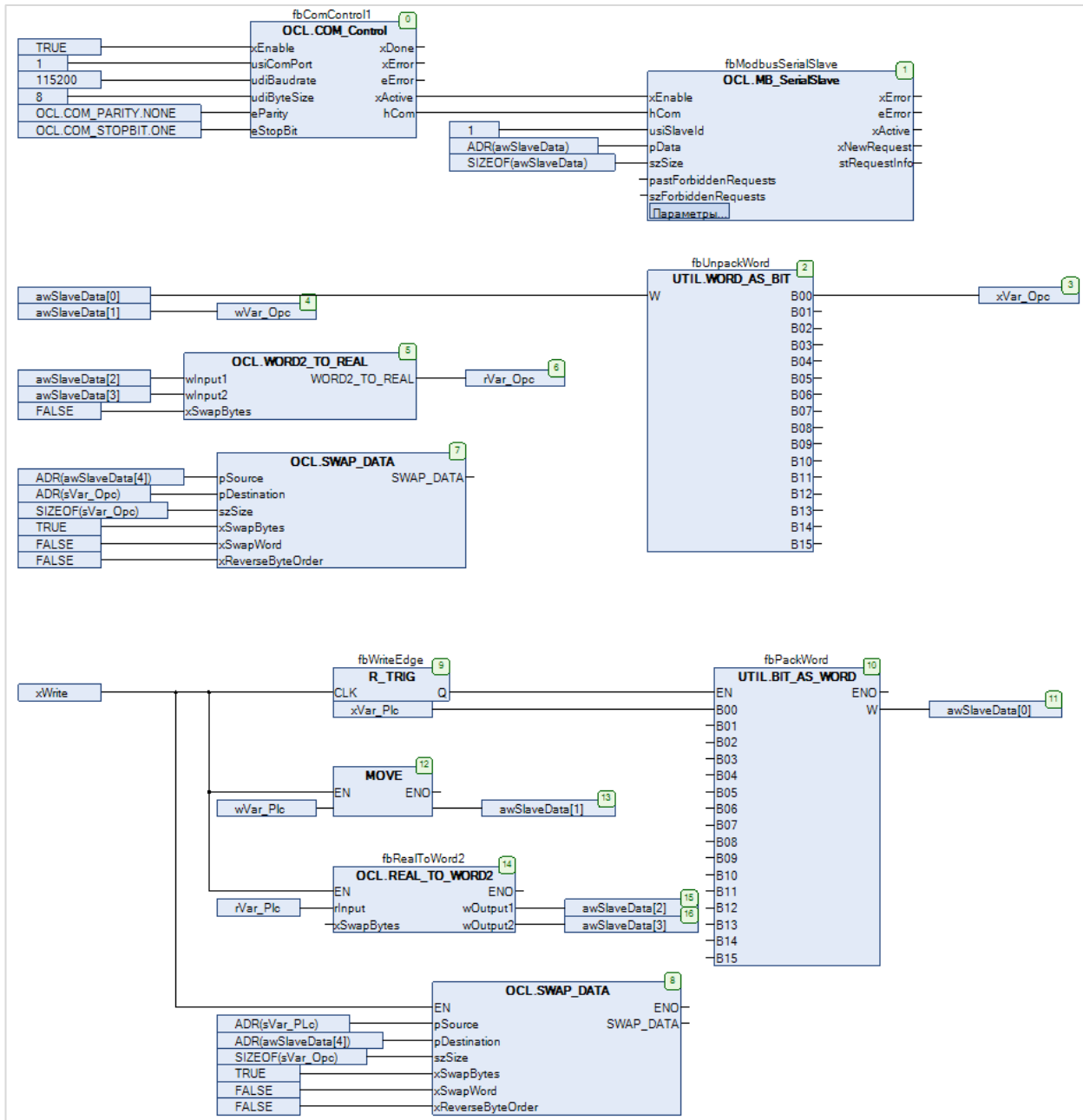


Рисунок 5.9.14 – Код программы на языке CFC

При первом вызове программы происходит открытие COM-порта с заданными настройками с помощью экземпляра ФБ [COM Control](#) (блок 0). После успешного открытия порта выход **xActive** принимает и сохраняет значение **TRUE**, что приводит к вызову экземпляра ФБ [MB SerialSlave](#) (блок 1), который выполняет функцию Modbus Slave с адресом 1 (**usiSlaveId**). Данные slave'a хранятся в массиве **awSlaveData**. В экземпляр ФБ передается указатель на этот массив (**pData**) и его размер в байтах (**szSize**).

В блоках 2–7 происходит копирование данных из регистров slave'a в переменные программы. Для выделения переменных типа **BOOL** из переменной типа **WORD** используется экземпляр ФБ [WORD\\_AS\\_BIT](#) из библиотеки [Util](#). Для преобразования двух переменных типа **WORD** в переменную типа **REAL** используется функция [WORD2 TO REAL](#). Для преобразования набора переменных типа **WORD** в переменную типа **STRING** используется функция [SWAP DATA](#). Для обеспечения порядка байт, принятого в OPC-сервере, в процессе копирования происходит перестановка байт в регистрах (**xSwapBytes=TRUE**).

**ПРИМЕЧАНИЕ**

Для определения размера строки используется оператор **SIZEOF**, который учитывает терминирующий ноль. В рамках примера этот способ работает корректно, так как размер строки является нечетным (15 символов), и с учетом терминирующего нуля равен 16 байт (8 регистров Modbus).

По переднему фронту переменной **xWrite** (блок 9) происходит запись переменных программы в регистры slave'a. Для записи переменной типа **BOOL** используется экземпляр ФБ **BIT\_AS\_WORD** из библиотеки **Util**. Для записи переменной типа **REAL** используется экземпляр ФБ **REAL TO WORD2**. Для записи переменной типа **STRING** используется функция **SWAP DATA**.

6. Код программы на языке ST будет выглядеть следующим образом:

```

2 // чтобы запустить пример на ST на контроллере требуется:
3 // 1. Удалить из задачи MainTask вызов программа PLC_PRG_CFC
4 // 2. Привязать к задаче MainTask вызов программы PLC_PRG_ST
5
6
7 fbComControl1
8 (
9     xEnable      := TRUE,
10    usiComPort   := 1,
11    udiBaudrate  := 115200,
12    udiByteSize  := 8,
13    eParity      := OCL.COM_PARITY.NONE,
14    eStopBit     := OCL.COM_STOPBIT.ONE
15 );
16
17 fbModbusSerialSlave
18 (
19    xEnable      :=fbComControl1.xActive,
20    hCom         := fbComControl1.hCom,
21    usiSlaveId   := 1,
22    pData       := ADR(awSlaveData),
23    szSize      := SIZEOF(awSlaveData)
24 );
25
26 // данные, полученные от OPC
27 xVar_Opc := awSlaveData[0].0;
28 wVar_Opc := awSlaveData[1];
29 rVar_Opc := OCL.WORD2_TO_REAL(awSlaveData[2], awSlaveData[3], FALSE);
30 OCL.SWAP_DATA( ADR(awSlaveData[4]), ADR(sVar_Opc), SIZEOF(sVar_Opc), TRUE, FALSE, FALSE );
31
32
33 // по команде записываем переменные из программы в регистры Modbus Slave
34 fbWriteEdge(CLK := xWrite);
35
36 IF fbWriteEdge.Q THEN
37
38     awSlaveData[0].0 := xVar_Plc;
39     awSlaveData[1]   := wVar_Plc;
40     fbRealToWorld2(rInput := rVar_Plc, wOutput1 => awSlaveData[2], wOutput2 => awSlaveData[3]);
41     OCL.SWAP_DATA( ADR(sVar_Plc), ADR(awSlaveData[4]), SIZEOF(sVar_Plc), TRUE, FALSE, FALSE );
42
43 END_IF

```

Рисунок 5.9.15 – Код программы на языке ST

При первом вызове программы происходит открытие COM-порта с заданными настройками с помощью экземпляра ФБ **COM Control**. После успешного открытия порта выход **xActive** принимает и сохраняет значение **TRUE**, что приводит к вызову экземпляра ФБ **MB SerialSlave**, который выполняет функцию Modbus Slave с адресом 1 (**usiSlaveId**). Данные slave'a хранятся в массиве **awSlaveData**. В экземпляре ФБ передается указатель на этот массив (**pData**) и его размер в байтах (**szSize**).



Данные из регистров slave'a копируются в переменные программы. Для преобразования двух переменных типа **WORD** в переменную типа **REAL** используется функция [WORD2 TO REAL](#). Для преобразования набора переменных типа **WORD** в переменную типа **STRING** используется функция [SWAP DATA](#). Для обеспечения порядка байт, принятого в OPC-сервере, в процессе копирования происходит перестановка байт в регистрах (**xSwapBytes=TRUE**).



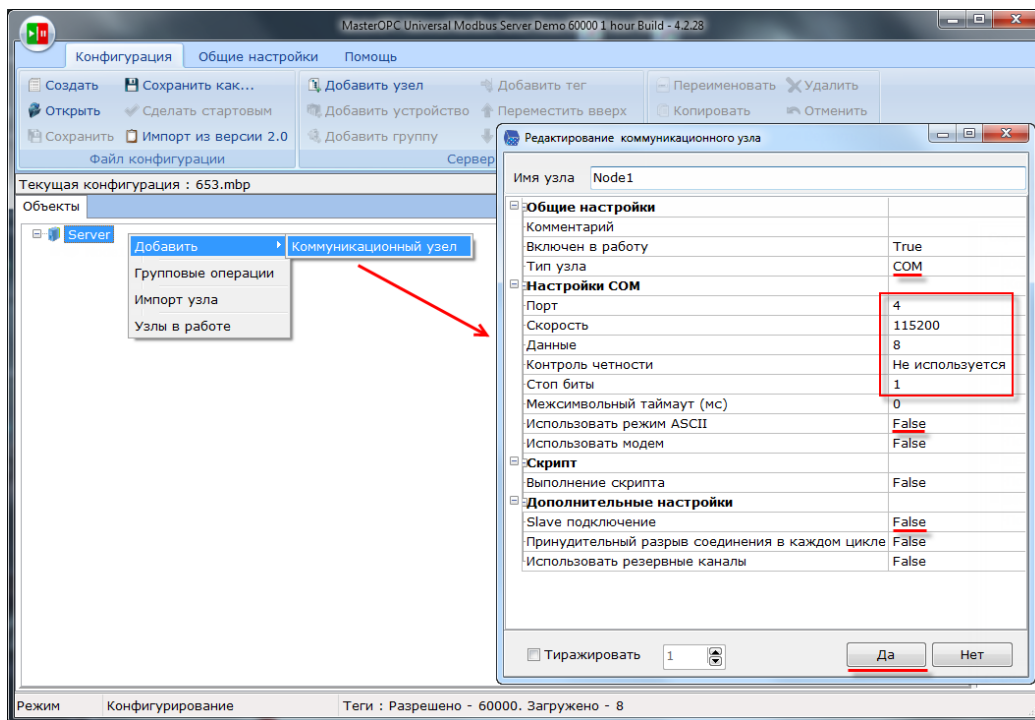
**ПРИМЕЧАНИЕ**

Для определения размера строки используется оператор **SIZEOF**, который учитывает [терминирующий ноль](#). В рамках примера этот способ работает корректно, так как размер строки является нечетным (15 символов), и с учетом терминирующего нуля равен 16 байт (8 регистров Modbus).

По переднему фронту переменной **xWrite** происходит запись переменных программы в регистры slave'a. Для записи переменной типа **REAL** используется экземпляр ФБ [REAL TO WORD2](#). Для записи переменной типа **STRING** используется функция [SWAP DATA](#).

7. Установить и запустить [MasterOPC Universal Modbus Server](#).

8. Нажать **ПКМ** на узел **Server** и добавить коммуникационный узел типа **COM**. В узле следует указать сетевые настройки в соответствии с [таблицей 5.9.2](#). Для работы OPC-сервера в режиме **Modbus RTU Master** параметры **Использовать режим ASCII** и **Slave подключение** должны иметь значение **FALSE**.



**Рисунок 5.9.16 – Добавление коммуникационного узла**

9. Нажать ПКМ на коммуникационный узел и добавить устройство с настройками по умолчанию (Slave ID = 1 в соответствии с [таблицей 5.9.2](#)).

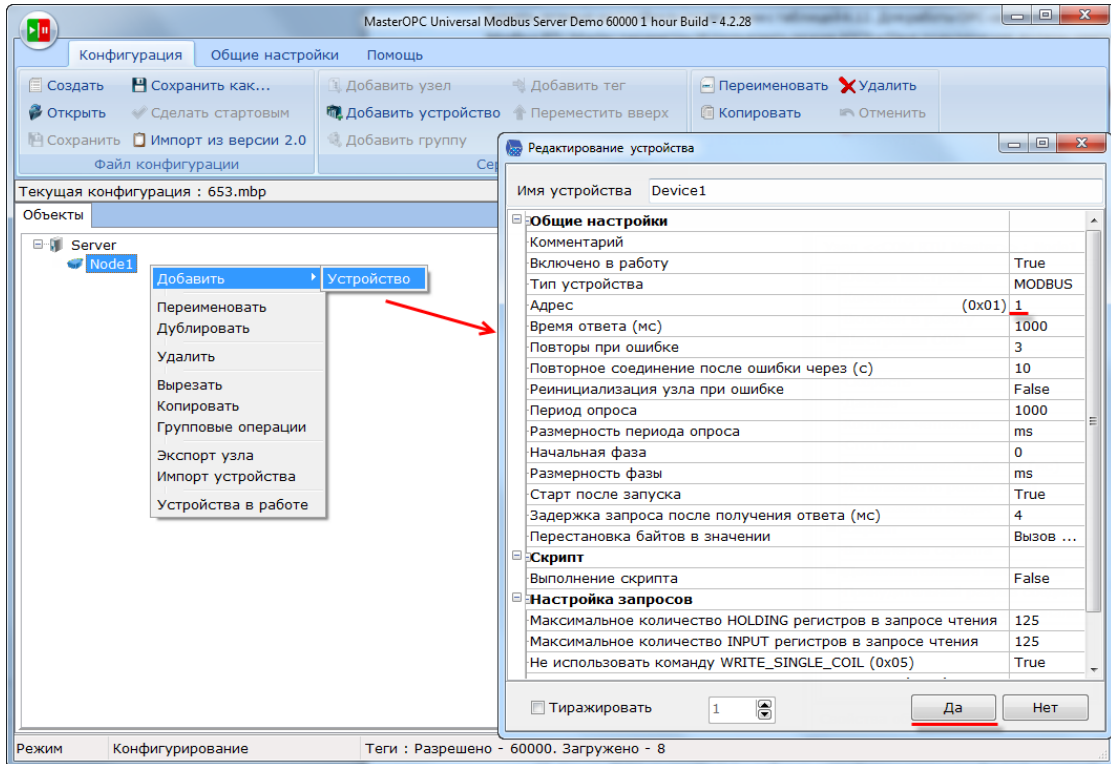


Рисунок 5.7.17 – Добавление устройства

10. Нажать ПКМ на устройство и добавить 4 тега. Число тегов соответствует числу переменных, считываемых/записываемых OPC-сервером. Настройки тегов приведены ниже.

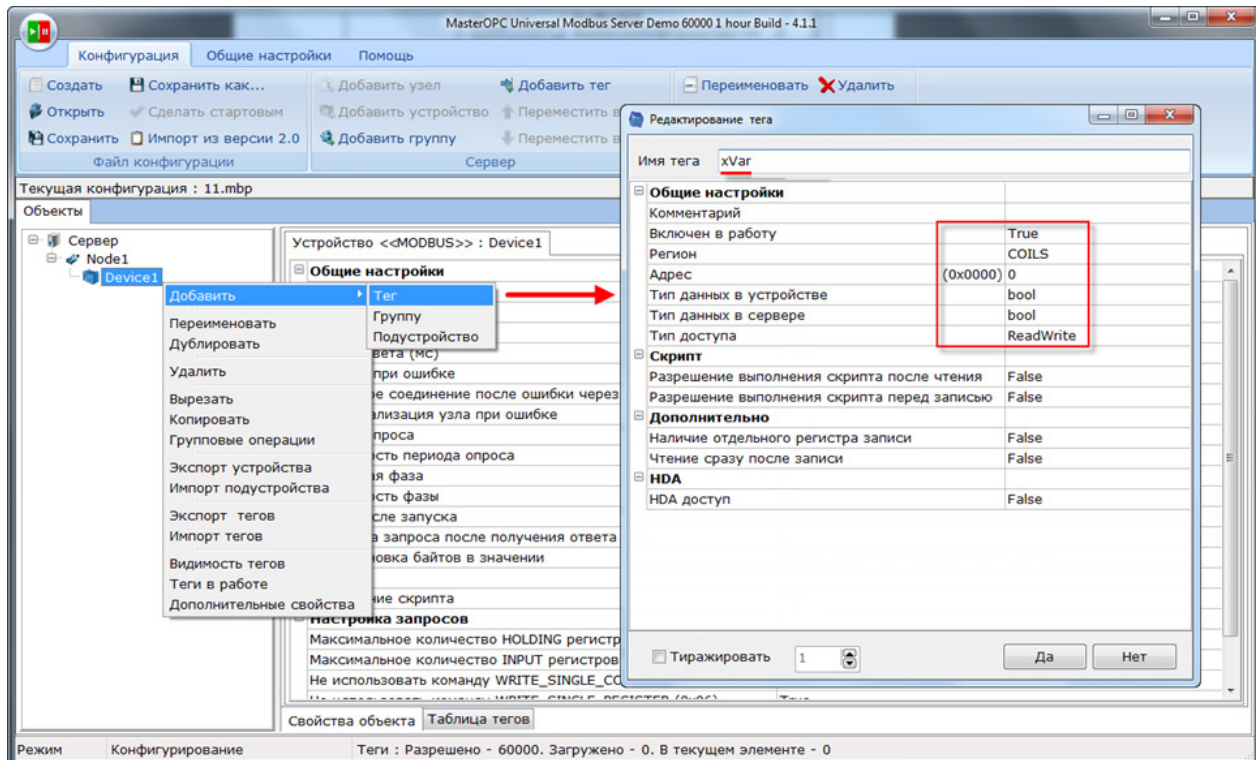


Рисунок 5.9.18 – Добавление тега xVar

Тег <<HOLDING_REGISTERS>> : wVar	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Адрес (0x0001)	1
Тип данных в устройстве	uint16
Тип данных в сервере	uint32
Тип доступа	ReadWrite
Использовать перестановку байтов устройства	True
Последний тег в групповом запросе	False
Пересчет (A*X + B)	False

Рисунок 5.9.19 – Добавление тега wVar

Тег <<HOLDING_REGISTERS>> : rVar	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Адрес (0x0002)	2
Тип данных в устройстве	float
Тип данных в сервере	float
Тип доступа	ReadWrite
Использовать перестановку байтов устройства	True
Последний тег в групповом запросе	False
Пересчет (A*X + B)	False

Рисунок 5.9.20 – Добавление тега rVar

Тег <<HOLDING_REGISTERS>> : sVar	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Адрес (0x0004)	4
Тип данных в устройстве	string
Тип данных в сервере	string
Количество байт для строкового типа	16
Тип строки для строкового типа	ascii
Тип доступа	ReadWrite
Использовать перестановку байтов устройства	True
Последний тег в групповом запросе	False
Пересчет (A*X + B)	False

Рисунок 5.9.21 – Добавление тега sVar

11. Загрузить проект в контроллер и запустить его. Запустить OPC-сервер для контроля значений переменных.

В OPC-сервере следует изменить значения тегов и наблюдать соответствующие значения в CODESYS. В CODESYS следует изменить значения **\_Plc** переменных и сгенерировать импульс в переменной **xWrite** для записи значений в регистры slave'a. Записанные значения будут прочитаны OPC-сервером.

Device.Application.PLC_PRG_ST		
Выражение	Тип	Значение
fbComControl1	OCL.COM_Control	
fbUnpackWord	UTIL.WORD_AS_BIT	
fbPackWord	UTIL.BIT_AS_WORD	
fbRealToWorld2	OCL.REAL_TO_WO...	
fbModbusSerialSlave	OCL.MB_SerialSlave	
awSlaveData	ARRAY [0..15] OF ...	
xVar_Opc	BOOL	TRUE
wVar_Opc	WORD	11
rVar_Opc	REAL	22.33
sVar_Opc	STRING(15)	'привет'
xVar_Plc	BOOL	TRUE
wVar_Plc	WORD	11
rVar_Plc	REAL	22.33
sVar_Plc	STRING(15)	'привет'
xWrite	BOOL	TRUE
fbWriteEdge	R_TRIG	

Объекты							
Устройство <<Device1>>							
Теги							
Имя	Регион	Адрес	Значение	Качество	Время (UTC)	Тип в сер...	
Node1.Device1.xVar	COILS	(0x00...	True	GOOD	2019-08-1...	bool	
Node1.Device1.wVar	HOL...	(0x00...	11	GOOD	2019-08-1...	uint32	
Node1.Device1.rVar	HOL...	(0x00...	22.330000	GOOD	2019-08-1...	float	
Node1.Device1.sVar	HOL...	(0x00...	привет	GOOD	2019-08-1...	string	

Рисунок 5.9.22 – Чтение и запись данных через OPC-сервер

### 5.9.3 СПК1хх [M01] (Modbus TCP Master) + модули Mx210

В качестве примера будет рассмотрена настройка обмена с модулями Mx210 (MB210-101 и МК210-301) с использованием библиотеки **OwenCommunication**. В примере используется библиотека версии **3.5.11.1**.

**Реализуемый алгоритм:** если значение первого аналогового входа модуля **MB210-101** превышает **30** и при этом первый дискретный вход модуля **МК210-301** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **МК210-301** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).

Структурная схема примера приведена на рисунке ниже:

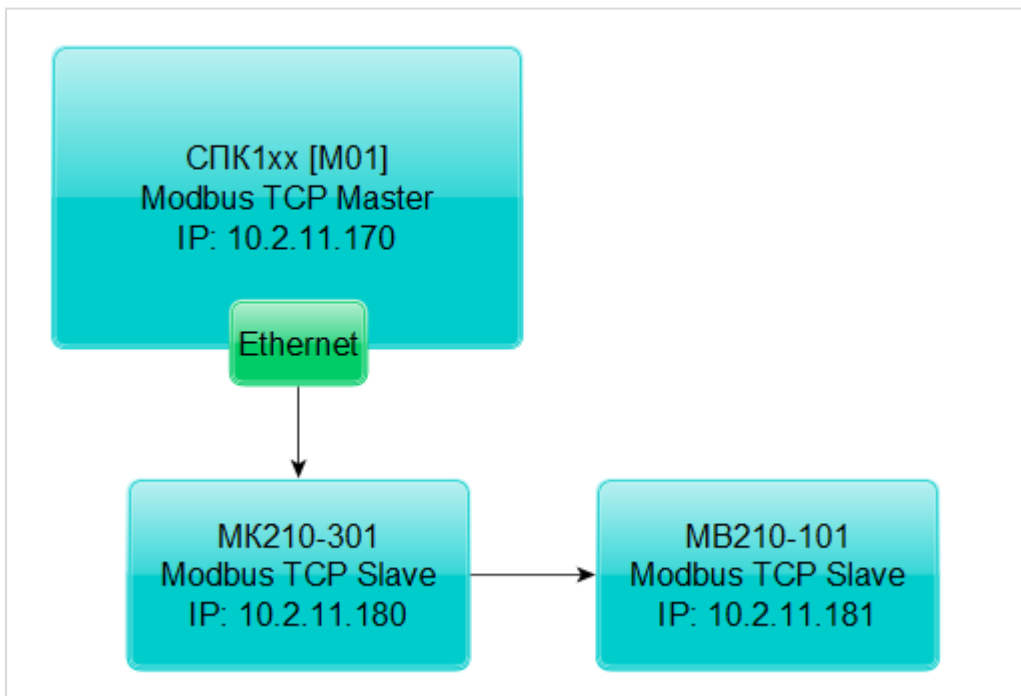


Рисунок 5.9.23 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (**ПКМ** на узел **Device** – **Обновить устройство**).

Пример доступен для скачивания:

[Example\\_OwenCommunicationModbusTcpMaster\\_3511v1.projectarchive](http://Example_OwenCommunicationModbusTcpMaster_3511v1.projectarchive)

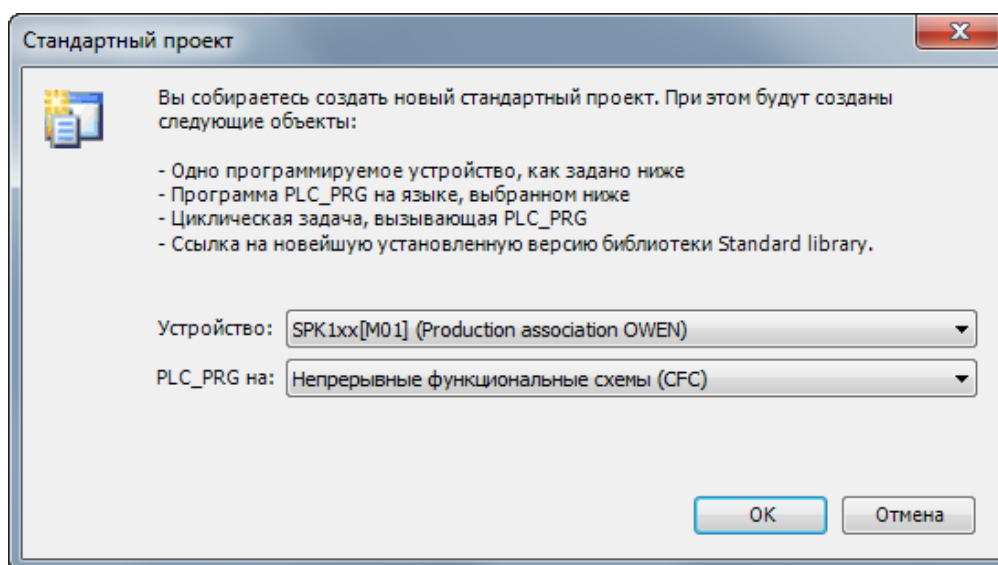
Сетевые параметры устройств приведены в таблице ниже:

**Таблица 5.9.4 – Сетевые параметры устройств**

Параметр	СПК1xx [M01]	МК210-301	МВ210-101
Режим работы	master	slave	slave
IP-адрес	10.2.11.170	10.2.11.180	10.2.11.181
Маска подсети	255.255.0.0		
IP-адрес шлюза	10.2.1.1		
Порт	502		
Unit ID	-	1	1

Для настройки обмена следует:

1. Настроить модули **Mx210** с помощью программы **ОВЕН Конфигуратор** в соответствии с таблицей 5.9.4 (см. руководство **Mx210. Примеры настройки обмена**). Подключить модули к контроллеру.
2. Установить в CODESYS библиотеку **OwenCommunication** (см. [п. 5.2](#)).
3. Создать новый проект **CODESYS** с программой на языке **ST** или **CFC**:



**Рисунок 5.9.24 – Создание проекта CODESYS**



**ПРИМЕЧАНИЕ**

Проект примера содержит программы на языках CFC и ST. По умолчанию используется программа на CFC (**PLC\_PRG\_CFC**). Для работы с программой на ST следует в конфигурации задач для задачи **MainTask** удалить вызов **PLC\_PRG\_CFC** и добавить вызов **PLC\_PRG\_ST**.

4. Добавить в проект библиотеки **OwenCommunication** и **Util** (**Менеджер библиотек – Добавить библиотеку**).



**ПРИМЕЧАНИЕ**

Библиотека **Util** используется только в программе на языке CFC.

5. Объявить в программе следующие переменные:

```

1  PROGRAM PLC_PRG_CFC
2  VAR
3      fbTcpClientMV210:  OCL.TCP_Client;      // ФБ TCP-подключения к модулю MB210-101
4      fbTcpClientMK210:  OCL.TCP_Client;      // ФБ TCP-подключения к модулю MK210-301
5      fbMV210_101_AI1:  OCL.MB_TcpRequest;    // ФБ опроса модуля MB210-101
6      fbMK210_301_DI:   OCL.MB_TcpRequest;    // ФБ опроса входов модуля MK210-301
7      fbMK210_301_DO:   OCL.MB_TcpRequest;    // ФБ опроса выходов модуля MK210-301
8
9      rAI1:              REAL;                // значение 1-го входа модуля MB210-101
10     wDiMask:           WORD;                // битовая маска входов модуля MK210-301
11     wDoMask:           WORD;                // битовая маска выходов модуля MK210-301
12     xDi0:              BOOL;                // значение 1-го входа модуля MK210-301
13     xDo0:              BOOL;                // значение 1-го выхода модуля MK210-301
14
15     awAI1:             ARRAY [0..1] OF WORD; // регистры, считанные с модуля MB210-101
16
17     fbUnpackWord:      UTIL.WORD_AS_BIT;     // ФБ распаковки битовой маски
18     fbPackWord:       UTIL.BIT_AS_WORD;     // ФБ упаковки битовой маски
19
20     iStateMV210:      INT;                  // шаг опроса модуля MB210-101
21     iStateMK210:      INT;                  // шаг опроса модуля MK210-301
22
23     fbConnectionTimeoutMV210:  TON;        // таймер таймаута соединения с модулем MB210-101
24     fbConnectionTimeoutMK210:  TON;        // таймер таймаута соединения с модулем MK210-301
25 END VAR
    
```

Рисунок 5.9.25 – Объявление переменных в программе



**ПРИМЕЧАНИЕ**

Переменные **fbUnpackWord** и **fbPackWord** используются только в программе на языке CFC. Переменные **iStateMV210** и **iStateMK210** используется только в программе на языке ST.

6. Нажать ПКМ на программу, выбрать команду **Добавление объекта – Действие** и добавить действия с названиями **MV210\_101** и **MK210\_301** (язык реализации действий соответствует языку программы). В рамках примера действия используются для повышения читабельности кода.

7. Код действий и программы на языке CFC будет выглядеть следующим образом:

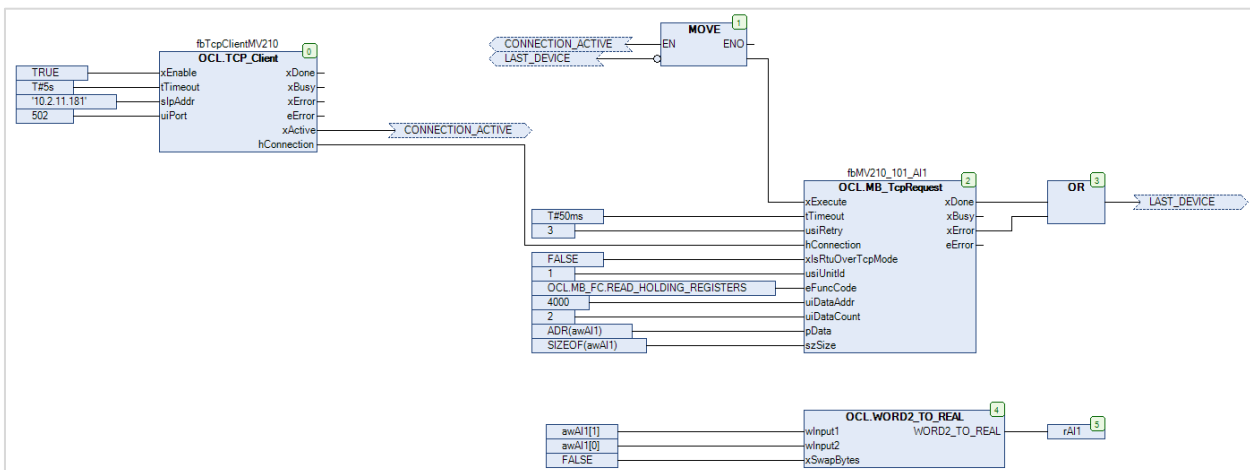


Рисунок 5.9.26 – Код действия MV210\_101

При первом вызове действия происходит установка соединения с модулем по заданному IP-адресу (**slpAddr**) и порту (**uiPort**) с помощью экземпляра ФБ **TCP\_Client** (блок 0). После установки соединения выход **xActive** принимает значение **TRUE**, что приводит к вызову экземпляра ФБ **MB\_TcpRequest** (блок 2), который производит опрос первого аналогового входа модуля **MB210-101** с

адресом **1 (usUnitId)**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Полученные данные помещаются в переменную **awAI1** типа **ARRAY [0..1] OF WORD**. С помощью функции **WORD2\_TO\_REAL** (блоки 4–5) этот массив преобразуется в переменную типа **rAI1** типа **REAL**.

После завершения работы экземпляра ФБ **MB\_TcpRequest** один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**. В любой из этих ситуаций с помощью блока **1** происходит вызов экземпляра ФБ **MB\_TcpRequest** со значением **FALSE** на входе **xExecute** (сброс блока), а потом – его следующий вызов, что приводит к началу следующего сеанса опроса.

Если соединение разорвано или экземпляр ФБ **TCP\_Client** находится в состоянии ошибки (выход **xActive** имеет значение **FALSE**), то опрос прекращается.



#### ПРИМЕЧАНИЕ

В рамках примера для повышения читабельности схемы вместо некоторых линий связи использованы **метки соединения**.

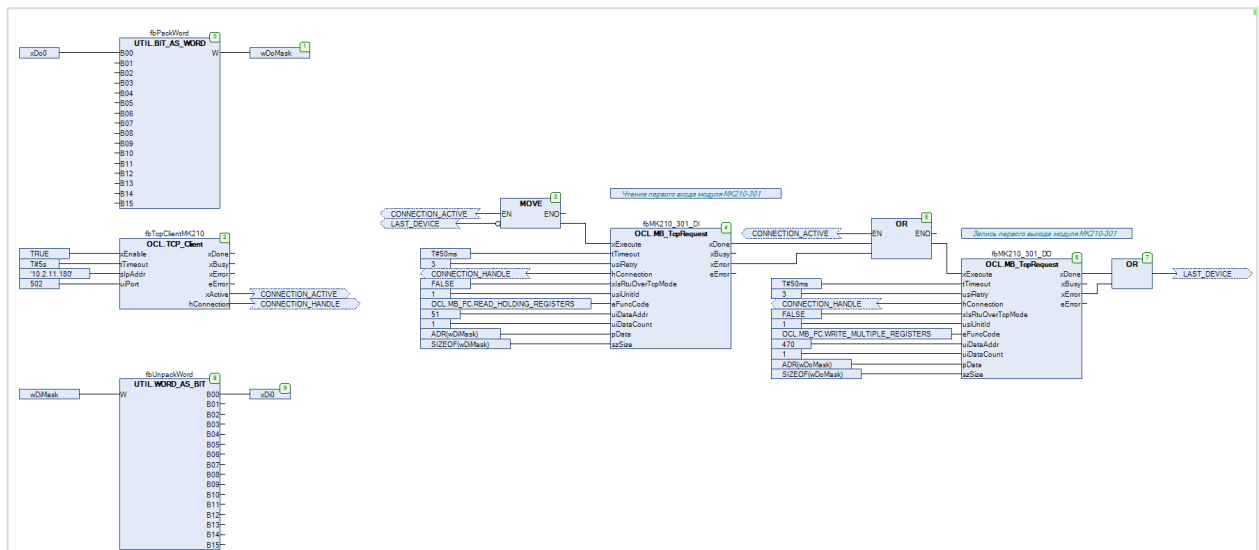


Рисунок 5.9.27 – Код действия MK210\_301

При первом вызове действия происходит установка соединения с модулем по заданному IP-адресу (**slpAddr**) и порту (**uiPort**) с помощью экземпляра ФБ **TCP\_Client** (блок 0). После установки соединения выход **xActive** принимает значение **TRUE**, что приводит к вызову экземпляра ФБ **MB\_TcpRequest** (блок 4), который производит опрос битовой маски дискретных входов модуля **MK210-301** с адресом **1 (usiUnitId)**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Полученные данные помещаются в переменную **wDiMask** типа **WORD** и разделяются на отдельные переменные типа **BOOL** с помощью экземпляра ФБ **WORD\_AS\_BIT** из библиотеки **Util**.

После завершения работы ФБ **MB\_TcpRequest** один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**. В любой из этих ситуаций начинается работа следующего блока **MB\_TcpRequest** (блоки 5–6), который производит запись переменной **wDoMask** в качестве битовой маски дискретных выходов модуля. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Экземпляр блока **BIT\_AS\_WORD** (блоки 0–1) позволяет упаковать отдельные переменные типа **BOOL** в битовую маску типа **WORD**.

После окончания последнего блока **MB\_TcpRequest** с помощью блока **3** происходит вызов первого ФБ **MB\_TcpRequest** со значением **FALSE** на входе **xExecute** (сброс блока), а потом – его следующий вызов, что приводит к началу следующего сеанса опроса.



Если соединение разорвано или экземпляр ФБ [TCP\\_Client](#) находится в состоянии ошибки (выход **xActive** имеет значение **FALSE**), то опрос прекращается.



**ПРИМЕЧАНИЕ**

Для переменных битовых масок используется тип **WORD** (хотя из-за небольшого числа входов/выходов подошел бы и **BYTE**), так как он соответствует по размеру регистру Modbus.



**ПРИМЕЧАНИЕ**

В рамках примера для повышения читабельности схемы вместо некоторых линий связи использованы **метки соединения**.

В программе **PLC\_PRG\_CFC** производится вызов действий **MV210\_101** и **MK210\_301**, а также выполнение алгоритма, описанного в условии примера.

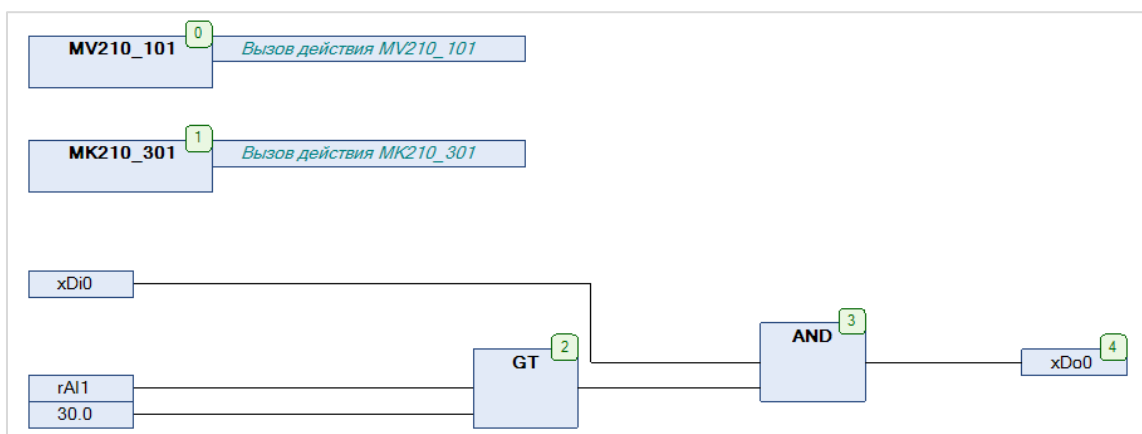


Рисунок 5.9.28 – Код программы PLC\_PRG\_CFC

## 8. Код действий и программы на языке ST будет выглядеть следующим образом:

```

1  CASE iStateMV210 OF
2
3      0: // подключение к модулю
4
5          fbConnectionTimeoutMV210(IN := TRUE, PT := T#10S);
6
7          fbTcpClientMV210
8          (
9              xEnable      := TRUE,
10             sIpAddr       := '10.2.11.181',
11             uiPort        := 502
12         );
13
14         IF fbTcpClientMV210.xActive THEN
15             iStateMV210 := 1;
16         END_IF
17
18         IF fbTcpClientMV210.xError OR fbConnectionTimeoutMV210.Q THEN
19             fbConnectionTimeoutMV210(IN := FALSE);
20             fbTcpClientMV210(xEnable := FALSE);
21         END_IF
22
23
24
25     1: // опрос MB210-101
26
27         fbMV210_101_AI1
28         (
29             xExecute       := TRUE,
30             tTimeout       := T#50MS,
31             usiRetry       := 3,
32             hConnection    := fbTcpClientMV210.hConnection,
33             xIsRtuOverTcpMode := FALSE,
34             usiUnitId      := 1,
35             eFuncCode      := OCL.MB_FC.READ_HOLDING_REGISTERS,
36             uiDataAddr     := 4000,
37             uiDataCount    := 2,
38             pData          := ADR(awAI1),
39             szSize         := SIZEOF(awAI1)
40         );
41
42         IF fbMV210_101_AI1.xDone OR fbMV210_101_AI1.xError THEN
43             // после выполнения блока его надо сбросить
44             fbMV210_101_AI1(xExecute := FALSE);
45
46             rAI1 := OCL.WORD2_TO_REAL(awAI1[1], awAI1[0], FALSE);
47
48             iStateMV210 := 2;
49         END_IF
50
51
52     2: // здесь можно добавить следующий запрос к модулю
53         // после выполнения последнего запроса проверяем соединение с модулем
54         iStateMV210 := 0;
55
56 END_CASE

```

Рисунок 5.9.29 – Код действия MV210\_101

При первом вызове действия в **шаге 0** происходит установка соединения с модулем по заданному IP-адресу (**sIpAddr**) и порту (**uiPort**) с помощью экземпляра ФБ [TCP\\_Client](#) (блок **0**). После установки соединения выход **xActive** принимает значение **TRUE**, что приводит к переходу на **шаг 1**. Если соединение не удастся установить в течение заданного времени (это определяются с помощью таймера **fbConnectionTimeoutMV210**) или при установке соединения происходит ошибка, то выполняется новая попытка соединения.

На **шаге 1** выполняется вызов экземпляра ФБ [MB\\_TcpRequest](#), который производит опрос первого аналогового входа модуля **MB210-101** с адресом **1** (**usiSlaveId**). Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Полученные данные помещаются в переменную **awAI1** типа **ARRAY [0..1] OF WORD**. С помощью функции [WORD2 TO REAL](#) этот массив преобразуется в переменную типа **rAI1** типа **REAL**.

После завершения работы экземпляра ФБ [MB\\_TcpRequest](#) один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**.

В любой из этих ситуаций происходит вызов экземпляра ФБ [MB\\_TcpRequest](#) со значением **FALSE** на входе **xExecute** (сброс блока) и переход на **шаг 2**.

В рамках примера на **шаге 2** происходит переход на **шаг 0**, что приводит к проверке состояния соединения и началу следующего сеанса опроса. В случае опроса нескольких параметров устройства число шагов можно увеличить. После выполнения последнего шага должен происходить переход на шаг опроса первого устройства.

```

1  CASE iStateMK210 OF
2
3      0: // подключение к модулю
4
5          fbConnectionTimeoutMK210(IN := TRUE, PT := T#10S);
6
7          fbTcpClientMK210
8          (
9              xEnable      := TRUE,
10             tTimeout     := T#5S,
11             sIpAddr      := '10.2.11.180',
12             uiPort       := 502
13         );
14
15         IF fbTcpClientMK210.xActive THEN
16             iStateMK210 := 1;
17         END_IF
18
19         IF fbTcpClientMK210.xError OR fbConnectionTimeoutMK210.Q THEN
20             fbConnectionTimeoutMK210(IN := FALSE);
21             fbTcpClientMK210(xEnable := FALSE);
22         END_IF
23
24     1: // опрос дискретных входов модуля MK210-301
25
26         fbMK210_301_DI
27         (
28             xExecute      := TRUE,
29             tTimeout     := T#50MS,
30             usiRetry      := 3,
31             hConnection   := fbTcpClientMK210.hConnection,
32             xIsRtuOverTcpMode := FALSE ,
33             usiUnitId     := 1,
34             eFuncCode     := OCL.MB_FC.READ_HOLDING_REGISTERS,
35             uiDataAddr    := 51,
36             uiDataCount   := 1,
37             pData         := ADR(wDiMask),
38             szSize        := SIZEOF(wDiMask)
39         );
40
41         IF fbMK210_301_DI.xDone OR fbMK210_301_DI.xError THEN
42             // после выполнения блока его надо сбросить
43             fbMK210_301_DI(xExecute := FALSE);
44
45             xDi0 := wDiMask.0;
46
47             iStateMK210 := 2;
48         END_IF
49
50     2: // запись дискретных выходов модуля MK210-301
51
52         wDoMask.0 := xDo0;
53
54         fbMK210_301_DO
55         (
56             xExecute      := TRUE,
57             tTimeout     := T#50MS,
58             usiRetry      := 3,
59             hConnection   := fbTcpClientMK210.hConnection,
60             xIsRtuOverTcpMode := FALSE ,
61             usiUnitId     := 1,
62             eFuncCode     := OCL.MB_FC.WRITE_MULTIPLE_REGISTERS,
63             uiDataAddr    := 470,
64             uiDataCount   := 1,
65             pData         := ADR(wDoMask),
66             szSize        := SIZEOF(wDoMask)
67         );
68
69         IF fbMK210_301_DO.xDone OR fbMK210_301_DO.xError THEN
70             // после выполнения блока его надо сбросить
71             fbMK210_301_DO(xExecute := FALSE);
72
73             // перед следующим сеансом опроса проверяем соединение с модулем
74             iStateMK210 := 0;
75         END_IF
76
77     END_CASE

```

Рисунок 5.9.30 – Код действия MK210\_301

При первом вызове действия в **шаге 0** происходит установка соединения с модулем по заданному IP-адресу (**slpAddr**) и порту (**uiPort**) с помощью экземпляра ФБ [TCP\\_Client](#) (**блок 0**). После установки соединения выход **xActive** принимает значение **TRUE**, что приводит к переходу на **шаг 1**. Если соединение не удастся установить в течение заданного времени (это определяются с помощью таймера **fbConnectionTimeoutMK210**) или при установке соединения происходит ошибка, то выполняется новая попытка соединения.

На **шаге 1** выполняется вызов экземпляра ФБ [MB\\_TcpRequest](#), который производит опрос битовой маски дискретных входов модуля **MK210-301** с адресом **1 (usiSlaveld)**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Полученные данные помещаются в переменную **wDiMask** типа **WORD**, из которой происходит копирование данных в нужные переменные типа **BOOL** с помощью побитового доступа (*переменная.номер\_бита*).

После завершения работы экземпляра ФБ [MB\\_TcpRequest](#) один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**. В любой из этих ситуаций происходит вызов экземпляра ФБ [MB\\_TcpRequest](#) со значением **FALSE** на входе **xExecute** (сброс блока) и переход на **шаг 2**.

На **шаге 2** выполняется вызов экземпляра ФБ [MB\\_TcpRequest](#), который производит запись переменной **wDoMask** в качестве битовой маски дискретных выходов модуля. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Битовая маска может формироваться из отдельных переменных типа **BOOL** с помощью побитового доступа.

В рамках примера на **шаге 2** происходит переход на **шаг 0**, что приводит к проверке соединения и началу следующего сеанса опроса. В случае опроса нескольких параметров устройства число шагов можно увеличить. После выполнения последнего шага должен происходить переход на шаг опроса первого устройства.

Если соединение разорвано или экземпляр ФБ [TCP\\_Client](#) находится в состоянии ошибки (выход **xActive** имеет значение **FALSE**), то опрос прекращается.

В программе **PLC\_PRG\_ST** производится вызов действий **MV210\_101** и **MK210\_301**, а также выполнение алгоритма, описанного в условии примера.

```
1 // чтобы запустить пример на ST на контроллере требуется:
2 // 1. Удалить из задачи MainTask вызов программы PLC_PRG_CFC
3 // 2. Привязать к задаче MainTask вызов программы PLC_PRG_ST
4
5 MV210_101();
6 MK210_301();
7
8 xDo0 := xDi0 AND (rAI1 > 30.0);
```

Рисунок 5.9.31 – Код программы PLC\_PRG\_ST

**9. Загрузить проект в контроллер и запустить его.**

В переменной **rAI1** будет отображаться текущее значение первого аналогового входа модуля **MB210-101**. В переменной **xDi0** будет отображаться текущее значение первого дискретного входа модуля **MK210-301**.

Если значение **rAI1** превысит **30** и при этом значение **xDi0** будет равно **TRUE**, то в переменную **xDo0** будет записано значение **TRUE**, что приведет к замыканию первого дискретного выхода модуля **MK210-301**. Если одно из условий перестанет выполняться, то выход будет разомкнут.

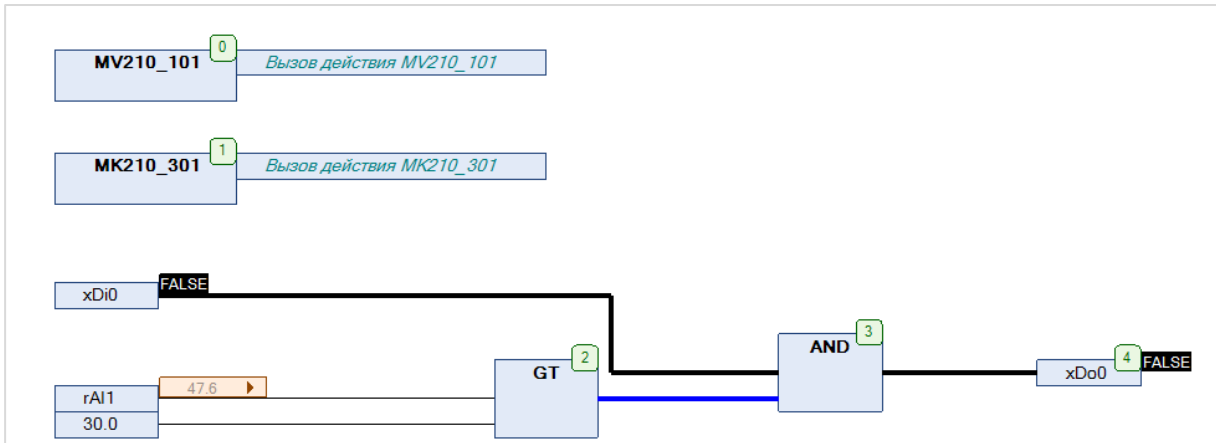


Рисунок 5.9.32 – Выполнение программы в режиме Online

### 5.9.4 СПК1хх [M01] (Modbus TCP Slave) + MasterOPC Universal Modbus Server

В качестве примера будет рассмотрена настройка обмена с OPC-сервером [Insat MasterOPC Universal Modbus Server](#), который будет использоваться в режиме **Modbus TCP Master**, с помощью библиотеки **OwenCommunication**. В примере используется библиотека версии **3.5.11.1**.

Структурная схема примера приведена на рисунке ниже:

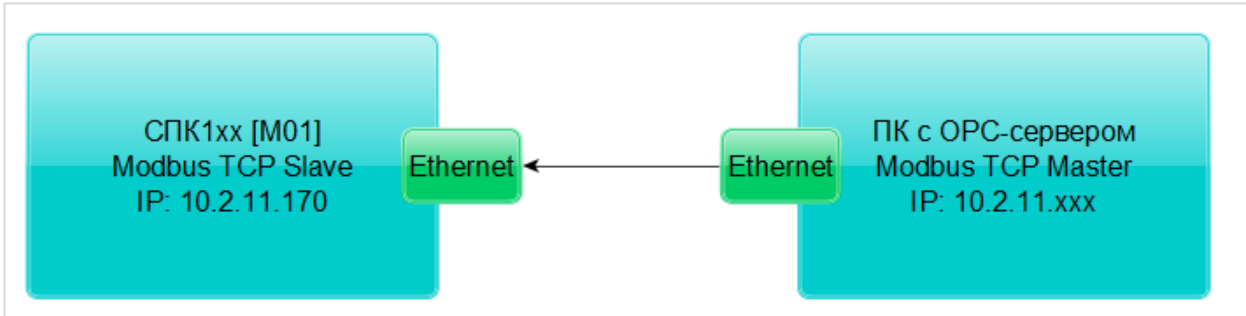


Рисунок 5.9.33 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел **Device** – **Обновить устройство**).

Пример доступен для скачивания: [Example OwenCommunicationModbusTcpSlave\\_3511v1.zip](#)

Сетевые параметры модулей приведены в таблице ниже:

Таблица 5.9.5 – Сетевые параметры устройств

Параметр	СПК1хх [M01]	ПК с OPC-сервером
IP-адрес	10.2.11.170	<i>любой адрес из сети, к которой подключен контроллер</i>
Порт	502	
Адрес (Unit ID)	1	-
Режим работы	slave	master

Переменные примера описаны в таблице ниже:

Таблица 5.9.6 – Список переменных примера

Имя	Тип	Область памяти Modbus	Адрес регистра/бита
xVar_Opc	BOOL	Coils	0/0
wVar_Opc	WORD	Holding регистры	1
rVar_Opc	REAL		2–3
sVar_Opc	STRING(15)		4–11



**ПРИМЕЧАНИЕ**

В рамках примера значения переменных slave'a могут быть изменены как из OPC, так и из программы контроллера (с помощью переменных с постфиксом **\_Pic**).

Для настройки обмена следует:

1. Подключить контроллер и ПК с общей локальной сети.
2. Создать новый проект **CODESYS** с программой на языке **ST** или **CFC**:

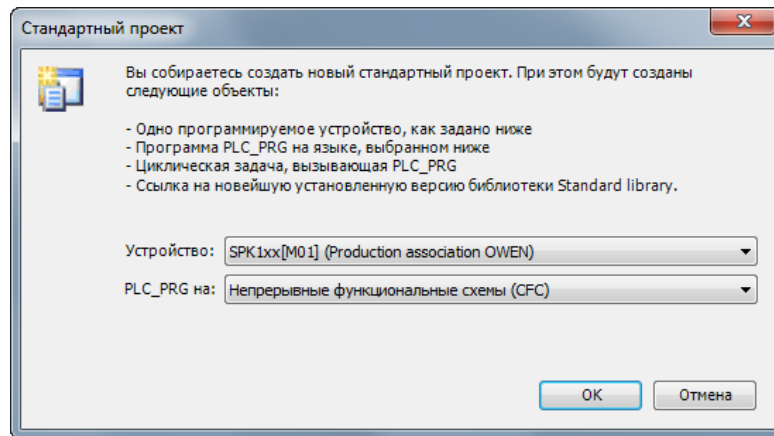


Рисунок 5.9.12 – Создание проекта CODESYS



#### ПРИМЕЧАНИЕ

Проект примера содержит программы на языках CFC и ST. По умолчанию используется программа на CFC (**PLC\_PRG\_CFC**). Для работы с программой на ST следует в конфигурации задач для задачи **MainTask** удалить вызов **PLC\_PRG\_CFC** и добавить вызов **PLC\_PRG\_ST**.

3. Добавить в проект библиотеки **OwenCommunication** и **Util** (**Менеджер библиотек – Добавить библиотеку**).



#### ПРИМЕЧАНИЕ

Библиотека **Util** используется только в программе на языке CFC.

4. Объявить в программе следующие переменные:

```

1  PROGRAM PLC_PRG_ST
2  VAR
3
4      fbUnpackWord:      UTIL.WORD_AS_BIT;      // #Б распаковки битовой маски
5      fbPackWord:        UTIL.BIT_AS_WORD;      // #Б упаковки битовой маски
6      fbRealToWord2:     OCL.REAL_TO_WORD2;     // #Б преобразования REAL в две переменные типа WORD
7
8
9      fbModbusTcpSlave:  OCL.MB_TcpSlave;       // #Б для реализации Modbus Slave
10
11     awSlaveData:        ARRAY [0..15] OF WORD; // буфер данных Modbus Slave
12
13     (* значения, полученные от OPC *)
14     xVar_Opc:            BOOL;
15     wVar_Opc:            WORD;
16     rVar_Opc:            REAL;
17     sVar_Opc:            STRING(15);
18
19     (* значения для передачи в OPC *)
20     xVar_Plc:            BOOL;
21     wVar_Plc:            WORD;
22     rVar_Plc:            REAL;
23     sVar_Plc:            STRING(15);
24
25     xWrite:              BOOL;                 // команда записи данных из программы в регистры Modbus Slave
26     fbWriteEdge:         R_TRIG;              // триггер для однократной записи
27 END_VAR
28

```

Рисунок 5.9.34 – Объявление переменных программы



#### ПРИМЕЧАНИЕ

Переменные **fbUnpackWord** и **fbPackWord** используются только в программе на языке CFC.



5. Код программы на языке CFC будет выглядеть следующим образом:

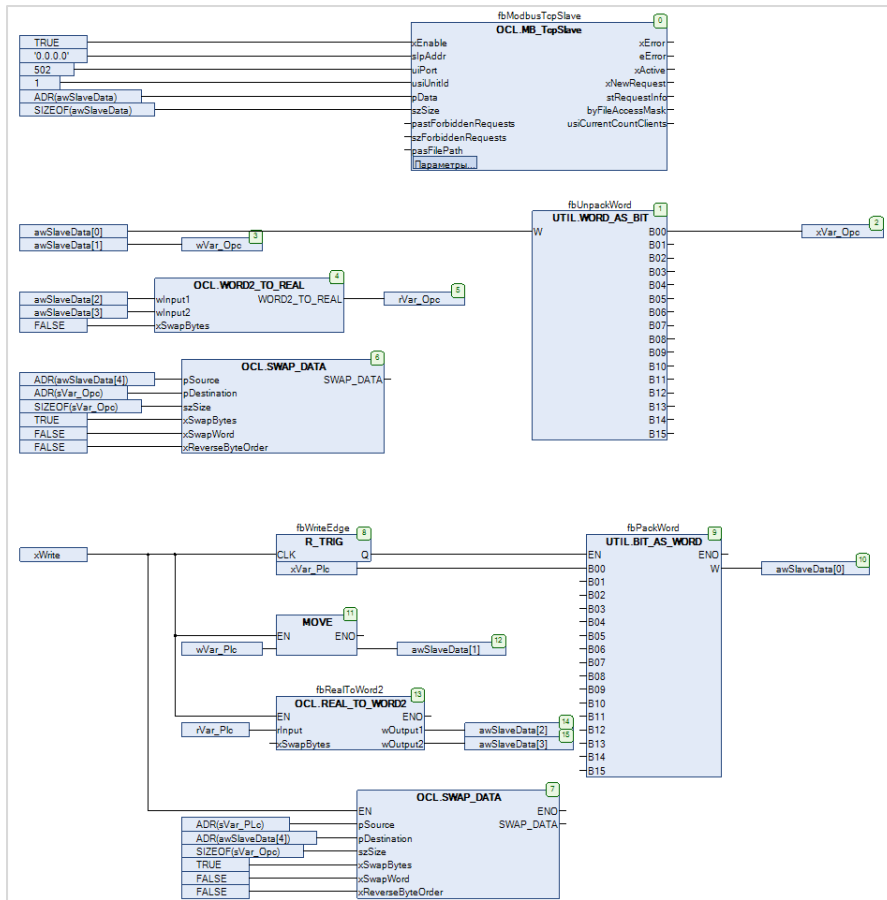


Рисунок 5.9.35 – Код программы на языке CFC

При запуске программы происходит вызов экземпляра ФБ [MB TcpSlave](#) (блок 0), который выполняет функцию Modbus Slave. В качестве IP-адреса (**slpAddr**), на котором используется slave, применяется специальный адрес [0.0.0.0](#), который соответствует адресам всех интерфейсов контроллера (то есть slave доступен по всем TCP/IP интерфейсам). Значения входов **uiPort** и **usiUnitId** соответствуют [таблице 5.9.5](#).

Данные slave'a хранятся в массиве **awSlaveData**. В экземпляр ФБ передается указатель на этот массив (**pData**) и его размер в байтах (**szSize**).

В **блоках 1–6** происходит копирование данных из регистров slave'a в переменные программы. Для выделения переменных типа **BOOL** из переменной типа **WORD** используется экземпляр ФБ **WORD\_AS\_BIT** из библиотеки **Util**. Для преобразования двух переменных типа **WORD** в переменную типа **REAL** используется функция [WORD2\\_TO\\_REAL](#). Для преобразования набора переменных типа **WORD** в переменную типа **STRING** используется функция [SWAP\\_DATA](#). Для обеспечения порядка байт, принятого в OPC-сервере, в процессе копирования происходит перестановка байт в регистрах (**xSwapBytes=TRUE**).



**ПРИМЕЧАНИЕ**

Для определения размера строки используется оператор **SIZEOF**, который учитывает [терминирующий ноль](#). В рамках примера этот способ работает корректно, так как размер строки является нечетным (15 символов), и с учетом терминирующего нуля равен 16 байт (8 регистров Modbus).

По переднему фронту переменной **xWrite** (блок 8) происходит запись переменных программы в регистры slave'a. Для записи переменной типа **BOOL** используется экземпляр ФБ **BIT\_AS\_WORD** из библиотеки **Util**. Для записи переменной типа **REAL** используется экземпляр ФБ **REAL\_TO\_WORD2**. Для записи переменной типа **STRING** используется функция **SWAP\_DATA**.

6. Код программы на языке ST будет выглядеть следующим образом:

```

2 // чтобы запустить пример на ST на контроллере требуется:
3 // 1. Удалить из задачи MainTask вызов программы PLC_PRG_CFC
4 // 2. Привязать к задаче MainTask вызов программы PLC_PRG_ST
5
6
7 fbModbusTcpSlave
8 (
9     xEnable      := TRUE,
10    sIpAddr      := '0.0.0.0',
11    uiPort       := 502,
12    usiUnitId    := 1,
13    pData        := ADR(awSlaveData),
14    szSize       := SIZEOF(awSlaveData)
15 );
16
17 // данные, полученные от OPC
18 xVar_Opc := awSlaveData[0].0;
19 wVar_Opc := awSlaveData[1];
20 rVar_Opc := OCL.WORD2_TO_REAL(awSlaveData[2], awSlaveData[3], FALSE);
21 OCL.SWAP_DATA( ADR(awSlaveData[4]), ADR(sVar_Opc), SIZEOF(sVar_Opc), TRUE, FALSE, FALSE );
22
23
24 // по команде записываем переменные из программы в регистры Modbus Slave
25 fbWriteEdge(CLK := xWrite);
26
27 IF fbWriteEdge.Q THEN
28
29     awSlaveData[0].0 := xVar_Plc;
30     awSlaveData[1]   := wVar_Plc;
31     fbRealToWord2(rInput := rVar_Plc, wOutput1 => awSlaveData[2], wOutput2 => awSlaveData[3]);
32     OCL.SWAP_DATA( ADR(sVar_Plc), ADR(awSlaveData[4]), SIZEOF(sVar_Plc), TRUE, FALSE, FALSE );
33
34 END_IF

```

Рисунок 5.9.36 – Код программы на языке ST

При запуске программы происходит вызов экземпляра ФБ **MB\_TcpSlave** (блок 0), который выполняет функцию Modbus Slave. В качестве IP-адреса (**sIpAddr**), на котором используется slave, применяется специальный адрес **0.0.0.0**, который соответствует адресам всех интерфейсов контроллера (то есть slave доступен по всем TCP/IP интерфейсам). Значения входов **uiPort** и **usiUnitId** соответствуют [таблице 5.9.5](#).

Данные из регистров slave'a копируются в переменные программы. Для преобразования двух переменных типа **WORD** в переменную типа **REAL** используется функция **WORD2\_TO\_REAL**. Для преобразования набора переменных типа **WORD** в переменную типа **STRING** используется функция **SWAP\_DATA**. Для обеспечения порядка байт, принятого в OPC-сервере, в процессе копирования происходит перестановка байт в регистрах (**xSwapBytes=TRUE**).



#### ПРИМЕЧАНИЕ

Для определения размера строки используется оператор **SIZEOF**, который учитывает **терминирующий ноль**. В рамках примера этот способ работает корректно, так как размер строки является нечетным (15 символов), и с учетом терминирующего нуля равен 16 байт (8 регистров Modbus).

По переднему фронту переменной **xWrite** происходит запись переменных программы в регистры slave'a. Для записи переменной типа **REAL** используется экземпляр ФБ [REAL TO WORD2](#). Для записи переменной типа **STRING** используется функция [SWAP DATA](#).

7. Установить и запустить [MasterOPC Universal Modbus Server](#).

8. Нажать **ПКМ** на узел **Server** и добавить коммуникационный узел типа **TCP/IP**. В узле следует указать сетевые настройки в соответствии с [таблицей 5.9.5](#). Для работы OPC-сервера в режиме **Modbus TCP Master** параметры **Modbus по верх TCP** и **Slave подключение** должны иметь значение **FALSE**.

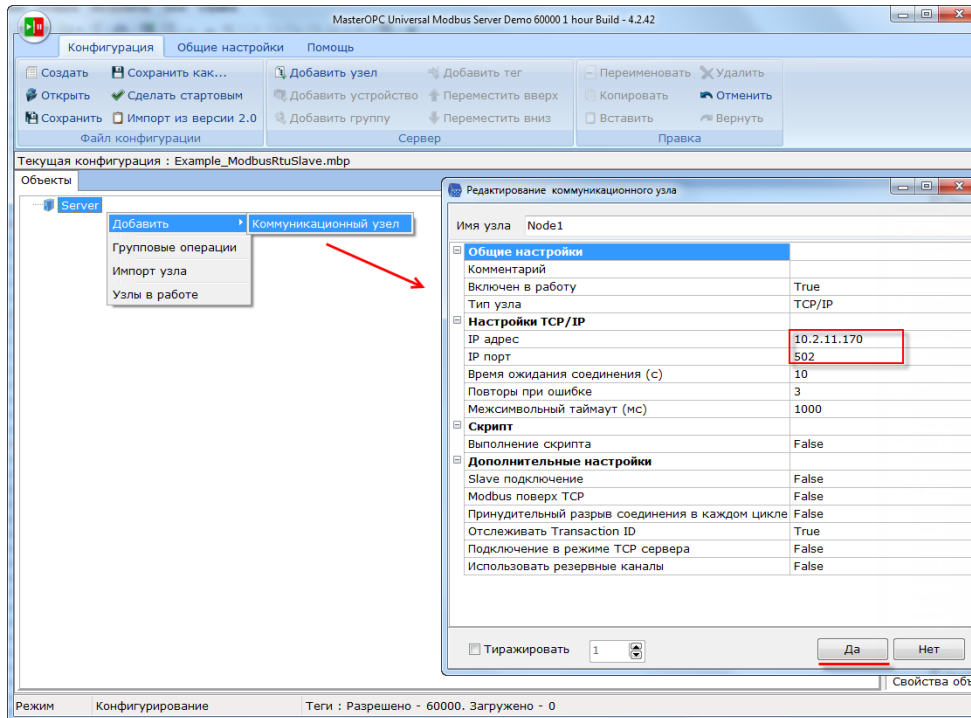


Рисунок 5.9.37 – Добавление коммуникационного узла

9. Нажать **ПКМ** на коммуникационный узел и добавить устройство с настройками по умолчанию.

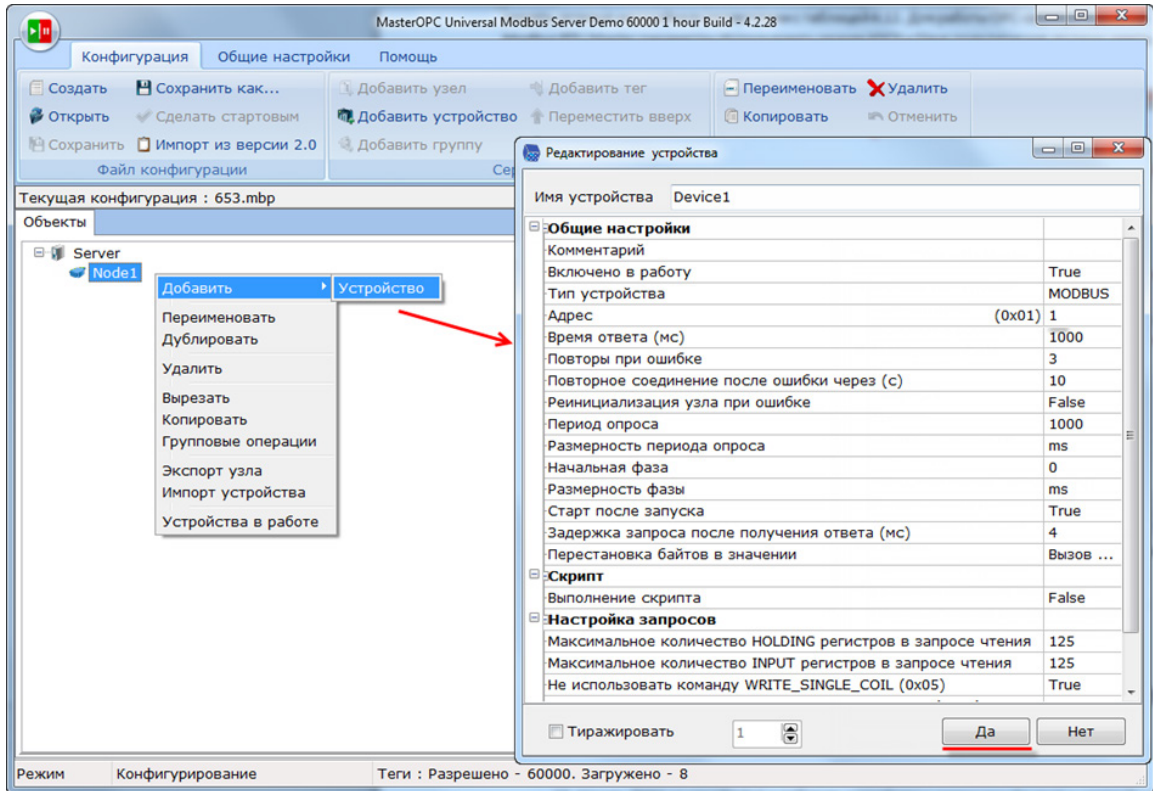


Рисунок 5.9.38 – Добавление устройства

10. Нажать **ПКМ** на устройство и добавить 4 тега. Число тегов соответствует числу переменных, считываемых/записываемых OPC-сервером. Настройки тегов приведены ниже.

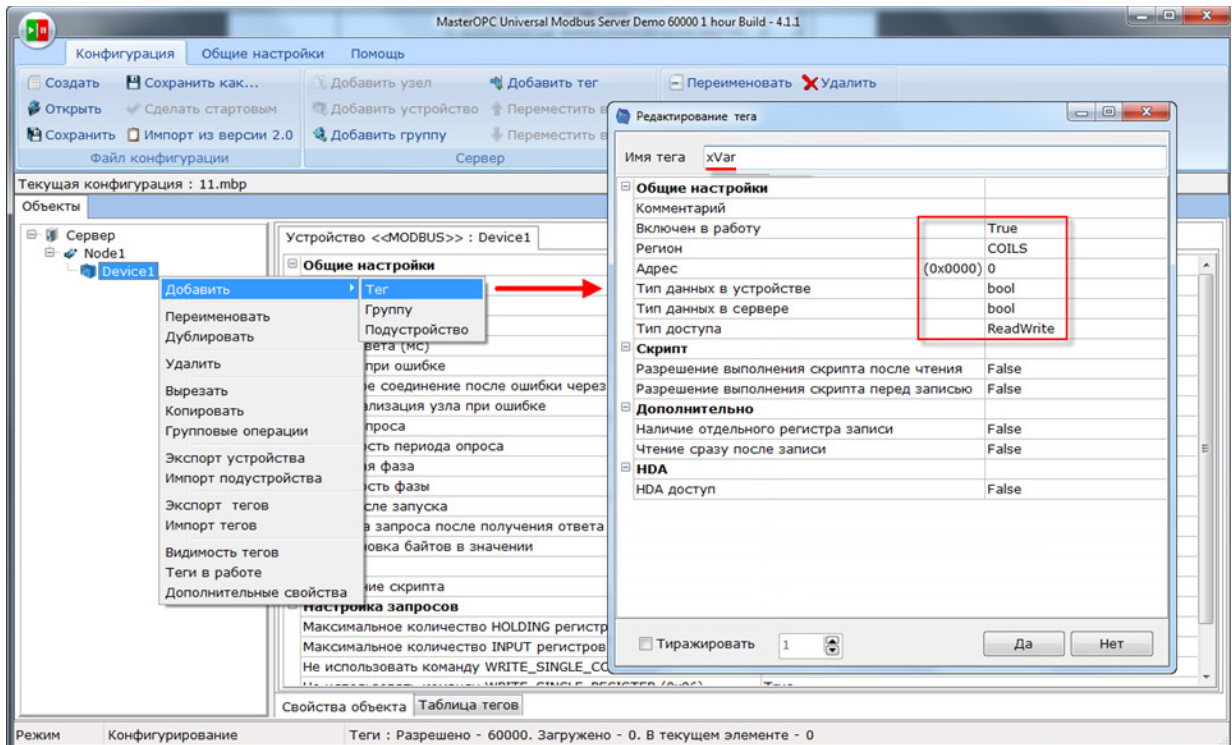


Рисунок 5.9.39 – Добавление тега xVar

Тег <<HOLDING_REGISTERS>> : wVar	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Адрес (0x0001)	1
Тип данных в устройстве	uint16
Тип данных в сервере	uint32
Тип доступа	ReadWrite
Использовать перестановку байтов устройства	True
Последний тег в групповом запросе	False
Пересчет (A*X + B)	False

Рисунок 5.9.40 – Добавление тега wVar

Тег <<HOLDING_REGISTERS>> : rVar	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Адрес (0x0002)	2
Тип данных в устройстве	float
Тип данных в сервере	float
Тип доступа	ReadWrite
Использовать перестановку байтов устройства	True
Последний тег в групповом запросе	False
Пересчет (A*X + B)	False

Рисунок 5.9.41 – Добавление тега rVar

Тег <<HOLDING_REGISTERS>> : sVar	
<b>Общие настройки</b>	
Комментарий	
Включен в работу	True
Адрес (0x0004)	4
Тип данных в устройстве	string
Тип данных в сервере	string
Количество байт для строкового типа	16
Тип строки для строкового типа	ascii
Тип доступа	ReadWrite
Использовать перестановку байтов устройства	True
Последний тег в групповом запросе	False
Пересчет (A*X + B)	False

Рисунок 5.9.42 – Добавление тега sVar

11. Загрузить проект в контроллер и запустить его. Запустить OPC-сервер для контроля значений переменных.

В OPC-сервере следует изменить значения тегов и наблюдать соответствующие значения в CODESYS. В CODESYS следует изменить значения **\_Plc** переменных и сгенерировать импульс в переменной **xWrite** для записи значений в регистры slave'a. Записанные значения будут прочитаны OPC-сервером.

Device.Application.PLC_PRG_ST			
Выражение	Тип	Значение	
fbUnpackWord	UTIL.WORD_AS_BIT		
fbPackWord	UTIL.BIT_AS_WORD		
fbRealToWorld2	OCL.REAL_TO_WO...		
fbModbusTcpSlave	OCL.MB_TcpSlave		
awSlaveData	ARRAY [0..15] OF ...		
xVar_Opc	BOOL	TRUE	
wVar_Opc	WORD	11	
rVar_Opc	REAL	22.33	
sVar_Opc	STRING(15)	'привет'	
xVar_Plc	BOOL	TRUE	
wVar_Plc	WORD	11	
rVar_Plc	REAL	22.33	
sVar_Plc	STRING(15)	'привет'	
xWrite	BOOL	TRUE	
fbWriteEdge	R_TRIG		

Объекты							
Устройство <<Device1>>							
Теги							
Имя	Регион	Адрес	Значение	Качество	Время (UTC)	Тип в сер...	
Node1.Device1.xVar	COILS	(0x00...	True	GOOD	2019-08-1...	bool	
Node1.Device1.wVar	HOL...	(0x00...	11	GOOD	2019-08-1...	uint32	
Node1.Device1.rVar	HOL...	(0x00...	22.330000	GOOD	2019-08-1...	float	
Node1.Device1.sVar	HOL...	(0x00...	привет	GOOD	2019-08-1...	string	

Рисунок 5.9.43 – Считывание и запись данных через OPC-сервер

### 5.9.5 СПК1хх [M01] (Modbus TCP Slave) – чтение файлов с помощью 20 функции Modbus

Функциональный блок [MB\\_TcpSlave](#) поддерживает 20 функцию Modbus (**Read File Record**), что позволяет считывать файлы контроллера. Устройство, которое является Modbus TCP Master'ом, должно также поддерживать эту функцию.

В качестве примера будет рассмотрено считывание файлов с контроллера OPC-сервером [Insat MasterOPC Universal Modbus Server](#), который поддерживает 20 функцию Modbus. Считанные из файлов значения могут быть переданы в SCADA-систему с помощью технологии **OPC HDA**.

Данный OPC-сервер включает скрипт, который позволяет считывать только файлы определенного формата (формата архива, поддержанного в контроллерах ОВЕН ПЛК1хх, режим ASCII). Описание этого формата доступно по [ссылке](#). Пользователь может отредактировать скрипт для поддержки другого формата. Скрипт имеет название **OwenPicHDA.lua** и расположен в директории **C:\ProgramData\InSAT\MasterOPC Universal Modbus Server\MODULES**. Описание принципов работы со скриптами OPC-сервера доступно по [ссылке](#).

В рамках примера считывается файл **Arc1.log** формата архива ПЛК1хх. Этот файл включен в состав файла примера. Файл включает в себя архив трех переменных с типами **UINT**, **UDINT** и **REAL** соответственно.

Структурная схема примера приведена на рисунке ниже:

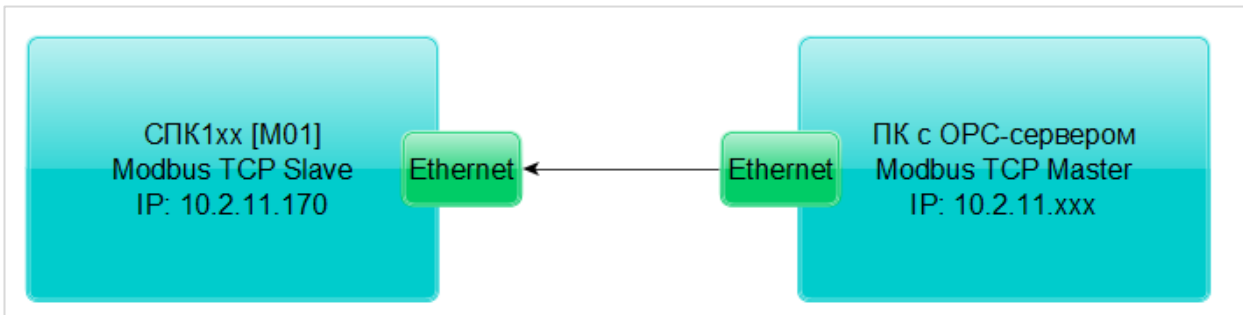


Рисунок 5.9.44 – Структурная схема примера

Пример создан в среде **CODESYS V3.5 SP11 Patch 5** и подразумевает запуск на **СПК1хх [M01]** с таргет-файлом **3.5.11.x**. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (**ПКМ** на узел **Device** – **Обновить устройство**).

Пример доступен для скачивания: [Example\\_OwenCommunicationModbusTcpSlaveFiles\\_3511v1.zip](#)

Сетевые параметры модулей приведены в таблице ниже:

Таблица 5.9.7 – Сетевые параметры устройств

Параметр	СПК1хх [M01]	ПК с OPC-сервером
IP-адрес	10.2.11.170	<i>любой адрес из сети, к которой подключен контроллер</i>
Порт	502	
Адрес (Unit ID)	1	-
Режим работы	slave	master

Для настройки обмена следует:

1. Подключить контроллер и ПК к общей локальной сети.
2. Убедиться, что в контроллере по пути `/mnt/ufs/home/root` находится файл **Arc1.log**. В рамках примера этот файл может быть загружен в контроллер вручную через утилиту [WinSCP](#) (см. руководство **CODESYS V3.5. FAQ**, п. 13.5).
3. Создать новый проект **CODESYS** с программой на языке **ST** или **CFC**:

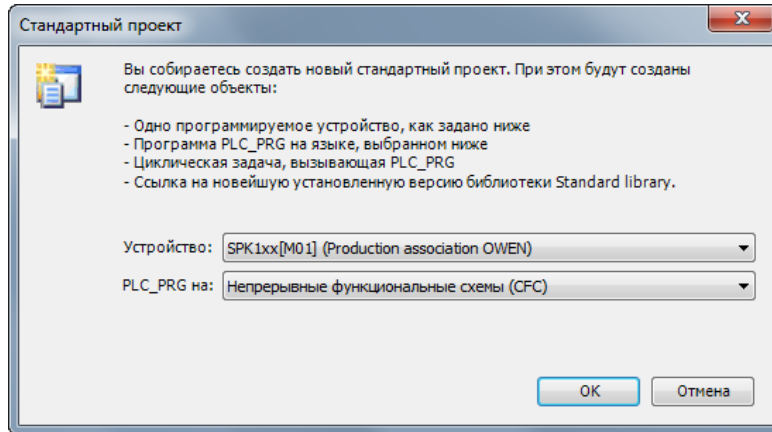


Рисунок 5.9.45 – Создание проекта CODESYS



#### ПРИМЕЧАНИЕ

Проект примера содержит программы на языках CFC и ST. По умолчанию используется программа на CFC (**PLC\_PRG\_CFC**). Для работы с программой на ST следует в конфигурации задач для задачи **MainTask** удалить вызов **PLC\_PRG\_CFC** и добавить вызов **PLC\_PRG\_ST**.

4. Объявить в программе следующие переменные:

```

1  PROGRAM PLC_PRG_CFC
2  VAR
3
4      fbModbusTcpSlave:    OCL.MB_TcpSlave;    // фБ для реализации Modbus Slave
5      awSlaveData:        ARRAY [0..15] OF WORD; // буфер данных Modbus Slave
6
7      // путь к файлам архивов
8      asFilePath:        ARRAY [1..8] OF STRING := ['/mnt/ufs/home/root/Arc1.log', 7('')];
9  END VAR

```

Рисунок 5.9.46 – Объявление переменных программы

5. Код программы будет выглядеть следующим образом:

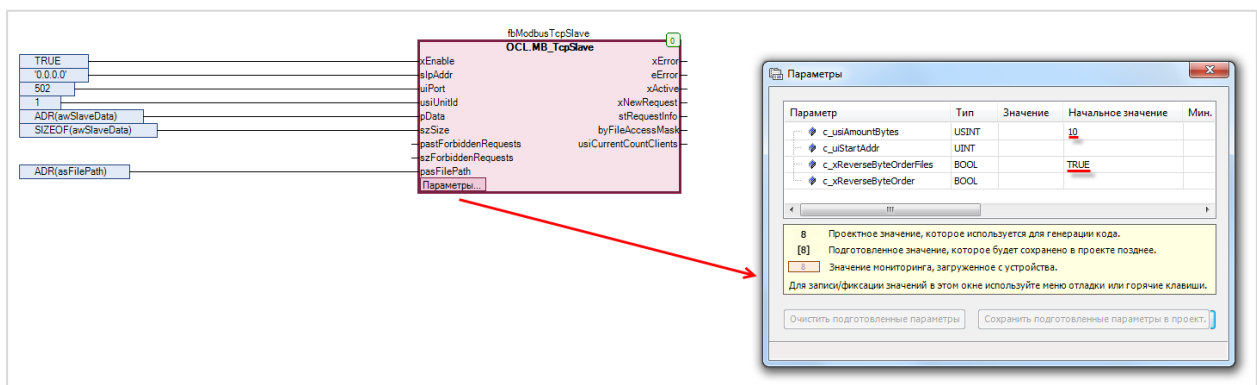


Рисунок 5.9.47 – Код программы на языке CFC



```

7   fbModbusTcpSlave
8   (
9       xEnable           := TRUE,
10      sIpAddr           := '0.0.0.0',
11      uiPort            := 502,
12      usiUnitId        := 1,
13      pData             := ADR(awSlaveData),
14      szSize            := SIZEOF(awSlaveData),
15      pasFilePath       := ADR(asFilePath),
16      c_xReverseByteOrderFiles := TRUE,
17      c_usiAmountBytes  := 10
18  );

```

Рисунок 5.9.48 – Код программы на языке ST

При запуске программы происходит вызов экземпляра ФБ **MB TcpSlave** (блок 0), который выполняет функцию Modbus Slave. В качестве IP-адреса (**sIpAddr**), на котором используется slave, применяется специальный адрес **0.0.0.0**, который соответствует адресам всех интерфейсов контроллера (то есть slave доступен по всем TCP/IP интерфейсам). Значения входов **uiPort** и **usiUnitId** соответствуют [таблице 5.9.7](#).

Данные slave'a хранятся в массиве **awSlaveData**. В экземпляр ФБ передается указатель на этот массив (**pData**) и его размер в байтах (**szSize**).

Пути к файлам архивов хранятся в массиве строк **asFilePath**. В блок передается указатель на эту переменную. ФБ поддерживает до 8 файлов. Нумерация файлов ведется с 1. В рамках примера используется только один (первый) файл.

Для работы с [Insat MasterOPC Universal Modbus Server](#) параметр **c\_xReverseByteOrderFiles** (порядок байт при передаче файлов) должен иметь значение **TRUE**, а параметр **c\_usiAmountBytes** (размер записи в файле) – значение **10**.

6. Установить и запустить [MasterOPC Universal Modbus Server](#).

7. Нажать **ПКМ** на узел **Server** и добавить коммуникационный узел типа **TCP/IP**. В узле следует указать сетевые настройки в соответствии с [таблицей 5.9.7](#). Для работы OPC-сервера в режиме **Modbus TCP Master** параметры **Modbus поверх TCP** и **Slave подключение** должны иметь значение **FALSE**.

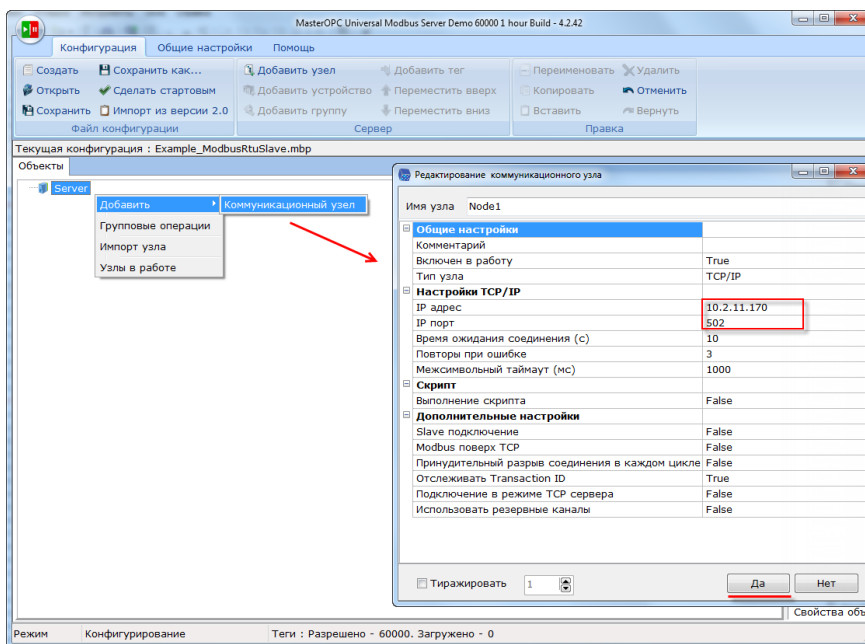


Рисунок 5.9.49 – Добавление коммуникационного узла

8. Нажать ПКМ на коммуникационный узел и добавить устройство с настройками по умолчанию.

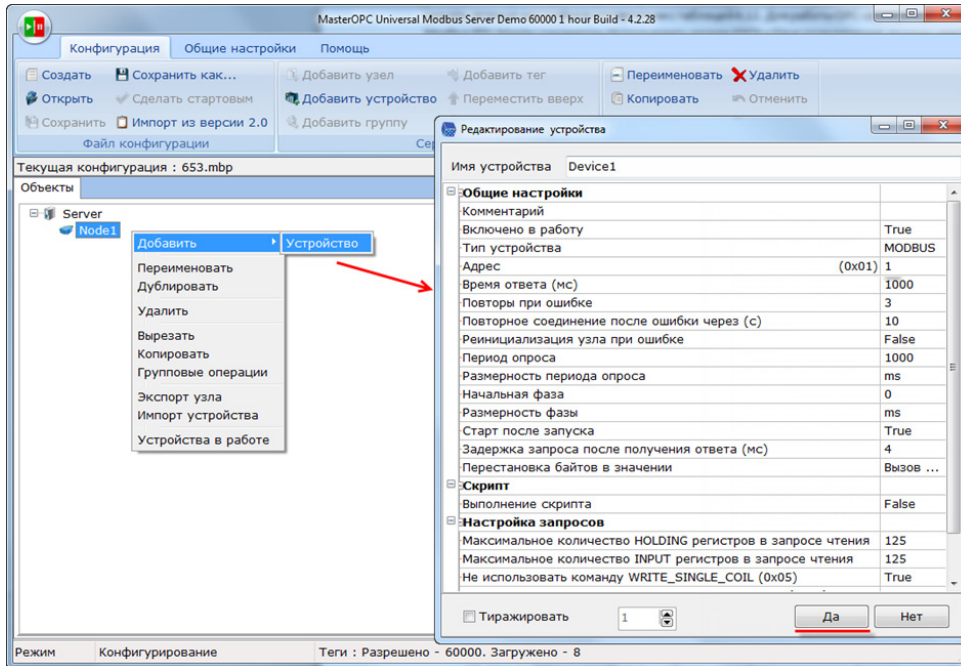


Рисунок 5.9.50 – Добавление устройства

9. Нажать ПКМ на устройство и импортировать подустройство Owen History HDA.ssd (по умолчанию оно находится в директории C:\ProgramData\InSAT\MasterOPC Universal Modbus Server\SERVEREXPORT\SUBDEVICE\_LIBRARY\OWEN).

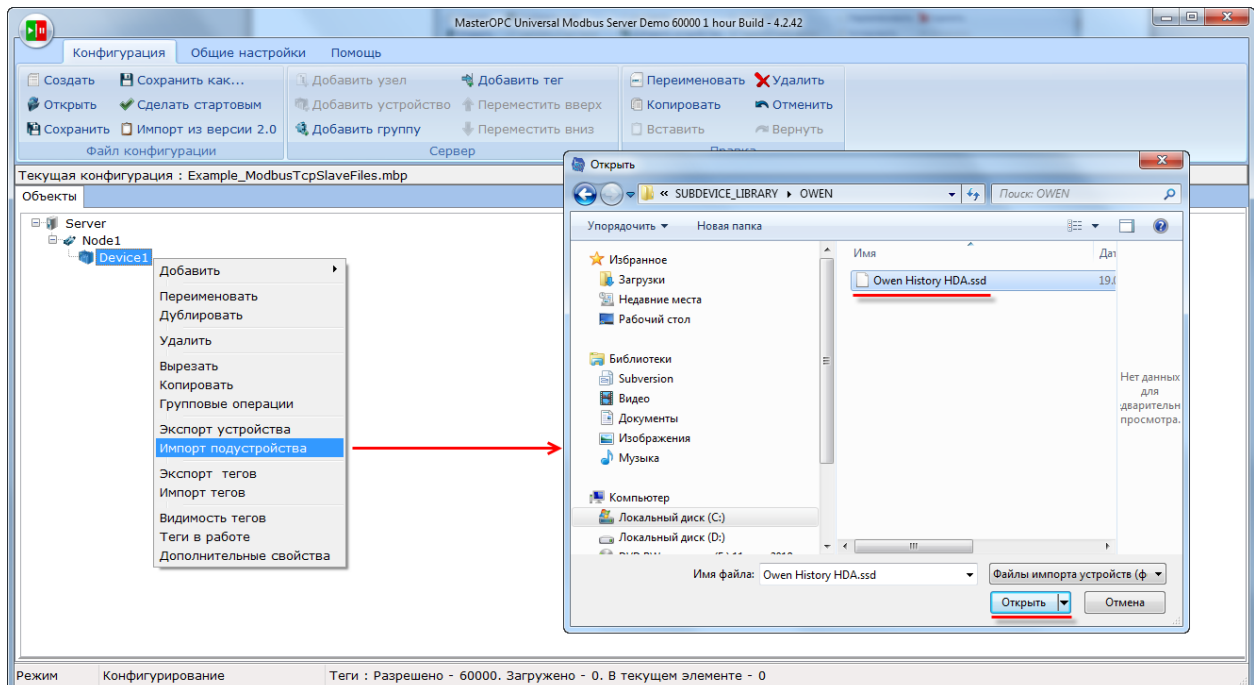


Рисунок 5.9.51 – Импорт подустройства

10. В появившемся подустройстве **Архив** указать номер файла. Нумерация файлов ФБ ведется с 1, и в примере используется только один файл – поэтому следует установить значение 1. В случае необходимости можно изменить другие настройки подустройства.

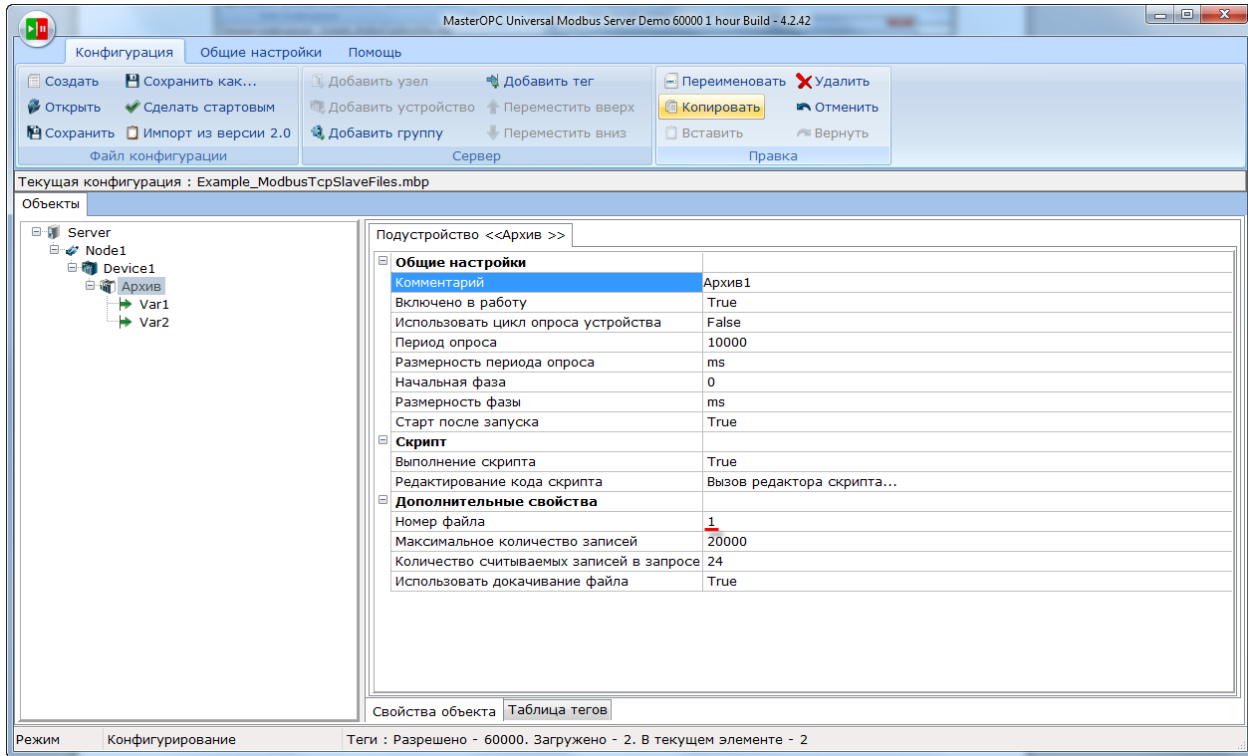


Рисунок 5.9.52 – Настройки подустройства

11. В рамках примера считывается файл **Arc1.log** формата архива ПЛК1xx. Этот файл включен в состав проекта примера. Файл содержит архив трех переменных с типами **UINT**, **UDINT** и **REAL** соответственно.

По умолчанию подустройство **Архив** содержит только два тега – типа **Uint32** (UDINT) и типа **Float** (REAL). Следует нажать **ПКМ** на подустройство и добавить тег с названием **Var0** (используется тип **Uint32**, так как в OPC-сервере нет типа **Uint16**) :

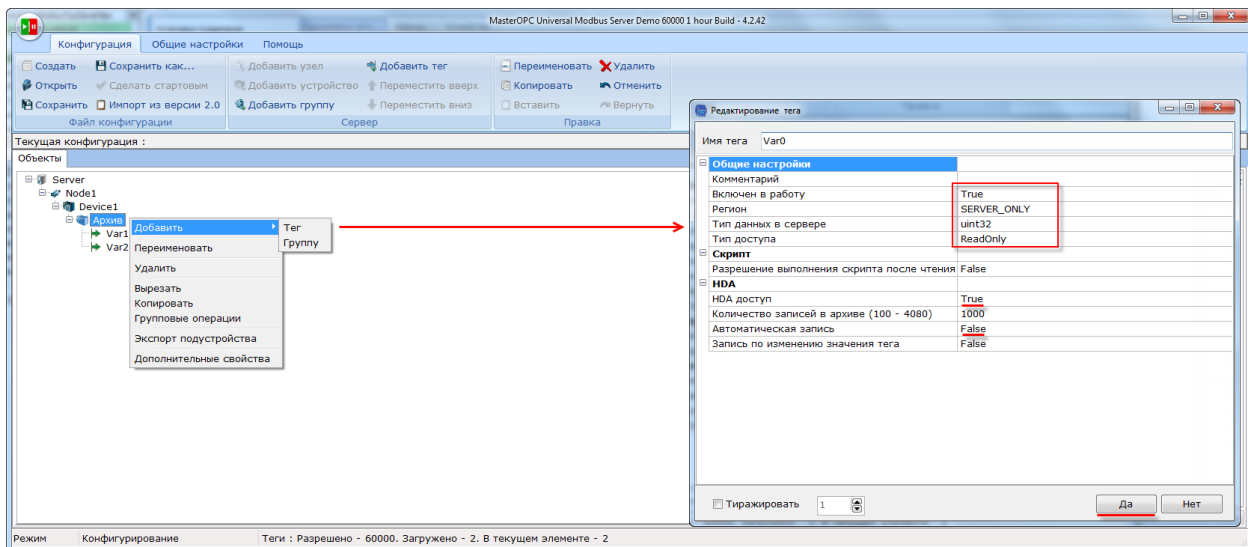


Рисунок 5.9.53 – Добавление тега в подустройство

12. Переместить тег **Var0** в дереве OPC-сервера с помощью кнопки **Переместить вверх**:

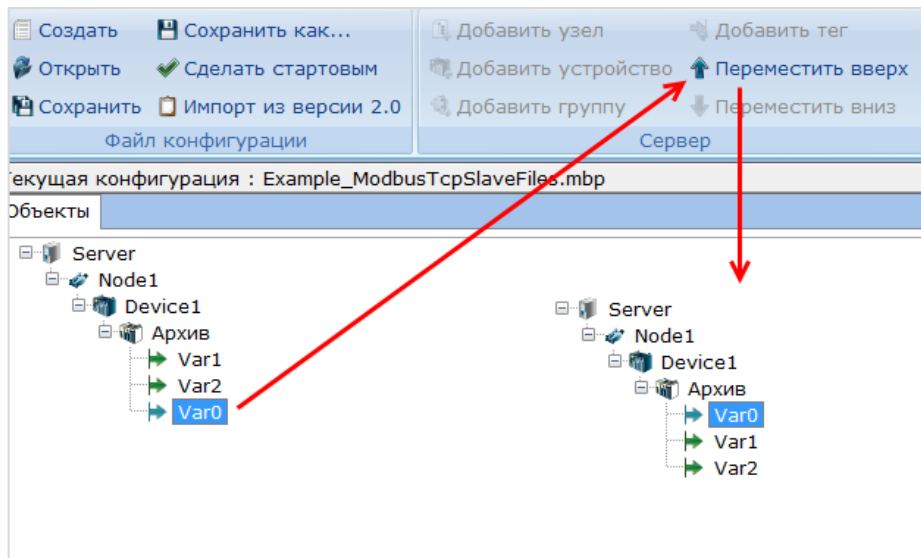


Рисунок 5.9.54 – Перемещение тегов в дереве OPC-сервера

13. Загрузить проект в контроллер и запустить его. Запустить OPC-сервер для контроля значений переменных. В тегах будут отображены последние значения, считанные из файла архива.

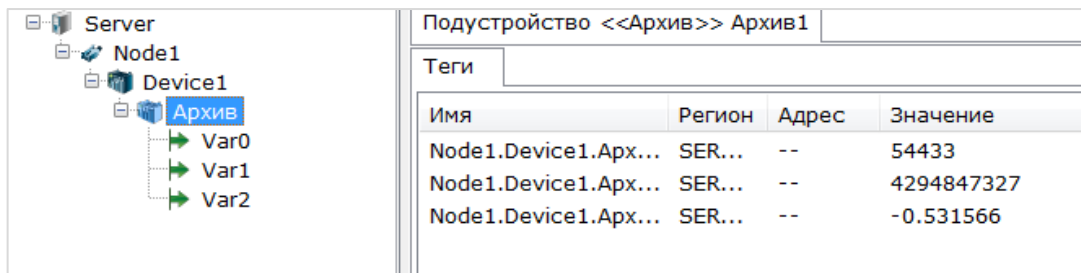


Рисунок 5.9.55 – Считывание и запись данных через OPC-сервер

5. Библиотека OwenCommunication

Для просмотра истории тега следует выделить его в дереве проекта и перейти на вкладку **HDA** (в OPC формат отображения **DEC**, порядок – от новых записей к старым. В файле архива формат отображения **HEX**, порядок – от старых записей к новым):

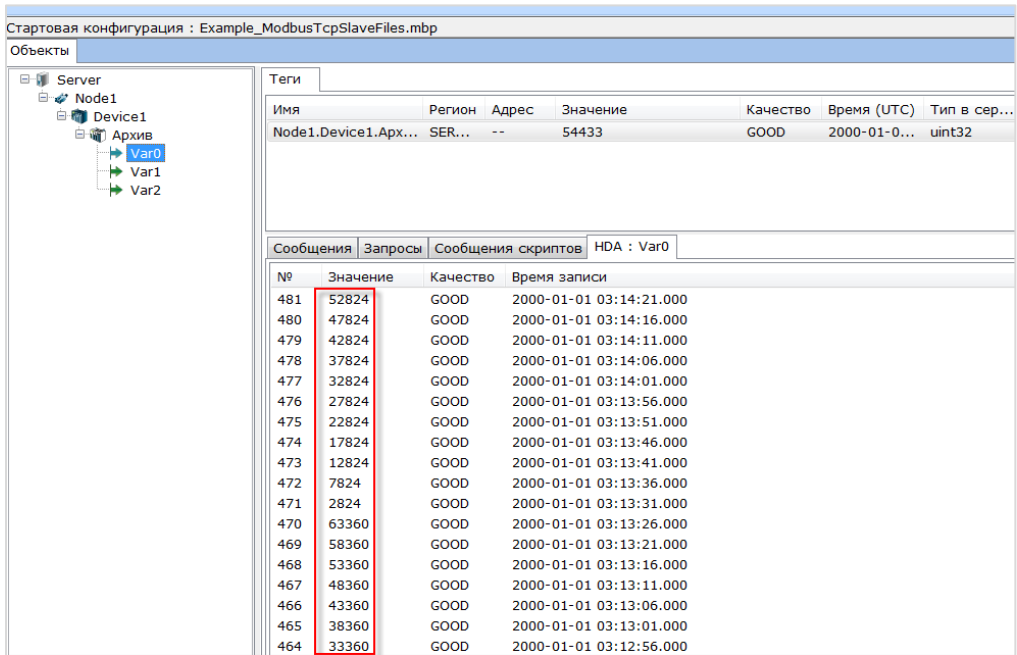


Рисунок 5.9.56 – Считывание архивных данных через OPC-сервер

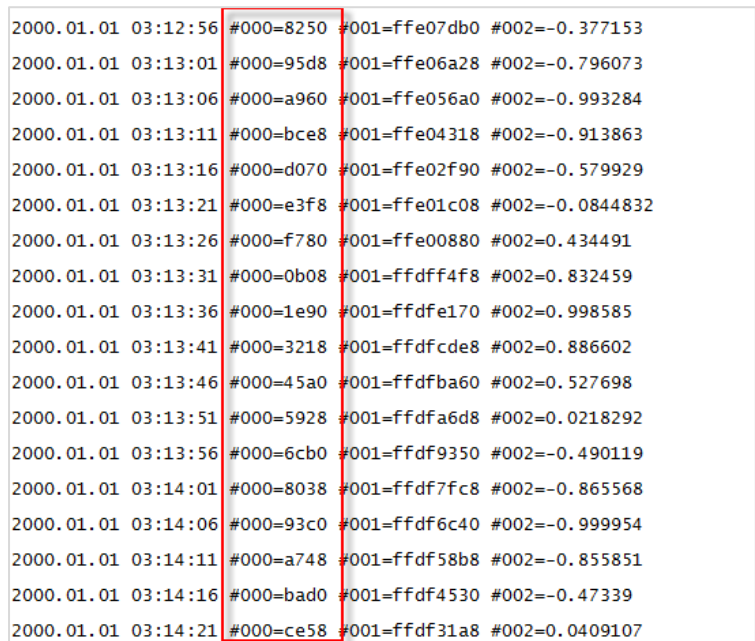


Рисунок 5.9.57 – Содержимое файла архива

Считанные исторические данные могут быть переданы в SCADA-систему с помощью технологии **OPC HDA**.

## 6 FAQ

### 6.1 Что делать, если не удастся наладить обмен по Modbus?

В случае возникновения проблем при настройке обмена рекомендуется:

1. Проверить (прозвонить) линию связи. Проверить распиновку кабеля (в случае использования кабелей с разъемом DB9). Проверить, что контакты **A** и **B** (или **RXD** и **TXD**) не перепутаны местами.
2. Проверить [используемый номер COM-порта](#) в **CODESYS**.
3. Проверить соответствие сетевых настроек контроллера и подключаемых приборов (скорость обмена, количество стоп-бит, адреса slave-устройств и т. д.).
4. Проверить [настройки опроса регистров](#): используемые коды функций, адреса регистров, типы данных и т. д.
5. Проверить, нет ли разрывов в карте регистров slave-устройства (в случае использования групповых запросов).
6. Проверить, что на шине находится только одно master-устройство (для **Modbus RTU/ASCII**).
7. Проверить, что в сети нет slave-устройств с одинаковыми адресами.
8. В случае опроса модулей **Mx110** – с помощью программы [Конфигуратор Mx110](#) проверить, что модулям заданы корректные сетевые настройки.
9. В случае использования [стандартных средств конфигурирования](#) – проверить, что на вкладке привязки переменных для параметра **Всегда обновлять переменные** установлено значение **Вкл. 2 (Всегда в задаче цикла шины)**.

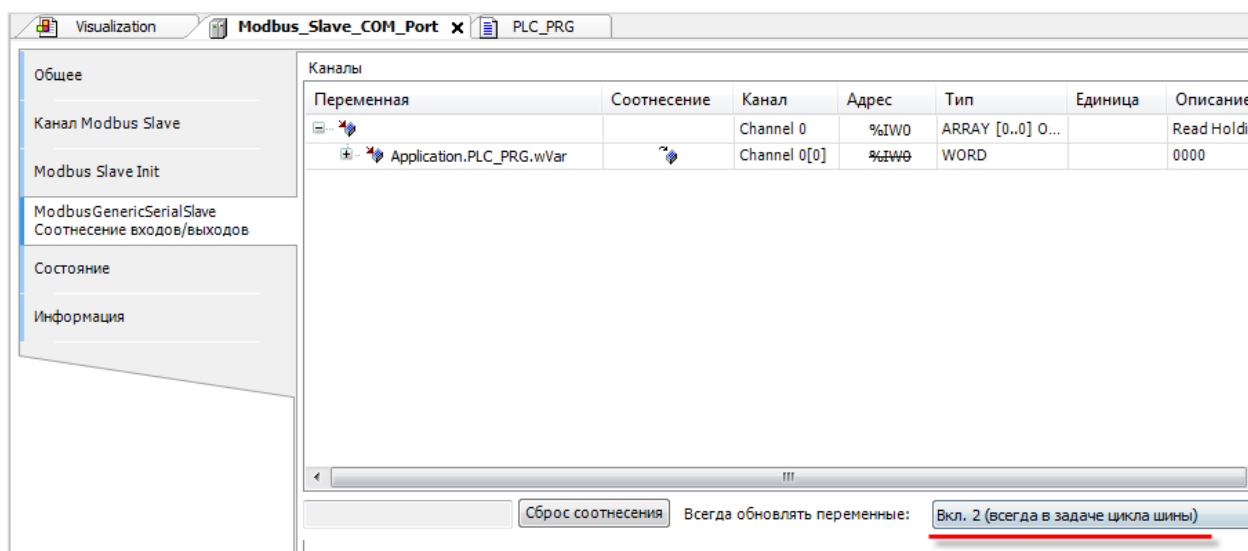


Рисунок 6.1 – Настройка параметра Всегда обновлять переменные

Если успешное выполнение всех вышеперечисленных пунктов не приведет к настройке обмена, то следует обратиться в [техническую поддержку компании Овен](#). При обращении следует предоставить указанную ниже информацию и материалы:

- модель и модификацию контроллера, версию прошивки, таргет-файла, среды CODESYS и используемых компонентов Modbus;
- **подробное** описание проблемы;
- структурную схему сети с указанием используемых портов и адресов;

- маркировку используемых кабелей, информацию об изоляции и заземлении, наличии согласующих резисторов (терминаторов) и повторителей, а также условиях, в которых находятся приборы (например, о присутствии в шкафу автоматики силового оборудования);
- архивы проектов для программируемого оборудования Овен (СПК, ПЛК, панели оператора и т. д.), скриншоты сетевых настроек конфигурируемых приборов (модули Mx110, ТРМ и т. д.) и приборов других производителей (а также карты регистров этих устройств).

### 6.2 Каким образом считать/передать значение с плавающей точкой (REAL)?

См. [п. 4.8](#) (перечисления, указатели) и [п. 5.8](#) (функции библиотеки **OwenCommunication**).

### 6.3 Каким образом считать/передать отрицательное значение (INT)?

Если необходимо считать отрицательное число, то после получения соответствующей переменной типа **WORD** следует преобразовать ее в **INT** с помощью стандартного оператора **WORD\_TO\_INT**.

Если необходимо записать значение типа **INT**, то следует преобразовать его в **WORD** с помощью стандартного оператора **INT\_TO\_WORD**. На устройстве, которое получит эти данные, необходимо будет произвести обратную операцию.

## 6.4 Вопросы по стандартным средствам конфигурирования

### 6.4.1 Какие версии компонентов рекомендуются к использованию?

Таблица рекомендуемых версий компонентов Modbus приведена в [приложении А](#).

Узнать используемую версию компонента можно на вкладке **Информация**:

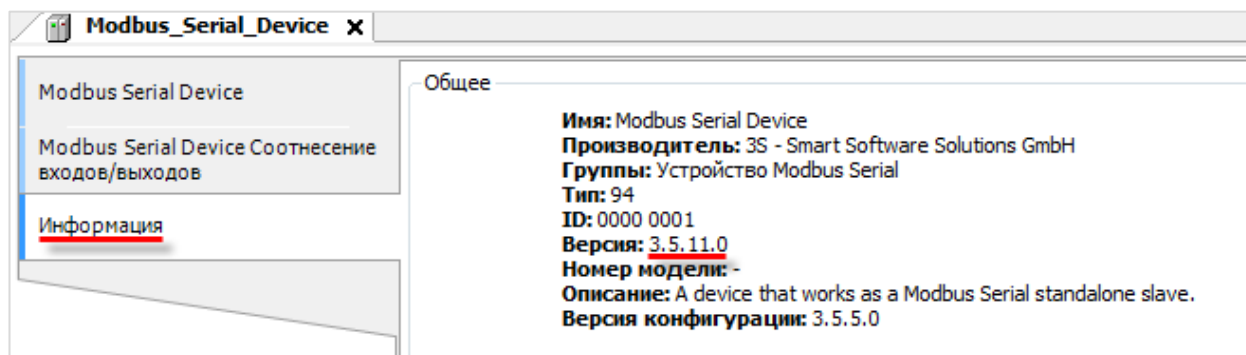


Рисунок 6.2 – Версия компонента Modbus Slave Serial Device

Чтобы изменить версию компонента следует нажать на нем **ПКМ** и выбрать команду **Обновить устройство**.

### 6.4.2 Modbus Serial Master: как реализовать чтение/запись по триггеру?

В настройках канала для параметра **Триггер** следует выбрать значение **Передний фронт**.

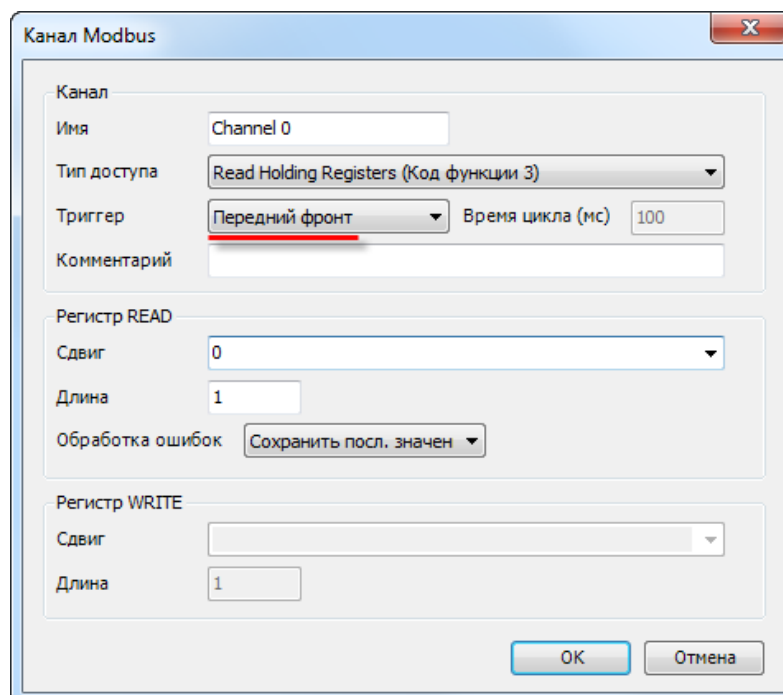


Рисунок 6.3 – Настройка опроса канала по триггеру

После этого на вкладке привязки переменных к каналу появится строка для триггерной переменной. Чтение/запись будет происходить по переднему фронту этой переменной.

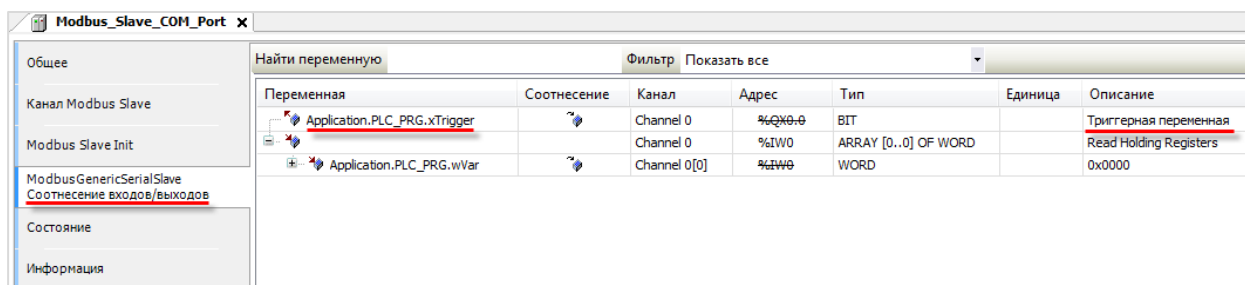


Рисунок 6.4 – Привязка триггерной переменной



### 6.4.3 Modbus RTU Slave: почему принятые значения сбрасываются в 0?

Если контроллер используется в режиме **Modbus RTU Slave**, а мастером является устройство, производящее запись по триггеру (например, панель оператора записывает введенное значение однократно, а не циклически), то следует отключить галочку **Сторожевой таймер**. В противном случае регистры slave'a будут обнуляться, если в течение заданного времени не будет получено ни одного запроса от мастера.

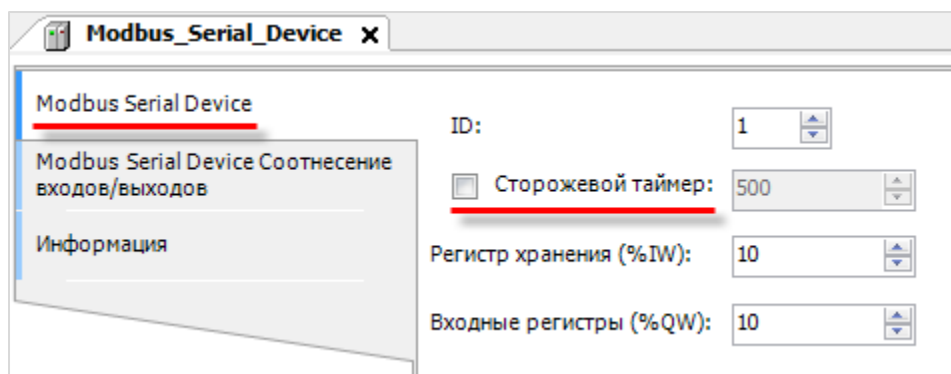


Рисунок 6.5 – Настройки компонента Modbus Serial Device

### 6.4.4 Можно ли менять данные holding регистров из программы?

Нет, стандартные средства конфигурирования **CODESYS** в явном виде запрещают запись значений в **holding** регистры из программы контроллера. Если это необходимо, то следует использовать ФБ [MB\\_SerialSlave](#) или [MB\\_TcpSlave](#) из библиотеки [OwenCommunication](#).

### 6.4.5 Как произвести диагностику обмена в программе?

См. [п. 4.6](#). В случае использования [шаблонов модулей Mx110 и 210](#) см. [п. 3.4](#).

### 6.4.6 Как расшифровываются пиктограммы статуса обмена?

Расшифровка пиктограмм статуса обмена приведена в таблице 6.1:

Таблица 6.1 – Расшифровка пиктограмм статуса обмена

Пиктограмма	Описание для компонентов Modbus Serial Master / Modbus TCP Master	Описание для остальных компонентов
	На запрос получен корректный ответ	Компонент работает корректно
	Отсутствует лицензия на компонент	
	Ожидание соединения	-
	На запрос получен ответ с кодом ошибки Modbus	Компонент работает некорректно (например, IP-адрес в компоненте Ethernet отличается от реального IP-адреса контроллера)
	Ответ не получен	Ошибка инициализации компонента (например, не удалось выделить память)

## 6.5 Вопросы по библиотеке OwenCommunication

### 6.5.1 В примерах работы с библиотекой используются действия. Как добавить их в проект?

Для добавления действия следует нажать **ПКМ** на нужный компонент (например, программу) и выбрать команду **Добавление объекта – Действие**.

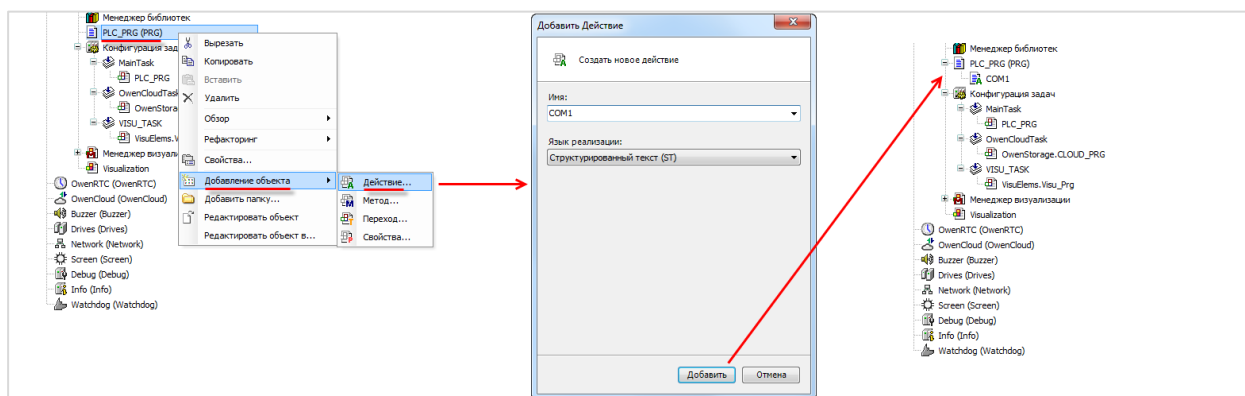


Рисунок 6.6 – Добавление действия в проект CODESYS

### 6.5.2 Позволяет ли библиотека организовать опрос с более высокой частотой по сравнению со стандартными средствами конфигурирования?

Нет. ФБ библиотеки работают асинхронно и построены на асинхронных блоках обмена из системных библиотек. Преимуществами библиотеки **OwenCommunication** являются расширенный функционал и возможность управления обменом из кода программы, но в большинстве случаев частота опроса будет ниже по сравнению со [стандартными средствами конфигурирования](#).

## Приложение А. Рекомендуемые версии компонентов Modbus

Различные версии **CODESYS** включают в себя разные версии Modbus-компонентов. В таблице ниже приведены рекомендуемые версии компонентов в зависимости от используемой в контроллере системы исполнения. Информация о соответствии версий прошивок контроллеров, таргет-файлов и CODESYS приведена в документе **CODESYS V3.5. FAQ**.

**Таблица А.1 – Рекомендуемые версии компонентов Modbus**

Система исполнения	<b>3.5.14.3</b>	<b>3.5.11.5</b>	<b>3.5.4.3</b>
Среда программирования	V3.5 SP14 Patch 3	V3.5 SP11 Patch 5 HF 4	V3.5 SP5 Patch 5
Modbus COM	3.5.11.20	3.5.11.20	3.4.0.0
Modbus RTU Master	3.5.14.0	3.5.11.20	3.5.5.0
Modbus RTU Slave	3.5.10.30	3.5.10.30	3.5.4.0
Шаблоны Mx110	3.5.11.x	3.5.11.x	3.5.4.13
Modbus RTU Serial Device	3.5.14.0	3.5.11.0	3.5.5.0
Ethernet-адаптер	3.5.14.0	3.5.11.0	3.4.2.0
Modbus TCP Master	3.5.14.0	3.5.11.30	3.5.5.0
Modbus TCP Slave	3.5.12.0	3.5.11.30	3.5.4.0
Шаблоны Mx210	3.5.11.x	3.5.11.x	-
Modbus TCP Device	3.5.14.0	3.5.11.0	3.5.2.0

## Приложение Б. Листинги примеров

### Б1 Листинг примера из п. 5.9.1

#### Б.1.1 Код программы PLC\_PRG\_ST

```
PROGRAM PLC_PRG_ST
VAR
    fbComControl1: OCL.COM_Control; // ФБ управления портом COM1
    fbComControl2: OCL.COM_Control; // ФБ управления портом COM2
    fbMV110_8A_AI1: OCL.MB_SerialRequest; // ФБ опроса модуля MB110-8A
    fbMV110_16D_DI: OCL.MB_SerialRequest; // ФБ опроса модуля MB110-16Д
    fbMU110_16R_DO: OCL.MB_SerialRequest; // ФБ опроса модуля МУ110-16Р

    rAI1: REAL; // значение 1-го входа модуля MB110-8A
    wDiMask: WORD; // битовая маска входов модуля MB110-16Д
    wDoMask: WORD; // битовая маска выходов модуля МУ110-16Р
    xDi0: BOOL; // значение 1-го входа модуля MB110-16Д
    xDo0: BOOL; // значение 1-го выхода модуля МУ110-16Р

    awAI1: ARRAY [0..1] OF WORD; // регистры, считанные с модуля MB110-8A

    iStateCom1: INT; // шаг опроса по порту COM1
    iStateCom2: INT; // шаг опроса по порту COM2
END_VAR

// чтобы запустить пример на ST на контроллере требуется:
// 1. Удалить из задачи MainTask вызов программы PLC_PRG_CFC
// 2. Привязать к задаче MainTask вызов программы PLC_PRG_ST

COM1();
COM2();

xDo0 := xDi0 AND (rAI1 > 30.0);
```

**Б.1.2 Код действия COM1**

## CASE iStateCom1 OF

0: // открытие COM-порта COM1

```
fbComControl1
(
    xEnable := TRUE,
    usiComPort := 1,
    udiBaudrate := 115200,
    udiByteSize := 8,
    eParity := OCL.COM_PARITY.NONE,
    eStopBit := OCL.COM_STOPBIT.ONE
);
```

```
IF fbComControl1.xDone THEN
    iStateCom1 := 1;
END_IF
```

1: // опрос MB110-8A

```
fbMV110_8A_AI1
(
    xExecute           := fbComControl1.xActive,
    tTimeout           := T#200MS,
    usiRetry            := 3,
    hCom               := fbComControl1.hCom,
    xlsAsciiMode       := FALSE,
    usiSlaveId         := 1,
    eFuncCode          := OCL.MB_FC.READ_HOLDING_REGISTERS,
    uiDataAddr         := 4,
    uiDataCount        := 2,
    pData              := ADR(awAI1),
    szSize             := SIZEOF(awAI1)
);
```

```
IF fbMV110_8A_AI1.xDone OR fbMV110_8A_AI1.xError THEN
    // после выполнения блока его надо сбросить
    fbMV110_8A_AI1(xExecute := FALSE);
    rAI1 := OCL.WORD2_TO_REAL(awAI1[0], awAI1[1], FALSE);
    iStateCom1 := 2;
END_IF
```

2: // здесь можно добавить опрос следующего устройства  
 // после опроса последнего устройства возвращаемся к опросу первого  
 iStateCom1 := 1;

END\_CASE

**Б.1.3 Код действия COM2**

CASE iStateCom2 OF

0: // открытие COM-порта COM2

```

fbComControl2
(
    xEnable := TRUE,
    usiComPort := 2,
    udiBaudrate := 115200,
    udiByteSize := 8,
    eParity := OCL.COM_PARITY.NONE,
    eStopBit := OCL.COM_STOPBIT.ONE
);

```

IF fbComControl2.xDone THEN

iStateCom2 := 1;

END\_IF

1: // опрос модуля MB110-16Д

```

fbMV110_16D_DI
(
    xExecute := fbComControl2.xActive,
    tTimeout := T#200MS,
    usiRetry := 3,
    hCom := fbComControl2.hCom,
    xIsAsciiMode := FALSE,
    usiSlaveId := 1,
    eFuncCode := OCL.MB_FC.READ_HOLDING_REGISTERS,
    uiDataAddr := 51,
    uiDataCount := 1,
    pData := ADR(wDiMask),
    szSize := SIZEOF(wDiMask)
);

```

IF fbMV110\_16D\_DI.xDone OR fbMV110\_16D\_DI.xError THEN

// после выполнения блока его надо сбросить

fbMV110\_16D\_DI(xExecute := FALSE);

xDi0 := wDiMask.0;

iStateCom2 := 2;

END\_IF

2: // опрос модуля МУ110-16Р

```
wDoMask.0 := xDo0;
```

```
fbMU110_16R_DO
```

```
(  
    xExecute           := fbComControl2.xActive,  
    tTimeout           := T#200MS,  
    usiRetry           := 3,  
    hCom               := fbComControl2.hCom,  
    xlsAsciiMode       := FALSE ,  
    usiSlaveId         := 17,  
    eFuncCode          := OCL.MB_FC.WRITE_MULTIPLE_REGISTERS,  
    uiDataAddr         := 50,  
    uiDataCount        := 1,  
    pData              := ADR(wDoMask),  
    szSize              := SIZEOF(wDoMask)  
);
```

```
IF fbMU110_16R_DO.xDone OR fbMU110_16R_DO.xError THEN
```

```
    // после выполнения блока его надо сбросить
```

```
    fbMU110_16R_DO(xExecute := FALSE);
```

```
    // возвращаемся к опросу первого модуля
```

```
    iStateCom2 := 1;
```

```
END_IF
```

```
END_CASE
```

**Б2 Листинг примера из п. 5.9.2**

```

PROGRAM PLC_PRG_ST
VAR
    fbComControl1: OCL.COM_Control;           // ФБ управления портом COM1

    fbRealToWorld2: OCL.REAL_TO_WORD2;       // ФБ преобразования REAL в две...
                                                // ...переменные типа WORD

    fbModbusSerialSlave: OCL.MB_SerialSlave;  // ФБ для реализации Modbus Slave

    awSlaveData: ARRAY [0..15] OF WORD; // буфер данных Modbus Slave

    (* значения, полученные от OPC *)
    xVar_Opc: BOOL;
    wVar_Opc: WORD;
    rVar_Opc: REAL;
    sVar_Opc: STRING(15);

    (* значения для передачи в OPC *)
    xVar_Plc: BOOL;
    wVar_Plc: WORD;
    rVar_Plc: REAL;
    sVar_Plc: STRING(15);

    xWrite: BOOL; // команда записи данных из программы в регистры Modbus Slave
    fbWriteEdge: R_TRIG; // триггер для однократной записи
END_VAR

// чтобы запустить пример на ST на контроллере требуется:
// 1. Удалить из задачи MainTask вызов программа PLC_PRG_CFC
// 2. Привязать к задаче MainTask вызов программы PLC_PRG_ST

fbComControl1
(
    xEnable := TRUE,
    usiComPort := 1,
    udiBaudrate := 115200,
    udiByteSize := 8,
    eParity := OCL.COM_PARITY.NONE,
    eStopBit := OCL.COM_STOPBIT.ONE
);

```



fbModbusSerialSlave

```
(  
    xEnable      :=fbComControl1.xActive,  
    hCom        := fbComControl1.hCom,  
    usiSlaveId  := 1,  
    pData       := ADR(awSlaveData),  
    szSize      := SIZEOF(awSlaveData)  
);
```

```
// данные, полученные от OPC
```

```
xVar_Opc := awSlaveData[0].0;
```

```
wVar_Opc := awSlaveData[1];
```

```
rVar_Opc := OCL.WORD2_TO_REAL(awSlaveData[2], awSlaveData[3], FALSE);
```

```
OCL.SWAP_DATA( ADR(awSlaveData[4]), ADR(sVar_Opc), SIZEOF(sVar_Opc), TRUE, FALSE, FALSE );
```

```
// по команде записываем переменные из программы в регистры Modbus Slave
```

```
fbWriteEdge(CLK := xWrite);
```

```
IF fbWriteEdge.Q THEN
```

```
    awSlaveData[0].0 := xVar_Plc;
```

```
    awSlaveData[1] := wVar_Plc;
```

```
    fbRealToWorld2(rInput := rVar_Plc, wOutput1 => awSlaveData[2], wOutput2 => awSlaveData[3]);
```

```
    OCL.SWAP_DATA( ADR(sVar_Plc), ADR(awSlaveData[4]), SIZEOF(sVar_Plc), TRUE, FALSE, FALSE );
```

```
END_IF
```

**Б3 Листинг примера из п. 5.9.3****Б.3.1 Код программы PLC\_PRG\_ST**

```

PROGRAM PLC_PRG_ST
VAR
    fbTcpClientMV210:    OCL.TCP_Client; // ФБ TCP-подключения к модулю MB210-101
    fbTcpClientMK210:    OCL.TCP_Client; // ФБ TCP-подключения к модулю MK210-301
    fbMV210_101_AI1:     OCL.MB_TcpRequest; // ФБ опроса модуля MB210-101
    fbMK210_301_DI:      OCL.MB_TcpRequest; // ФБ опроса входов модуля MK210-301
    fbMK210_301_DO:      OCL.MB_TcpRequest; // ФБ опроса выходов модуля MK210-301

    rAI1:                REAL; // значение 1-го входа модуля MB210-101
    wDiMask:             WORD; // битовая маска входов модуля MK210-301
    wDoMask:             WORD; // битовая маска выходов модуля MK210-301
    xDi0:                BOOL; // значение 1-го входа модуля MK210-301
    xDo0:                BOOL; // значение 1-го выхода модуля MK210-301

    awAI1:              ARRAY [0..1] OF WORD; // регистры, считанные с модуля MB210-101

    iStateMV210:        INT; // шаг опроса модуля MB210-101
    iStateMK210:        INT; // шаг опроса модуля MK210-301
END_VAR

// чтобы запустить пример на ST на контроллере требуется:
// 1. Удалить из задачи MainTask вызов программы PLC_PRG_CFC
// 2. Привязать к задаче MainTask вызов программы PLC_PRG_ST

MV210_101();
MK210_301();

xDo0 := xDi0 AND (rAI1 > 30.0);

```

**Б.3.2 Код действия MV210\_101**

CASE iStateMV210 OF

```

0:      // подключение к модулю

        fbTcpClientMV210
        (
            xEnable := TRUE,
            tTimeout      := T#5S,
            slpAddr       := '10.2.11.181',
            uiPort        := 502
        );

        IF fbTcpClientMV210.xActive THEN
            iStateMV210 := 1;
        END_IF

1:      // опрос MB210-101

        fbMV210_101_AI1
        (
            xExecute           := fbTcpClientMV210.xActive,
            tTimeout           := T#50MS,
            usiRetry           := 3,
            hConnection        := fbTcpClientMV210.hConnection,
            xlsRtuOverTcpMode  := FALSE ,
            usiUnitId          := 1,
            eFuncCode          := OCL.MB_FC.READ_HOLDING_REGISTERS,
            uiDataAddr         := 4000,
            uiDataCount        := 2,
            pData              := ADR(awAI1),
            szSize              := SIZEOF(awAI1)
        );

        IF fbMV210_101_AI1.xDone OR fbMV210_101_AI1.xError THEN
            // после выполнения блока его надо сбросить
            fbMV210_101_AI1(xExecute := FALSE);

            rAI1 := OCL.WORD2_TO_REAL(awAI1[1], awAI1[0], FALSE);

            iStateMV210 := 2;
        END_IF

2:      // здесь можно добавить следующий запрос к модулю
        // после выполнения последнего запроса возвращаемся к первому
        iStateMV210 := 1;

```

END\_CASE

**Б.3.3 Код действия МК210\_301**

## CASE iStateMK210 OF

```

0:    // подключение к модулю

      fbTcpClientMK210
      (
          xEnable := TRUE,
          tTimeout := T#5S,
          slpAddr  := '10.2.11.180',
          uiPort   := 502
      );

      IF fbTcpClientMV210.xActive THEN
          iStateMK210 := 1;
      END_IF

1:    // опрос дискретных входов модуля МК210-301

      fbMK210_301_DI
      (
          xExecute           := fbTcpClientMK210.xActive,
          tTimeout           := T#50MS,
          usiRetry           := 3,
          hConnection       := fbTcpClientMK210.hConnection,
          xIsRtuOverTcpMode := FALSE ,
          usiUnitId         := 1,
          eFuncCode         := OCL.MB_FC.READ_HOLDING_REGISTERS,
          uiDataAddr        := 51,
          uiDataCount       := 1,
          pData             := ADR(wDiMask),
          szSize            := SIZEOF(wDiMask)
      );

      IF fbMK210_301_DI.xDone OR fbMK210_301_DI.xError THEN
          // после выполнения блока его надо сбросить
          fbMK210_301_DI(xExecute := FALSE);

          xDi0 := wDiMask.0;

          iStateMK210 := 2;
      END_IF

```

2: // запись дискретных выходов модуля МК210-301

```
wDoMask.0 := xDo0;
```

```
fbMK210_301_DO
```

```
(  
    xExecute           := fbTcpClientMK210.xActive,  
    tTimeout          := T#50MS,  
    usiRetry           := 3,  
    hConnection       := fbTcpClientMK210.hConnection,  
    xlsRtuOverTcpMode := FALSE ,  
    usiUnitId         := 1,  
    eFuncCode         := OCL.MB_FC.WRITE_MULTIPLE_REGISTERS,  
    uiDataAddr        := 470,  
    uiDataCount       := 1,  
    pData             := ADR(wDoMask),  
    szSize            := SIZEOF(wDoMask)  
);
```

```
IF fbMK210_301_DO.xDone OR fbMK210_301_DO.xError THEN
```

```
    // после выполнения блока его надо сбросить
```

```
    fbMK210_301_DO(xExecute := FALSE);
```

```
    // возвращаемся к первому запросу
```

```
    iStateMK210 := 1;
```

```
END_IF
```

```
END_CASE
```

**Б4 Листинг примера из п. 5.9.4**

```

PROGRAM PLC_PRG_ST
VAR

    fbRealToWorld2: OCL.REAL_TO_WORD2;           // ФБ преобразования REAL в две...
                                                    // ...переменные типа WORD

    fbModbusTcpSlave: OCL.MB_TcpSlave;           // ФБ для реализации Modbus Slave
    awSlaveData:     ARRAY [0..15] OF WORD; // буфер данных Modbus Slave

    (* значения, полученные от OPC *)
    xVar_Opc:        BOOL;
    wVar_Opc:        WORD;
    rVar_Opc:        REAL;
    sVar_Opc:        STRING(15);

    (* значения для передачи в OPC *)
    xVar_Plc:        BOOL;
    wVar_Plc:        WORD;
    rVar_Plc:        REAL;
    sVar_Plc:        STRING(15);

    xWrite:          BOOL; // команда записи данных из программы в регистры Modbus Slave
    fbWriteEdge:     R_TRIG; // триггер для однократной записи
END_VAR

// чтобы запустить пример на ST на контроллере требуется:
// 1. Удалить из задачи MainTask вызов программы PLC_PRG_CFC
// 2. Привязать к задаче MainTask вызов программы PLC_PRG_ST

fbModbusTcpSlave
(
    xEnable      := TRUE,
    slpAddr      := '0.0.0.0',
    uiPort       := 502,
    usiUnitId    := 1,
    pData        := ADR(awSlaveData),
    szSize       := SIZEOF(awSlaveData)
);

// данные, полученные от OPC
xVar_Opc := awSlaveData[0].0;
wVar_Opc := awSlaveData[1];
rVar_Opc := OCL.WORD2_TO_REAL(awSlaveData[2], awSlaveData[3], FALSE);
OCL.SWAP_DATA( ADR(awSlaveData[4]), ADR(sVar_Opc), SIZEOF(sVar_Opc), TRUE, FALSE, FALSE );

```

```
// по команде записываем переменные из программы в регистры Modbus Slave
```

```
fbWriteEdge(CLK := xWrite);
```

```
IF fbWriteEdge.Q THEN
```

```
    awSlaveData[0].0 := xVar_Plc;
```

```
    awSlaveData[1] := wVar_Plc;
```

```
    fbRealToWorld2(rInput := rVar_Plc, wOutput1 => awSlaveData[2], wOutput2 => awSlaveData[3]);
```

```
    OCL.SWAP_DATA( ADR(sVar_Plc), ADR(awSlaveData[4]), SIZEOF(sVar_Plc), TRUE, FALSE, FALSE );
```

```
END_IF
```

## Б5 Листинг примера из п. 5.9.5

```
PROGRAM PLC_PRG_ST
VAR
    fbModbusTcpSlave:    OCL.MB_TcpSlave;           // ФБ для реализации Modbus Slave
    awSlaveData:         ARRAY [0..15] OF WORD;    // буфер данных Modbus Slave

    // пути к файлам архивов
    asFilePath:         ARRAY [1..8] OF STRING := ['/mnt/ufs/home/root/Arc1.log', 7('')];
END_VAR

// чтобы запустить пример на ST на контроллере требуется:
// 1. Удалить из задачи MainTask вызов программы PLC_PRG_CFC
// 2. Привязать к задаче MainTask вызов программы PLC_PRG_ST

fbModbusTcpSlave
(
    xEnable              := TRUE,
    slpAddr              := '0.0.0.0',
    uiPort               := 502,
    usiUnitId            := 1,
    pData               := ADR(awSlaveData),
    szSize               := SIZEOF(awSlaveData),
    pasFilePath          := ADR(asFilePath),
    c_xReverseByteOrderFiles := TRUE,
    c_usiAmountBytes     := 10
);
```