2016

СПК

Настройка обмена по Modbus

Руководство для начинающих и продвинутых пользователей

Версия: 1.0
Дата: 15.08.2016

Оглавление

1. Цель документа. Способы работы с Modbus в CODESYS 3.5 5
2. Общие сведения 7
2.1. Основные сведения об интерфейсе RS-485 7
2.2. Основные сведения о протоколе Modbus 8
2.3. Особенности работы с COM 10
2.3.1. Соответствие нумерации COM-портов СПК и CODESYS 10
2.3.2. Распиновка COM1-портов СПК 10
2.3.3. Выбор режима работы порта 11
2.4. Особенности работы с модулями Mx110 12
3. Шаблоны модулей Mx110 14
3.1. Установка шаблонов модулей в среду CODESYS 14
3.2. Добавление шаблонов модулей в проект 16
3.3. Использование шаблонов модулей в проекте 20
3.4. Пример: СПК207 + модули Mx110 (MB110-8A, MB110-16Д, МУ110-16Р) 22
4. Стандартные средства конфигурирования (Modbus RTU) 29
4.1. Общая методика конфигурирования интерфейсов 29
4.2. Настройка СПК в режиме Modbus RTU Master 30
4.3. Настройка СПК в режиме Modbus RTU Slave 38
4.4. Преобразование данных (REAL, DWORD, STRING) 43
4.4.1. Использование объединений (UNION) 44
4.4.2. Использование указателей 48
4.5. Диагностика обмена 51
4.6. Пример: СПК207 + модули Mx110 (MB110-8A, MB110-16Д, МУ110-16Р) 52
4.7. Пример: СПК207 (master) + СПК207 (slave) 65
4.7.1. Настройка СПК207 (master) 67
4.7.2. Настройка СПК207 (slave) 73
4.7.3. Запуск примера 79
5. Библиотека ModulsOwenLib 80
5.1. Установка библиотеки 80
5.2. Добавление библиотеки в проект CODESYS 82
5.3. Описание ФБ библиотеки 83
5.3.1. Блок ComConn 83

5.3.2. Блоки входов и выходов модулей 84
5.3.3. Блоки UniRead/UniWrite 85
5.4. Пример: СПК207 + модули Mx110 (MB110-8A, MB110-16Д, МУ110-16Р) 86
5.4.1. Описание реализации на языке CFC 88
5.4.2. Описание реализации на языке ST 94
5.5. Рекомендации по использованию библиотеки 100
6. Библиотека Modbus 101
6.1. Установка библиотеки 101
6.2. Добавление библиотеки в проект CODESYS 103
6.3. Описание ФБ библиотеки 105
6.3.1. Блок ComService (библиотека ComService) 105
6.3.2. Блоки функций Modbus 106
6.4. Пример: СПК207 + модули Mx110 (MB110-8A, MB110-16Д, МУ110-16Р) 110
6.4.1. Описание реализации на языке CFC 112
6.4.2. Описание реализации на языке ST 119
6.5. Рекомендации по использованию библиотеки 126
7. Библиотека Modbus Slave 127
7.1. Установка библиотеки 127
7.2. Добавление библиотеки в проект CODESYS 129
7.3. Описание ФБ библиотеки 131
7.3.1. Блок ComService (библиотека ComService) 131
7.3.2. Блок MB_SLAVE 131
7.4. Пример: СПК207 (master) + СПК207 (slave) 132
7.4.1. Настройка СПК207 (master) 134
7.4.2. Настройка СПК207 (slave) 140
7.4.3. Запуск примера 145
8. Modbus TCP (стандартные средства конфигурирования 147
8.1. Особенности работы с Modbus TCP 147
8.2. Настройка СПК в режиме Modbus TCP Master 147
8.3. Настройка СПК в режиме Modbus TCP Slave 151
8.4. Пример: СПК207 (master) + СПК207 (slave) 154
8.4.1. Настройка СПК207 (master) 156
8.4.2. Настройка СПК207 (slave) 162
8.4.3. Запуск примера 168
9. FAQ 169

9.1. Что делать, если не удается наладить обмен по Modbus? 169
9.2. Каким образом считать/передать значение с плавающей точкой (REAL)? 170
9.3. Каким образом считать/передать отрицательное значение (INT)? 171
9.4. Как работать с примерами? 171
9.5. Стандартные средства конфигурирования 172
9.5.1. Какие версии Modbus компонентов рекомендуются к использованию? 172
9.5.2. [СПК - master] Как реализовать чтение/запись по триггеру? 172
9.5.3. [СПК – slave] Почему принятые значения сбрасываются в 0? 173
9.5.4. [СПК – slave] Как работать с Coils/Discrete Inputs? 174
9.5.5. [СПК – slave] Можно ли менять данные holding регистров из программы? 174
9.5.6. Как произвести диагностику обмена в программе? 174
9.5.7. Как расшифровываются пиктограммы статуса обмена? 174
Приложение 175
А. Список переменных шаблонов модулей Mx110 175
Б. Список переменных входов/выходов модулей в ФБ библиотеки ModulsOwenLib 179
В. Список переменных диагностики Modbus 183
Табл. В1. Список переменных диагностики 183
Табл. В2. Коды ошибок обмена 185
Г. Рекомендуемые версии компонентов Modbus 186
Д. Листинги программ 187
Д.1. Листинг программы COM2 из п. 5.4.2 187
Д.2. Листинг программы COM3 из п. 5.4.2 188
Д.3. Листинг программы COM2 из п. 6.4.2 190
Д.4. Листинг программы COM3 из п. 6.4.2 192

1. Цель документа. Способы работы с Modbus в CODESYS 3.5

Этот документ представляет собой руководство по настройке обмена данными с использованием протокола **Modbus** для панельных контроллеров Овен СПК в среде **CODESYS V3.5**. Подразумевается, что читатель обладает базовыми навыками работы с **CODESYS** и **СПК**, поэтому общие вопросы (например, создание и загрузка проектов) в данном документе не рассматриваются; они подробно описаны в документах **СПК. Первый старт** и **СПК. FAQ**, которые доступны на сайте Овен в разделе **CODESYS V3/Документация**.

При работе с **СПК** у пользователя, в зависимости от его квалификации и потребностей, имеется возможность выбрать наиболее подходящий ему способ организации связи по протоколу **Modbus**:

1. для начинающих пользователей, работающих с модулями Mx110 – **шаблоны модулей**. Шаблоны представляют собой сконфигурированные компоненты **CODESYS**, добавляемые в проект несколькими кликами мыши, для которых необходимо указать только адрес в сети **Modbus**.

Преимущества:
- простота использования;
- быстрое создание проекта;
- нет необходимости в дополнительном программировании;
- не надо разбираться с картами регистров модулей.

Недостатки:
- строго заданная конфигурация регистров и интервалов опроса без возможности редактирования.

2. для продвинутых пользователей – **стандартные средства конфигурирования CODESYS**. С их помощью можно достаточно просто настроить обмен с любым устройством, в т.ч. по **Modbus TCP**.

Преимущества:
- возможность создания гибкой конфигурации;
- возможность выбора регистров и функций для опроса;
- ручная настройка таймаутов для обеспечения максимально возможной скорости интерфейса;
- нет необходимости в дополнительном программировании.

Недостатки:
- требуются глубокие знания спецификации протокола;
- требуется знание особенностей работы модулей Mx110.

3. для опытных пользователей, работающих с модулями Mx110 – библиотека **ModulsOwenLib**. В отличие от шаблонов (пп. 1), она предоставляет более широкий функционал – например, исключение/возвращение модулей из опроса, изменение настроек COM-порта в процессе работы программы и т.д. Кроме того, библиотека позволяет работать с протоколом **Modbus ASCII**.

Преимущества:

- практически неограниченные возможности для работы с протоколом Modbus;

- создание любых конфигураций и методов опроса модулей и других устройств;

- удобно при программировании модульных систем (в разные промежутки времени в работе находится разное оборудование).

Недостатки:

- требуются профессиональные навыки программирования;

- требуется доскональное знание работы интерфейсов и протокола Modbus.

4. для профессионалов – библиотеки **Modbus** и **ModbusSlave**. Эти библиотеки позволяют настроить обмен с любым устройством, но, в отличие от стандартных средств (пп. 2), предоставляют более широкий функционал – например, исключение/возвращение устройств из опроса, изменение настроек COM-порта в процессе работы программы и т.д. Кроме того, библиотека позволяет работать с протоколом **Modbus ASCII**.

Преимущества:

- практически неограниченные возможности для работы с протоколом Modbus;

- создание любых конфигураций и методов опроса модулей и других устройств;

- удобно при программировании модульных систем (в разные промежутки времени в работе находится разное оборудование).

Недостатки:

- требуются профессиональные навыки программирования;

- требуется доскональное знание работы интерфейсов и протокола Modbus.

2. Общие сведения

2.1. Основные сведения об интерфейсе RS-485

1. Интерфейс RS-485 подразумевают использование исключительно топологии «шина» (топологии «звезда» и «кольцо» не поддерживаются).

2. В сети может присутствовать только одно master-устройство, которое опрашивает подчиненные slave-устройства.

3. Число slave-устройств на шине не должно превышать 32-х. На практике это значение может быть увеличено до 247 устройств при использовании повторителей интерфейса (после каждых 32-х устройств), но нужно учитывать, что так как опрос всех устройств происходит последовательно, время одного полного цикла опроса может значительно увеличиться.

4. На первом и последнем устройстве шины должен быть установлен 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.

2.2. Основные сведения о протоколе Modbus

Modbus – открытый коммуникационный протокол, основанный на архитектуре **Master-Slave** (ведущий-ведомый).

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

Slave (слэйв, подчиненное устройство) не может инициализировать обмен.

Существуют две основные реализации протокола:

1. **Modbus Serial** для передачи данных с использованием последовательных интерфейсов RS-232/422/485;

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 ASCII символов; в CODESYS поддержан в библиотеках Овен, см. п. 5, п. 6 и п. 7).

При работе с **Modbus** по интерфейсам **RS-232/485** в сети может находиться **только одно** master-устройство и несколько slave-устройств (согласно стандарту – до 32-х без использования повторителей, до 247-ми с использованием повторителя после каждых 32-х устройств). Адрес **0** используется для широковещательной рассылки (рассылки, которую получат все slave-устройств).

В сети **Modbus TCP** может присутствовать более одного ведущего устройства. При этом любое устройство может быть одновременно и ведущим, и подчиненным. В сети могут также присутствовать специальные шлюзы (**gateway**), которые предоставляют устройствам из сети **TCP/IP** доступ к устройствам, объединенным последовательной линией связи, или ведущему устройству с последовательным интерфейсом доступ к сети **TCP/IP**.

Запрос master-устройства к slave-устройству содержит:

1. **Slave ID** (адрес slave-устройства);

2. **Код функции,** применяемый к slave-устройству;

3. Данные – адрес первого регистра и их количество (в случае записи – также их значения);

4. Контрольную сумму.

Ответ slave-устройства имеет аналогичную структуру.

Спецификация протокола определяет четыре области данных для устройств, участвующих в обмене:

Табл. 2.1. Области данных протокола **Modbus**

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

Запрос, с помощью которого ведущее устройство обращается к подчиненному, должен содержать код функции. Список наиболее часто используемых функций приведен в табл. 2.2:

Табл. 2.2. Основные функции протокола **Modbus**

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

Описание стандарта **Modbus** доступно на сайте modbus.org.

2.3. Особенности работы с COM

2.3.1. Соответствие нумерации COM-портов СПК и CODESYS

При настройке интерфейсов RS-232/485 в **CODESYS** необходимо указывать номера портов. Программная нумерация портов отличается от нумерации на корпусе устройств. Соответствие между номерами портов на корпусе СПК и в CODESYS приведено в табл. 2.3:

Табл. 2.3. Соответствие нумерации COM-портов СПК и CODESYS

Нумерация портов на корпусе прибора	**Нумерация портов в CODESYS**				
	СПК105*	СПК107	СПК110	СПК2xx.03	СПК2xx.04
COM1	2 (RS-485) 3 (RS-232)	2 (RS-232/485)		2 (RS-232)	
COM2	-	3 (RS-232/485)		3 (RS-232/485)	3 (RS-232)
COM3	-	-		4 (RS-232/485)	

** В СПК105 интерфейсы RS-485 и RS-232 выведены на один порт COM1 и, в отличие от остальных СПК, поддерживается их одновременная работа*

2.3.2. Распиновка COM1-портов СПК

Распиновка COM1-порта СПК1xx приведена в табл. 2.4. Распиновка COM-порта СПК2xx приведена в табл. 2.5.

Табл. 2.4. Распиновка COM1 **СПК1xx**

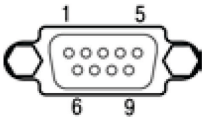
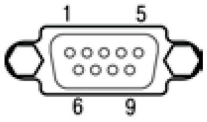
Номер контакта 	Наименование сигнала
1	RS-485 A
2	RXD
3	TXD
4	-
5	GND
6	RS-485 B
7	-
8	-
9	-

Табл. 2.5. Распиновка COM1 **СПК2xx**

Номер контакта 	Наименование сигнала
1	DCD
2	RXD
3	TXD
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI

2.3.3. Выбор режима работы порта

Некоторые COM-порты СПК являются универсальными, и могут использоваться как в режиме RS-232, так и RS-485. Переключение режимов осуществляется в конфигураторе СПК. Подробно этот процесс описан в **РЭ** соответствующего контроллера.

Рис. 2.1. Выбора режима COM-порта СПК в конфигураторе

При реализации обмена с помощью библиотек (п. 5, п. 6, п. 7) можно переключать режим работы порта из программы с помощью ФБ ComConn (п. 5) и ComService (п. 6, п. 7).

2.4. Особенности работы с модулями Mx110

Перед тем, как подключать модули Mx110 к СПК, их необходимо сконфигурировать с помощью программы **Конфигуратор Mx110**. Программа находится на диске из комплекта поставки, а также доступна на сайте ОВЕН на странице любого из модулей.

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

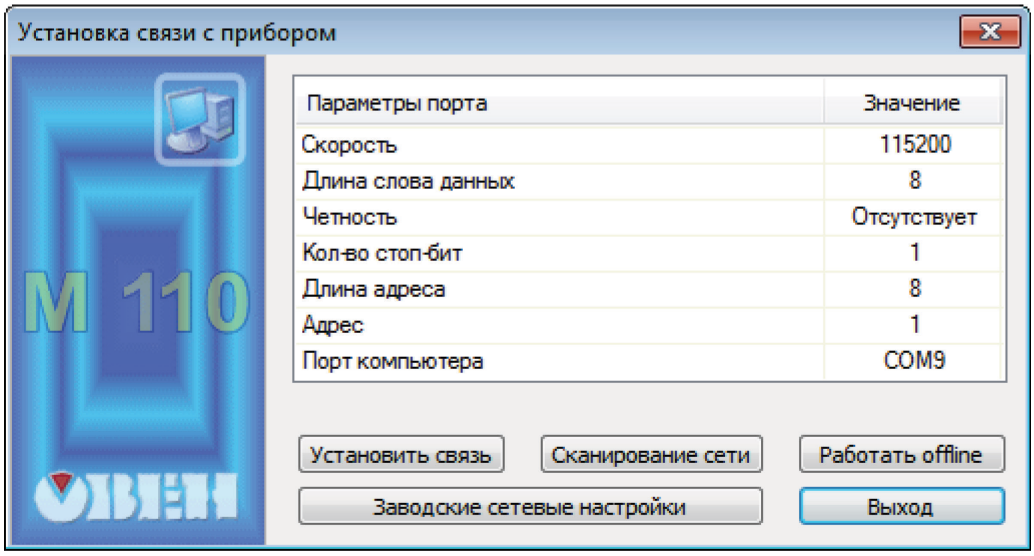


Рис. 2.2. Подключение к модулю Mx110 с помощью программы **Конфигуратор Mx110**

В конфигураторе задаются сетевые настройки модулей и параметры входов/выходов. ***Обратите внимание***, что в пределах одной сети все модули должны иметь одинаковые сетевые настройки, за исключением адресов.

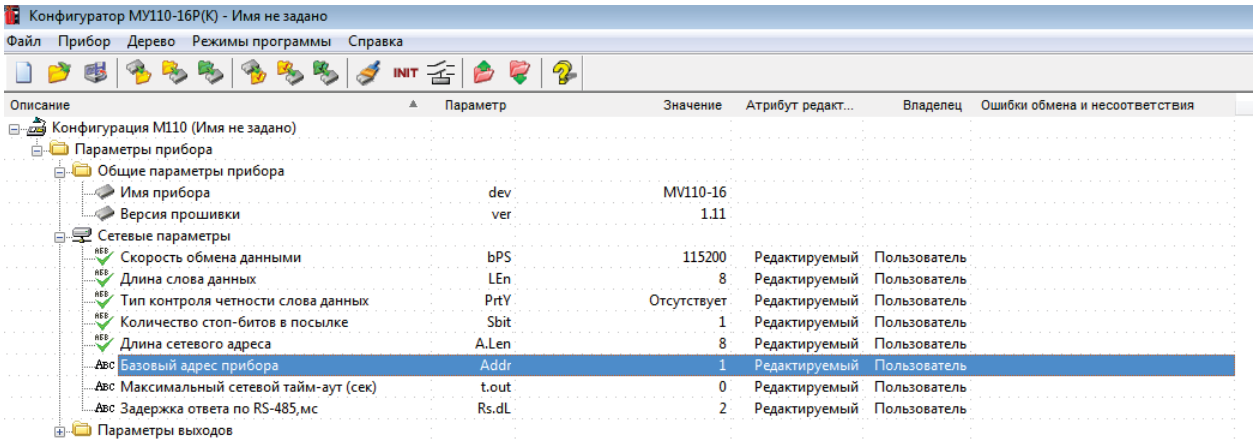


Рис. 2.3. Настройка модуля с помощью конфигуратора

Обратите внимание, что конфигурирование модулей происходит по протоколу **Овен**. В связи с особенностями протокола, при конфигурировании каждый модуль занимает количество адресов, равное количеству его каналов; это приводит к тому, что при 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.

3. Шаблоны модулей Mx110

3.1. Установка шаблонов модулей в среду CODESYS

Для работы с шаблонами модулей нужно установить в среду программирования пакет **Mx110_drivers_3.5.4.5** (или его более свежую версию).

Пакет доступен на диске с ПО, входящем в комплект поставки, а также на сайте компании ОВЕН в разделе **CODESYS V3/Библиотеки**.

Для установки пакета в **CODESYS** в меню **Инструменты** выберите пункт **Менеджер пакетов**, после чего укажите путь к файлу пакета и нажмите **Установить**:

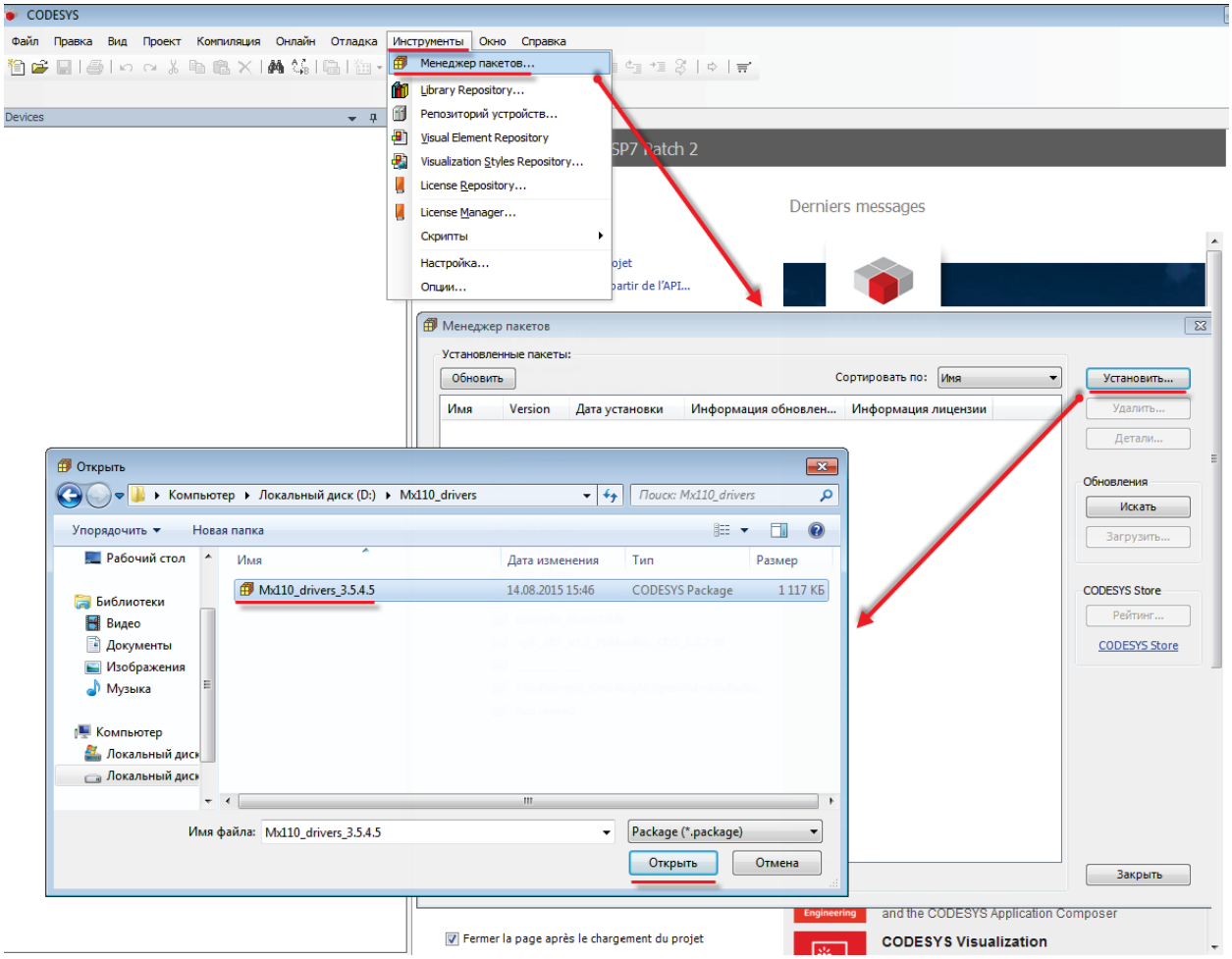


Рис. 3.1. Установка пакета **Mx110_drivers_3.5.4.5** в среду **CODESYS**

В появившемся диалоговом окне выберите пункт **Полная установка**, после чего нажмите кнопку **Next**:

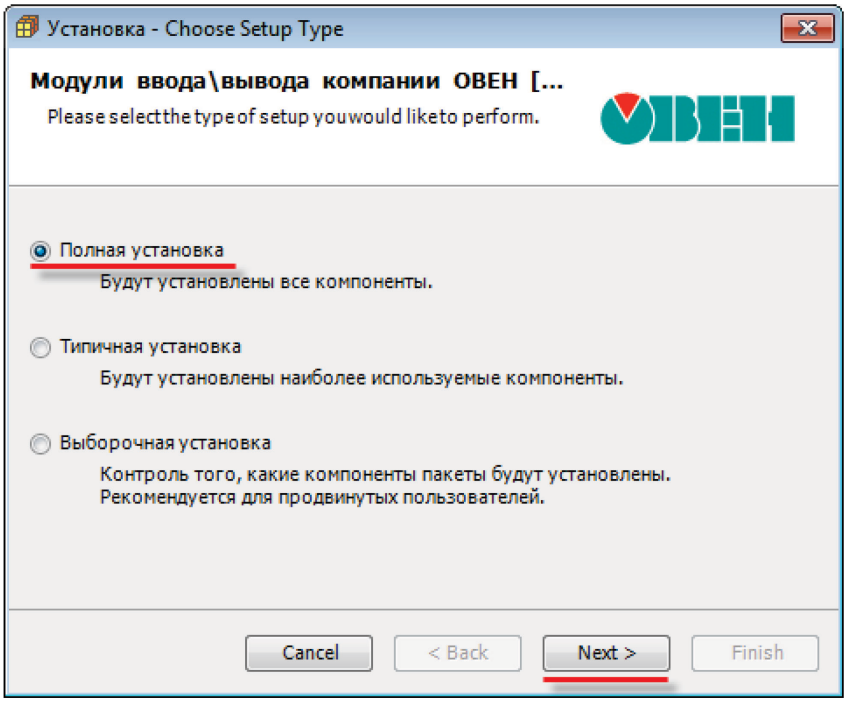


Рис. 3.2. Начало установки шаблонов модулей

После завершения установки закройте диалоговое окно с помощью кнопки **Finish**:

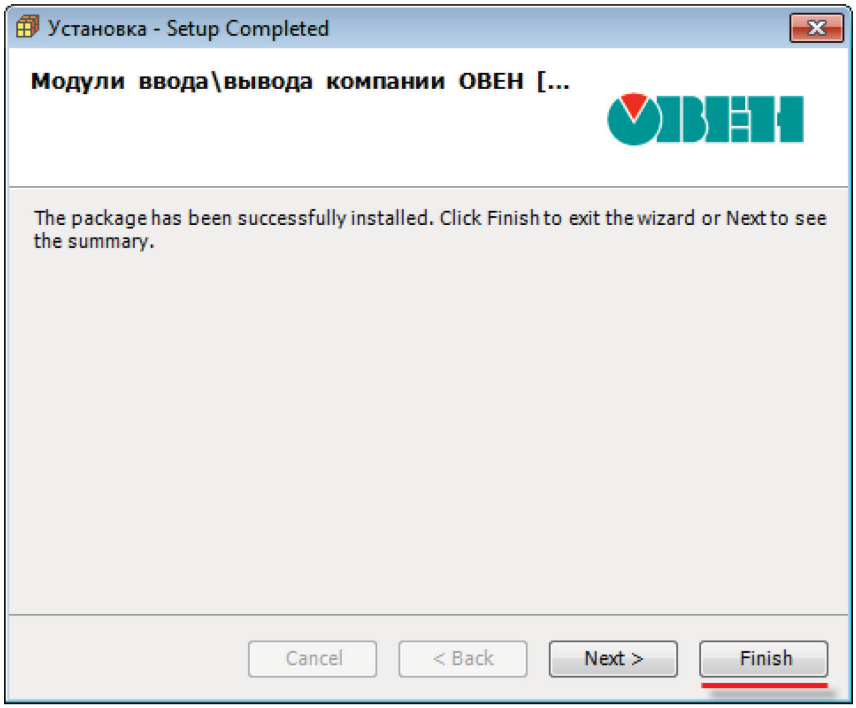


Рис. 3.3. Завершение установки шаблонов модулей

3.2. Добавление шаблонов модулей в проект

Для того чтобы добавить шаблоны модулей в проект **CODESYS**, необходимо выполнить следующие действия:

1. Нажмите **ПКМ** на компонент **Device** и добавьте компонент **Modbus COM**, расположенный во вкладке **Промышленные сети/Modbus/Порт Modbus Serial**:

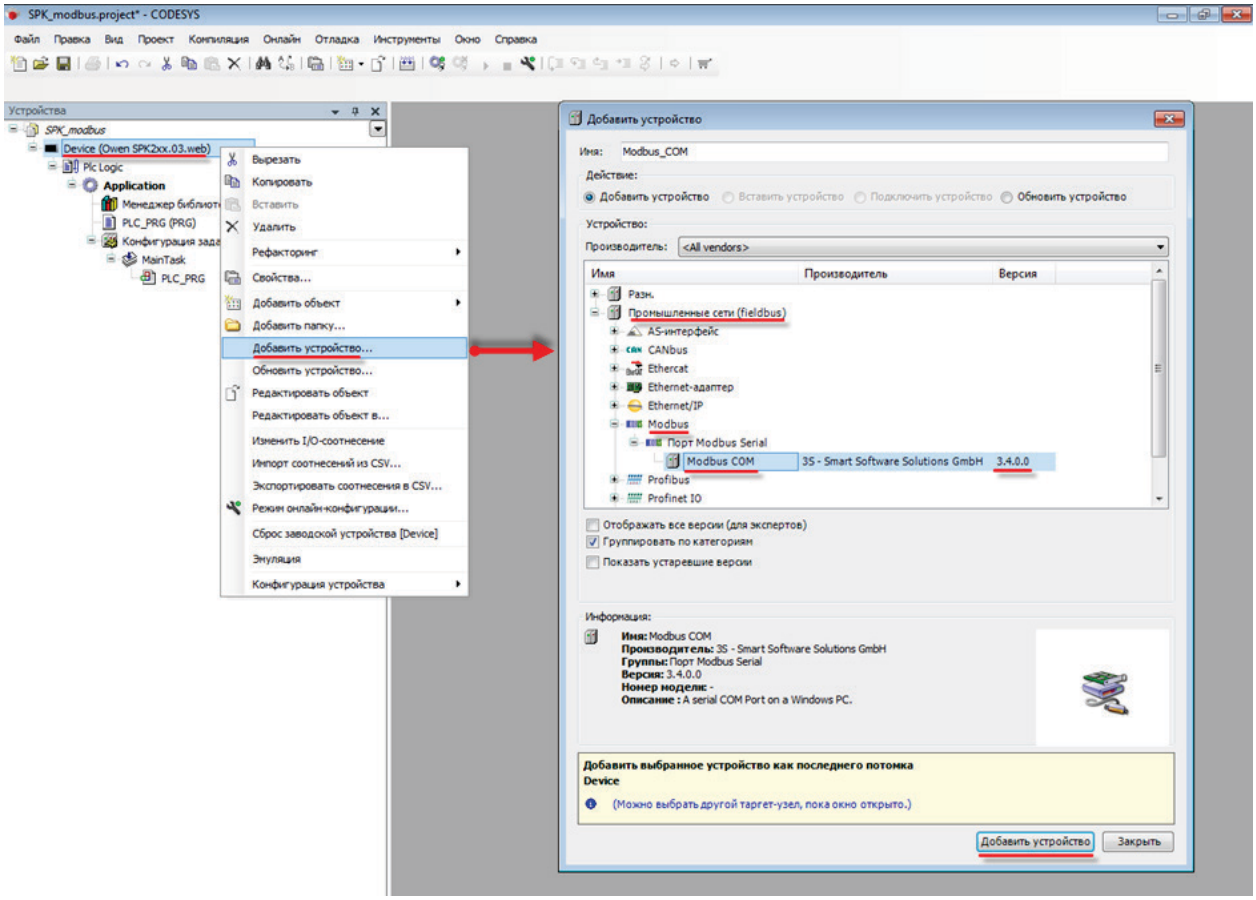


Рис. 3.4. Добавление компонента **Modbus COM**

В настройках компонента на вкладке **Общее** необходимо указать номер COM-порта СПК, к которому будут подключены модули Mx110, а также сетевые настройки модулей. ***Обратите внимание***, что программная нумерация COM-портов отличается от приведенной на корпусе СПК (см. п. 2.3.1).

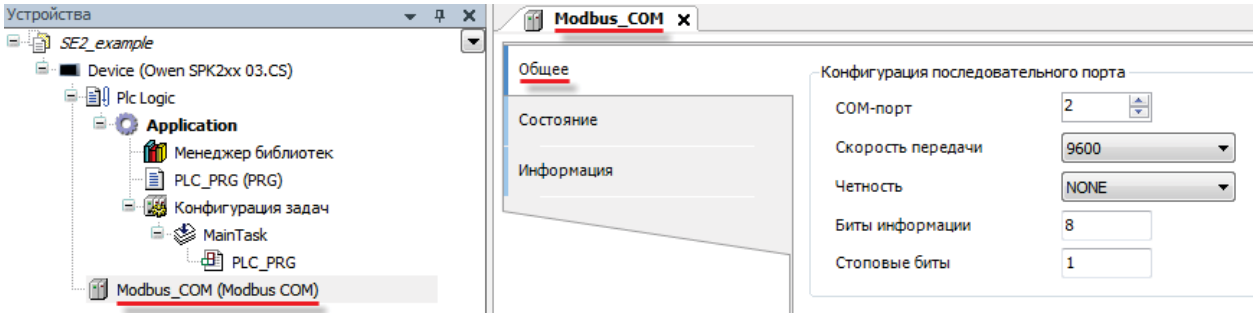


Рис. 3.5. Настройки компонента **Modbus COM**

Сетевые настройки модулей, подключенных к одному COM-порту, должны совпадать; настроить модули можно с помощью программы **Конфигуратор Mx110**, которая доступна на диске с ПО из комплекта поставки модуля, а также на сайте компании ОВЕН (на странице любого из модулей).

По умолчанию любой из модулей имеет следующие заводские сетевые настройки:

1. Скорость передачи – **9600**;
2. Чётность – **NONE**;
3. Информационные биты – **8**;
4. Стоповые биты – **1**.

2. Нажмите **ПКМ** на компоненте **Modbus COM** и добавьте компонент **Modbus Master**, расположенный во вкладке **Промышленные сети/Modbus/Мастер Modbus Serial**.

Обратите внимание, что версия компонента не должна превышать версию **target-файла** СПК. Для отображения предыдущих версий компонента поставьте галочку **Отображать все версии**. См. рекомендации в приложении Г.

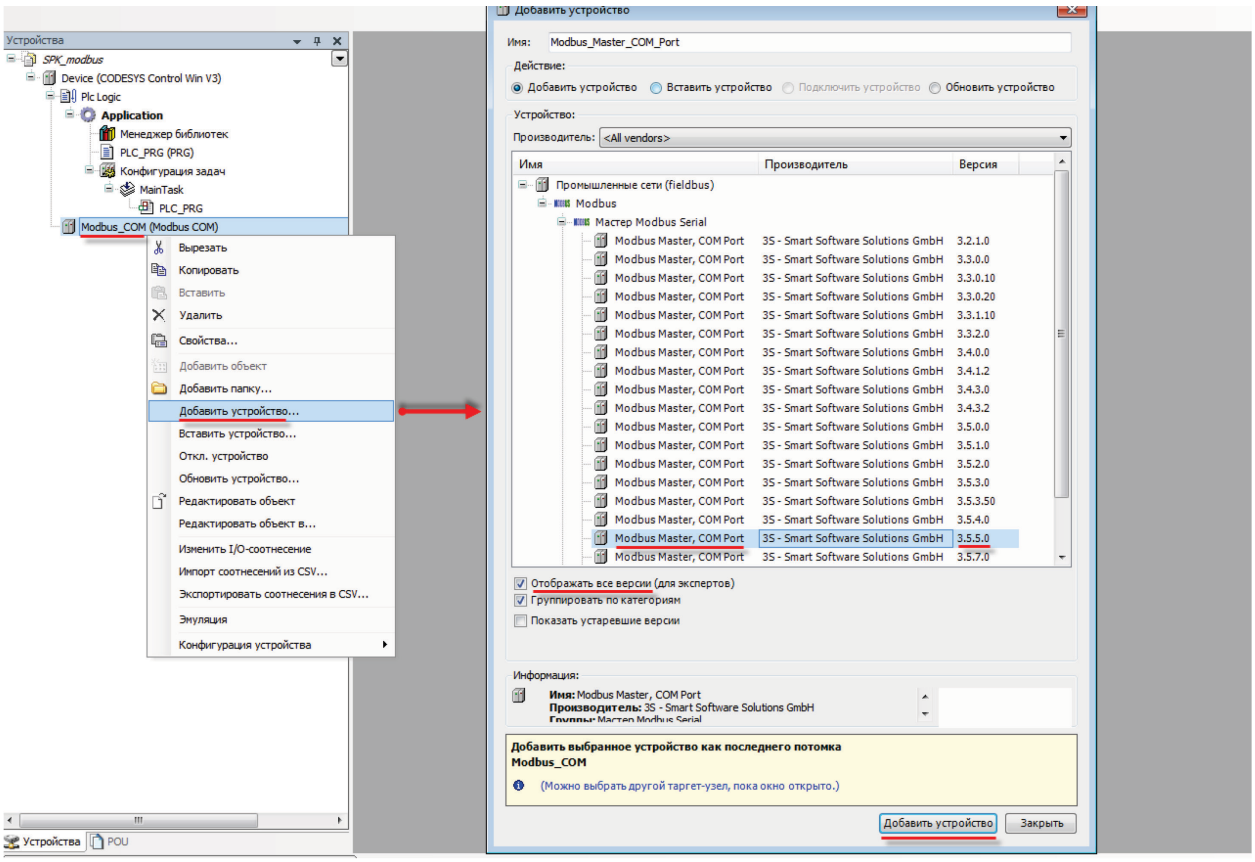


Рис. 3.6. Добавление компонента **Modbus Master**

В настройках компонента на вкладке **Общее** рекомендуется поставить галочку **Автоперезапуск соединения** – это позволит СПК автоматически восстанавливать связь с модулем в случае разрыва и восстановления канала связи.

В параметре **Время между фреймами** рекомендуется выставить значение **20 мс**.

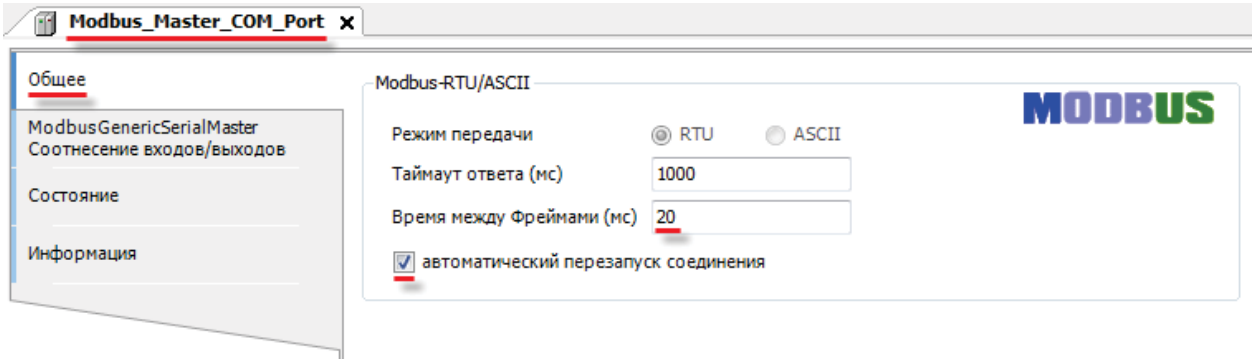


Рис. 3.7. Настройки компонента **Modbus Master**

3. Нажмите **ПКМ** на компоненте **Modbus Master** и добавьте нужные модули:

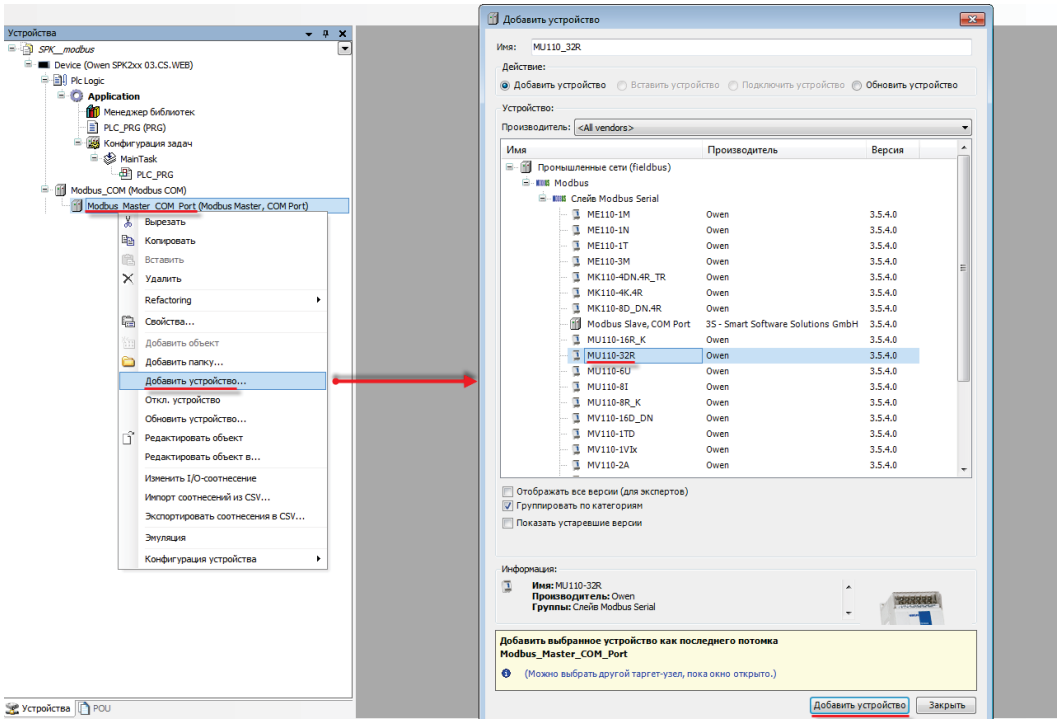


Рис. 3.8. Добавление в проект шаблонов модулей

В настройках каждого модуля укажите его **адрес**. По умолчанию он равен **16**, изменить его можно с помощью программы **Конфигуратор Mx110**, которая доступна на диске с ПО из комплекта поставки модуля, а также на сайте компании ОВЕН (на странице любого из модулей). ***Обратите внимание,*** что в пределах одной сети не должно быть двух модулей с совпадающими адресами.

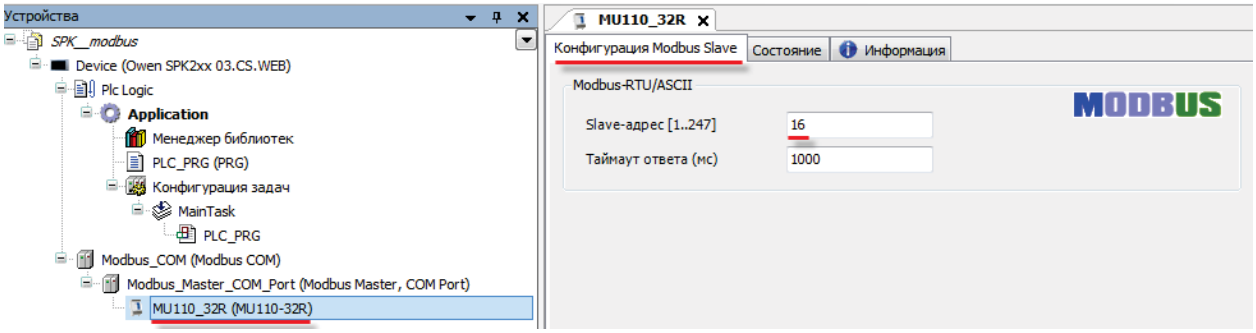


Рис. 3.9. Настройка шаблона модуля

На этом настройка шаблона завершена. Привязка переменных программы к каналам модуля не требуется, поскольку каждый шаблон уже содержит готовые переменные. Процесс их использования описан в п. 3.3.

3.3. Использование шаблонов модулей в проекте

После того, как модули добавлены в проект, пользователь может работать с их входами/выходами как с соответствующими переменными функционального блока.

Для этого в нужном месте программы введите имя модуля из дерева проекта, поставьте точку и из выпадающего списка выберите нужную переменную:

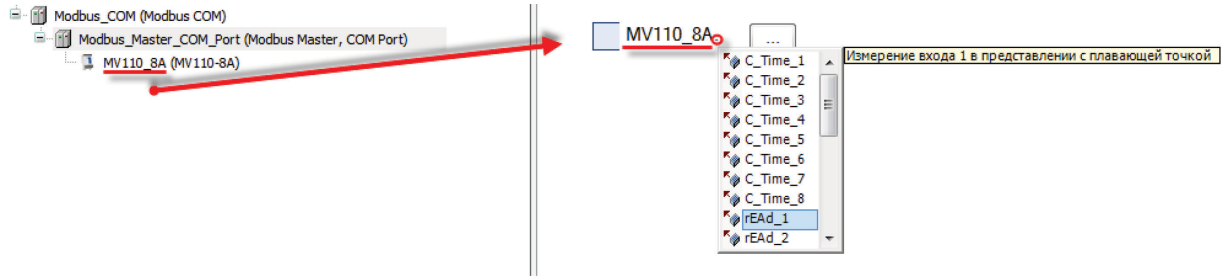


Рис. 3.10. Использование переменных модуля в программе

Ниже приведен пример присваивания значения первого входа модуля **MB110-8A** с именем **MV110_8A** переменной **rValue1** типа **REAL** на языках **CFC** и **ST**:

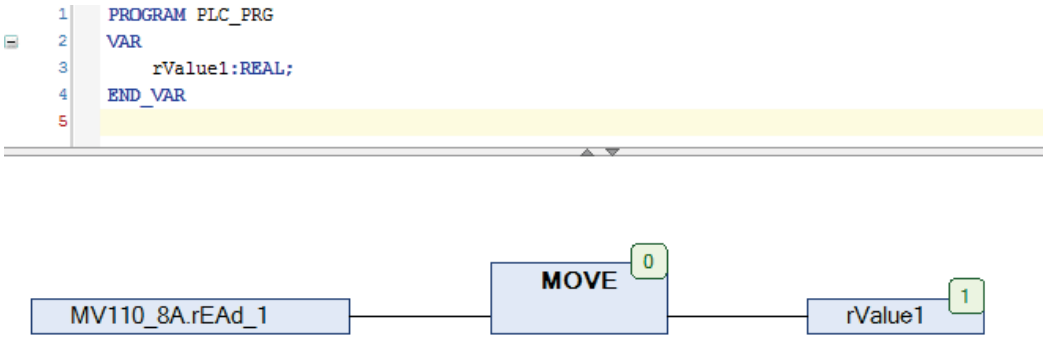


Рис. 3.11. Пример работы с переменными модулей на языке **CFC**

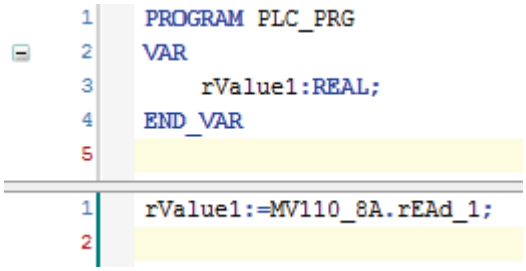


Рис. 3.12. Пример работы с переменными модулей на языке **ST**

Полный список всех переменных всех шаблонов модулей приведен в приложении А.

В случае необходимости контролировать состояние обмена с модулем, можно воспользоваться переменными диагностики. Для этого в нужном месте программы введите имя модуля из дерева проекта с постфиксом **_modbus**, поставьте точку и из выпадающего списка выберите нужную переменную диагностики:

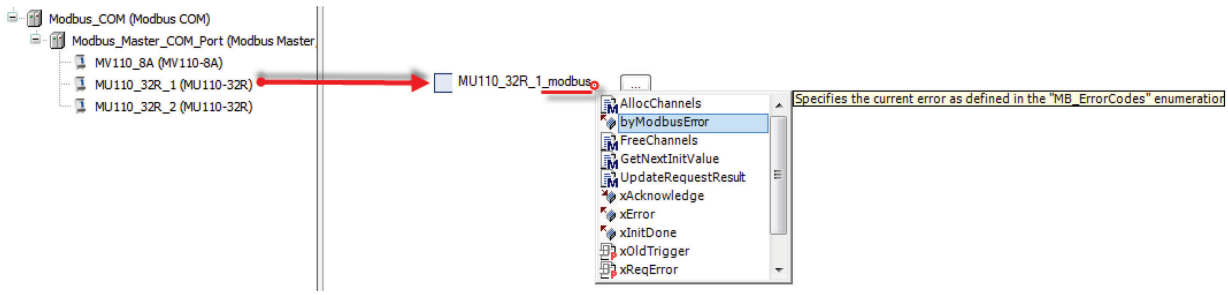


Рис. 3.13. Переменные диагностики модулей

При работе с переменными диагностики компонентов **Modbus COM** и **Modbus Master** постфикс **_modbus** не требуется – достаточно ввести имя компонента и поставить точку.

Список переменных диагностики приведен в приложении В.

3.4. Пример: СПК207 + модули Mx110 (MB110-8A, MB110-16Д, МУ110-16Р)

Рассмотрим пример настройки обмена с **модулями Mx110** с использованием **шаблонов**. В нем мы наладим связь между контроллером **СПК207.03** (master) и тремя модулями (slave-устройствами).

Реализуемый алгоритм: если значение 1-го аналогового входа модуля **MB110-8A** превышает 30 и при этом 1-ый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то присвоить 1-му дискретному выходу модуля **МУ110-16Р** значение **TRUE** (замкнут). Во всех остальных случаях присвоить дискретному выходу значение **FALSE** (разомкнут).

Структурная схема примера приведена на рис. 3.14:

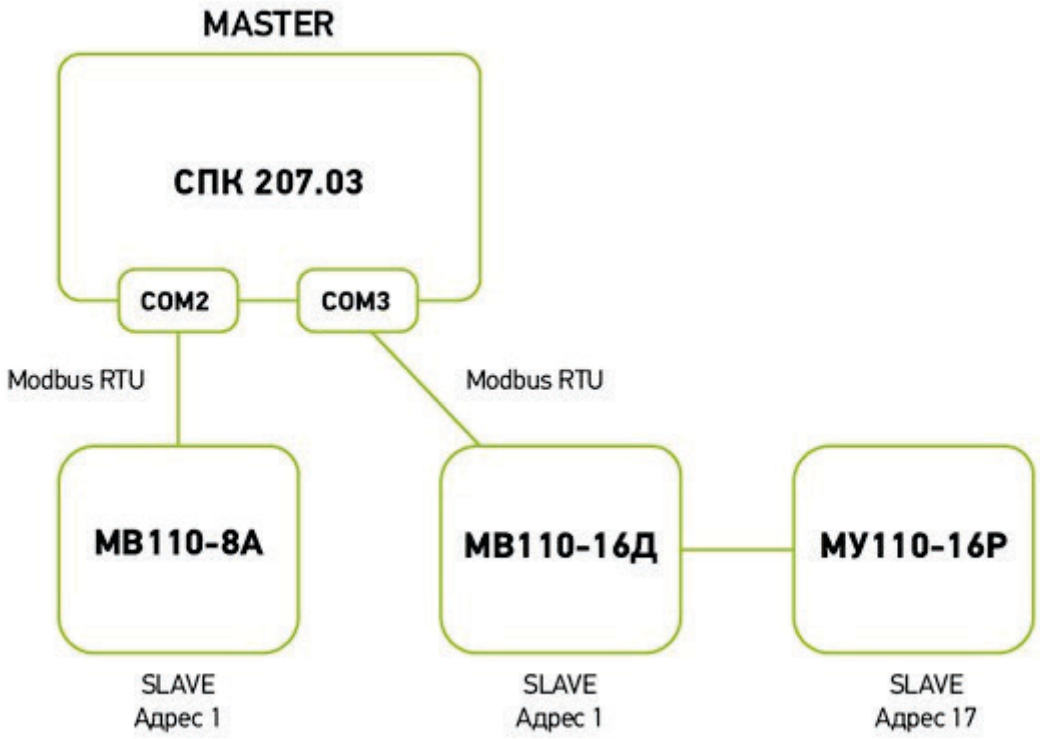


Рис. 3.14. Структурная схема примера **Шаблоны модулей Mx110**

Пример создан в среде **CODESYS 3.5 SP7 Patch4** и подразумевает запуск на **СПК207.03.CS(-WEB)**.

Пример доступен для скачивания: Example_Templates_Mx110.projectarchive

Сетевые параметры модулей приведены в табл. 3.1.

Табл. 3.1. Сетевые параметры модулей **Mx110**

Параметр	**МВ110-8А**	**МВ110-16Д**	**МУ110-16Р**
COM-порт СПК, к которому подключен модуль	*COM2*	*COM3*	
Адрес модуля	1	1	17
Скорость обмена	115200		
Количества бит данных	8		
Контроль четности	отсутствует		
Количество стоп-бит	1		

1. Настройте модули **Mx110** с помощью программы **Конфигуратор Mx110** в соответствии с табл. 3.1. Подключите модули к COM-портам СПК (в соответствии с рис. 3.14).

2. Выберите режим работы COM-портов **RS-485** в конфигураторе СПК (см. п. 2.3.3).

3. Установите пакет шаблонов модулей CODESYS (см. п. 3.1).

4. Создайте новый проект **CODESYS** для **СПК207.03** с программой **PLC_PRG** на языке **CFC**:

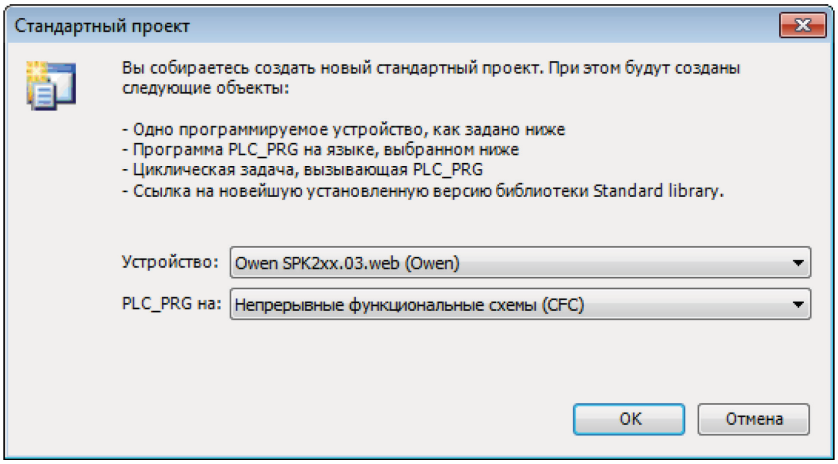


Рис. 3.15. Создание проекта **CODESYS**

5. Добавьте в проект два устройства **Modbus COM** с названиями **COM2** и **COM3**:

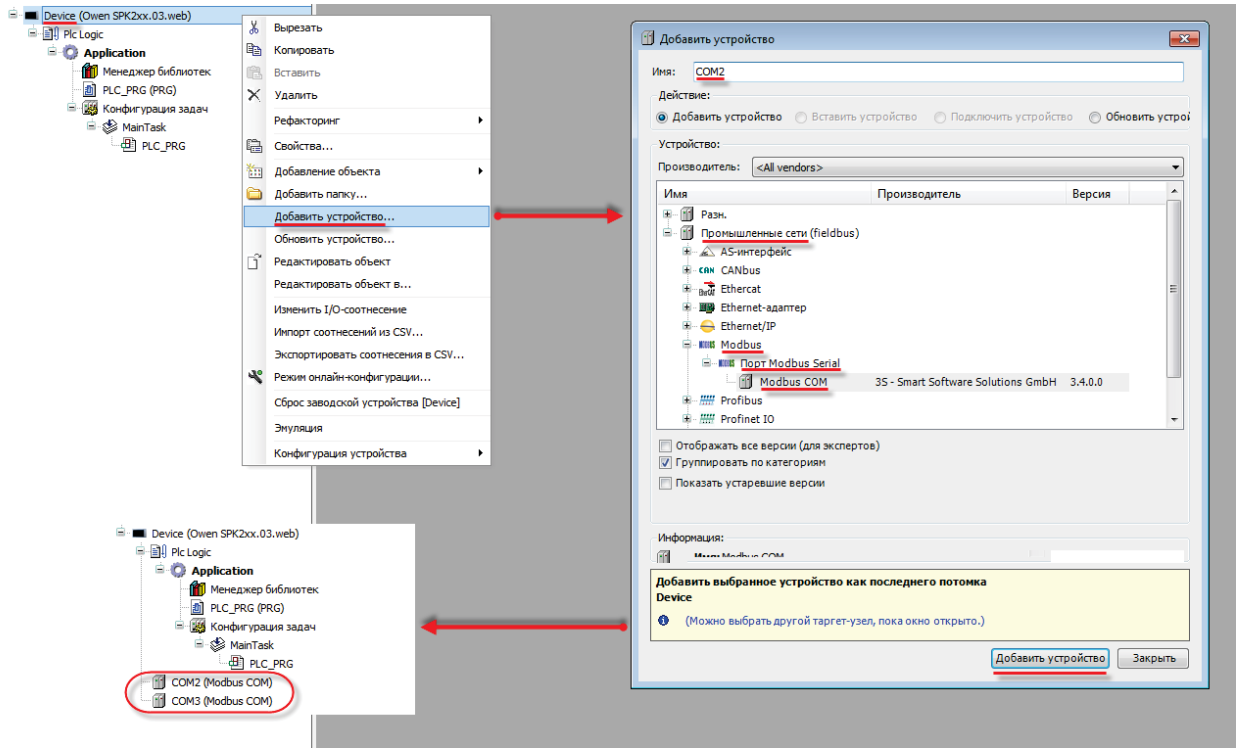


Рис. 3.16. Добавление устройства **Modbus COM**

В конфигурации COM-портов укажите сетевые настройки в соответствии с табл. 3.1, а также номера портов – **COM2** будет соответствовать номер **3**, **COM3** – номер **4** (см. п. 2.3.1).

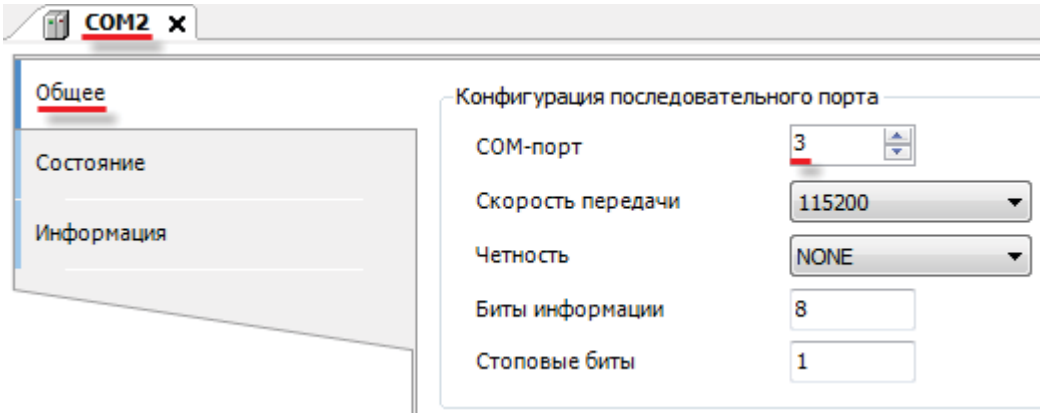


Рис. 3.17. Настройки COM-порта **COM2**

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

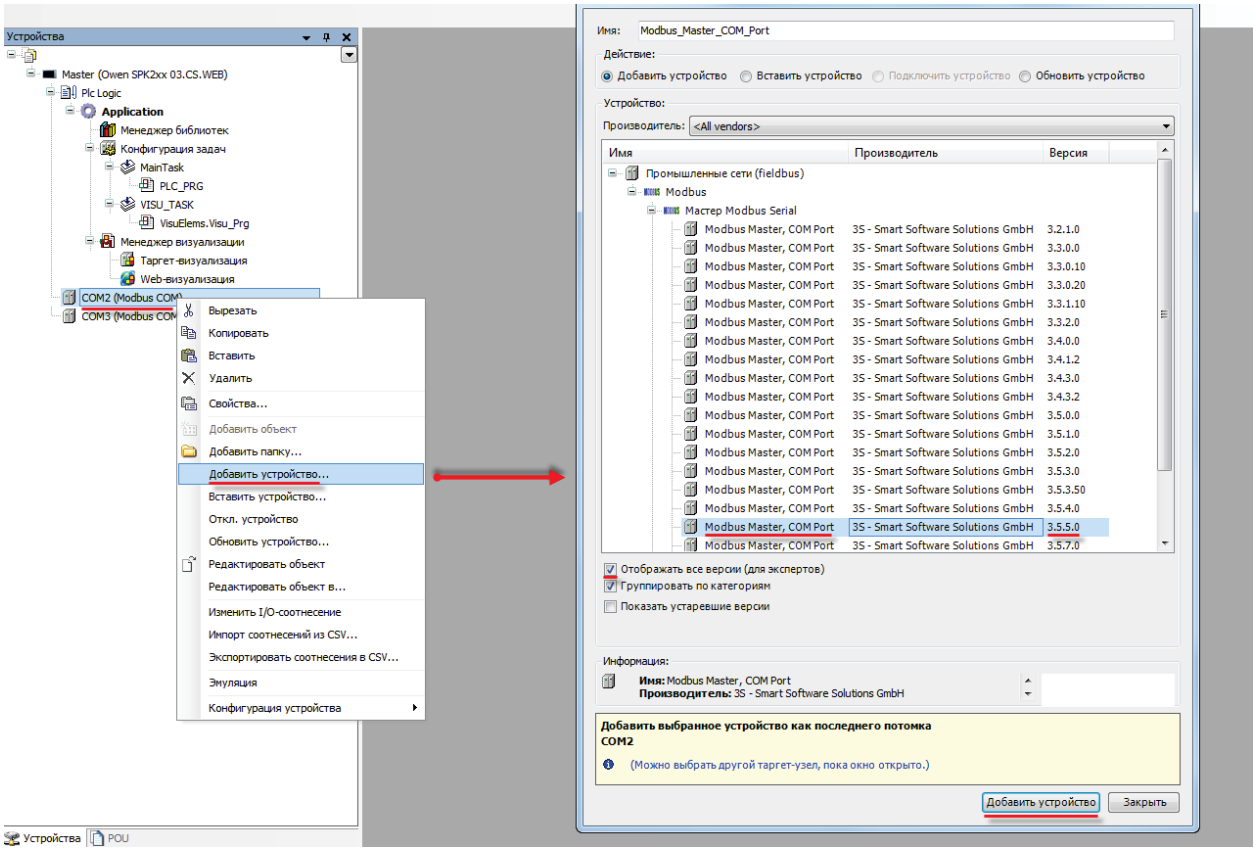


Рис. 3.18. Добавление компонента **Modbus Master**

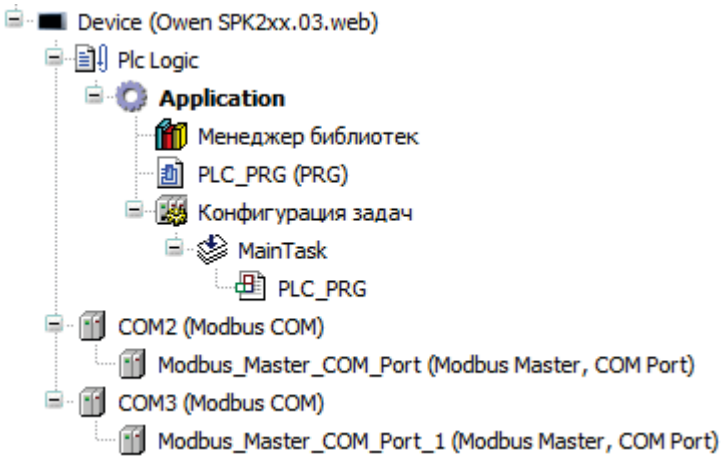


Рис. 3.19. Внешний вид дерева проекта после добавления **Modbus Master**

В настройках компонентов поставьте галочку **Автоперезапуск соединения**. В параметре **Время между фреймами** установите значение **20 мс**.

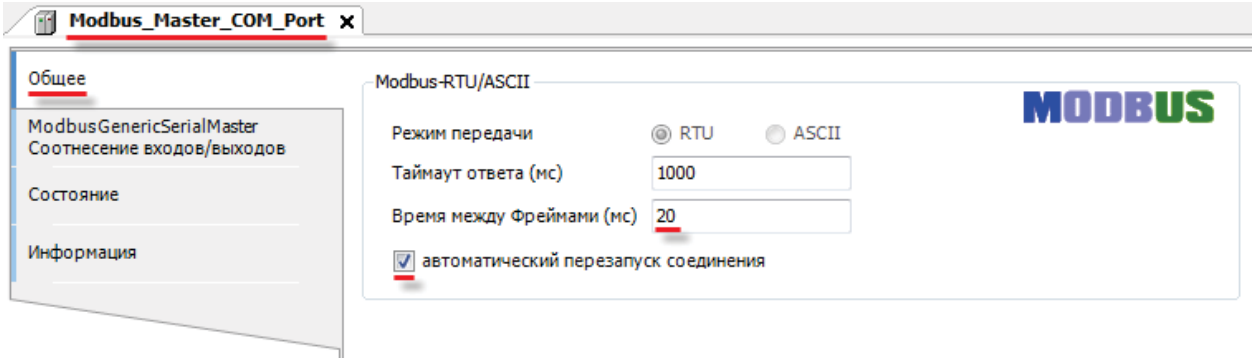


Рис. 3.20. Настройка компонентов **Modbus Master**

7. В компонент **Modbus Master** порта **COM2** добавьте модуль **MB110-8A,** а в **Modbus Master** порта **COM3 – MB110-16Д** и **МУ110-16Р**:

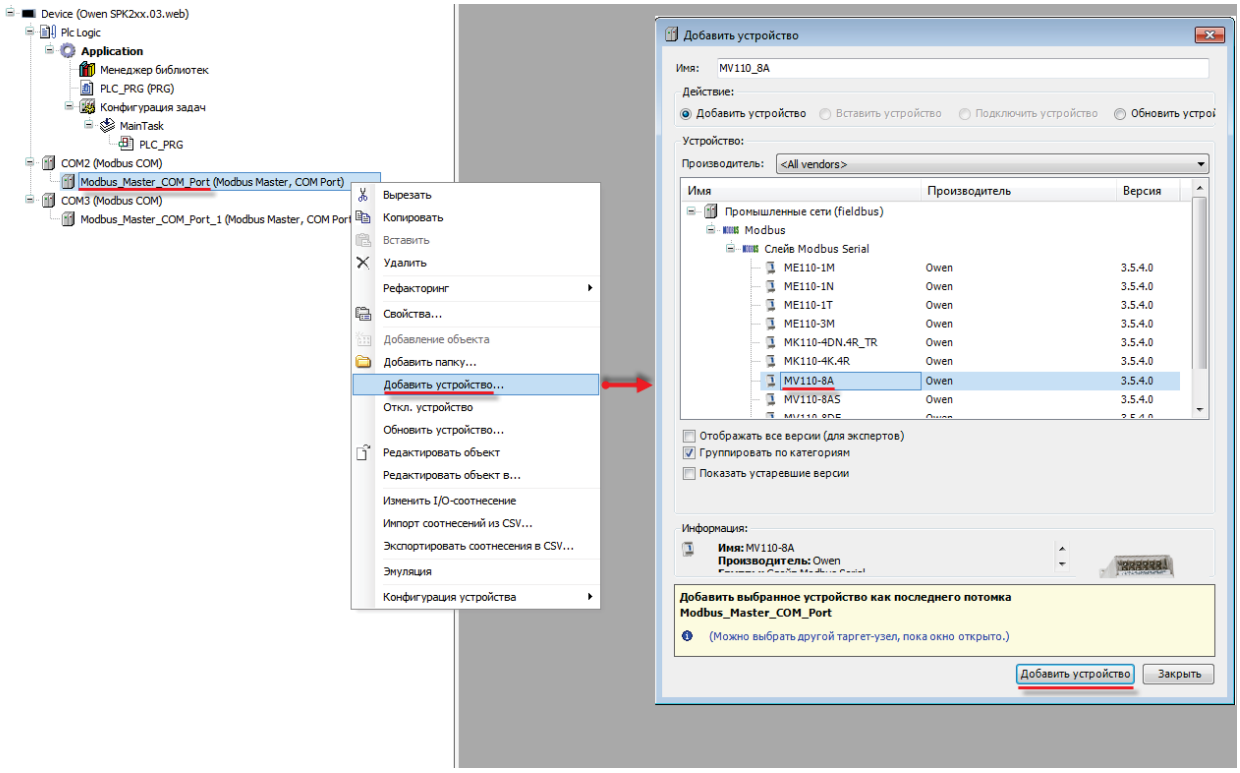


Рис. 3.21. Добавление шаблонов модулей в проект **CODESYS**

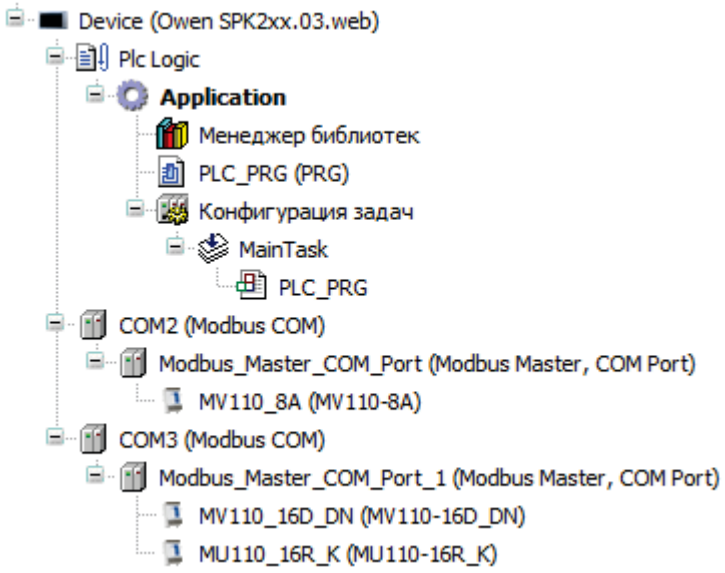


Рис. 3.22. Внешний вид дерева проекта после добавления шаблонов модулей

В настройках модулей укажите их адреса согласно табл. 3.1 (МВ110-8А – адрес **1**, МВ110-16Д – адрес **1**, МУ110-16Р – адрес **17**):

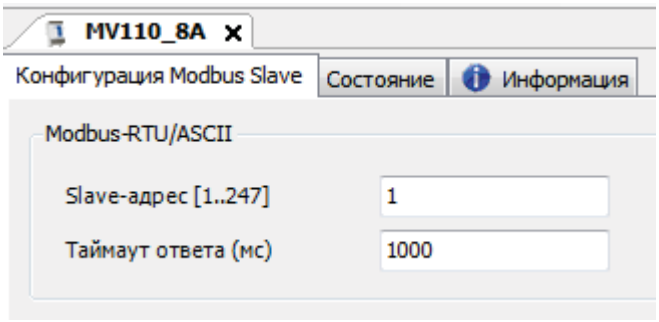


Рис. 3.23. Настройка модуля **MV110_8A**

8. Воспользуйтесь переменными шаблонов модулей в программе **PLC_PRG**. Для этого в нужном месте программы введите имя модуля из **дерева проекта**, поставьте точку и выберите нужную переменную модуля:

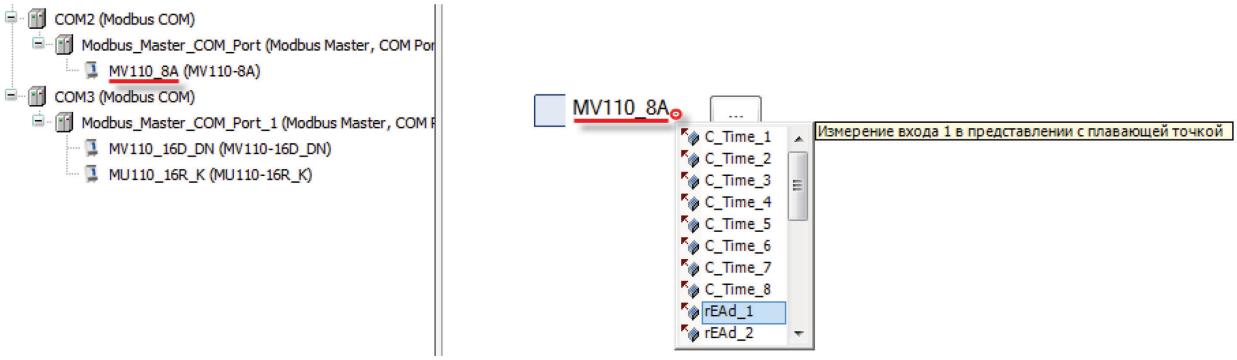


Рис. 3.24. Использование переменных модулей в программе

Код программы будет выглядеть следующим образом:

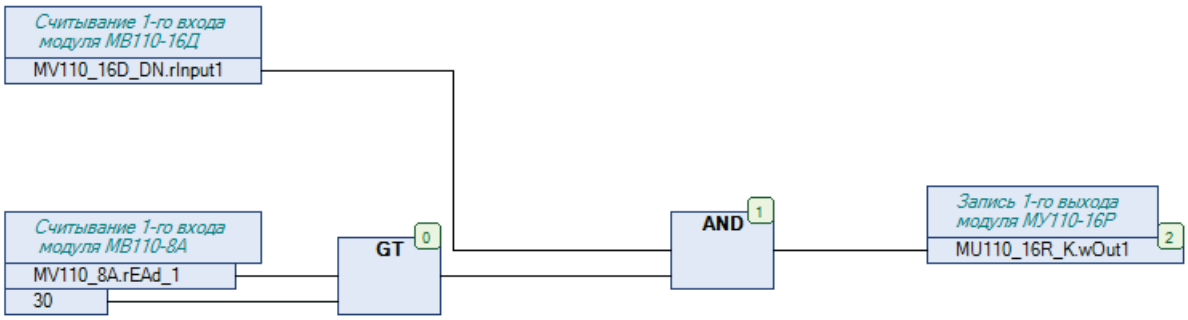


Рис. 3.25. Код программы на языке **CFC**

9. Загрузите проект в СПК и запустите его.

В переменной **MV110_8A.rEAd_1** будет отображаться текущее значение 1-го аналогового входа модуля **MB110-8A**. В переменной **MV110_16D_DN.rEAd_1** будет отображаться текущее значение 1-го дискретного входа модуля **MB110-16Д**.

Если значение **MV110_8A.rEAd_1** превысит 30 и при этом значение **MV110_16D_DN.rEAd_1** будет равно **TRUE**, то в переменную **MU110_16R.wOut1** будет записано значение **TRUE,** что приведет к замыканию 1-го дискретного выхода модуля **МУ110-16P**. Если одно из условий перестанет выполняться, то выход будет разомкнут.

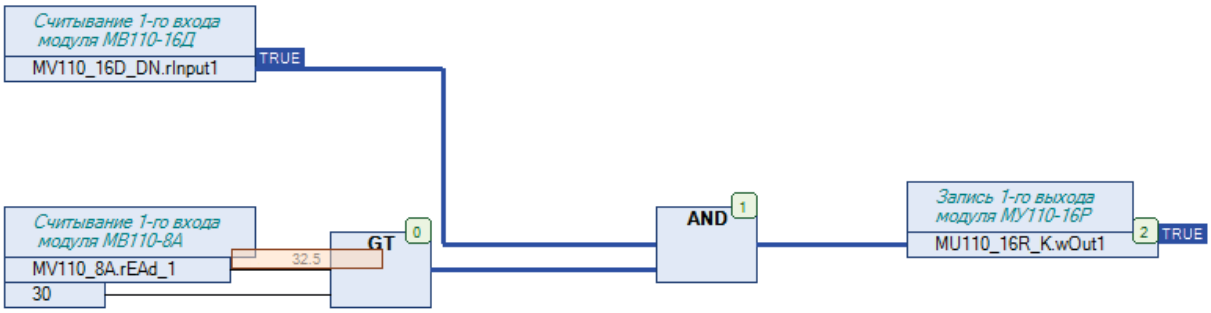


Рис. 3.26. Выполнение программы в режиме **Online**

4. Стандартные средства конфигурирования (Modbus RTU)

4.1. Общая методика конфигурирования интерфейсов

Настройка интерфейсов и протоколов обмена в **CODESYS** имеет следующую последовательность действий:

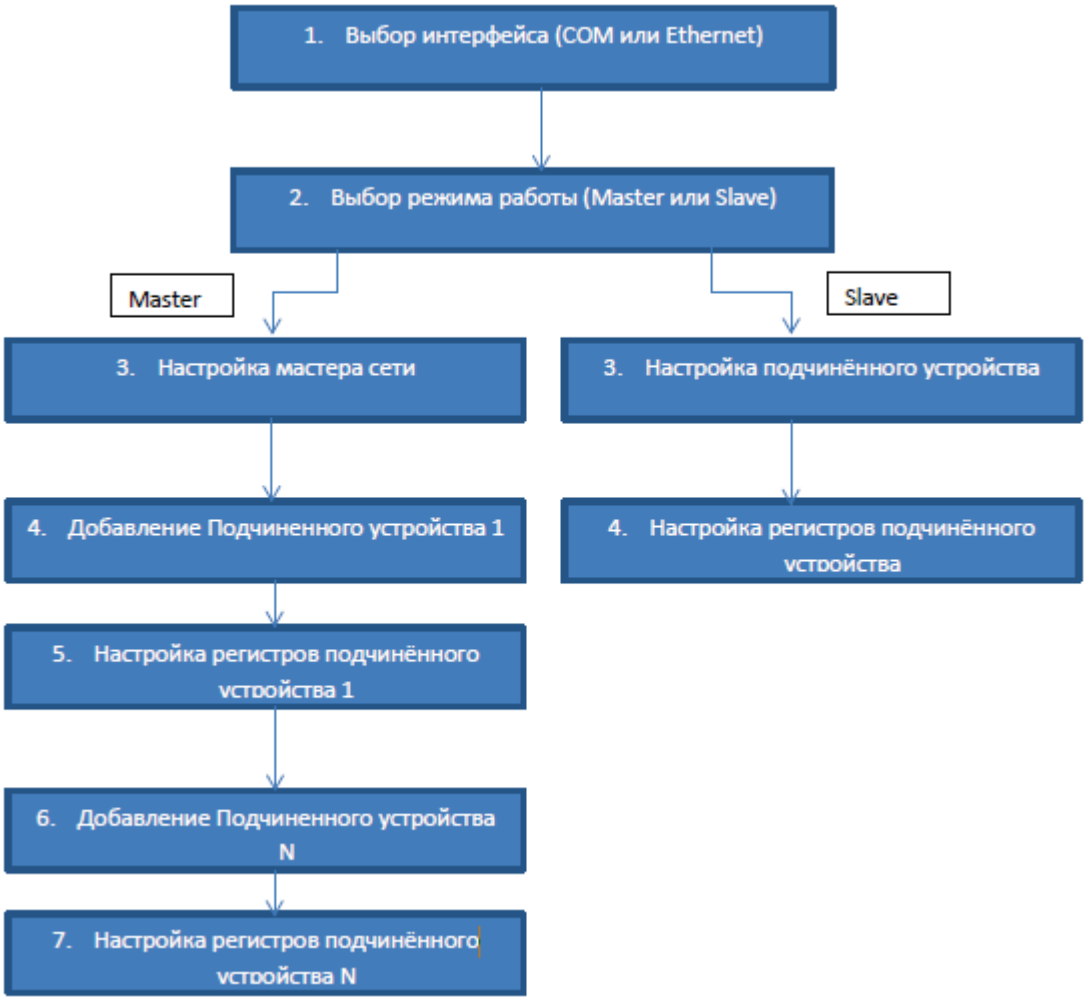


Рис. 4.1. Последовательность конфигурирования Modbus в **CODESYS**

Таким образом, сначала необходимо добавить и настроить интерфейс. После этого выбрать режим работы интерфейса – **Master** или **Slave**. Если порт работает в режиме мастера, то необходимо добавить все slave-устройства и указать для них адреса и опрашиваемые/записываемые регистры. Если порт работает в режиме slave, то достаточно указать перечень регистров, которые будут участвовать в обмене.

4.2. Настройка СПК в режиме Modbus RTU Master

1. Нажмите **ПКМ** на компонент **Device** и добавьте компонент **Modbus COM**, расположенный во вкладке **Промышленные сети/Modbus/Порт Modbus Serial**. ***Обратите внимание***, что версия компонента не должна превышать версию **target-файла** СПК. Для отображения предыдущих версий компонента поставьте галочку **Отображать все версии**. См. рекомендации в приложении Г.

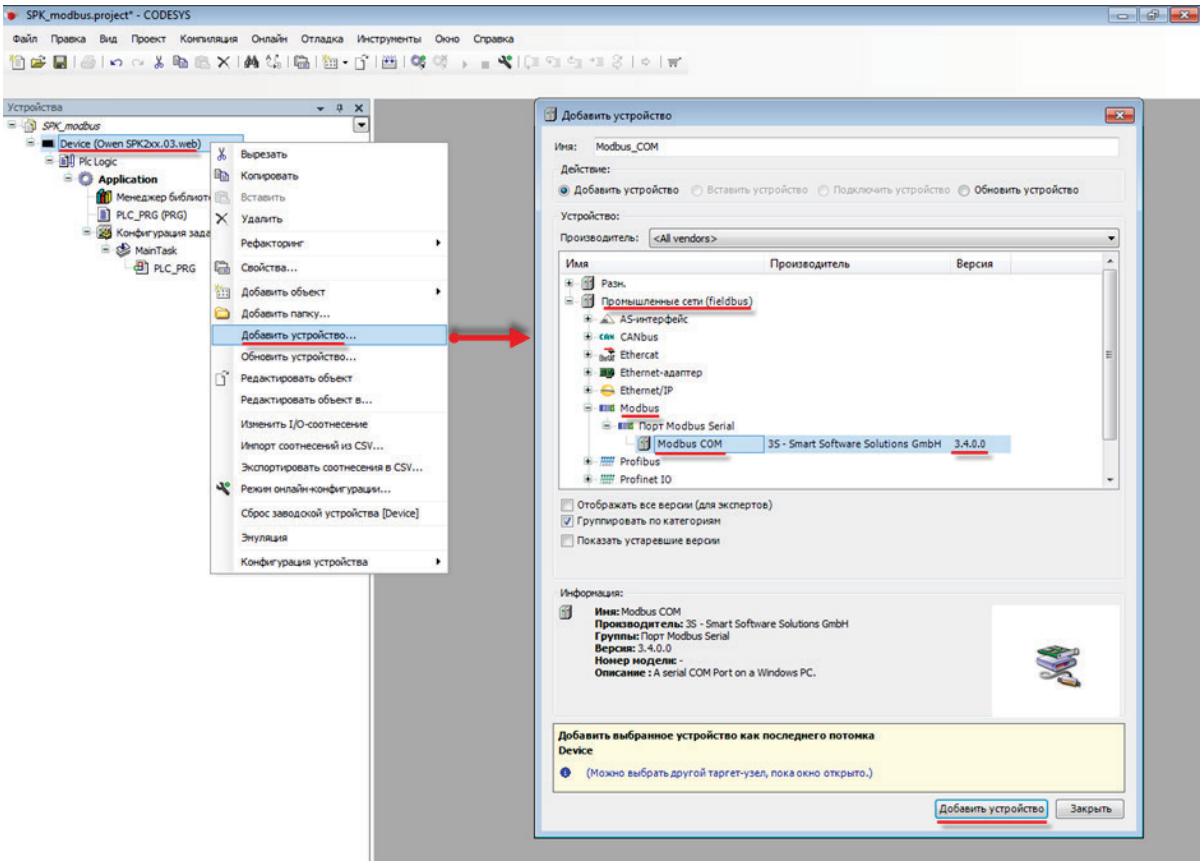


Рис. 4.2. Добавление компонента **Modbus COM**

В настройках компонента на вкладке **Общее** необходимо указать номер COM-порта СПК и его сетевые настройки. ***Обратите внимание***, что программная нумерация COM-портов отличается от приведенной на корпусе СПК (см. п. 2.3.1).

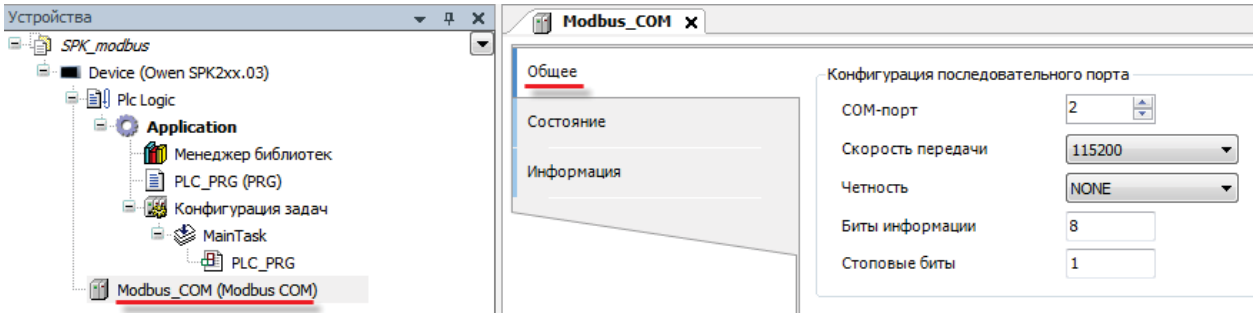


Рис. 4.3. Настройки компонента **Modbus COM**

COM-порт – номер COM-порта, используемого для передачи данных. Номер порта в CODESYS ***не соответствует*** номеру порта на корпусе СПК (см. п. 2.3.1);

Скорость передачи – измеряется в бодах, возможные значения: **4800/9600/19200/38400/57600/115200**;

Четность – режим контроля паритета: **EVEN** – четный, **ODD** – нечетный, **NONE** – отсутствует;

Информационные биты – количество информационных бит в передаваемых/принимаемых байтах, рекомендуемое значение – **8**;

Стоповые биты – количество стоповых бит (возможные значения: **1** или **2)**.

2. Нажмите **ПКМ** на компоненте **Modbus COM** и добавьте компонент **Modbus Master**, расположенный во вкладке **Промышленные сети/Modbus/Мастер Modbus Serial**.

Обратите внимание, что версия компонента не должна превышать версию **target-файла** СПК. Для отображения предыдущих версий компонента поставьте галочку **Отображать все версии**. См. рекомендации в приложении Г.

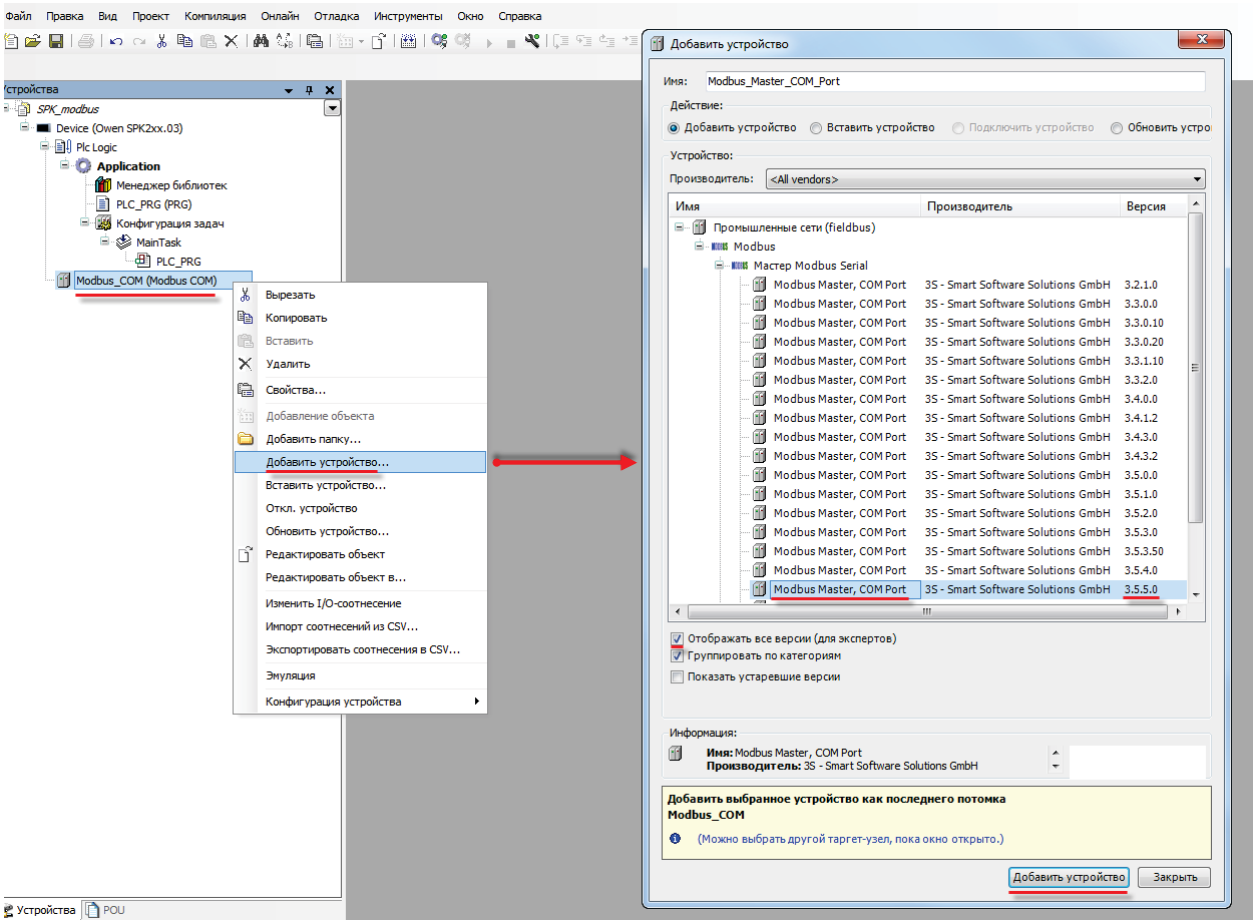


Рис. 4.4. Добавление компонента **Modbus Master**

В настройках компонента на вкладке **Общее** необходимо задать сетевые настройки мастера.

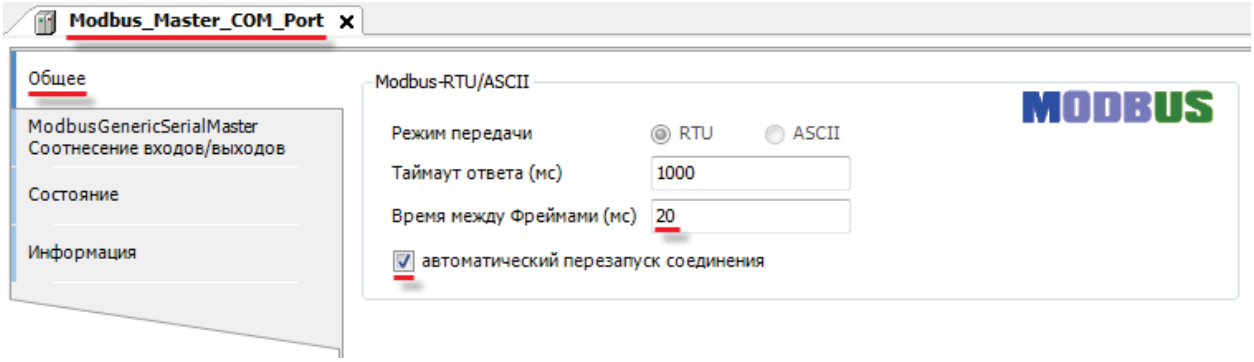


Рис. 4.5. Настройки компонента **Modbus Master**

Таймаут ответа – время, которое master дает slave-устройству на ответ. По истечению этого времени, master делает паузу на **время между фреймами** и переходит к опросу следующего slave-устройства. Значение, введенное здесь, будет по умолчанию использоваться для всех slave-устройств. На вкладке **Конфигурация Modbus Slave** (см. рис. 4.7) для каждого устройства можно задать индивидуальный таймаут отклика;

Время между фреймами – время между окончанием ответа slave-устройства и началом опроса следующего. Чем выше скорость, тем меньшим может быть это значение (на скорости 115200 бит/с – 5-10 мс). В то же время, некоторые устройства в течение определенного времени (например, **СМИ2** - 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 **Mx110** рекомендуется использовать значение **20 мс**;

Автоперезапуск соединения – при **отсутствии** галочки, не ответившее slave-устройство исключается из дальнейшего опроса***. Настоятельно рекомендуется*** всегда включать эту опцию.

3. Нажмите **ПКМ** на компоненте **Modbus Master** и добавьте компонент **Modbus Slave**, расположенный во вкладке **Промышленные сети/Modbus/Слейв Modbus Serial**. Число компонентов должно соответствовать числу slave-устройств, подключенных к COM-порту.

Обратите внимание, что версия компонента не должна превышать версию **target-файла** СПК. Для отображения предыдущих версий компонента поставьте галочку **Отображать все версии**. См. рекомендации в приложении Г.

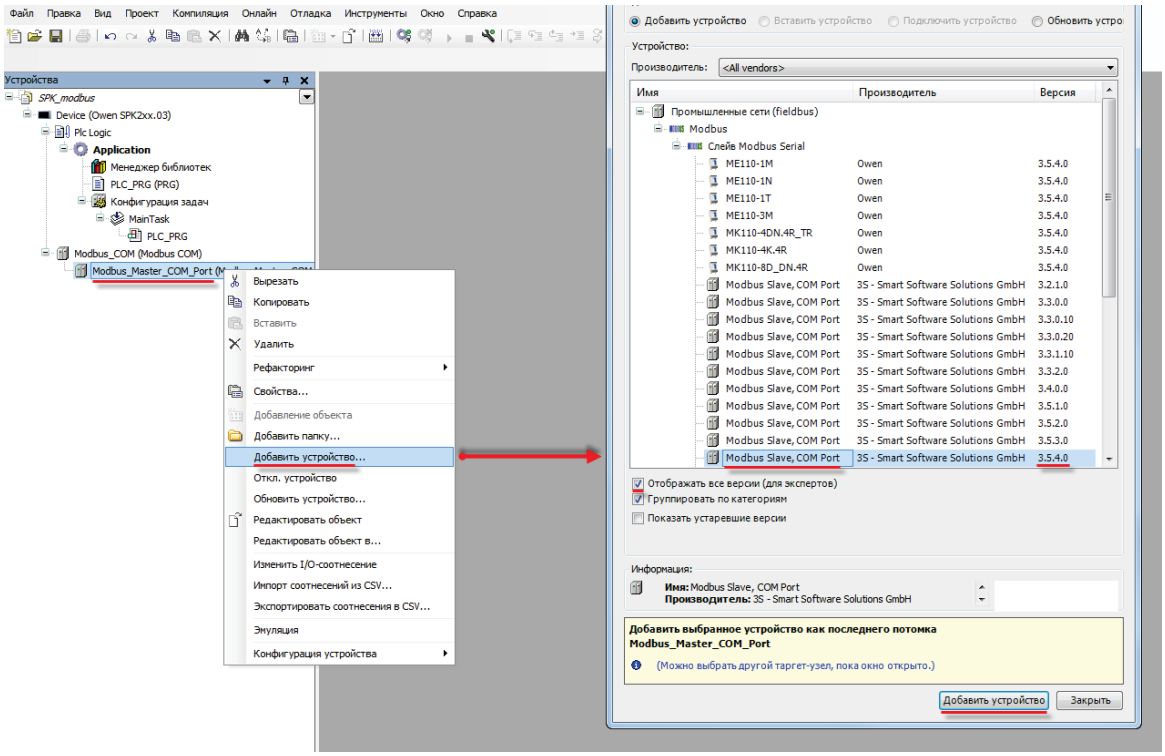


Рис. 4.6. Настройки компонента **Modbus Slave**

В настройках компонента на вкладке **Конфигурация Modbus Slave** укажите адрес slave-устройства. При необходимости можно указать индивидуальный таймаут ответа – он будет иметь приоритет по сравнению с установленным в настройках **Modbus Master** (см. рис. 4.5).

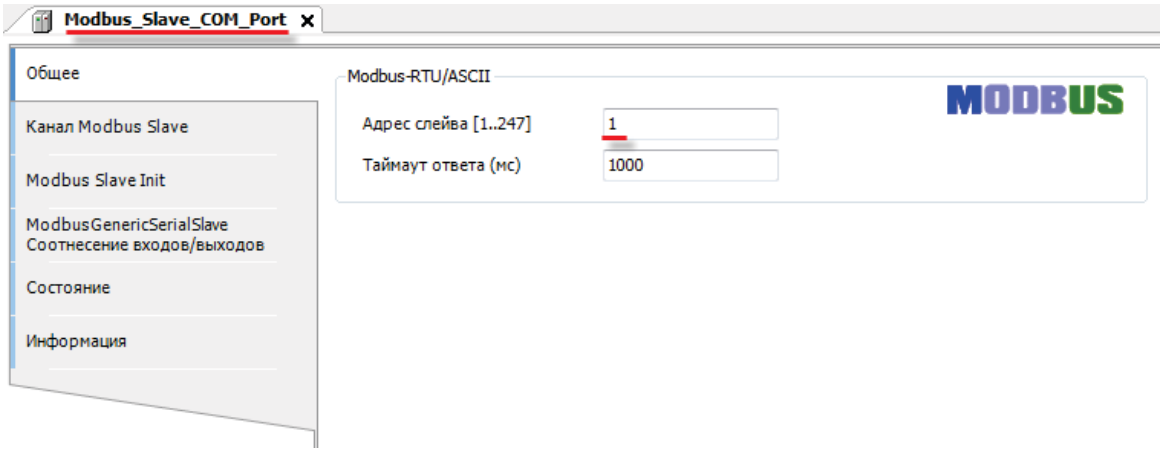


Рис. 4.7. Настройки компонента **Modbus Slave**

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

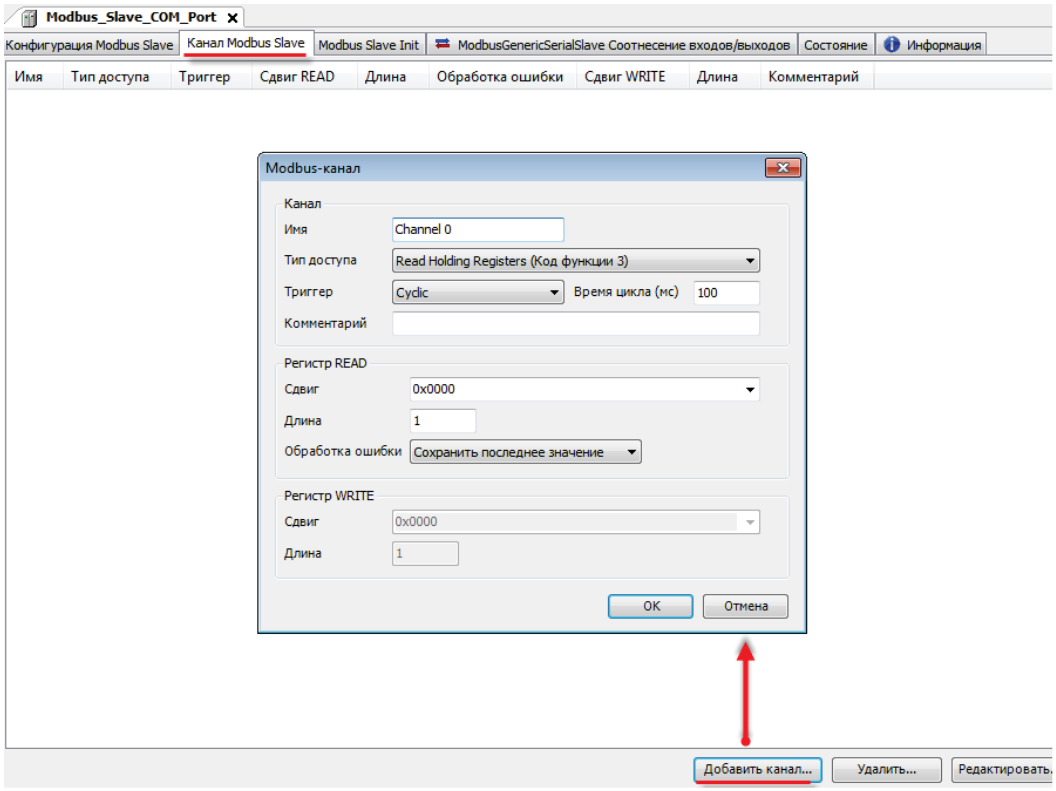


Рис. 4.8. Добавление канала **Modbus Slave**

Имя – название канала;

Тип доступа – функция, применяемая к регистрам slave-устройства (см. табл. 2.2);

Триггер – тип обращения к регистрам slave-устройства: циклически или по фронту логической переменной;

Время цикла - частота опроса канала slave-устройства. Должно быть равным или кратным времени цикла приложения. Также время цикла должно выбираться в зависимости от опрашиваемого устройства – например, для модулей MB110-8A время обновления данных одного канала для термопары типа ТХК составляет 0.4 секунды, 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 16-ричном, так и в 10-ном виде;

Длина – количество последовательных регистров, к которым применяется заданная функция (для операций группового чтения/записи).

Обработка ошибок (только для считываемых регистров) – позволяет выбрать операцию, производимую с регистром в случае возникновения ошибки обмена – сохранить последнее полученное значение или сбросить значение в **0**.

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

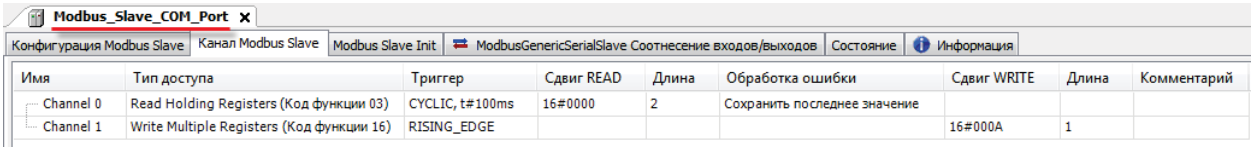


Рис. 4.9. Пример настройки каналов **Modbus Slave**

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

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

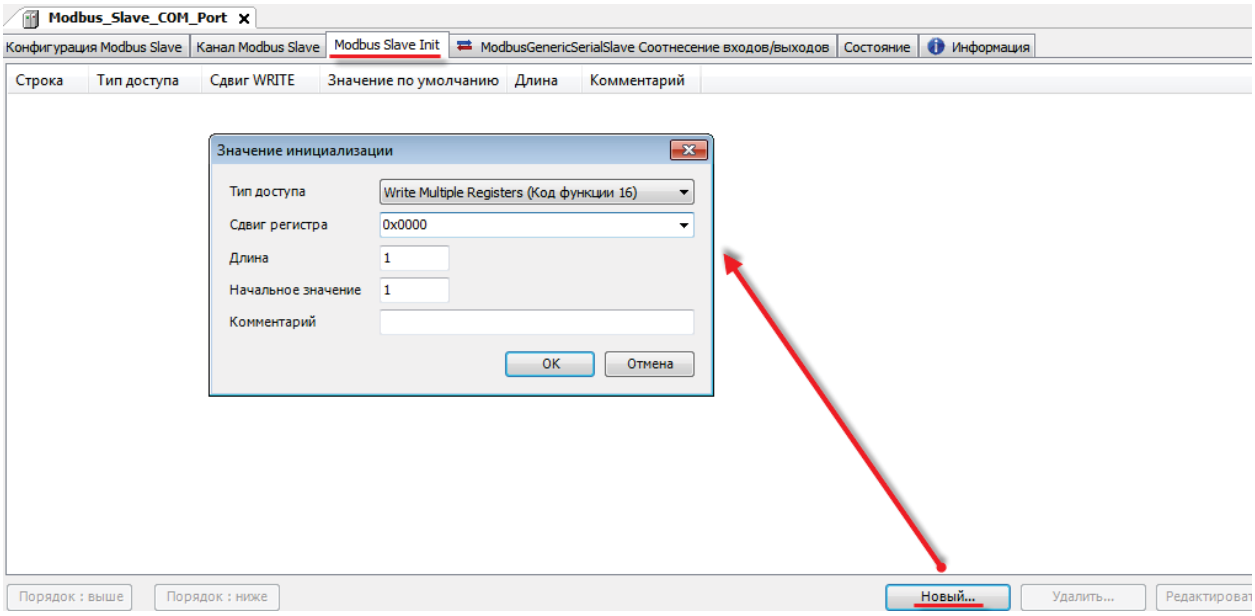


Рис. 4.10. Настройки вкладки **Modbus Slave Init**

На вкладке **ModbusGenericSerialSlave Соотнесение входов/выходов** осуществляется привязка переменных программы к каналам. 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.

Обратите внимание, что для корректного обмена данными во вкладке **Всегда обновлять переменные** необходимо выбрать значение **Включено 2 (Всегда в задаче цикла шины)**.

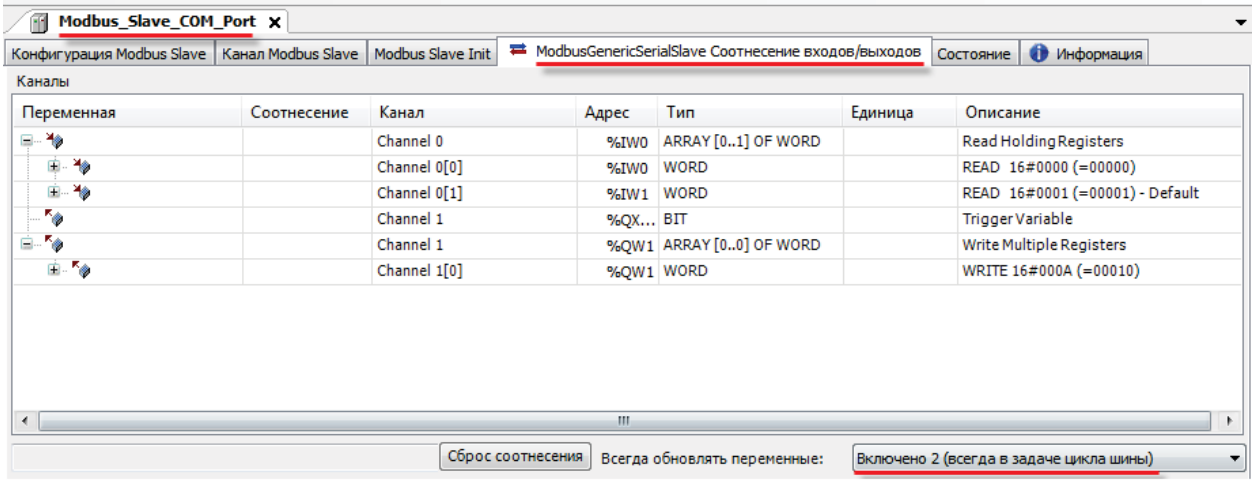


Рис. 4.11. Настройки вкладки **ModbusGenericSerialSlave Соотнесение входов/выходов**

Для привязки переменных два раза нажмите **ЛКМ** на ячейку столбца **Переменная**, после чего выберите необходимую переменную программы с помощью **Ассистента ввода**:

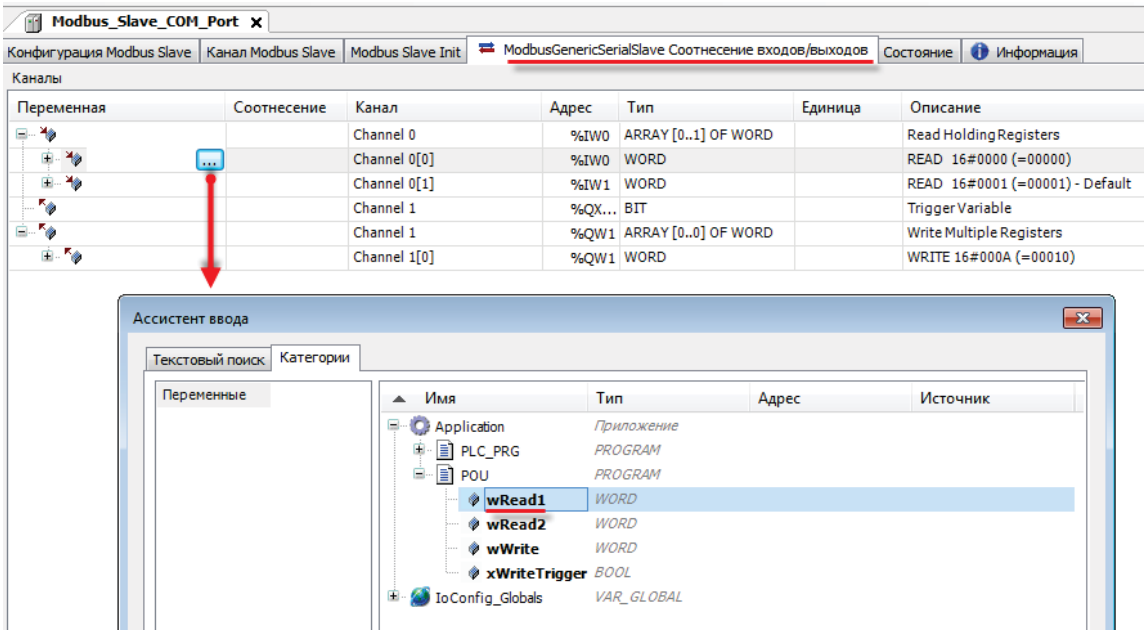


Рис. 4.12. Привязка переменных программы к каналам **Modbus Slave**

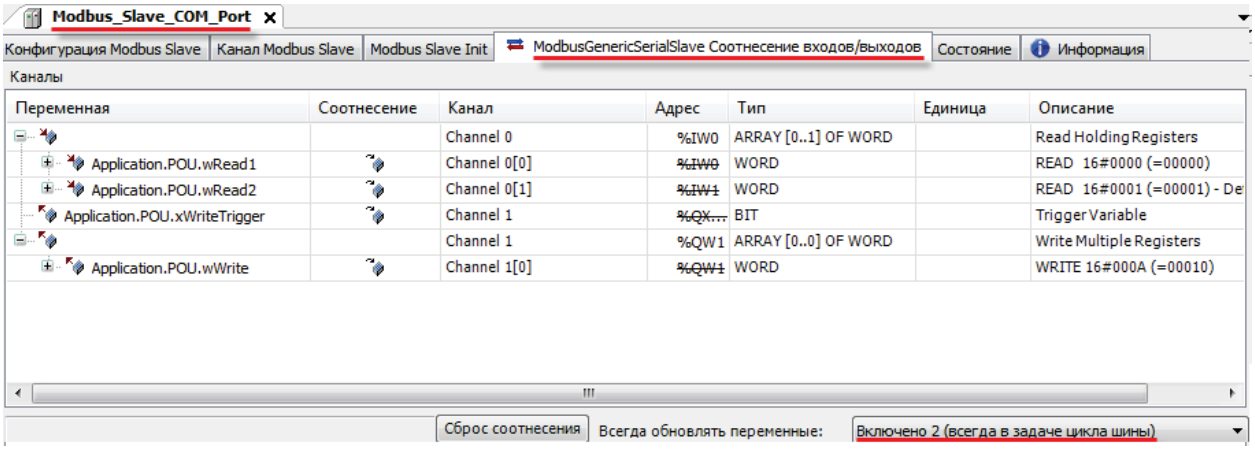


Рис. 4.13. Привязка переменных программы к каналам **Modbus Slave**

На этом настройка СПК в режиме **Modbus RTU Master** завершена.

Пример настройки СПК как **Modbus RTU Master** для опроса модулей **Mx110** приведен в п. 4.6.

4.3. Настройка СПК в режиме Modbus RTU Slave

1. Нажмите **ПКМ** на компонент **Device** и добавьте компонент **Modbus COM**, расположенный во вкладке **Промышленные сети/Modbus/Порт Modbus Serial**:

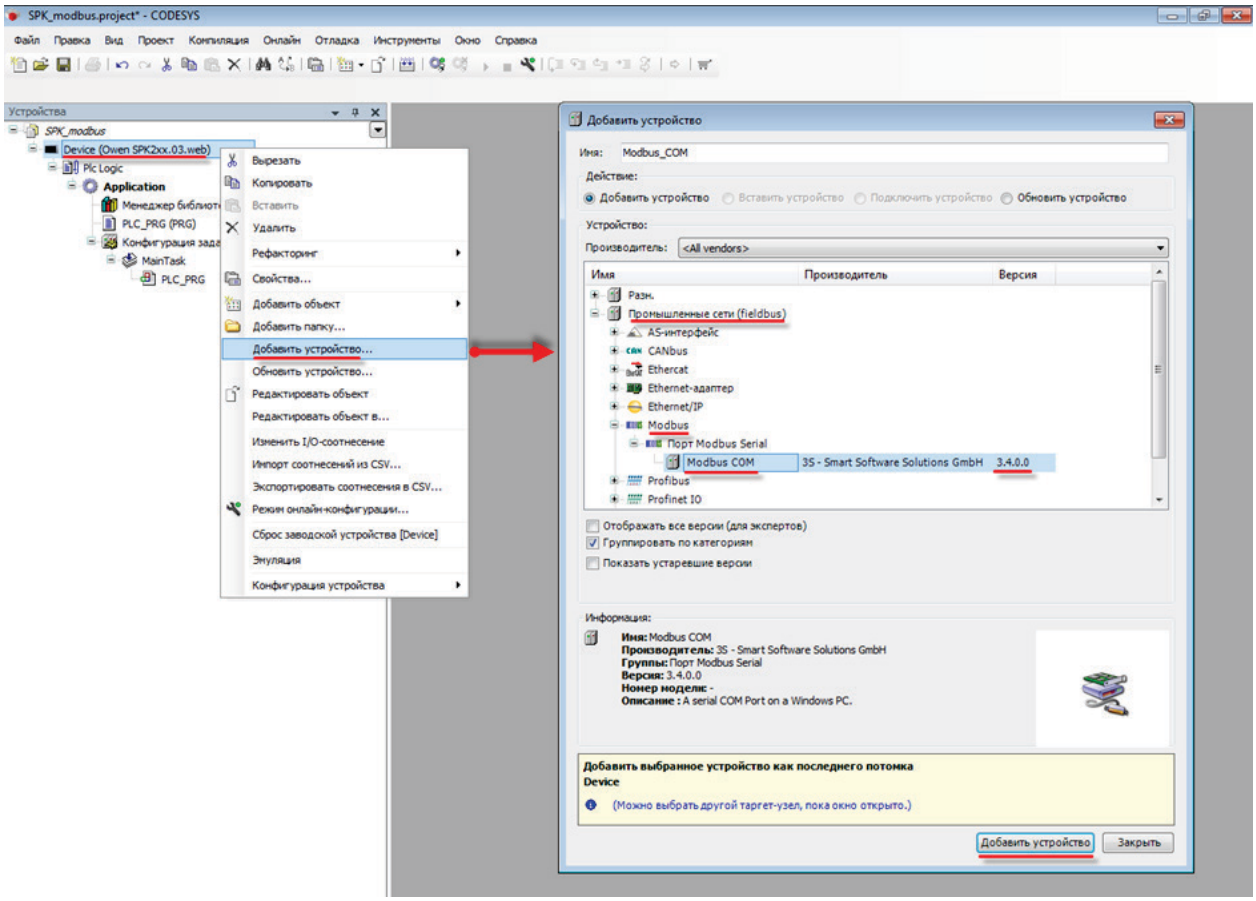


Рис. 4.14. Добавление компонента **Modbus COM**

В настройках компонента на вкладке **Общее** необходимо указать номер COM-порта СПК и его сетевые настройки. ***Обратите внимание***, что программная нумерация COM-портов отличается от приведенной на корпусе СПК (см. п. 2.3.1).

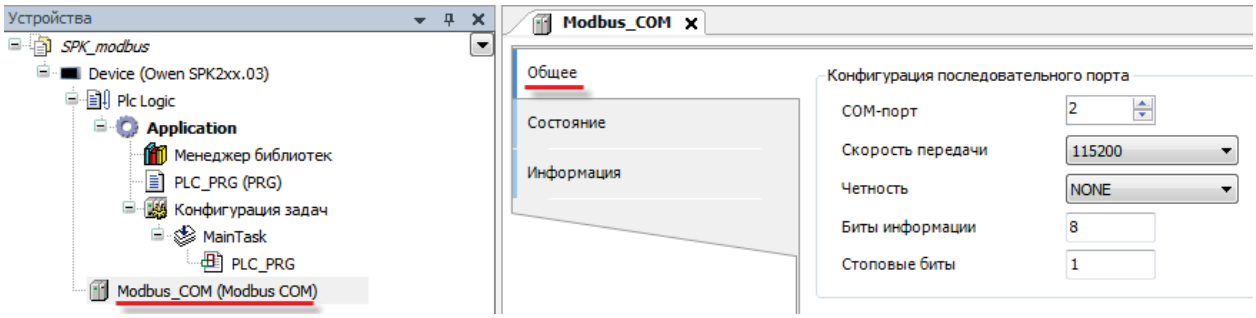


Рис. 4.15. Настройки компонента **Modbus COM**

COM-порт – номер COM-порта, используемого для передачи данных. Номер порта в CODESYS ***не соответствует*** номеру порта на задней панели СПК (см. п. 2.3.1);

Скорость передачи – измеряется в бодах, возможные значения: **4800/9600/19200/38400/57600/115200**;

Четность – режим контроля паритета: **EVEN** – четный, **ODD** – нечетный, **NONE** – отсутствует;

Информационные биты – количество информационных бит в передаваемых/принимаемых байтах, рекомендуемое значение – **8**;

Стоповые биты – количество стоповых бит (возможные значения: **1** или **2)**.

2. Нажмите **ПКМ** на компоненте **Modbus COM** и добавьте компонент **Modbus Serial Device**, расположенный во вкладке **Промышленные сети/Modbus/Устройство Modbus Serial**.

Обратите внимание, что версия компонента не должна превышать версию **target-файла** СПК. Для отображения предыдущих версий компонента поставьте галочку **Отображать все версии**. См. рекомендации в приложении Г.

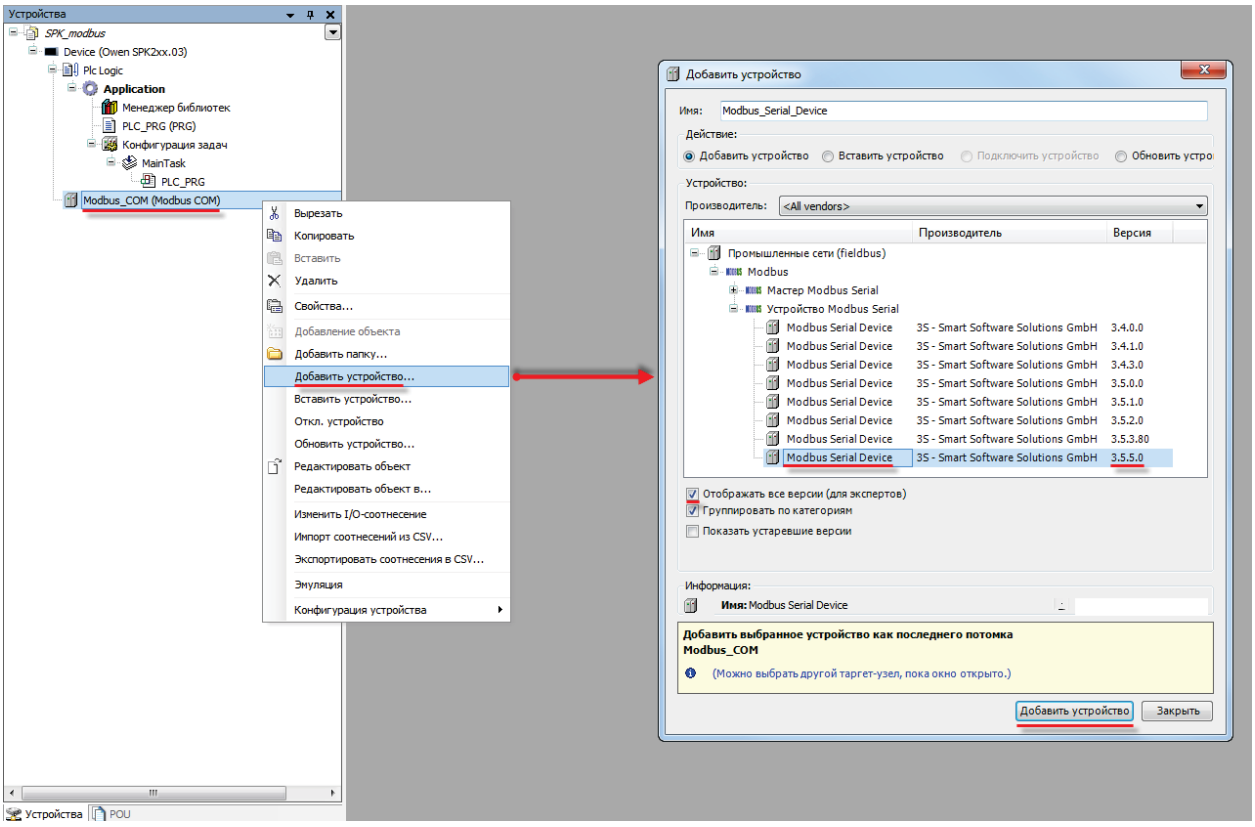


Рис. 4.16. Добавление компонента **Modbus Serial Device**

На вкладке **Modbus Serial Device** укажите настройки slave-устройства:

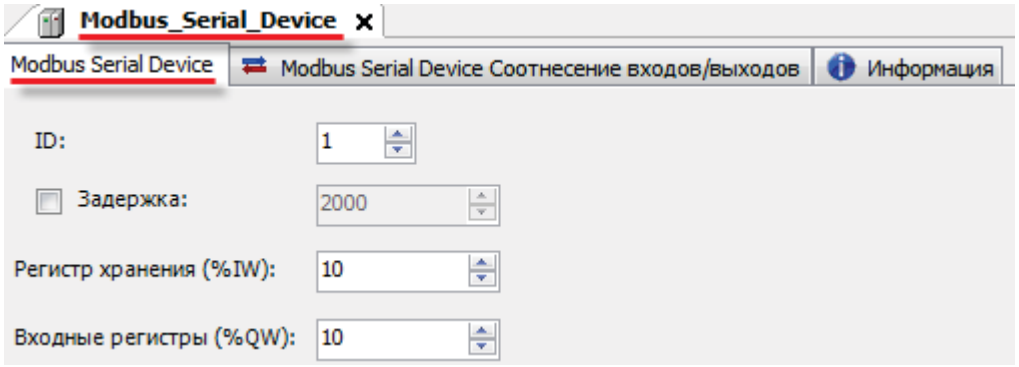


Рис. 4.17. Настройки компонента **Modbus Serial Device**

ID – адрес (**Slave ID**), который будет назначен данному COM-порту СПК;

Задержка – время ожидания (в мс) запроса от master-устройства. Если за это время запроса не происходит, то данные в регистрах обнуляются. ***При отсутствии галочки*** обнуления данных ***не происходит***;

Регистры хранения – число регистров хранения (holding registers) для данного slave-устройства;

Входные регистры – число входных регистров (input registers) для данного slave-устройства.

Обратите внимание на различия входных регистров и регистров хранения.

Область данных	Обозначение	Тип данных	Тип доступа	Изменение из программы
Holding Registers (Регистры хранения)	4x (%IW)	WORD	чтение/запись	невозможно
Input Registers (Регистры ввода)	3x (%QW)	WORD	только чтение	возможно

В определенных ситуациях требуется возможность считывать/записывать данные в slave-устройство, и при этом иметь возможность изменять их из программы slave-устройства. Как видно из таблицы, в **стандартных средствах конфигурирования CODESYS** такой функционал отсутствует. В этом случае следует использовать библиотеку ModbusSlave.

Обратите внимание, что область памяти **1x** (**Discrete Inputs**) наложена на **3x** (**Input Registers**), а область **0x** (**Coils**) – на **4x** (**Holding Registers**). Поддержаны все функции работы с битами.

На вкладке **Modbus Serial Slave Соотнесение входов/выходов** осуществляется привязка переменных программы к **holding** и **input** 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.

Обратите внимание, что для корректного обмена данными во вкладке **Всегда обновлять переменные** необходимо выбрать значение **Включено 2 (Всегда в задаче цикла шины)**.

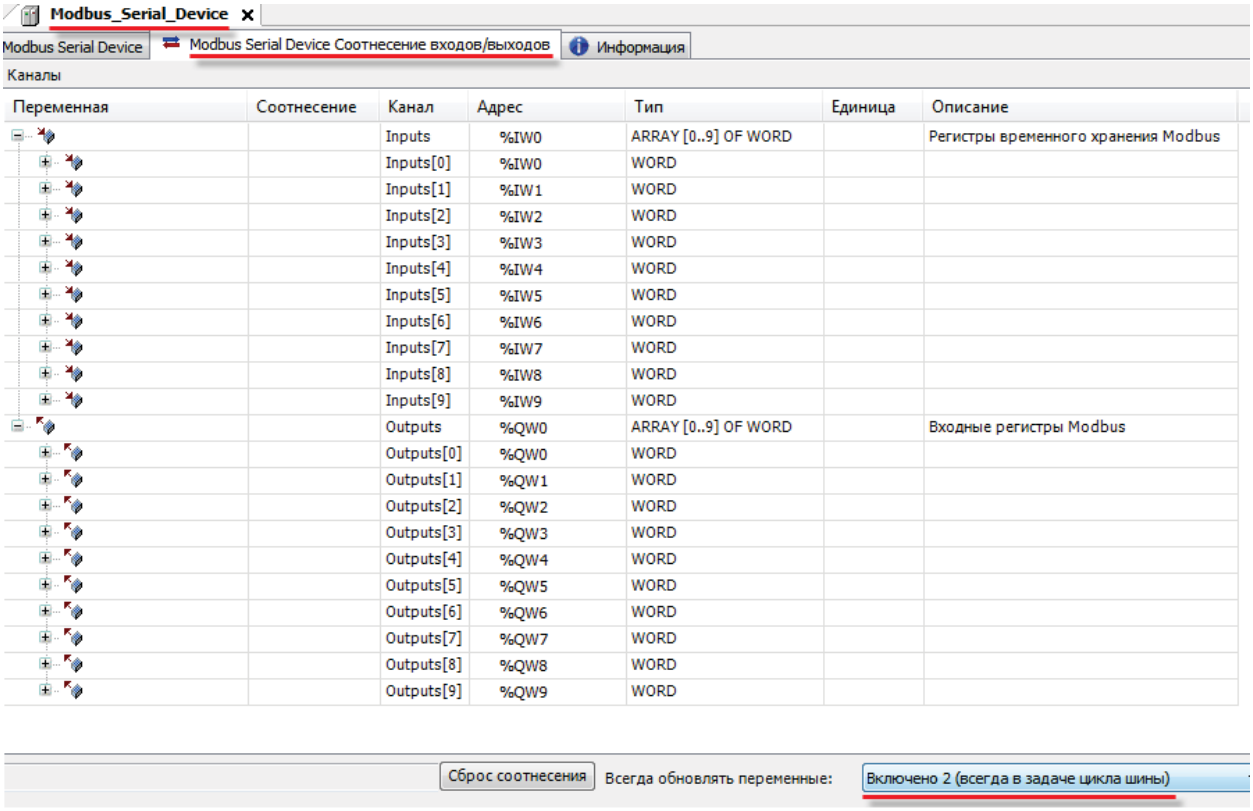


Рис. 4.18. Настройки вкладки **Modbus Serial Slave Соотнесение входов/выходов**

Для привязки переменных два раза нажмите **ЛКМ** на ячейку столбца **Переменная**, после чего выберите необходимую переменную программы с помощью **Ассистента ввода**:

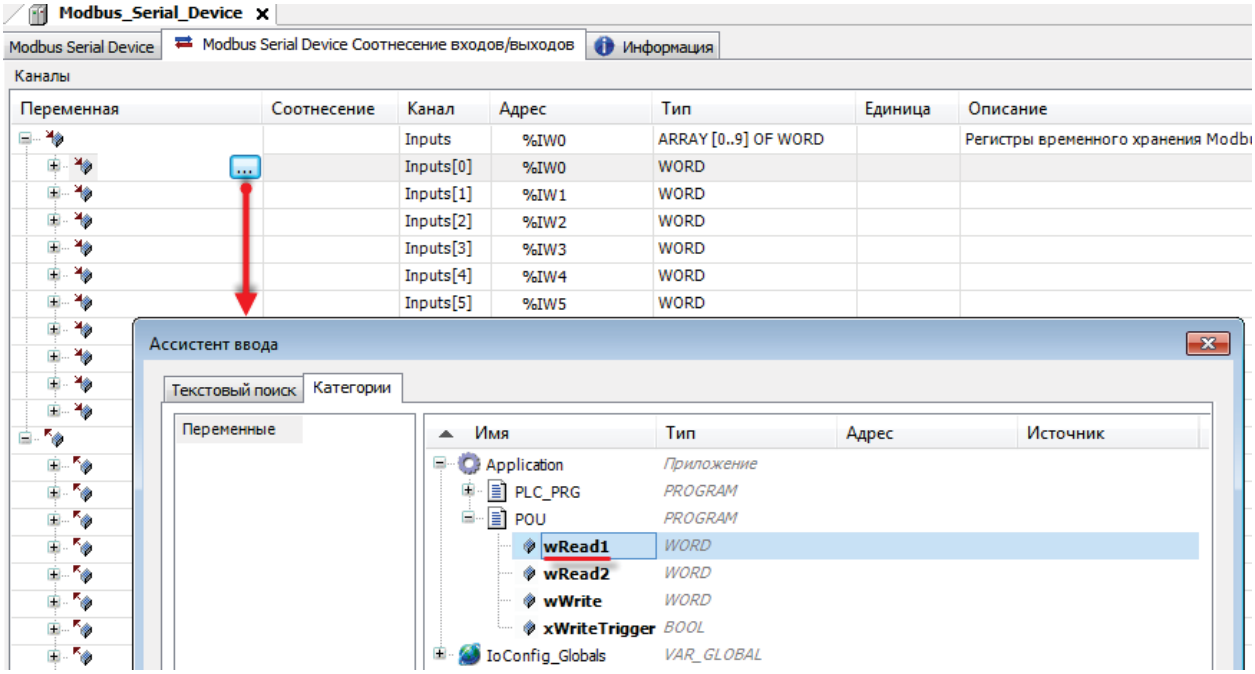


Рис. 4.19. Привязка переменных программы к регистрам СПК

На этом настройка СПК в режиме **Modbus RTU Slave** завершена.

Пример настройки СПК как **Modbus RTU Slave** приведен в п. 4.7.

4.4. Преобразование данных (REAL, DWORD, STRING)

Стандарт **Modbus** предусматривает только два типа данных, участвующих в обмене – **BOOL** и **WORD**. Достаточно часто возникает потребность передать данные других типов, например, **REAL** и **STRING**. В этом случае на устройстве, которое отравляет данные, необходимо преобразовать их в последовательность **WORD** регистров. Соответственно, на устройстве, получающем данные, должно быть выполнено обратное преобразование.

Значение переменных типа **REAL** и **DWORD** будет занимать **два WORD**; значение переменной типа **STRING** будет занимать количество **WORD**, равное **половине длины строки** (каждый WORD содержит два байта, каждый символ занимает байт).

В **CODESYS V3.5** используются два основных способа подобных преобразований: объединения и указатели.

4.4.1. 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.

Рассмотрим конвертацию значения с плавающей точкой, хранящегося в двух **WORD**, в переменную типа **REAL**:

1. Нажмите **ПКМ** на приложение **Application** и добавьте объект **DUT** типа **объединение** с названием **Real_Word**:

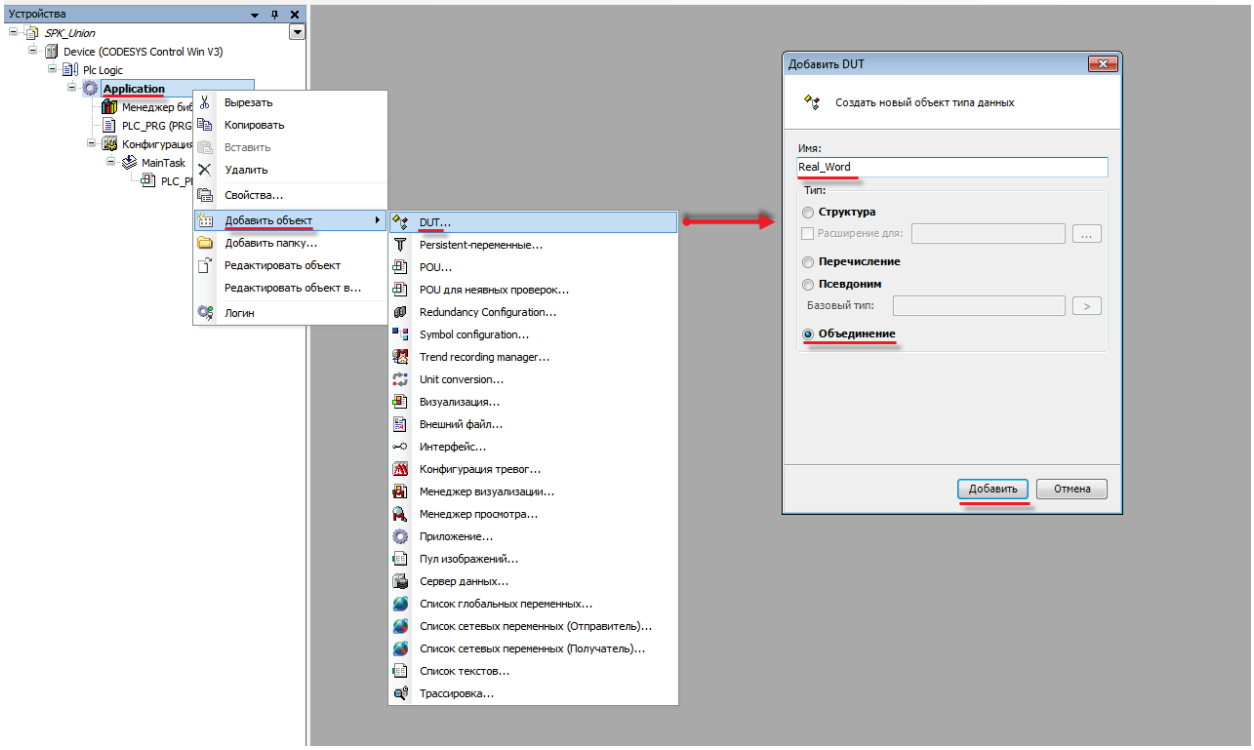


Рис. 4.20. Добавление в проект объединения

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

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

Рис. 4.21. Объявление переменных объединения

3. В программе объявим экземпляр объединения **Real_Word** с названием **_2WORD_TO_REAL**:

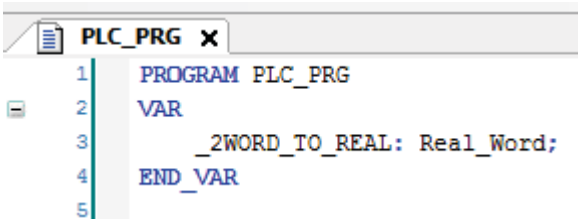


Рис. 4.22. Объявление экземпляра объединения в программе

Для использования переменных объединения в нужном месте программы введите имя экземпляра объединения и нажмите точку, после чего выберите из списка нужную переменную:

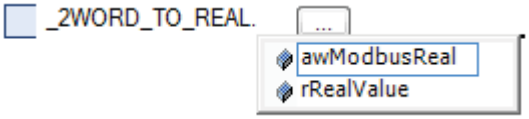


Рис. 4.23. Работа с переменными объединения в программе

4. Переменные массива **awModbusReal** будут привязаны к регистрам при настройке **Modbus**, а переменная **rRealValue** будет использоваться в программе.

Ниже приведен следующий пример: СПК является мастером и считывает значение типа **REAL** из 10 и 11 holding регистров slave-устройства в переменную **rRealValue** объединения **_2WORD_TO_REAL**:

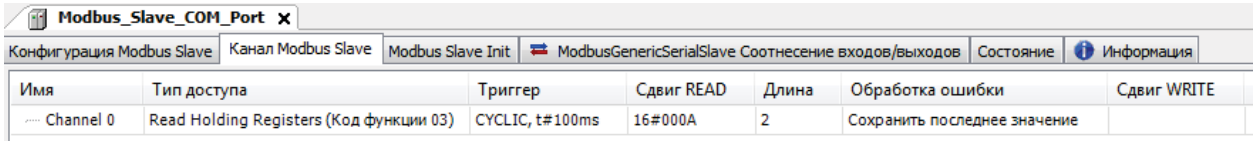


Рис. 4.24. Настройка канала для считывания значения с плавающей точкой

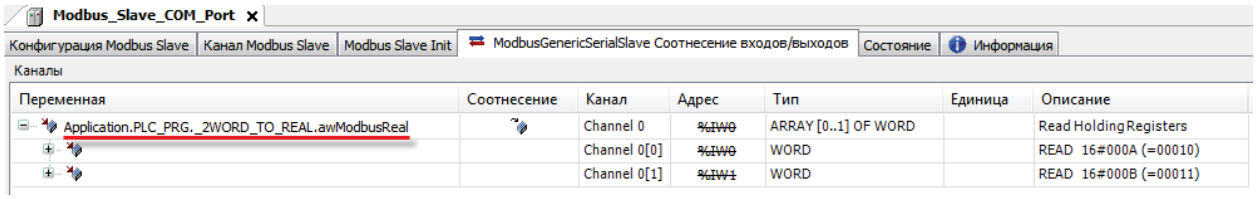


Рис. 4.25. Привязка переменной объединения к каналу

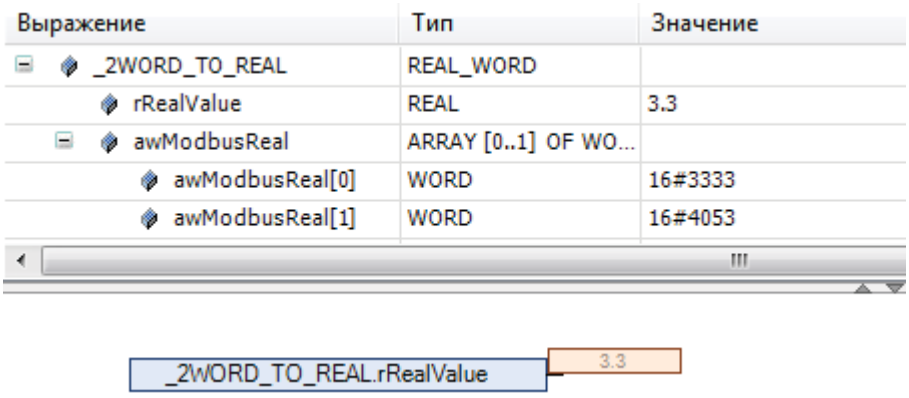


Рис. 4.26. Использование **REAL** переменной объединения в программе

Очевидно, что 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** и **STRING** происходит совершенно аналогично – в приведенном выше примере достаточно изменить тип переменной объединения (вместо **rRealValue** использовать **dwDwordValue** типа **DWORD** или **sStringValue** типа **STRING**). Напоминаем, что количество **WORD** переменных, необходимых для передачи **STRING**, равно половине длины передаваемой строки.

5. Передача **REAL** по протоколу **Modbus** не стандартизирована; как уже упоминалось, значения с плавающей точкой передаются как два **WORD**, но порядок этих **WORD** (или даже их байт) может отличаться. В этом случае необходимо привести их к нужному виду.

Порядок **WORD** можно менять на этапе привязки переменных к регистрам – например, сравните рис. 4.25 и рис. 4.27:

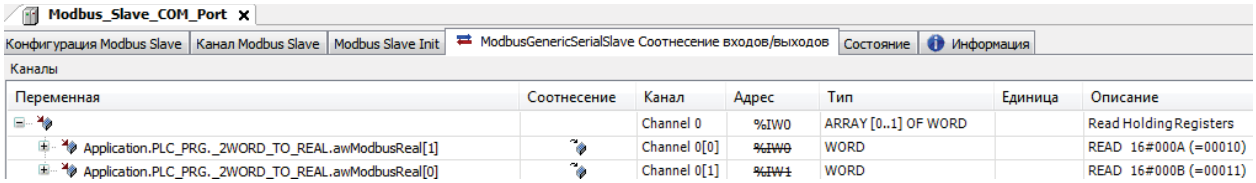


Рис. 4.27. Привязка элементов массива объединения к регистрам канала

.

При необходимости менять порядок байт можно создать два объединения – в первом будет происходить конвертация полученных по **Modbus** значений **WORD** в массив байт, а во втором – конвертация нового массива байт (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):

Word_Bytes

```
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
```

Bytes_Real

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

Рис. 4.28. Объявление двух объединений

Выражение	Тип	Значение	Подготовленное ...	Адрес	Комментарий
_2WORD_TO_4BYTES	Word_Bytes				
awModbusReal	ARRAY [0..1] OF WO...				
awModbusReal[0]	WORD	16#5340			
awModbusReal[1]	WORD	16#3333			
abyModbusReal	ARRAY [0..3] OF BYTE				
abyModbusReal[0]	BYTE	16#40			
abyModbusReal[1]	BYTE	16#53			
abyModbusReal[2]	BYTE	16#33			
abyModbusReal[3]	BYTE	16#33			
_4BYTES_TO_REAL	Bytes_Real				
abyModbusReal	ARRAY [0..3] OF BYTE				
abyModbusReal[0]	BYTE	16#33			
abyModbusReal[1]	BYTE	16#33			
abyModbusReal[2]	BYTE	16#53			
abyModbusReal[3]	BYTE	16#40			
rRealValue	REAL	3.3			

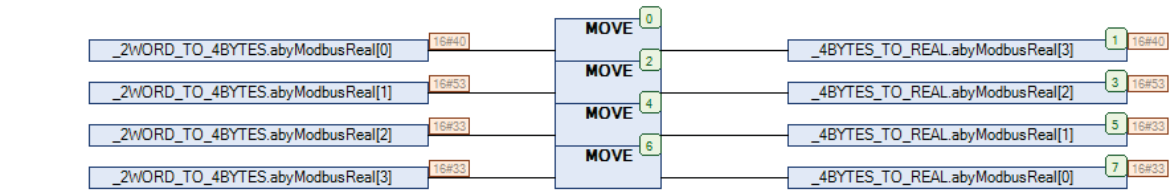


Рис. 4.29. Пример работы с объединениями на языке **CFC**. Перестановка байт

4.4.2. 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 интерпретацию записанных в нее данных.

Обратите внимание, что работа с указателями рекомендуется лишь квалифицированным программистам. Некорректное использование указателей может привести к зависанию программы и контроллера.

Рассмотрим конвертацию значения с плавающей точкой, хранящегося в двух **WORD**, в переменную типа **REAL**:

1. Объявите 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
```

Рис. 4. 30. Объявление указателя

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

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

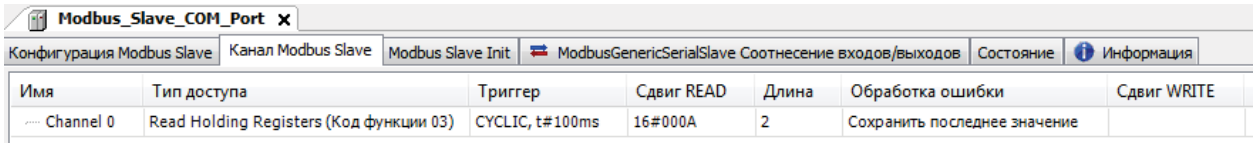


Рис. 4.31. Настройка канала для считывания значения с плавающей точкой

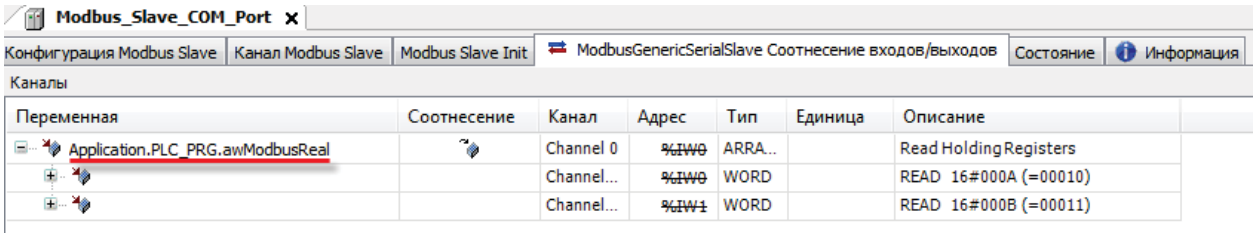


Рис. 4.32. Привязка переменной к каналу

3. В программе запишите в указатель адрес массива **awModbusReal**, после чего присвойте переменной **rRealValue** значение, хранящееся по указателю:

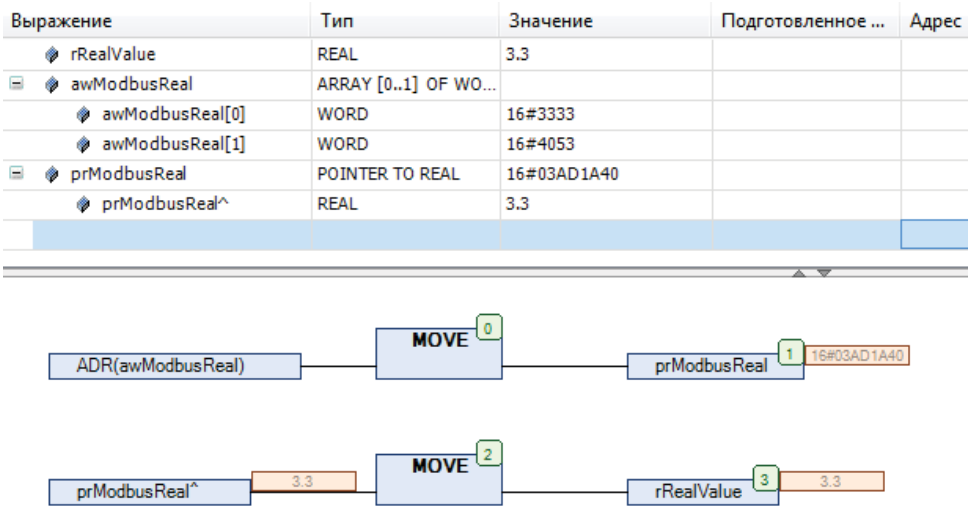


Рис. 4.33. Пример работы с указателями на языке **CFC** (конвертация **2 WORD** в **REAL**)

Очевидно, что обратное преобразование (**REAL** в два **WORD**) выполняется аналогичным способом:

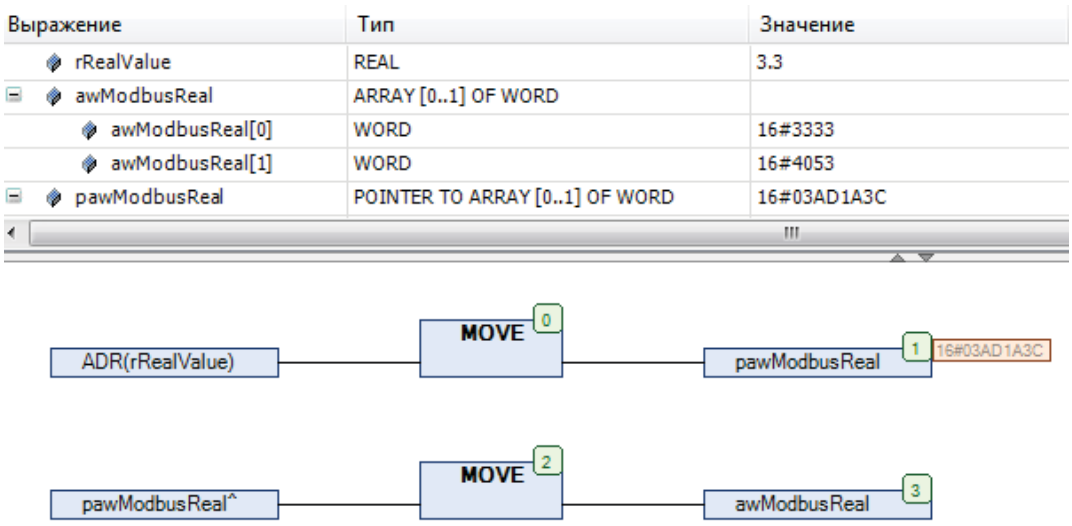


Рис. 4.34. Пример работы с указателями на языке **CFC** (конвертация **REAL** в 2 **WORD**)

Работа с **DWORD** и **STRING** происходит совершенно аналогично – в приведенном выше примере достаточно изменить тип переменной объединения и ее указателя (например, использовать **dwDwordValue** типа **DWORD** и **pdwModbusDword** типа **POINTER TO DWORD**). Напоминаем, что количество **WORD** переменных, необходимых для передачи **STRING**, равно половине длины передаваемой строки

4. Передача **REAL** по протоколу **Modbus** не стандартизирована; как уже упоминалось, значения с плавающей точкой передаются как два **WORD**, но порядок этих **WORD** (или даже их байт) может отличаться. В этом случае необходимо привести их к нужному виду.

Порядок **WORD** можно менять на этапе привязки переменных к каналам или регистрам – например, сравните рис. 4.32 и рис. 4.35:

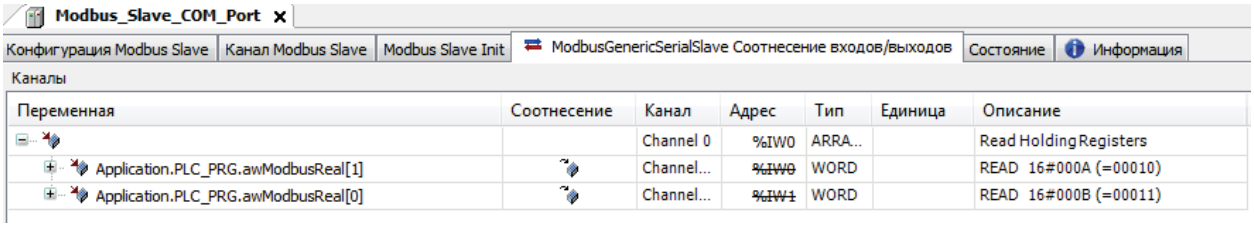


Рис. 4.35. Привязка элементов массива к регистрам канала

При необходимости менять порядок байт нужно 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):

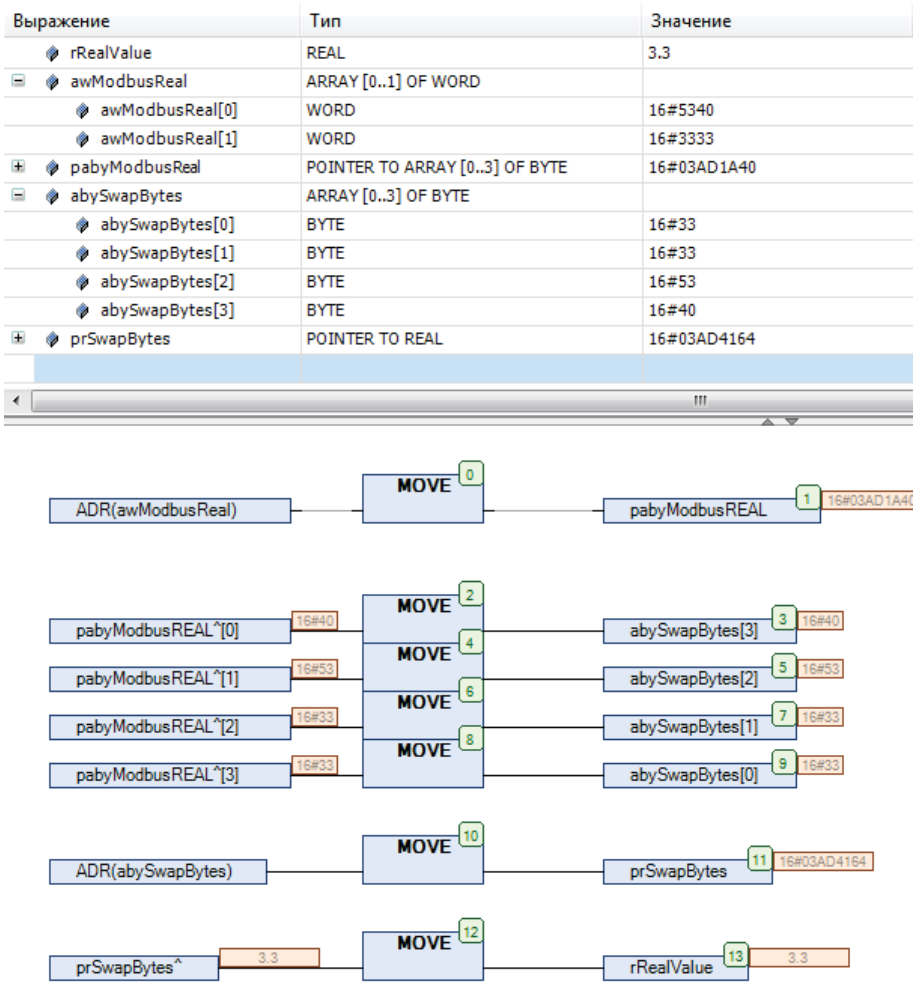


Рис. 4.36. Пример работы с указателями на языке **CFC**. Перестановка байт

4.5. Диагностика обмена

В случае необходимости контролировать процесс обмена данными, можно воспользоваться переменными диагностики. Для этого в нужном месте программы введите имя компонента из дерева проекта, поставьте точку и из выпадающего списка выберите нужную переменную диагностики:

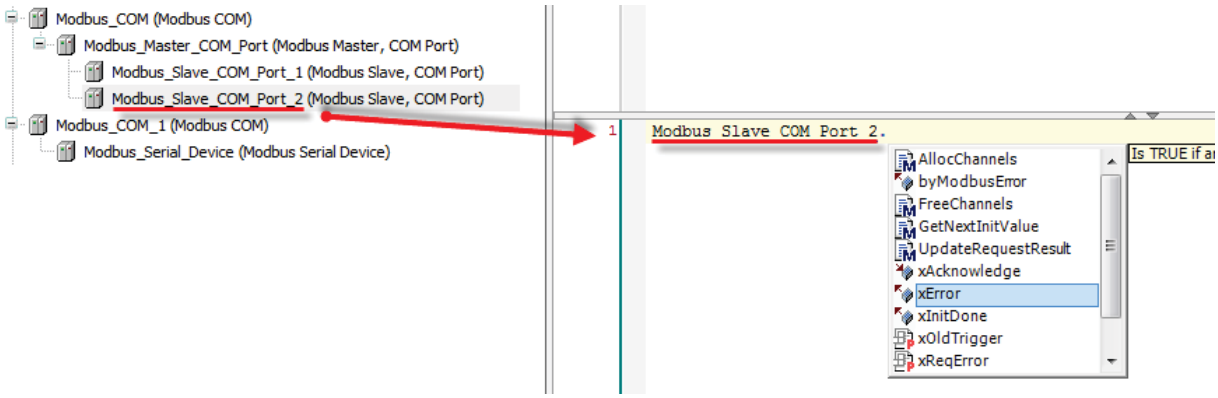


Рис. 4.37. Использование переменных диагностики в программе

Список переменных диагностики приведен приложении В.

4.6. Пример: СПК207 + модули Mx110 (MB110-8A, MB110-16Д, МУ110-16Р)

Рассмотрим пример настройки обмена с **модулями Mx110** с использованием **стандартных средств CODESYS**. В нем мы наладим связь между контроллером **СПК207.03** (master) и тремя модулями (slave-устройствами).

Реализуемый алгоритм: если значение 1-го аналогового входа модуля **MB110-8A** превышает 30 и при этом 1-ый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то произвести однократную запись в 1-й дискретный выход модуля **МУ110-16Р** значения **TRUE** (замкнут). Во всех остальных случаях присвоить дискретному выходу значение **FALSE** (разомкнут).

Структурная схема примера приведена на рис. 4.38:

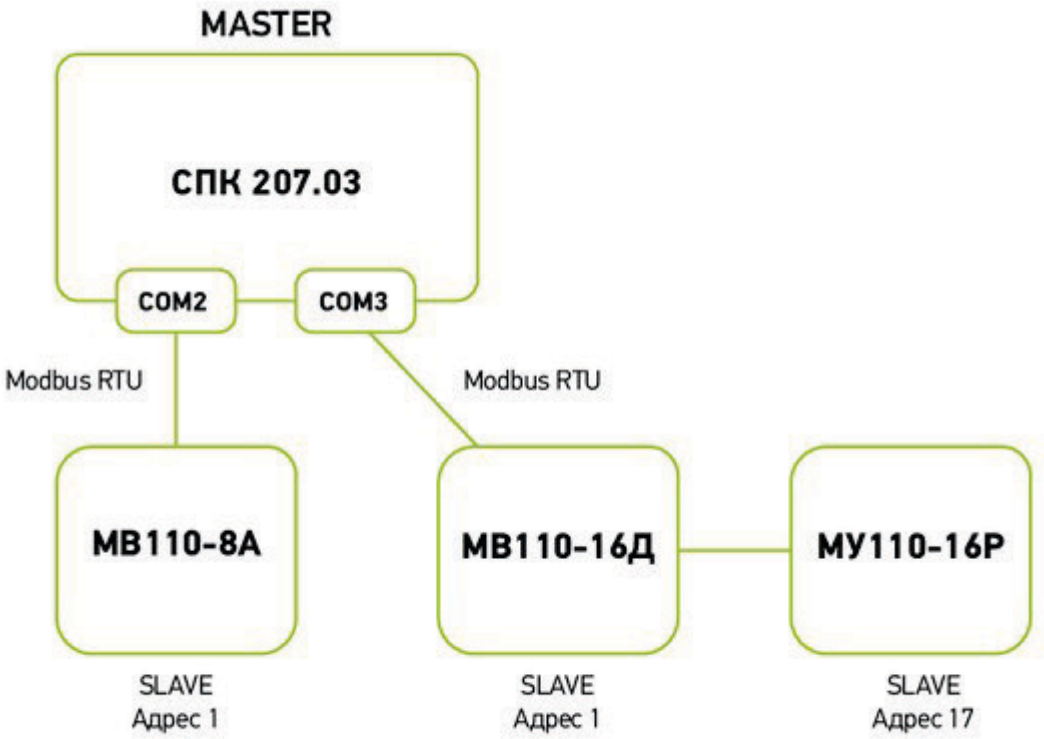


Рис. 4.38. Структурная схема примера **Стандартные средства конфигурирования (СПК+модули Mx110)**

Пример создан в среде **CODESYS 3.5 SP7 Patch4** и подразумевает запуск на **СПК207.03.CS(-WEB)**.

Пример доступен для скачивания: Example_StandardConfiguration_Mx110.projectarchive

Сетевые параметры модулей приведены в табл. 4.1.

Список переменных, используемых в примере, приведен в табл. 4.2.

Табл. 4.1. Сетевые параметры модулей Мх110

Параметр	**МВ110-8А**	**МВ110-16Д**	**МУ110-16Р**
COM-порт СПК, к которому подключен модуль	*COM2*	*COM3*	
Адрес модуля	1	1	17
Скорость обмена	115200		
Количества бит данных	8		
Контроль четности	отсутствует		
Количество стоп-бит	1		

Табл. 4.2. Список переменных программы

Модуль	**Имя переменной**	**Тип**	**Описание**
МВ110-8А	awModbusReal	ARRAY [0..1] OF WORD	Значение температуры в виде двух **WORD**, считываемое с модуля.
	rRealValue	REAL	Значение температуры в виде числа с плавающей точкой для использования в программе.
МВ110-16Д	wDI	WORD	Значение дискретных входов в виде битовой маски. При обращении к отдельным входам указывается их номер, начиная с 0: **wDI.0** – состояние первого входа (TRUE/FALSE) **wDI.1** – 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: **wDO.0** – состояние первого выхода (TRUE/FALSE) **wDO.1** – 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).

1. Настройте модули **Mx110** с помощью программы **Конфигуратор Mx110** в соответствии с табл. 4.1. Подключите модули к COM-портам СПК (в соответствии с рис. 4.38).

2. Выберите режим работы COM-портов **RS-485** в конфигураторе СПК (см. п. 2.3.3).

3. Создайте новый проект **CODESYS** для **СПК207.03** с программой **PLC_PRG** на языке **CFC**:

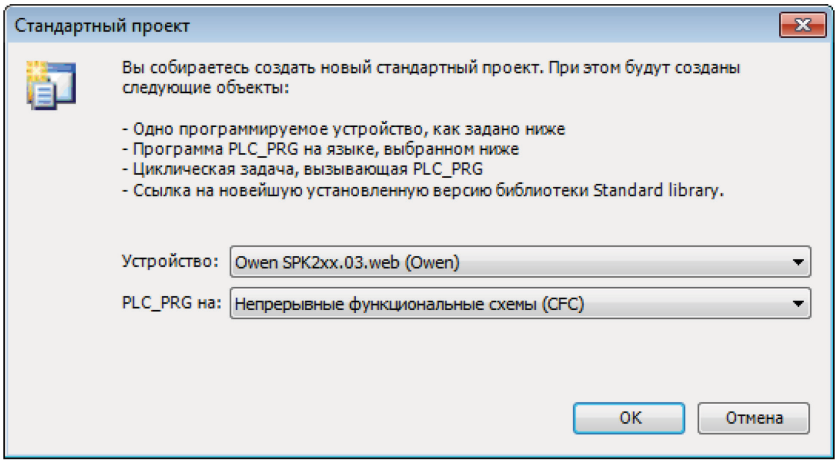


Рис. 4.39. Создание проекта **CODESYS**

4. Добавьте в проект объединение с именем **Real_Word**:

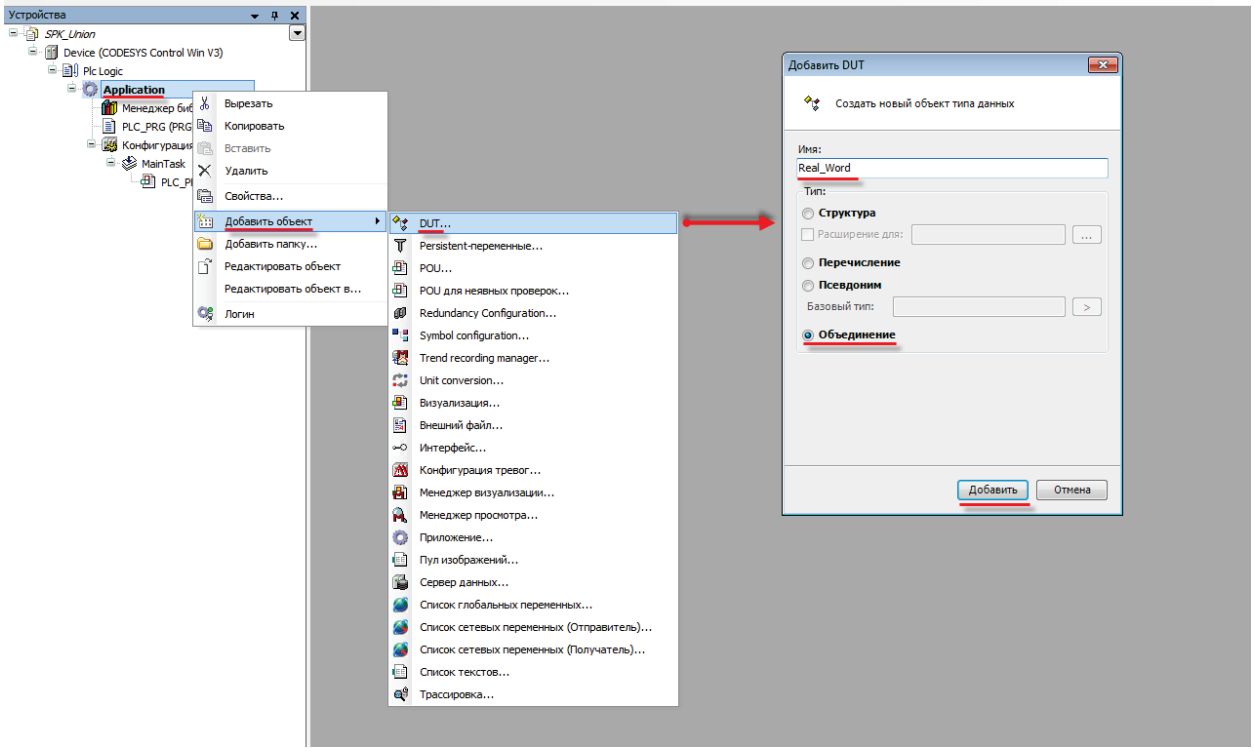


Рис. 4.40. Добавление в проект объединения

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

Real_Word

```
TYPE Real_Word :
UNION
    rRealValue      :REAL;
    awModbusReal    :ARRAY [0..1] OF WORD;
END_UNION
END TYPE
```

Рис. 4.41. Объявление переменных объединения

6. В программе **PLC_PRG** объявите 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 табл. 4.2.

```
PROGRAM PLC_PRG
VAR
    _2WORD_TO_REAL  :Real_Word;
    wDI             :WORD;      // маска дискетных входов
    wDO             :WORD;      // маска дискретных выходов
    wPrevDO         :WORD;      // маска дискретных выходов из предыдущего цикла
    xTrigger        :BOOL;      // триггер записи
END VAR
```

Рис. 4.42. Объявление переменных программы

Код программы будет выглядеть следующим образом:

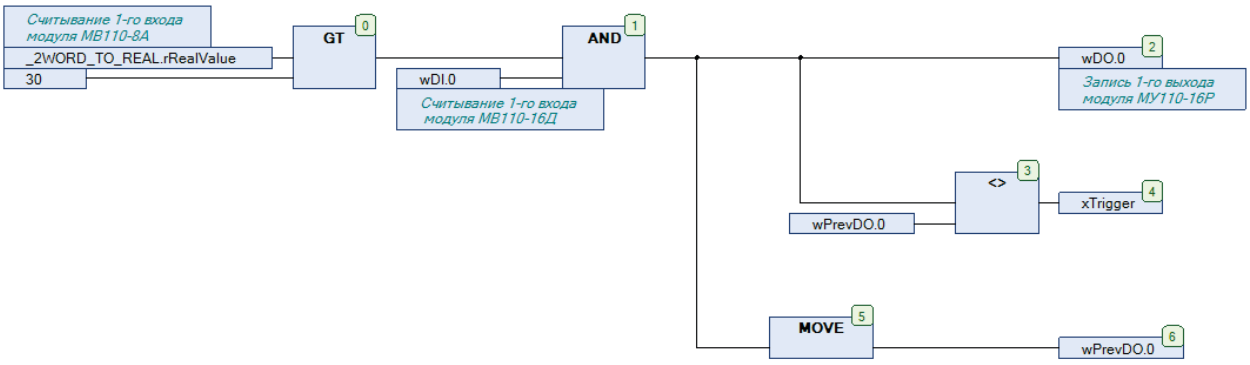


Рис. 4.43. Код программы **PLC_PRG**

Программа работает следующим образом: если значение переменной **rRealValue** (связанной с 1-м аналоговым входом модуля **MB110-8A)** превышает 30 и при этом значение нулевого бита переменной **wDI** (связанной с 1-м 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 1-й дискретный выход модуля **MУ110-16Р**.

7. Добавьте в проект два устройства **Modbus COM** с названиями **COM2** и **COM3**:

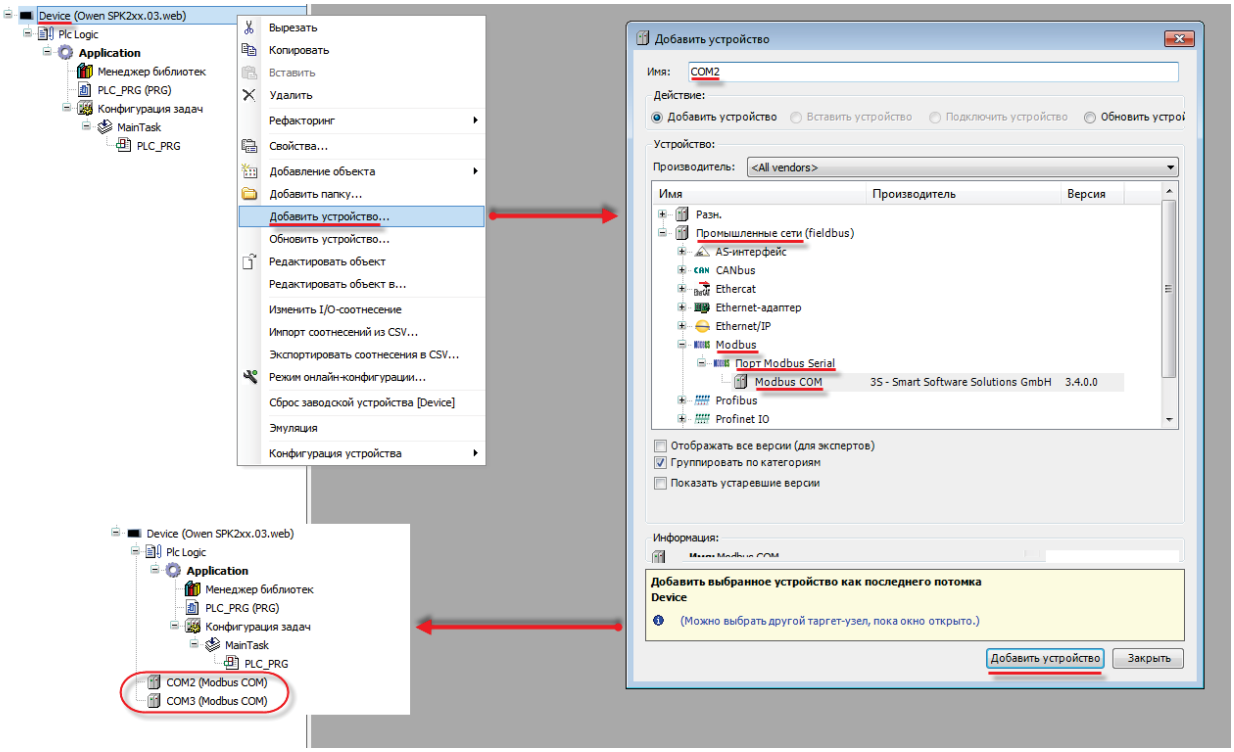


Рис. 4.44. Добавление устройства **Modbus COM**

8. В конфигурации COM-портов укажите сетевые настройки в соответствии с табл. 4.1, а также номера портов – **COM2** будет соответствовать номер **3**, **COM3** – номер **4** (см. п. 2.3.1).

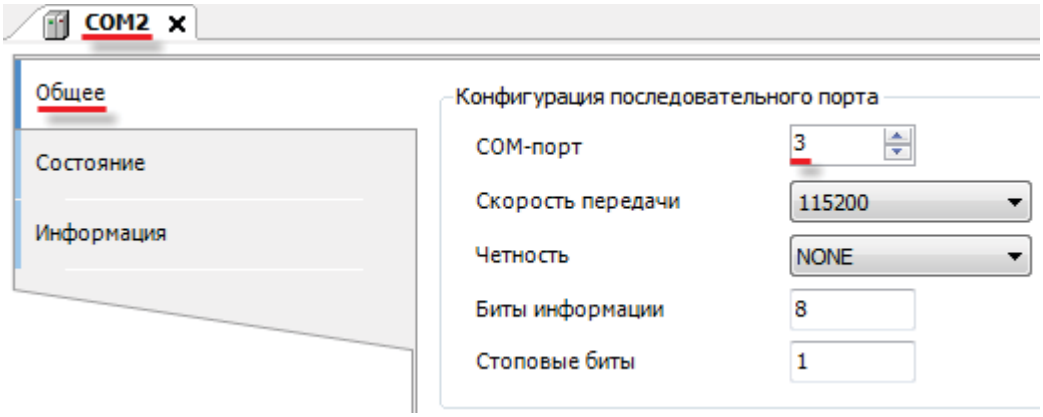


Рис. 4.45. Настройки COM-порта **COM2**

9. В каждый из COM-портов добавьте компонент **Modbus Master:**

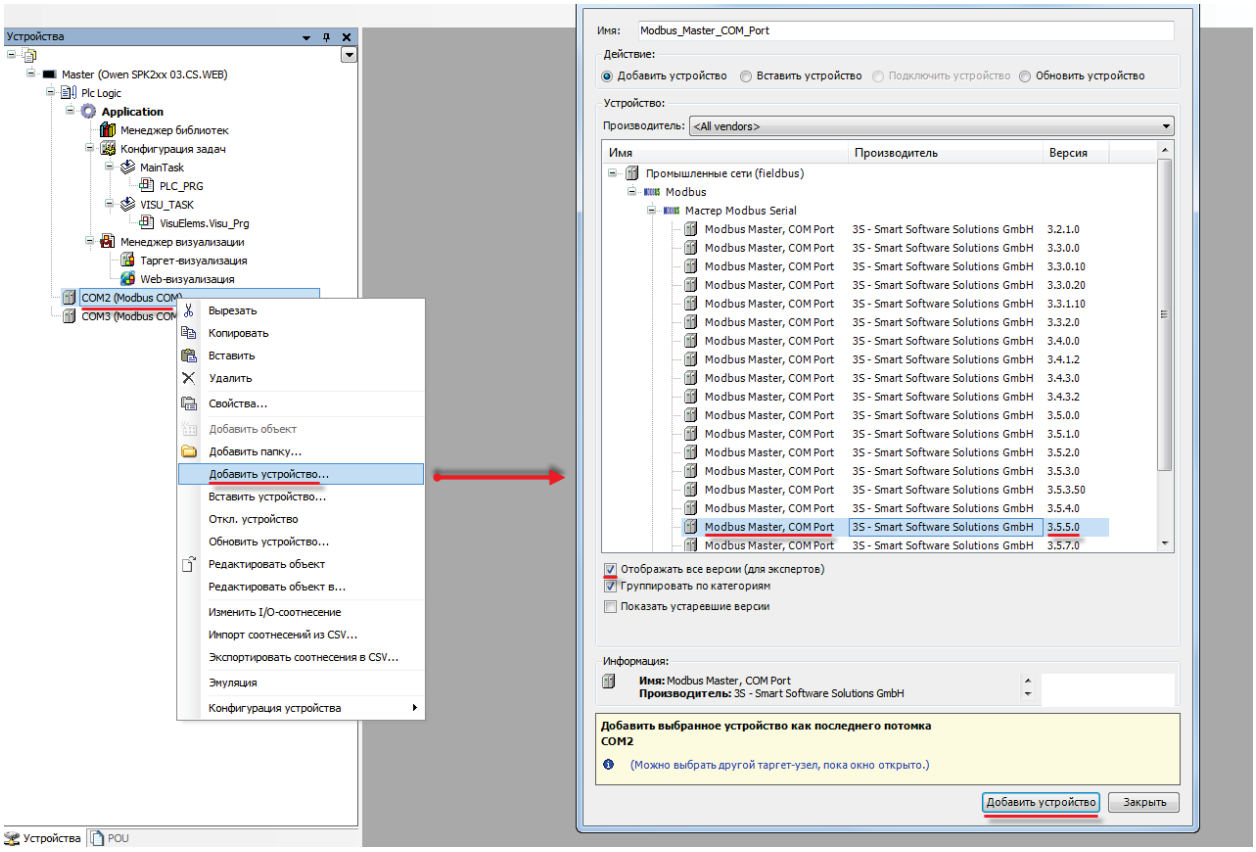


Рис. 4.46. Добавление компонента **Modbus Master**

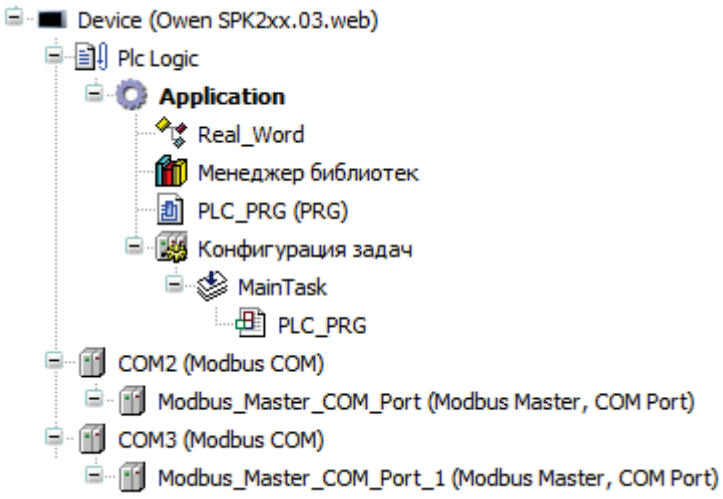


Рис. 4.47. Внешний вид дерева проекта после добавления **Modbus Master**

10. В настройках компонентов поставьте галочку **Автоперезапуск соединения**. В параметре **Время между фреймами** установите значение **20 мс**.

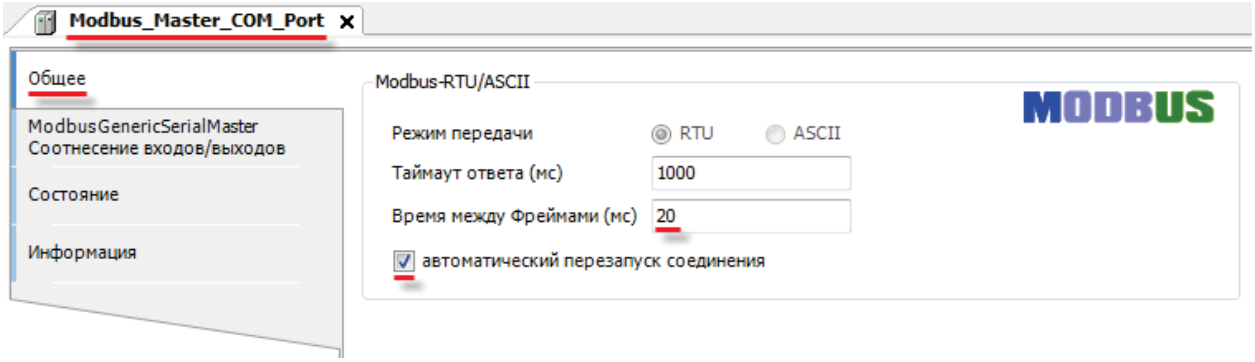


Рис. 4.48. Настройка компонентов **Modbus Master**

11. В **Modbus Master** порта **COM2** добавьте компонент **Modbus Slave** с именем **MV110_8A,** а в **Modbus Master** порта **COM3** – компоненты **Modbus Slave** с именами **MV110_16D** и **MU110_16R**:

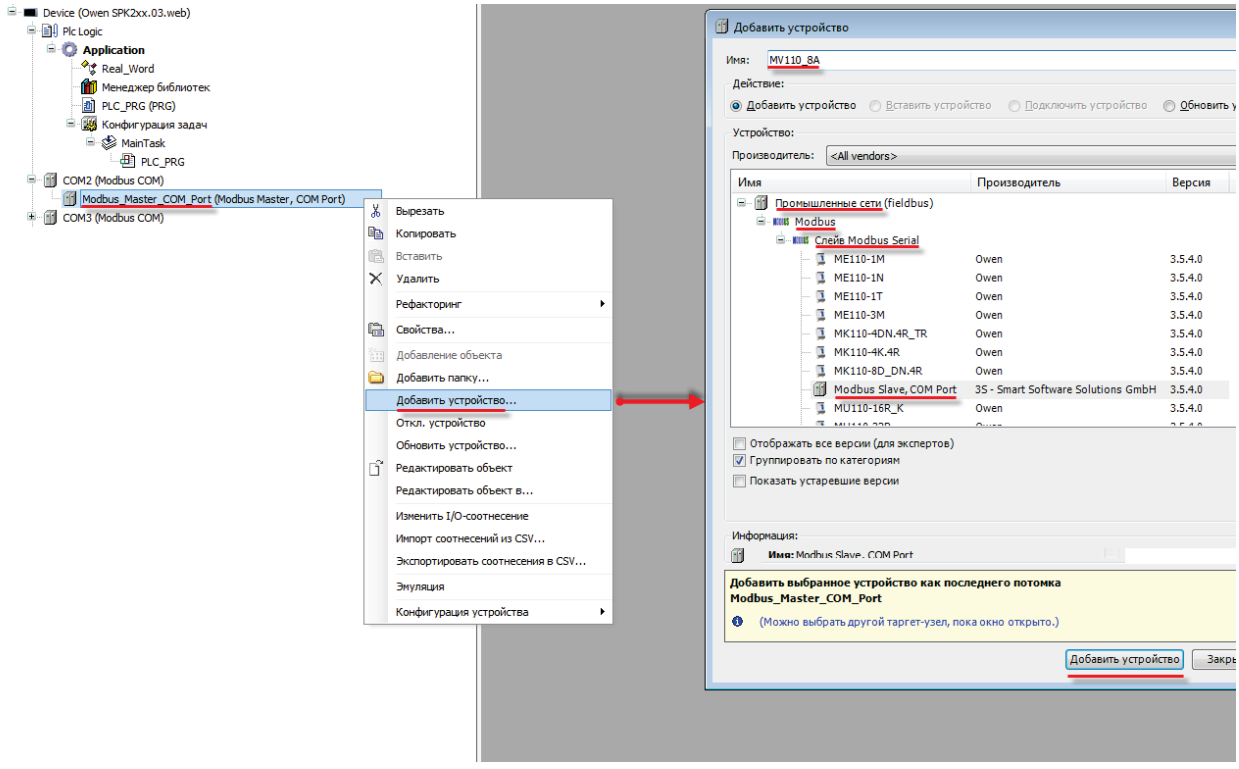


Рис. 4.49. Добавление компонента **Modbus Slave** в проект

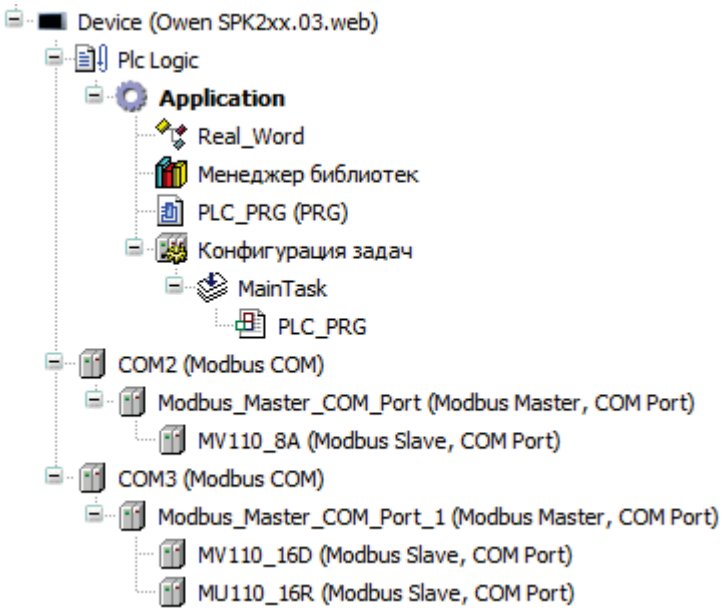


Рис. 4.50. Внешний вид дерева проекта после добавления компонентов **Modbus Slave**

12. В настройках компонента **MV110_8A** на вкладке **Конфигурация Modbus Slave** укажите адрес **1** (в соответствии с табл. 4.1):

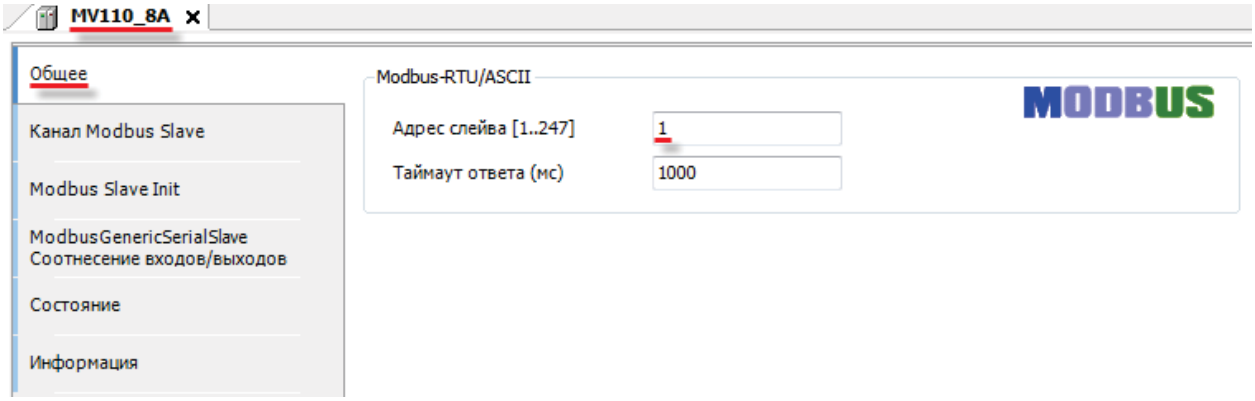


Рис. 4.51. Настройка slave-устройства **MV110_8A**

На вкладке **Канал Modbus Slave** добавьте канал, в котором с помощью функции **Read Holding Registers** будет считываться значение **4-го** и **5-го** регистров модуля. В этих регистрах содержится значение входа 1 в представлении с плавающей точкой. Таблица регистров модуля и поддерживаемые функции приведены в **РЭ**.

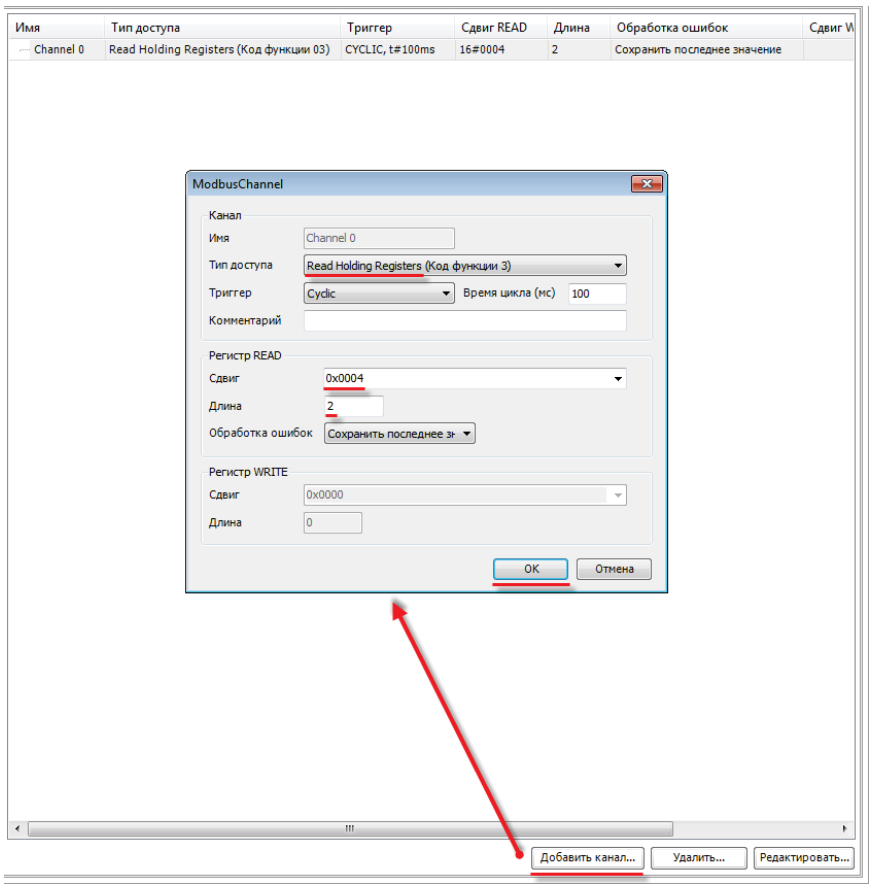


Рис. 4.52. Добавление канала в конфигурацию slave-устройства **MV110_8A**

На вкладке **ModbusGenericSerialSlave Соотнесение входов/выходов** привяжите к каналу элементы объединения **_2WORD_TO_REAL**. ***Обратите внимание***, что первый считываемый регистр присваивается первому элементу массива, а второй – нулевому. Это связано с тем, что порядок **WORD** в **REAL** у модуля и СПК отличается.

Не забудьте у параметра **Всегда обновлять переменные** выставить значение **Включено 2**.

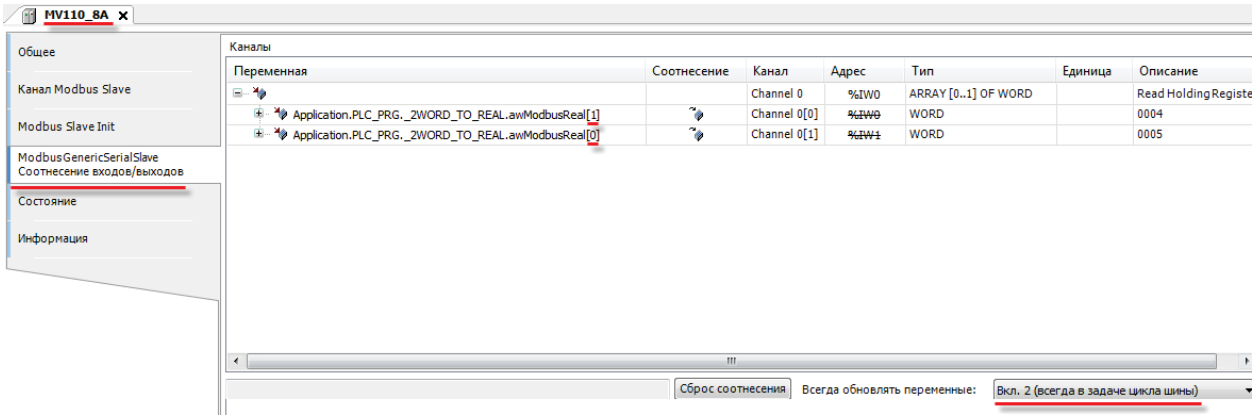


Рис. 4.53. Привязка переменных к каналу

13. В настройках компонента **MV110_16D** на вкладке **Конфигурация Modbus Slave** укажите адрес **1** (в соответствии с табл. 4.1):

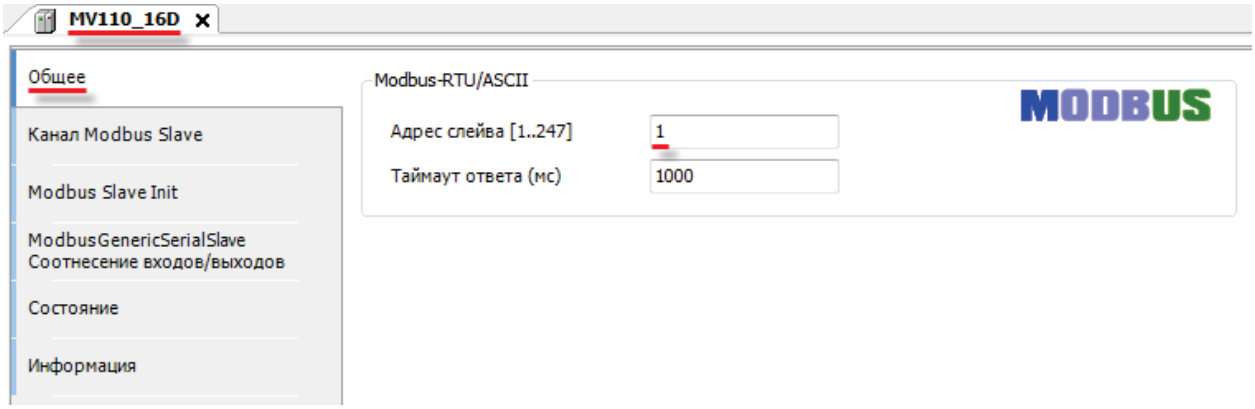


Рис. 4.54. Настройка slave-устройства **MV110_16D**

На вкладке **Канал Modbus Slave** добавьте канал, в котором с помощью функции **Read Holding Registers** будет считываться значение регистра **0x0033**. В этом регистре содержится битовая маска состояний дискретных входов.

Таблица регистров модуля и поддерживаемые функции приведены в **РЭ**.

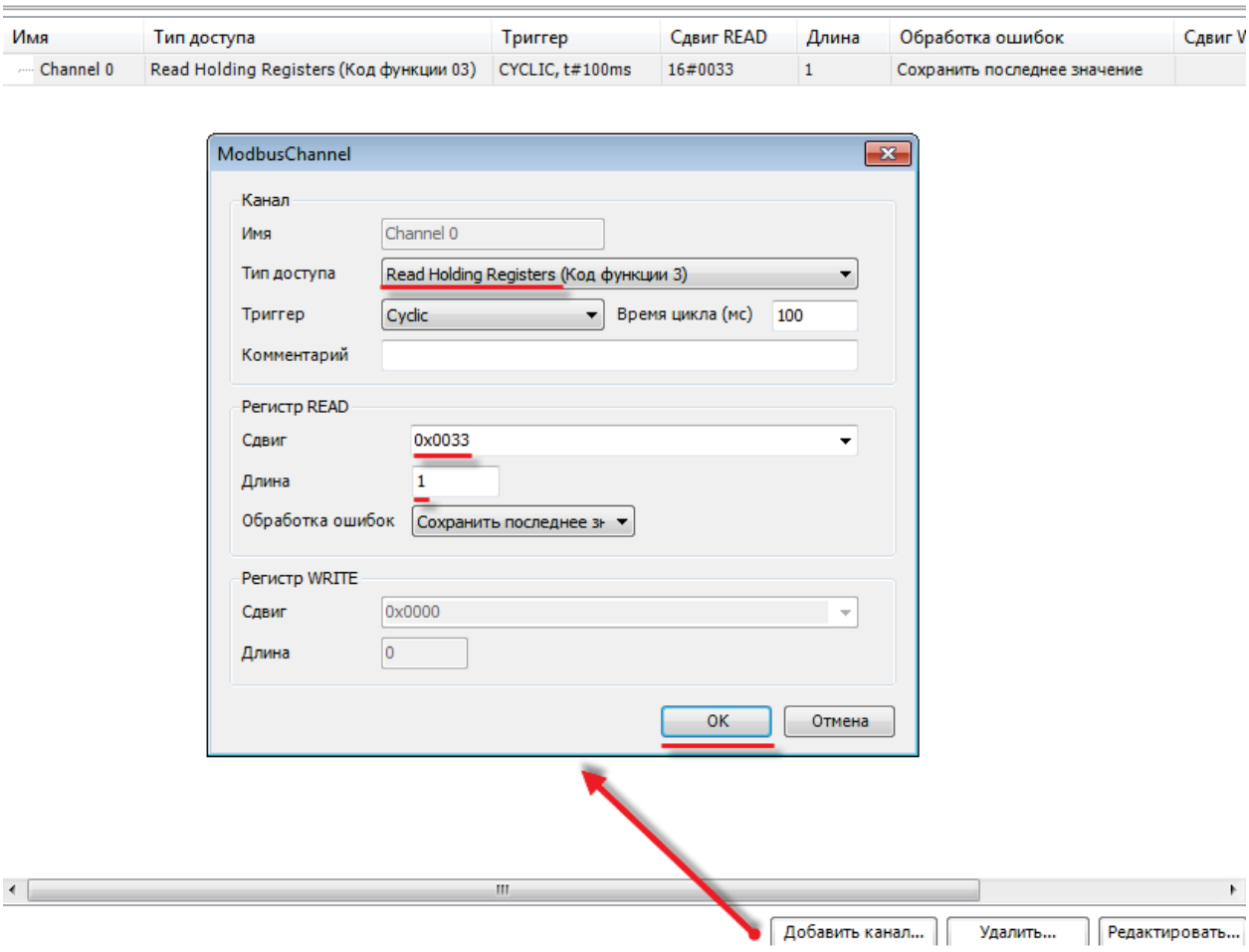


Рис. 4.55. Добавление канала в конфигурацию slave-устройства **MV110_16D**

На вкладке **ModbusGenericSerialSlave Соотнесение входов/выходов** привяжите к каналу переменную **wDI**.

Не забудьте у параметра **Всегда обновлять переменные** выставить значение **Включено 2.**

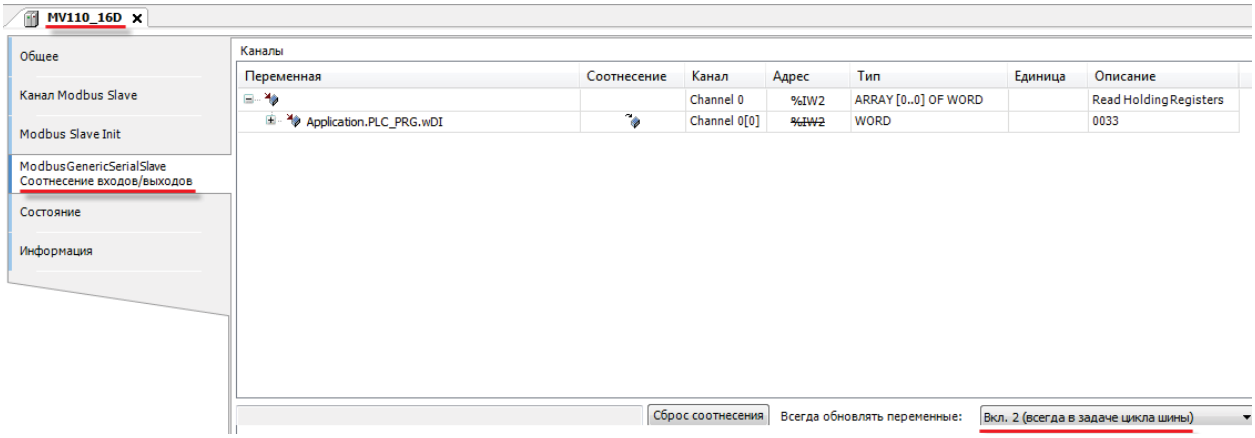


Рис. 4.56. Привязка переменных к каналу

14. В настройках компонента **MU110_16R** на вкладке **Общее** укажите адрес **17** (в соответствии с табл. 4.1):

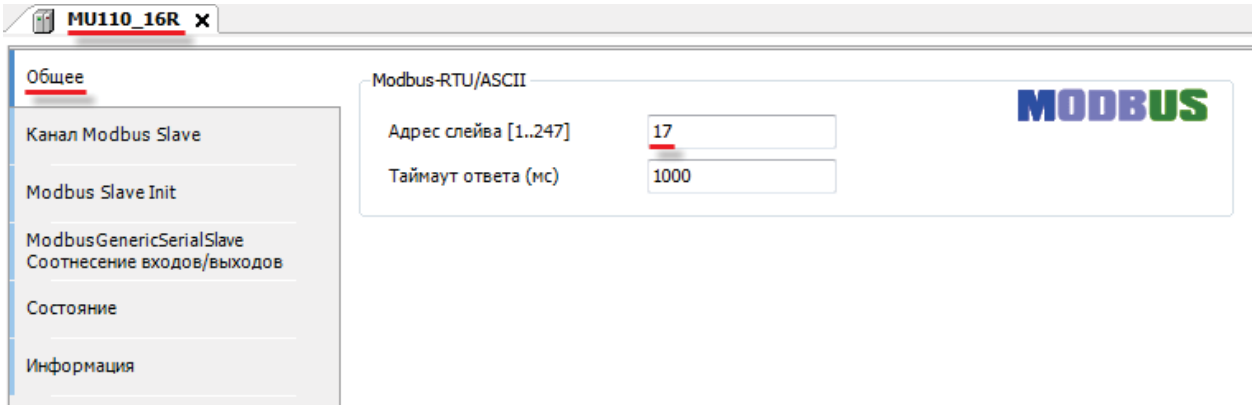


Рис. 4.57. Настройка slave-устройства **MU110_16R**

На вкладке **Канал Modbus Slave** добавьте канал, в котором с помощью функции **Write Multiple Registers** будет записываться значение в регистр **0x0032**. В этом регистре содержатся значения выходов модуля в виде битовой маски. У параметра **Триггер** поставьте значение **Rising Edge**, чтобы иметь возможность управлять записью в модуль с помощью логической переменной.

Таблица регистров модуля и поддерживаемые функции приведены в **РЭ**.

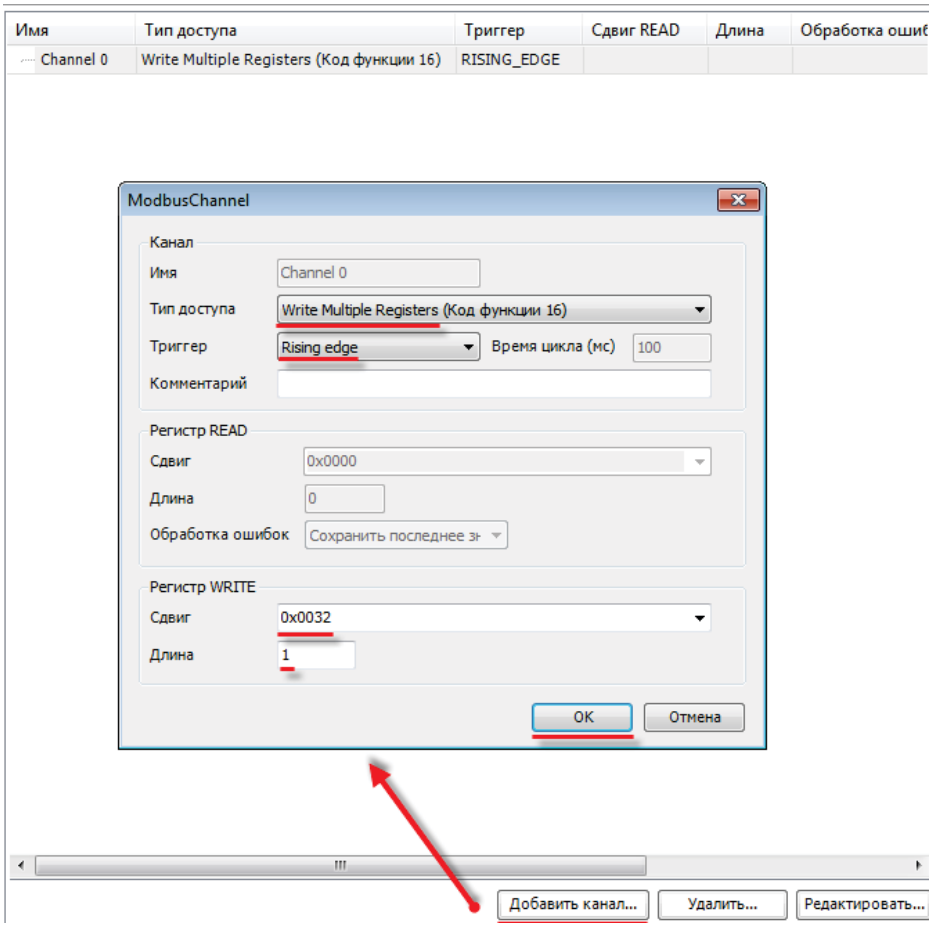


Рис. 4.58. Добавление канала в конфигурацию slave-устройства **MU110_16R**

На вкладке **ModbusGenericSerialSlave Соотнесение входов/выходов** привяжите к каналу переменную **wDO** и триггерную переменную **xTrigger**.

Не забудьте у параметра **Всегда обновлять переменные** выставить значение **Включено 2.**

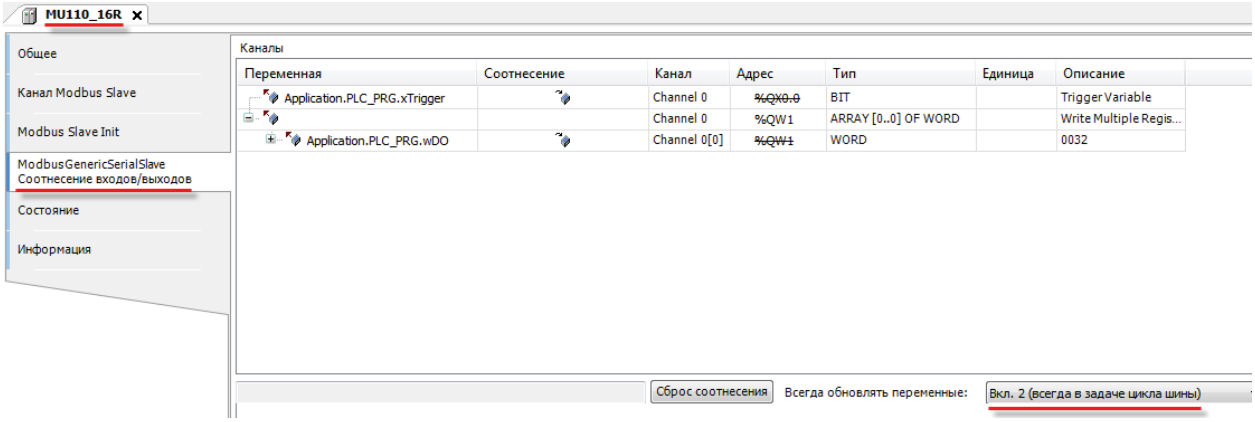
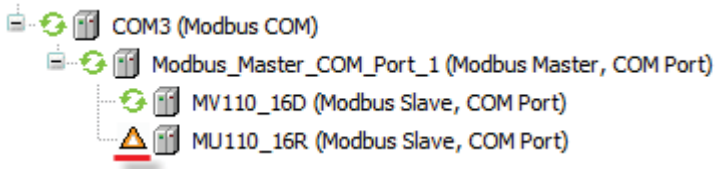


Рис. 4.59. Привязка переменных к каналу

15. Загрузите проект в СПК и запустите его.

Обратите внимание, что в дереве проекта рядом с модулем **MU110_16R** будет отображаться пиктограмма «ожидание связи». Это связано с тем, что запись в модуль ведется по триггеру, и в данный момент обмен с модулем отсутствует. После первой записи в модуль статус связи изменится на «связь установлена».



В переменной **_2WORD_TO_REAL.rRealValue** будет отображаться текущее значение 1-го аналогового входа модуля **MB110-8A**. В нулевом бите переменной **wDI (wDI.0)** будет отображаться текущее значение 1-го 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 1-го дискретного выхода модуля **МУ110-16Р**. Если одно из условий перестанет выполняться, то выход будет разомкнут.

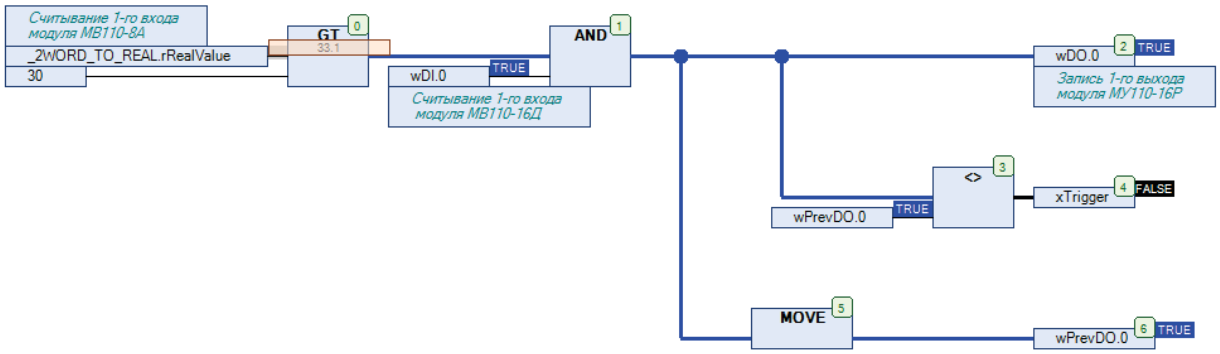


Рис. 4.60. Выполнение программы в режиме **Online**

4.7. Пример: СПК207 (master) + СПК207 (slave)

Рассмотрим пример настройки обмена между двумя контроллерами **СПК207**, один из которых выполняется функцию **master**, а другой – **slave**.

Формулировка задачи: с помощью СПК207 (master) выполнить следующие действия:

1. считать из СПК207 (slave) переменную типа **BOOL**;
2. считать из СПК207 (slave) переменную типа **WORD**;
3. записать в СПК207 (slave) переменную типа **REAL**;
4. записать в СПК207 (slave) переменную типа **STRING**.

Структурная схема примера приведена на рис. 4.61:

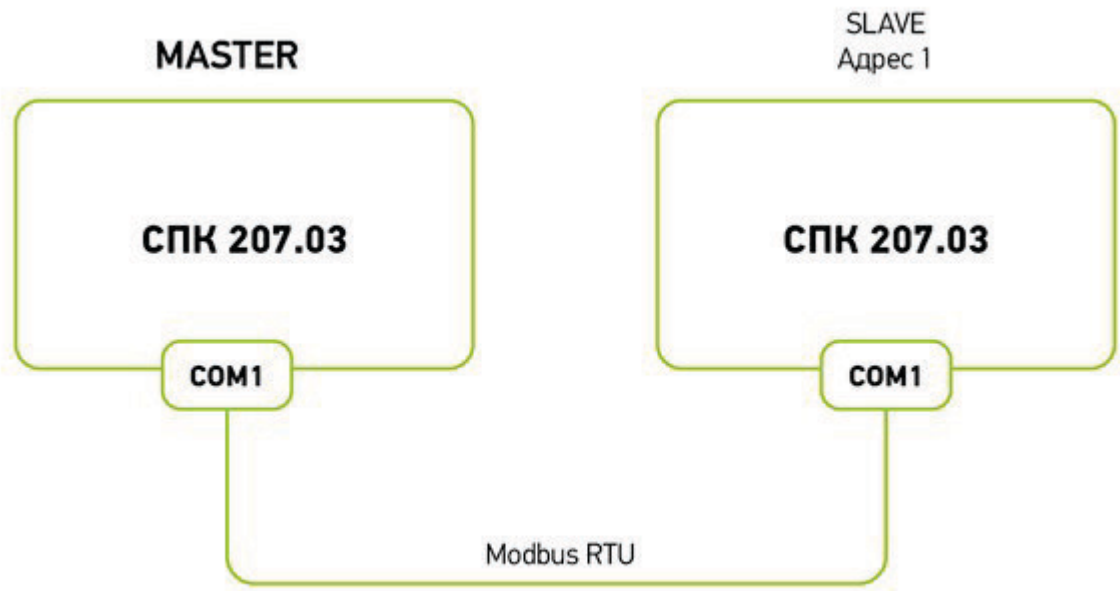


Рис. 4.61. Структурная схема примера **Стандартные средства конфигурирования (СПК+СПК)**

Пример создан в среде **CODESYS 3.5 SP7 Patch4** и подразумевает запуск на **СПК207.03.CS(-WEB)**.

Пример доступен для скачивания: Example_StandardConfiguration_Slave.projectarchive

Сетевые параметры модулей приведены в табл. 4.3.

Список переменных, используемых в примере, приведен в табл. 4.4.

Табл. 4.3. Сетевые параметры СПК

Параметр	**СПК207 (master)**	**СПК207 (slave)**
Используемый COM-порт	*COM1*	*COM1*
Адрес (Slave ID)	-	1
Скорость обмена	115200	
Количества бит данных	8	
Контроль четности	отсутствует	
Количество стоп-бит	1	

Табл. 4.4. Список переменных программы

		СПК207 (master)	**СПК207 (slave)**	
Переменная	**Тип**	**Функция**	**Область памяти**	**Регистры**
xVar	BOOL	Read Discrete Inputs	Input Registers	0
wVar	WORD	Read Input Registers	Input Registers	1
rRealValue	REAL	Write Multiple Registers	Holding Registers	0-1
sStringValue	STRING	Write Multiple Registers	Holding Registers	2-4

4.7.1. Настройка СПК207 (master)

1. Создайте новый проект **CODESYS** для **СПК207** с программой **PLC_PRG** на языке **CFC**.

2. Добавьте в проект объединение с именем **Real_Word**:

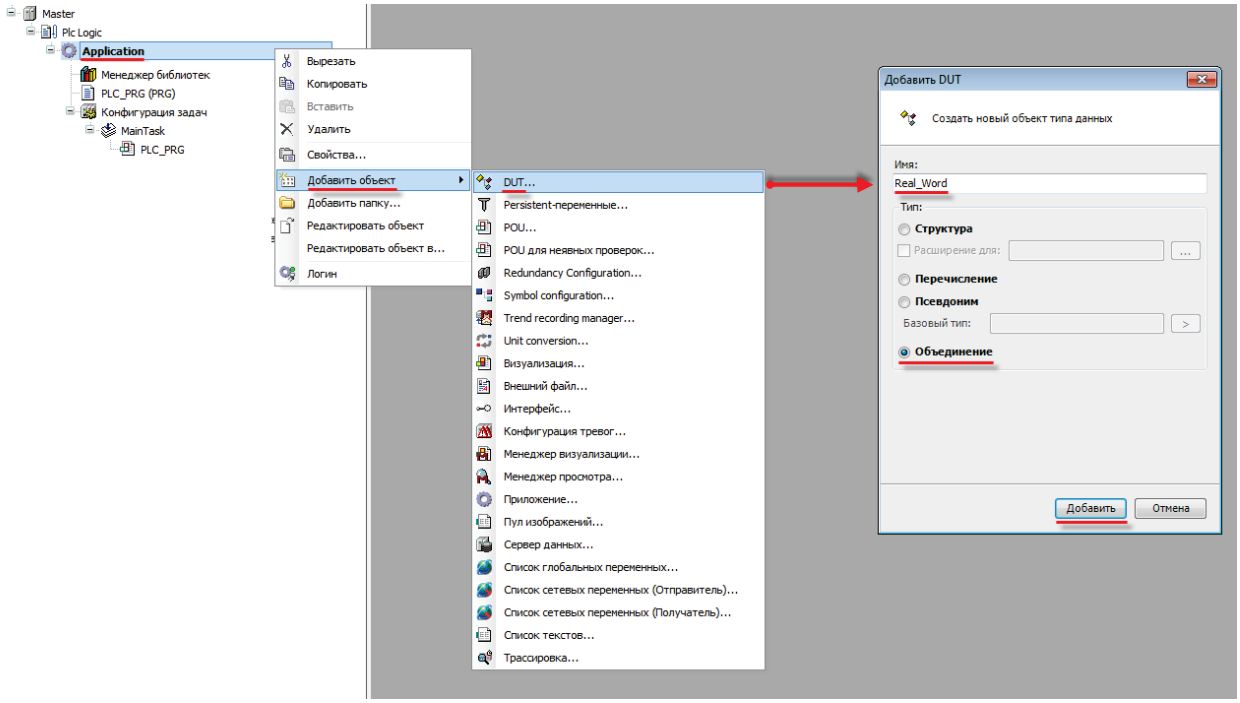


Рис. 4.62. Добавление в проект объединения

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

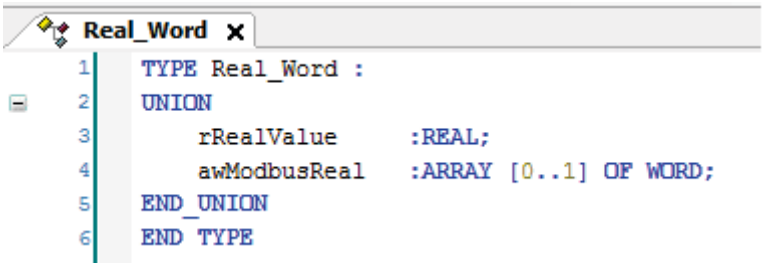


Рис. 4.63. Объявление переменных объединения

3. Добавьте в проект объединение с именем **String_Word**.

В объединении объявите переменную **sStringValue** типа **STRING** (с ограничением на размер в 6 символов) и массив **awModbusString** типа **WORD**, содержащий три элемента (**STRING** сможет содержать до 6 символов, поскольку каждый **WORD** может содержать два символа):

```
TYPE String_Word :
UNION
    awModbusString      :ARRAY [0..2] OF WORD;
    sStringValue        :STRING;
END_UNION
END_TYPE
```

Рис. 4.64. Объявление переменных объединения

4. Объявите в программе переменную **xVar** типа **BOOL** и **wVar** типа **WORD**, а также экземпляр объединения **Real_Word** с именем **_REAL_TO_2WORD** и экземпляр объединения **String_Word** с именем **_STRING_TO_3WORD**.

PLC_PRG

```
PROGRAM PLC_PRG
VAR
    xVar                :BOOL;
    wVar                :WORD;
    _REAL_TO_2WORD      :Real_Word;
    _STRING_TO_3WORD    :String_Word;
END VAR
```

Рис. 4.65. Объявление переменных программы

5. Код программы будет выглядеть следующим образом:

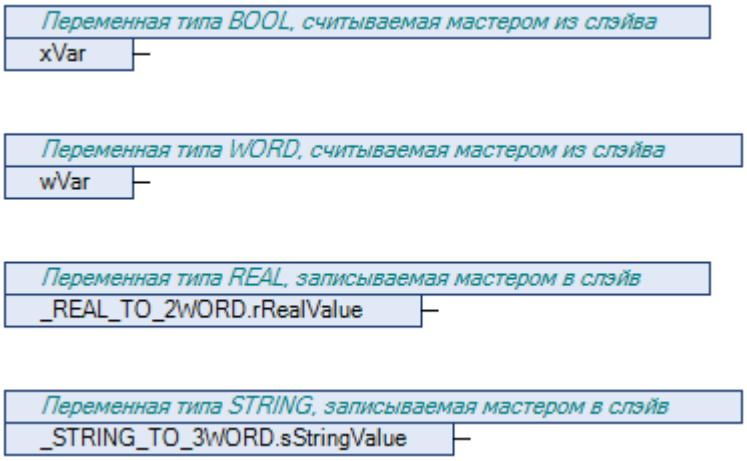


Рис. 4.66. Код программы на языке **CFC**

6. Добавьте в проект устройство **Modbus COM**:

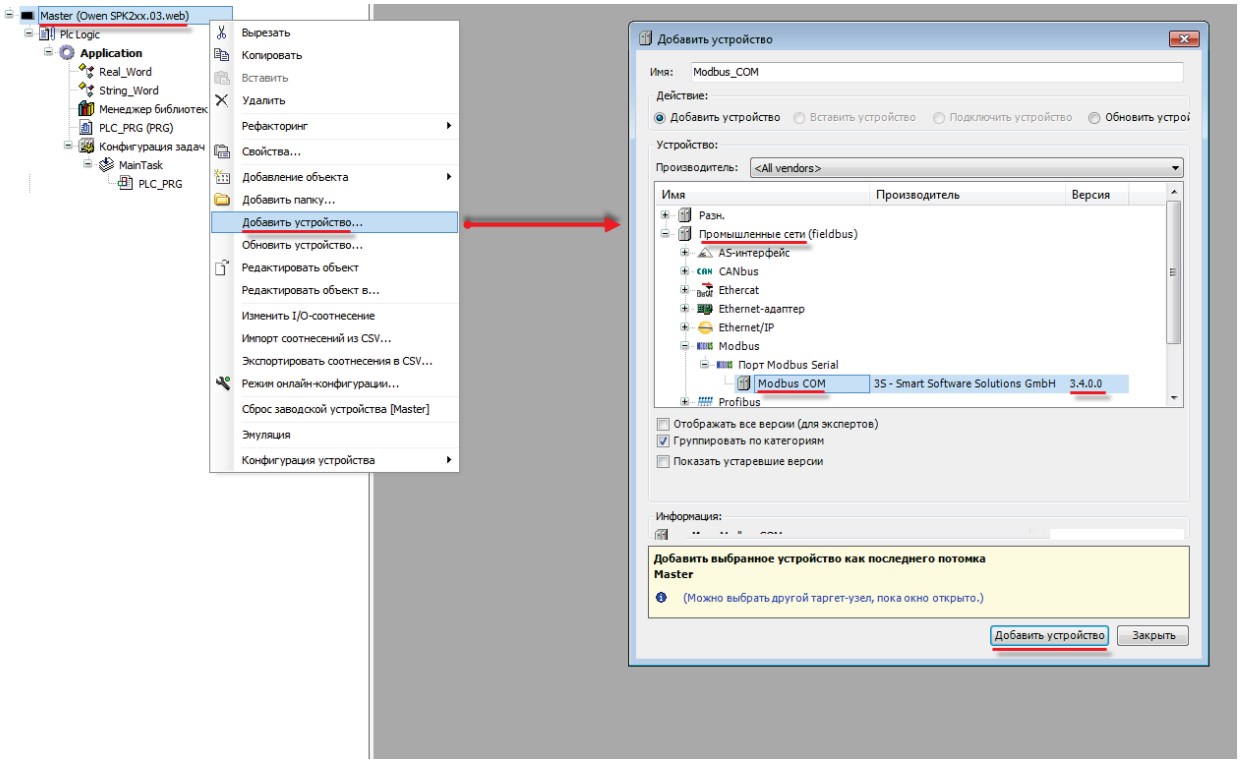


Рис. 4.67. Добавление устройства **Modbus COM**

В конфигурации COM-порта укажите сетевые настройки в соответствии с табл. 4.3, а также номер порта. COM-порту **1** будет соответствовать номер **2** (см. п. 2.3.1).

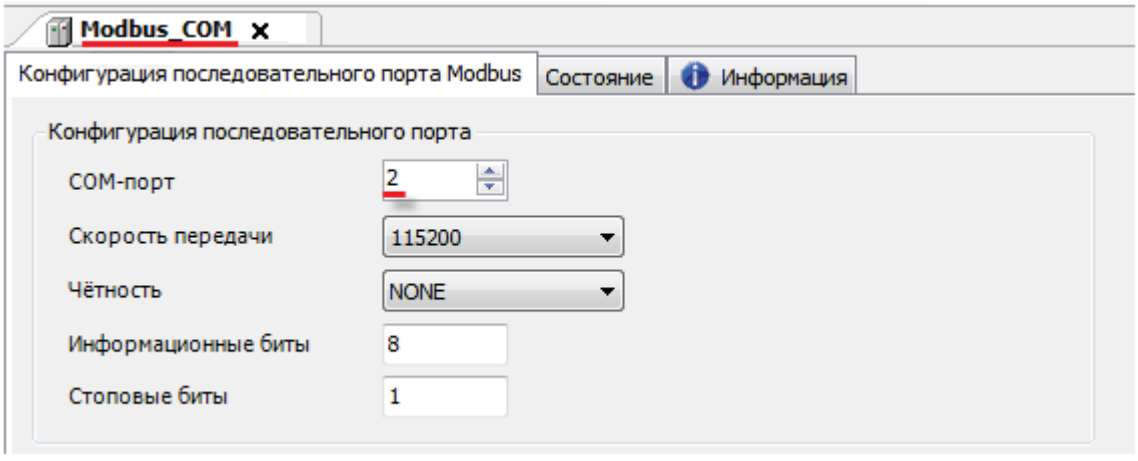


Рис. 4.68. Настройки COM-порта **COM1**

7. В COM-порт добавьте компонент **Modbus Master**:

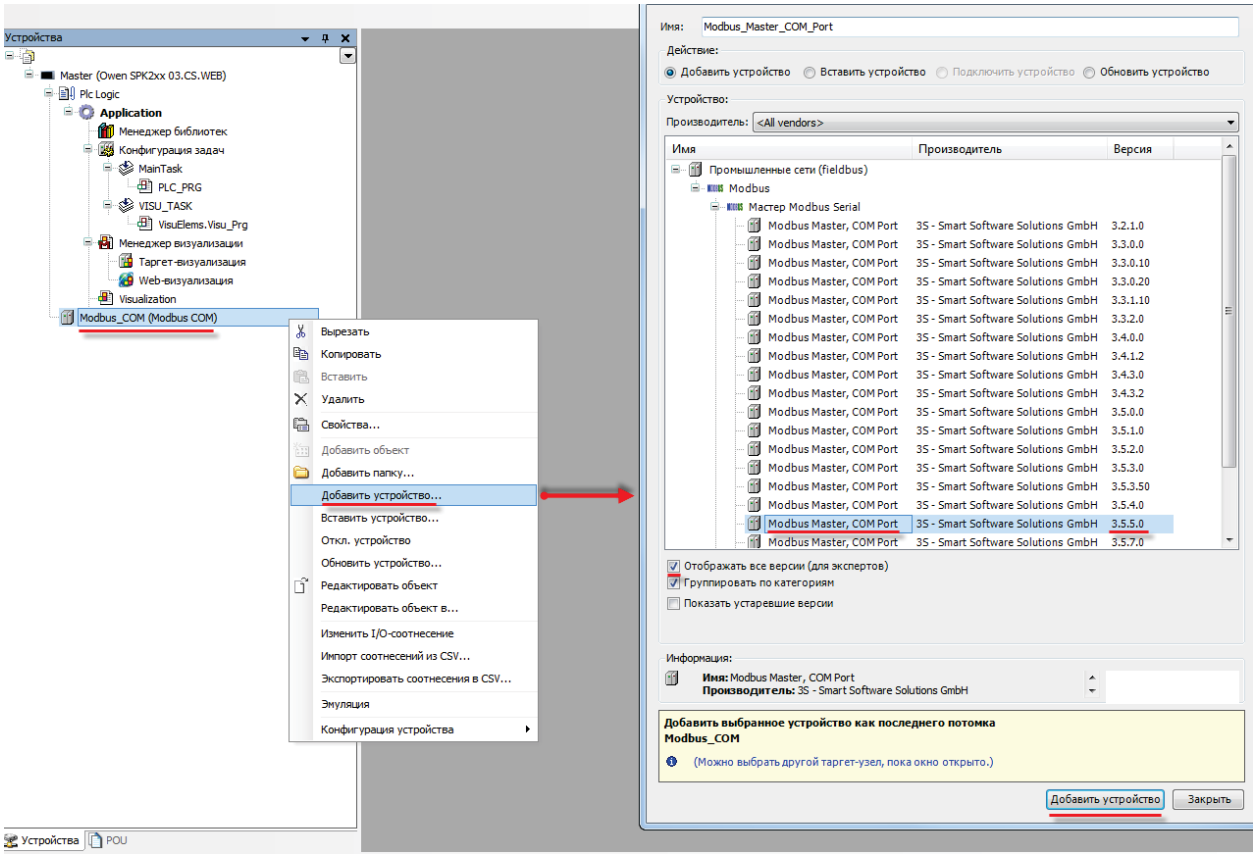


Рис. 4.69. Добавление компонента **Modbus Master**

В настройках компонента поставьте галочку **Автоперезапуск соединения**.

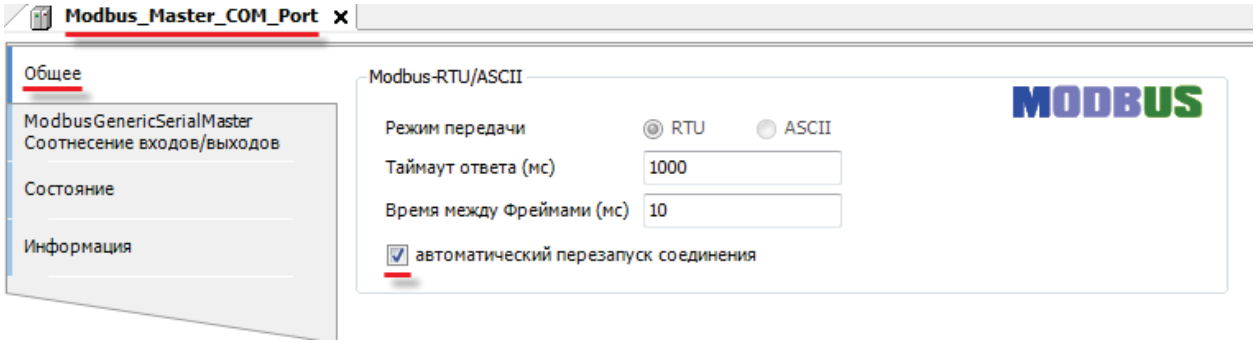


Рис. 4.70. Настройка компонентов **Modbus Master**

8. В **Modbus Master** добавьте компонент **Modbus Slave**:

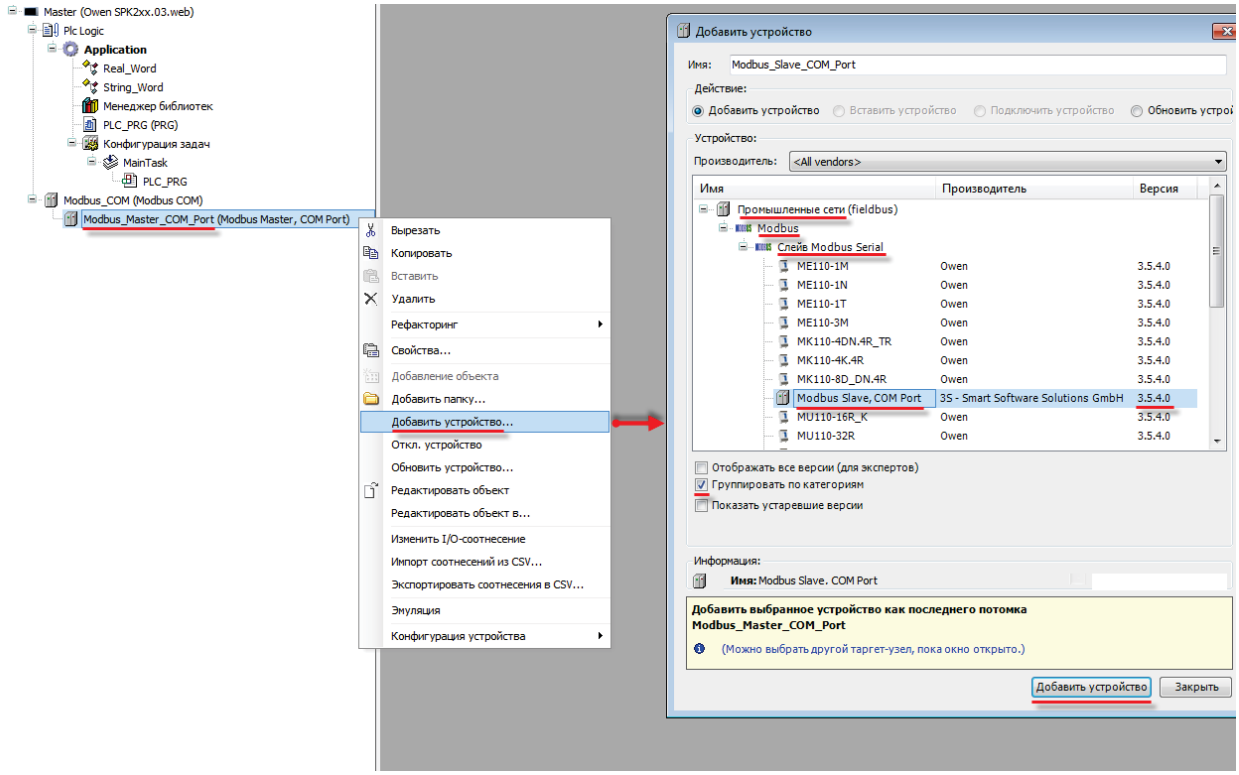


Рис. 4.71. Добавление компонента **Modbus Slave** в проект

В настройках компонента на вкладке **Общее** укажите адрес slave-устройства в соответствии с табл. 4.3.

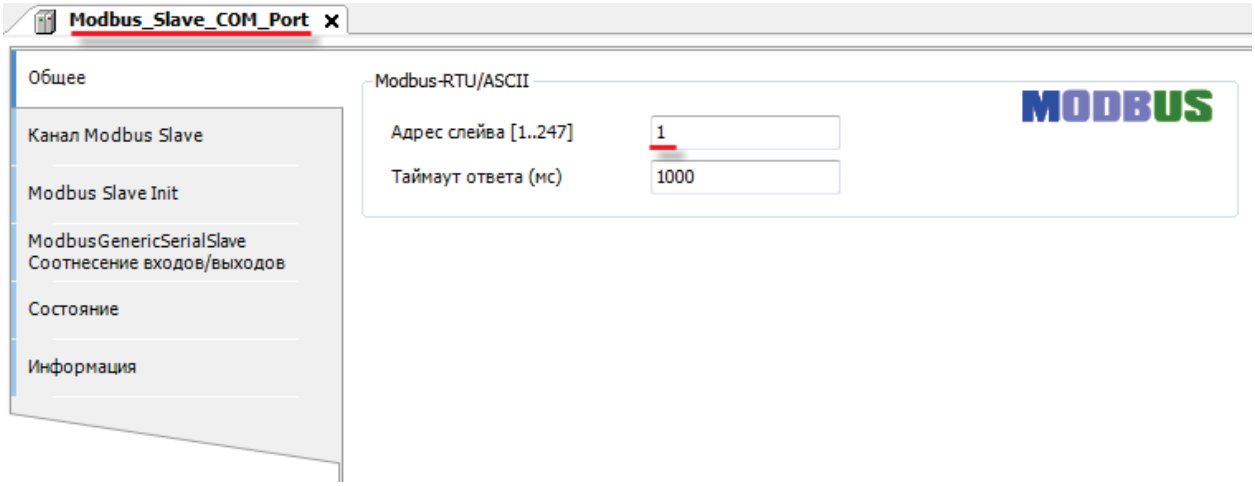


Рис. 4.72. Настройки компонента **Modbus Slave** в проект

На вкладке **Канал Modbus Slave** создайте 4 канала – по одному на каждую из переменных. Переменные типа **BOOL** и **WORD** СПК (master) будут считывать с помощью функций **Read Discrete Inputs** и **Read Input Registers** из ***Input*** регистров СПК (slave). Переменные типа **REAL** и **STRING** СПК (master) будет записывать с помощью функции **Write Multiple Registers** в ***Holding*** регистры СПК (slave). ***Обратите внимание***, что ***Input*** и ***Holding*** регистры – разные области памяти СПК (slave).

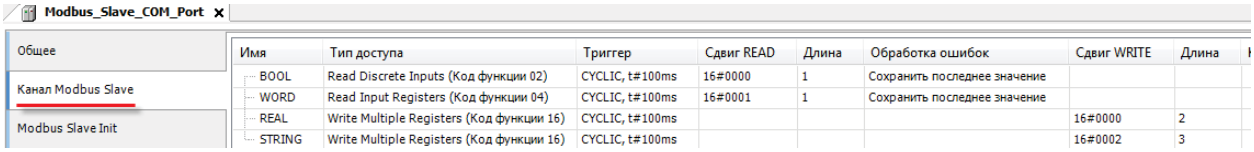


Рис. 4.73. Настройка каналов **Modbus Slave**

На вкладке **ModbusGenericSerialSlave Соотнесение входов/выходов** привяжите к каналам переменные программы в соответствии с табл. 4.4. Не забудьте у параметра **Всегда обновлять переменные** выставить значение **Включено 2**.

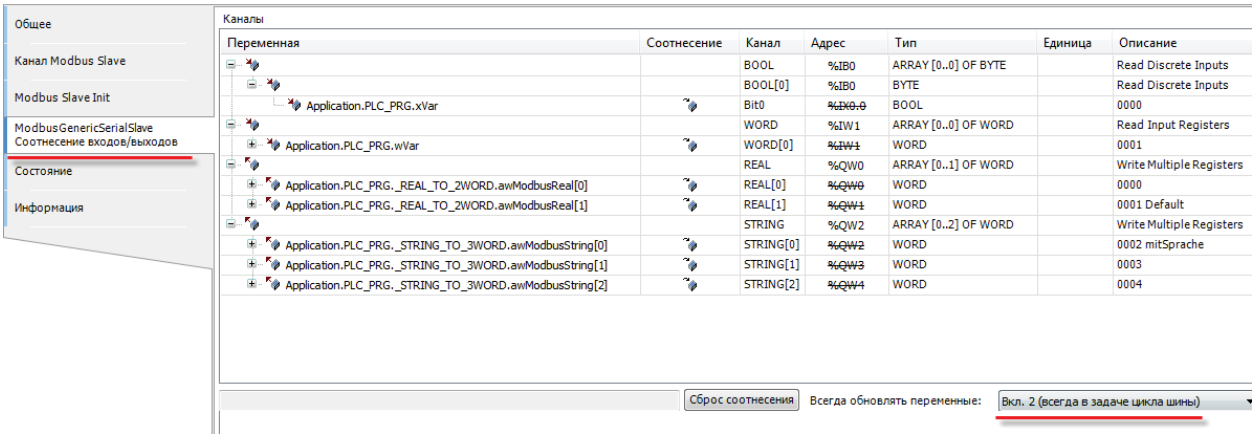


Рис. 4.74. Привязка переменных к каналу

На этом настройка **СПК (master)** завершена.

4.7.2. Настройка СПК207 (slave)

1. В проект CODESYS, содержащий СПК (master), добавьте контроллер СПК (slave).

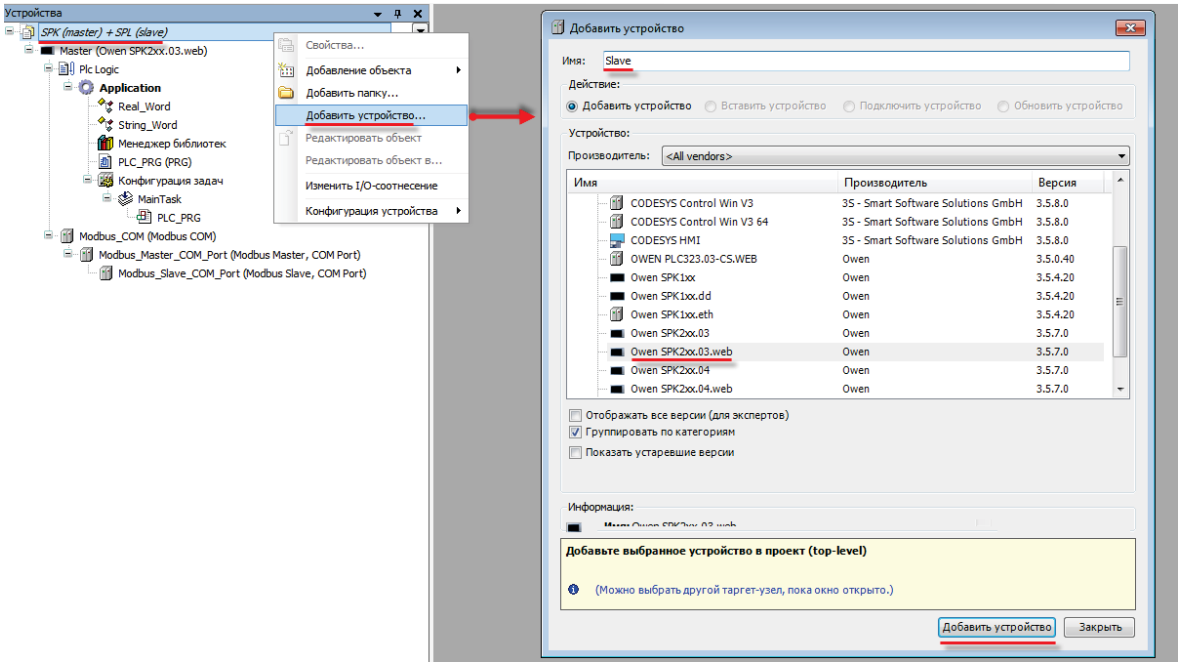


Рис. 4.75. Добавление СПК (slave) в проект **CODESYS**

В приложение **Application** устройства СПК (slave) добавьте программу с именем **PLC_PRG** на языке **CFC** и компонент **Конфигурация задач**. Привяжите программу **PLC_PRG** к задаче **MainTask**. После этого дерево проекта будет выглядеть следующим образом:

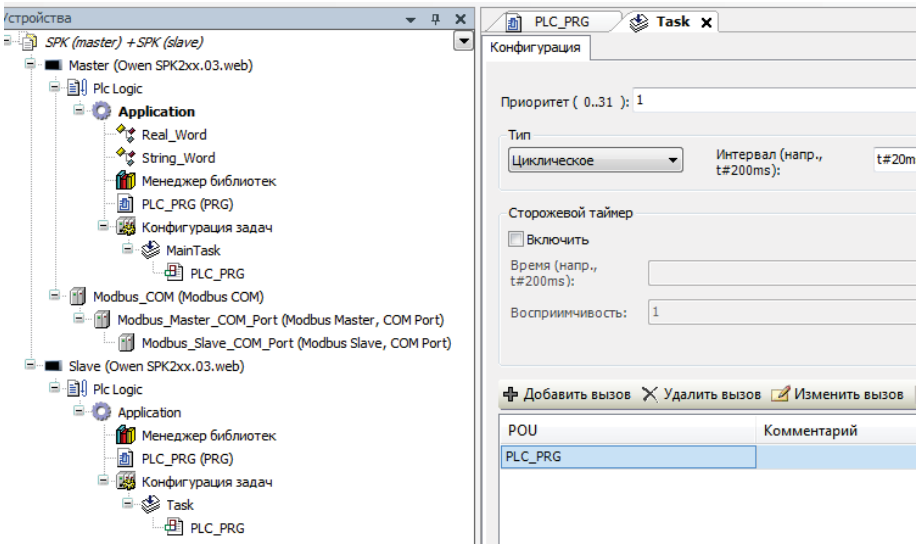


Рис. 4.76. Внешний вид дерева проекта после привязки программы к задаче

2. Добавьте в проект объединение с именем **Real_Word**:

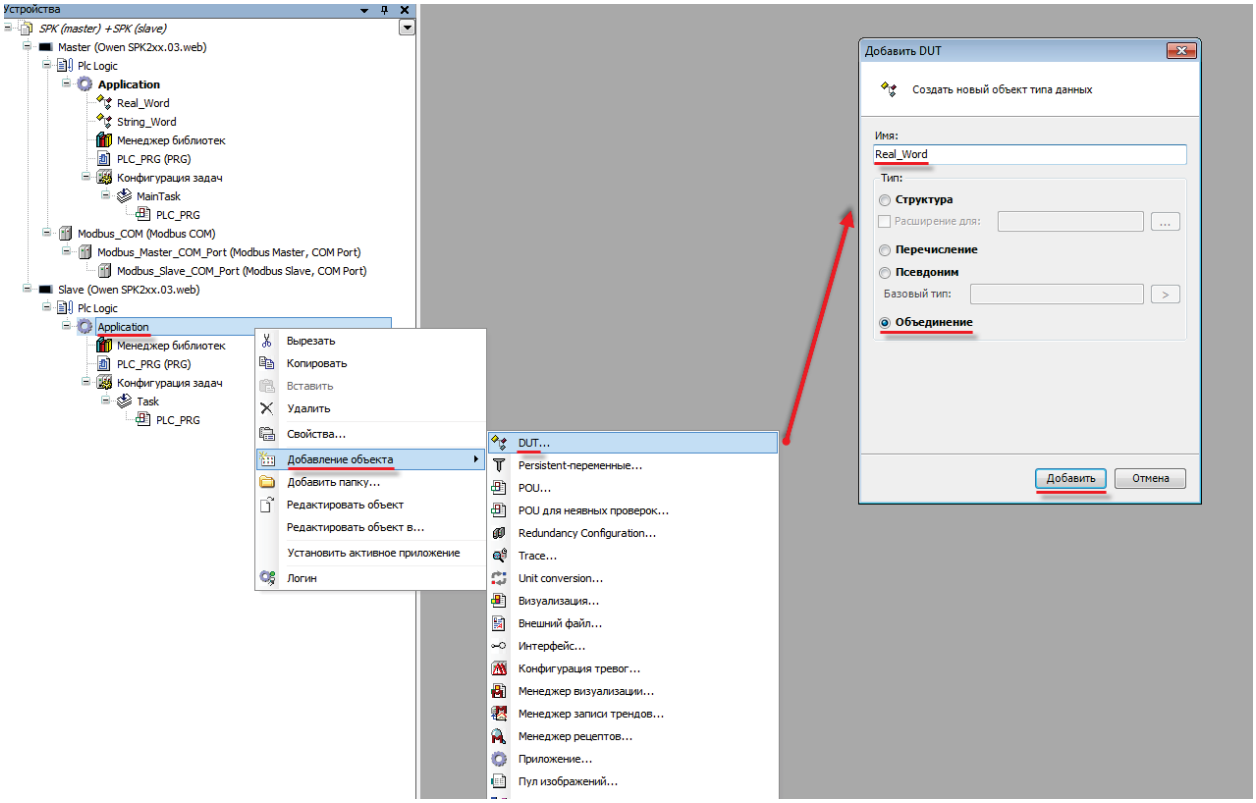


Рис. 4.77. Добавление объединения в проект

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

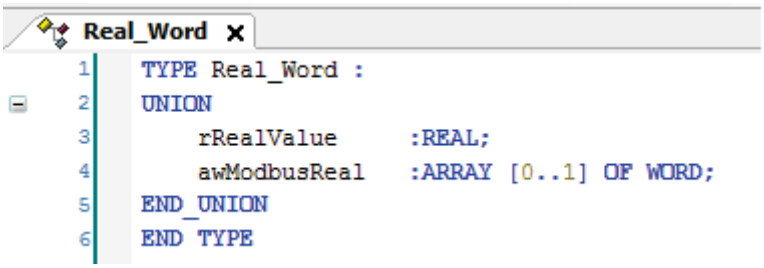


Рис. 4.78. Объявление переменных объединения

3. Добавьте в проект объединение с именем **String_Word**.

В объединении объявите переменную **sStringValue** типа **STRING** (с ограничением на размер в 6 символов) и массив **awModbusString** типа **WORD**, содержащий три элемента (**STRING** сможет содержать до 6 символов, поскольку каждый **WORD** может содержать два символа):

```
TYPE String_Word :
UNION
    awModbusString      :ARRAY [0..2] OF WORD;
    sStringValue        :STRING;
END_UNION
END_TYPE
```

Рис. 4.79. Объявление переменных объединения

4. Объявите в программе переменную **xVar** типа **BOOL** и **wVar** типа **WORD**, а также экземпляр объединения **Real_Word** с именем **_2WORD_TO_REAL** и экземпляр объединения **String_Word** с именем _ **3WORD_TO_STRING**.

```
PROGRAM PLC_PRG
VAR
    xVar                :BOOL;
    wVar                :WORD;
    _2WORD_TO_REAL      :Real_Word;
    _3WORD_TO_STRING    :String_Word;
END VAR
```

Рис. 4.80. Объявление переменных программ

5. Код программы будет выглядеть следующим образом:

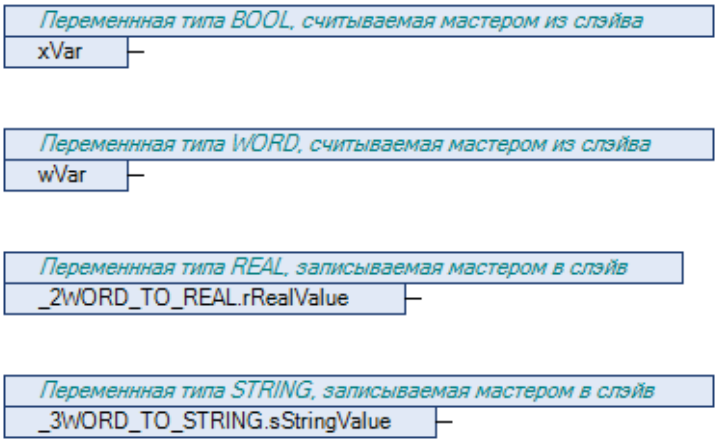


Рис. 4.81. Код программы на языке **CFC**

6. Добавьте в проект устройство **Modbus COM**:

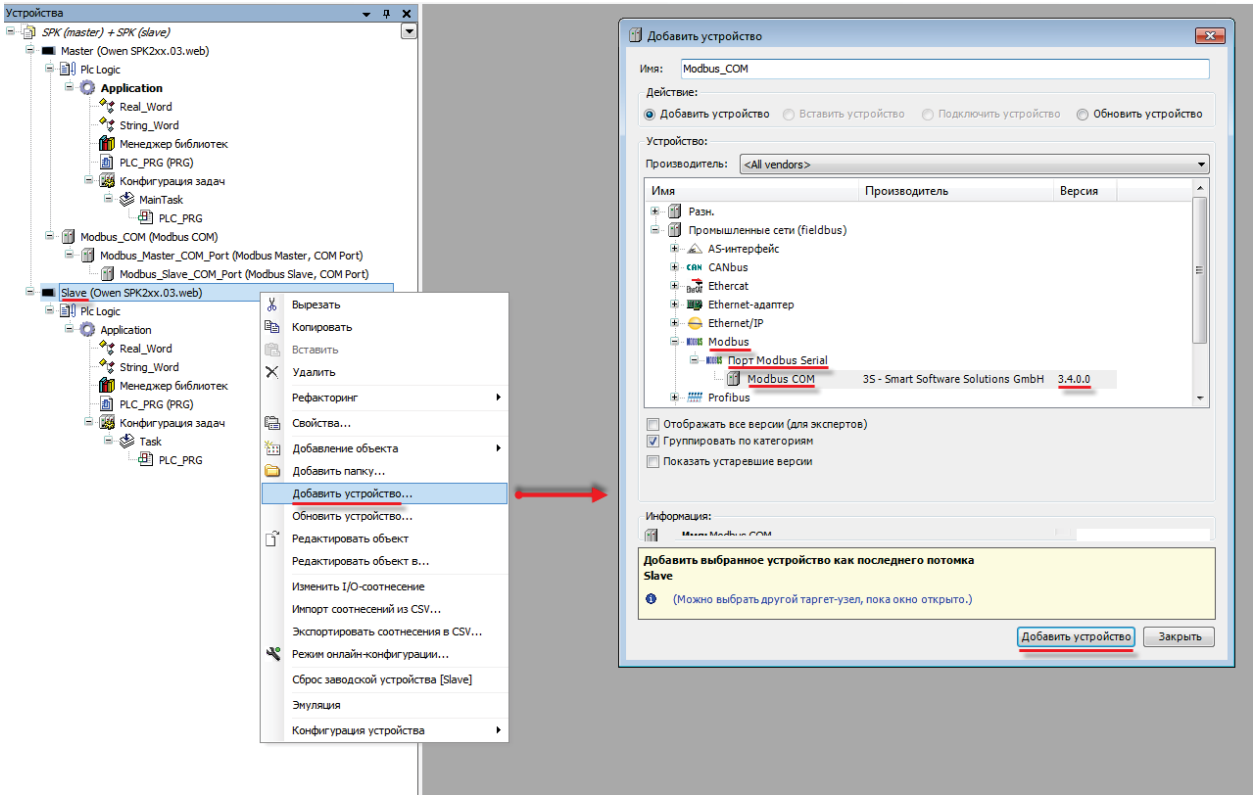


Рис. 4.82. Добавление в проект устройства **Modbus COM**

В конфигурации COM-порта укажите сетевые настройки в соответствии с табл. 4.3, а также номер порта. COM-порту **1** будет соответствовать номер **2** (см. п. 2.3.1).

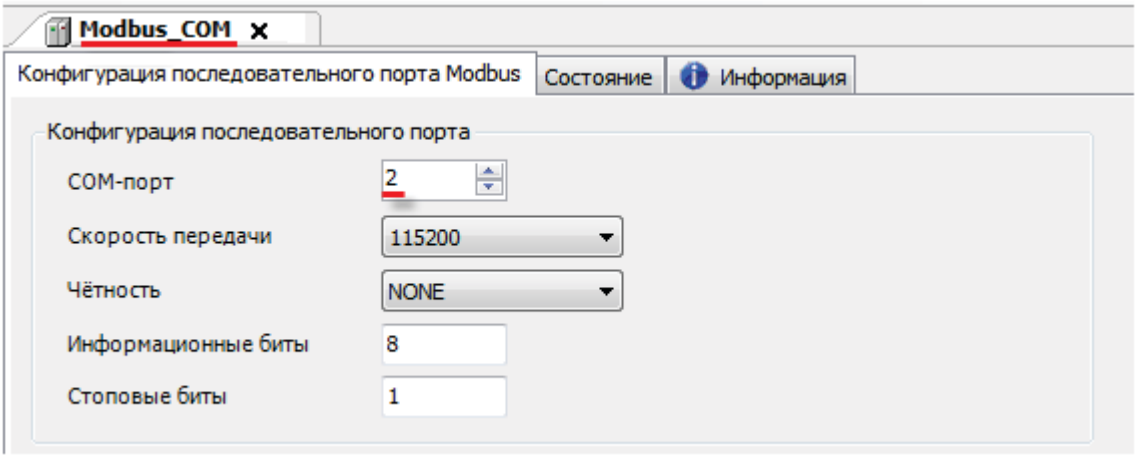


Рис. 4.83. Настройки COM-порта **COM1**

7. В COM-порт добавьте компонент **Modbus Serial Device:**

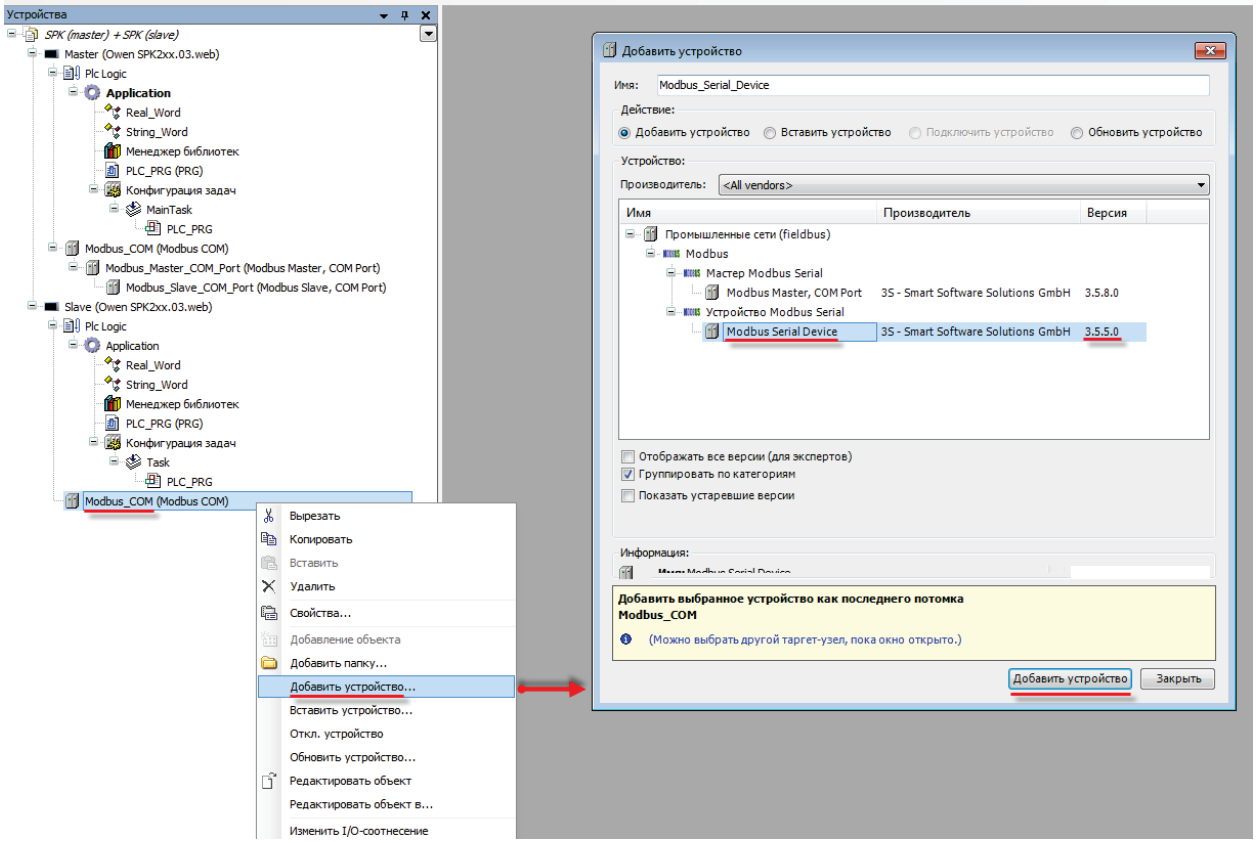


Рис. 4.84. Добавление компонента **Modbus Serial Device**

В настройках компонента на вкладке **Modbus Serial Device** укажите адрес slave-устройства (**1** в соответствии с табл. 4.3).

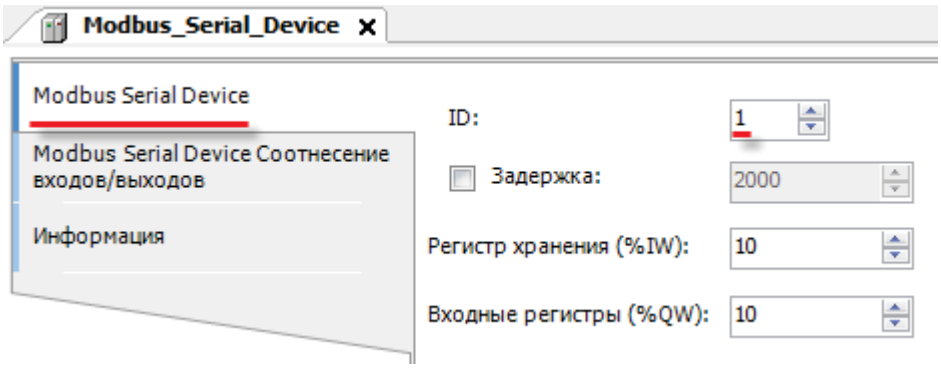


Рис. 4.85. Настройки компонента **Modbus Serial Device**

На вкладке **Modbus Serial Device Соотнесение входов/выходов** привяжите к регистрам переменные программы в соответствии с табл. 4.4.

Обратите внимание, что канал **Inputs** содержит **Holding регистры**, а канал **Outputs** – **Input регистры**.

Не забудьте у параметра **Всегда обновлять переменные** выставить значение **Включено 2**.

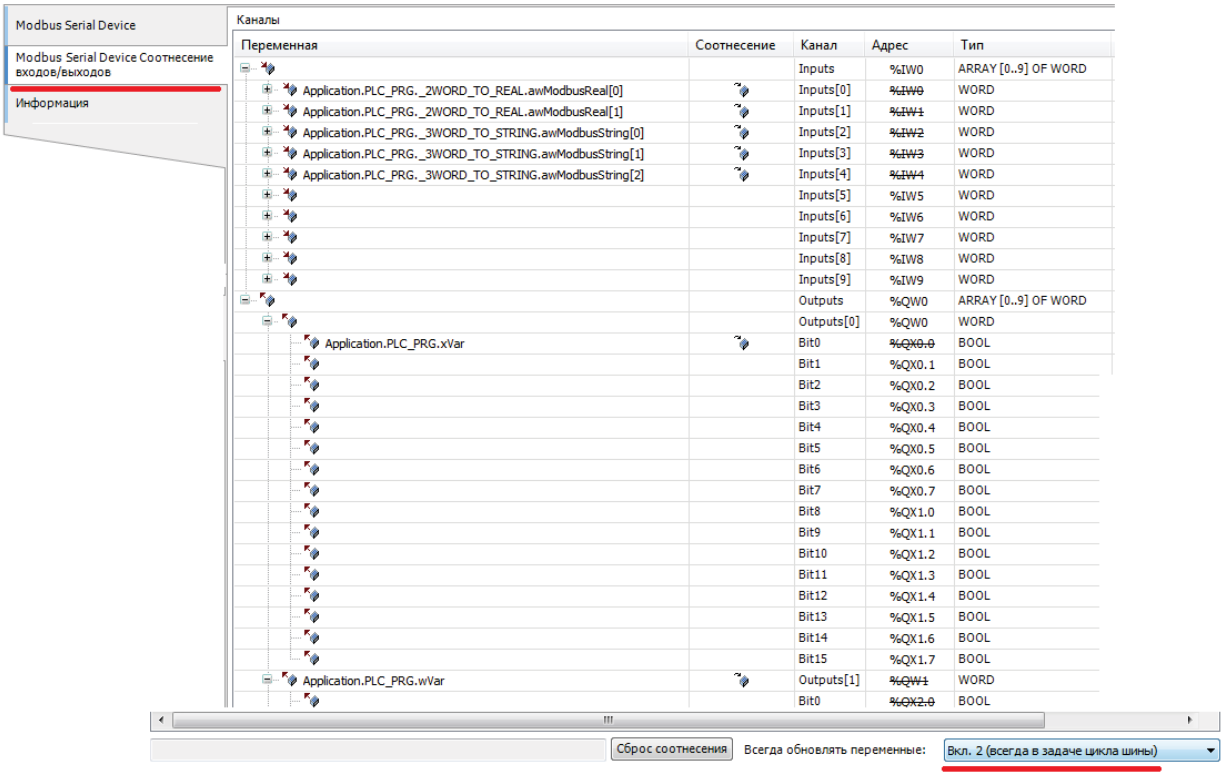


Рис. 4.86. Привязка переменных к регистрам slave-устройства

На этом настройка **СПК (slave)** завершена.

4.7.3. Запуск примера

1. Соедините первый COM-порт СПК (master) с первым COM-портом СПК (slave). См. п. 2.3.2.

2. Загрузите проекты в каждый из СПК и запустите их.

3. Изменяйте в программе СПК (slave) значения переменных **xVar** и **wVar** и наблюдайте соответствующие изменения в программе СПК (master).

Изменяйте в программе СПК (master) значения переменных **rRealValue** и **sStringValue** и наблюдайте соответствующие изменения в программе СПК (slave).

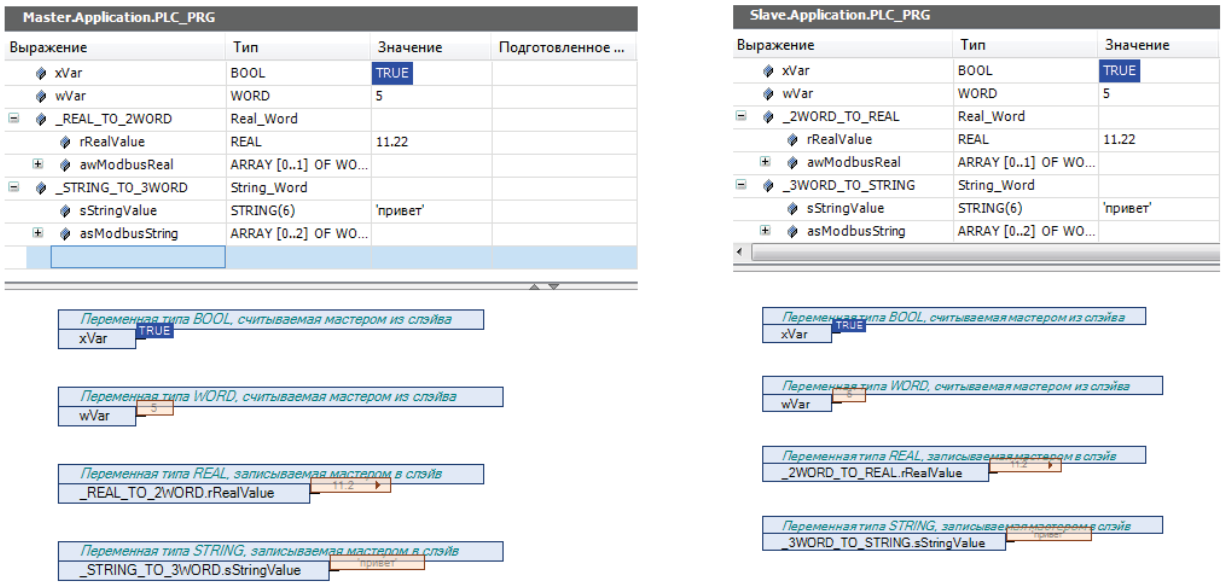


Рис. 4.87. Выполнение программ в режиме **Online**

5. Библиотека ModulsOwenLib

5.1. Установка библиотеки

Библиотека **ModulsOwenLib** доступна на диске с ПО, входящем в комплект поставки, а также на сайте компании ОВЕН в разделе **CODESYS V3/Библиотеки**. Она распространяется как отдельно, так и в составе пакета библиотек Овен **LibInstall.**

Для установки пакета в **CODESYS** в меню **Инструменты** выберите пункт **Менеджер пакетов**, после чего укажите путь к файлу пакета и нажмите **Установить**:

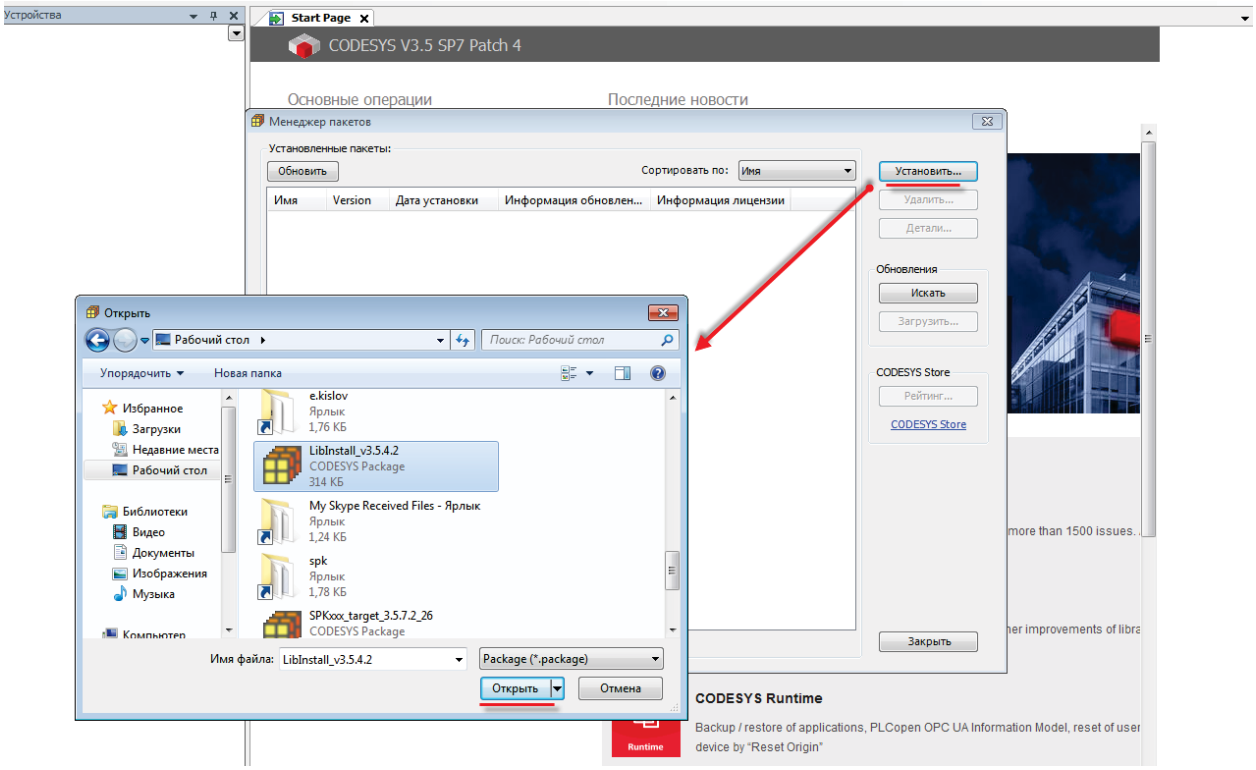


Рис. 5.1. Установка пакета библиотек Овен в среду **CODESYS**

В появившемся диалоговом окне выберите пункт **Полная установка**, после чего нажмите кнопку **Next**:

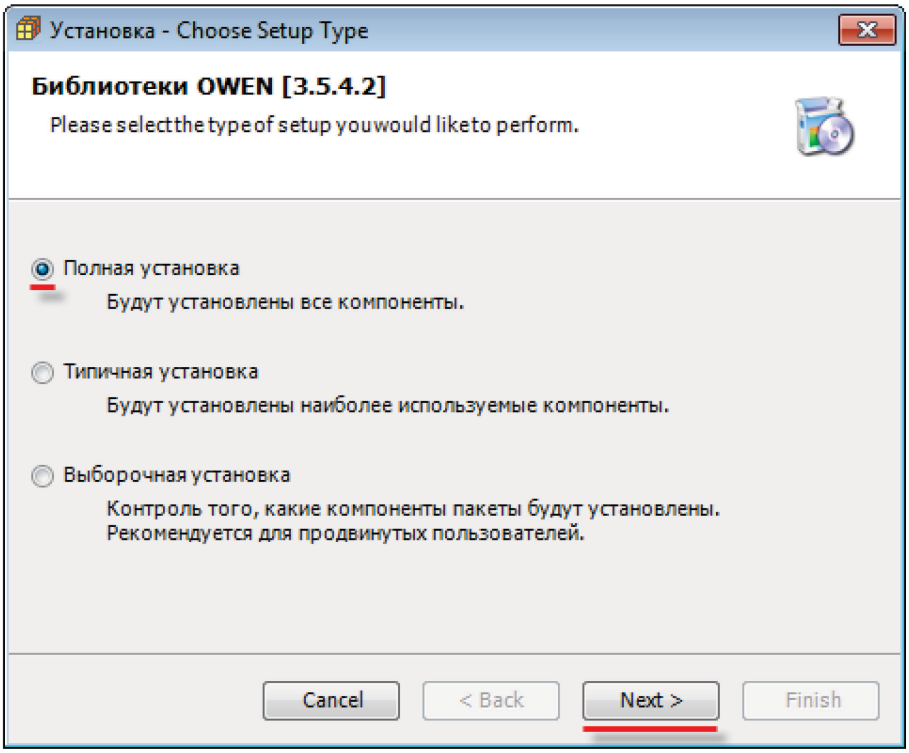


Рис. 5.2. Начало установки пакета библиотек

После завершения установки закройте диалоговое окно с помощью кнопки **Finish**:

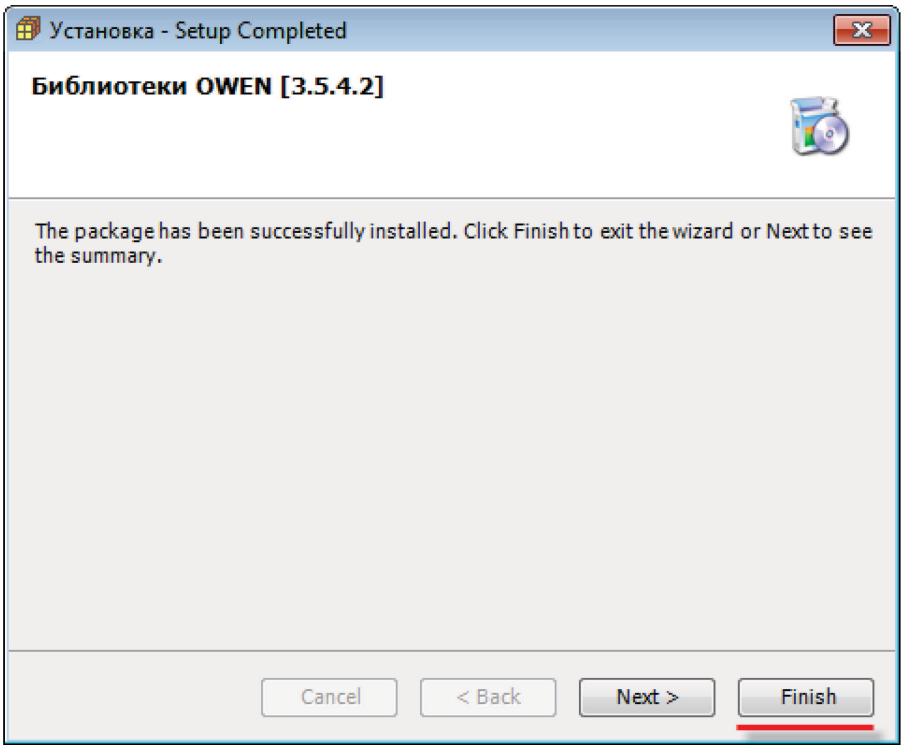


Рис. 5.3. Завершение установки пакета библиотек

5.2. Добавление библиотеки в проект CODESYS

Для добавления библиотеки **ModulsOwenLib** в проект **CODESYS**, в **Менеджере библиотек** нажмите кнопку **Добавить библиотеку** и в строке поиска введите **owen**, после чего выберите из списка нужную библиотеку и нажмите **ОК**.

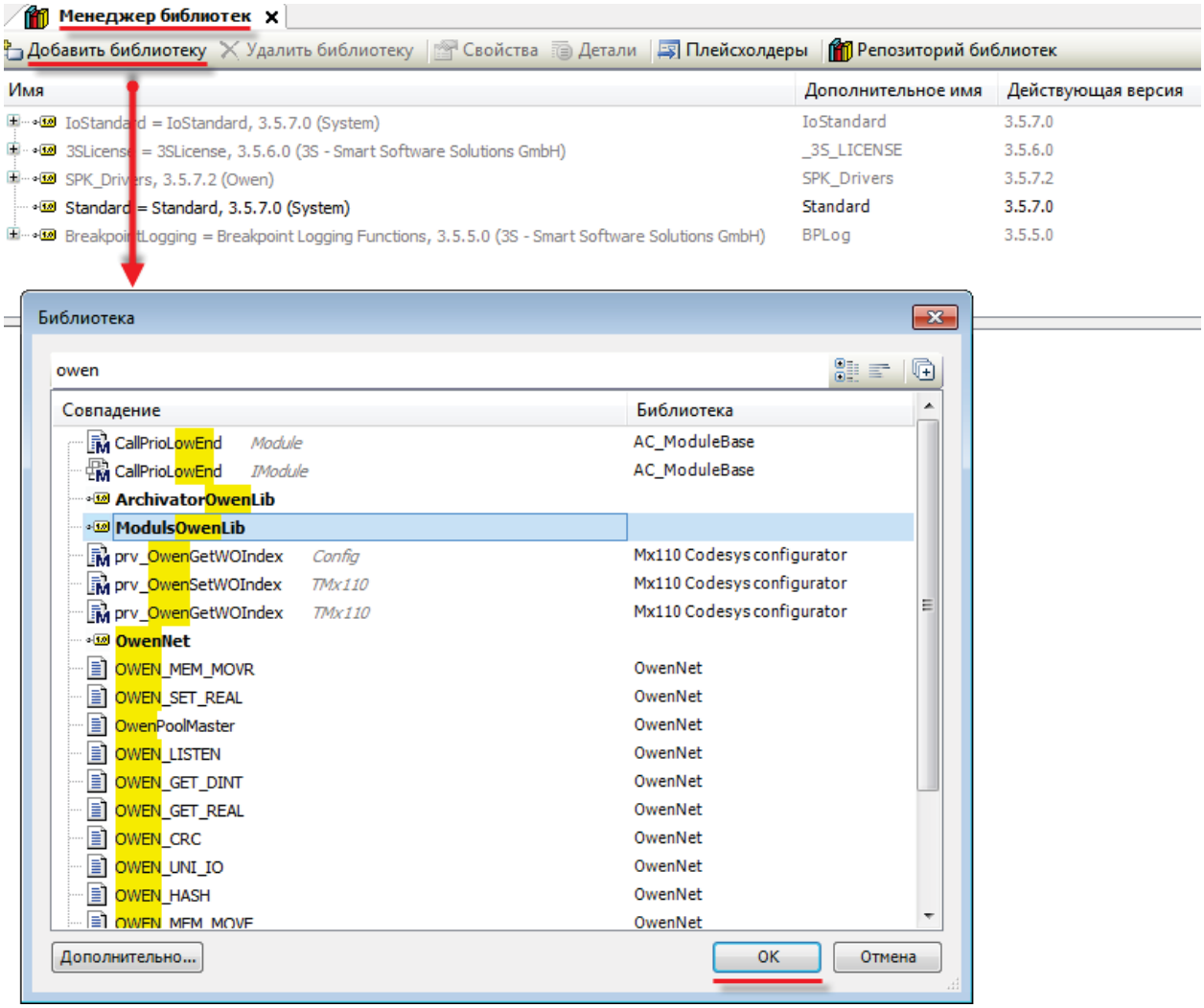


Рис. 5.4. Добавление библиотеки **ModulsOwenLib**

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

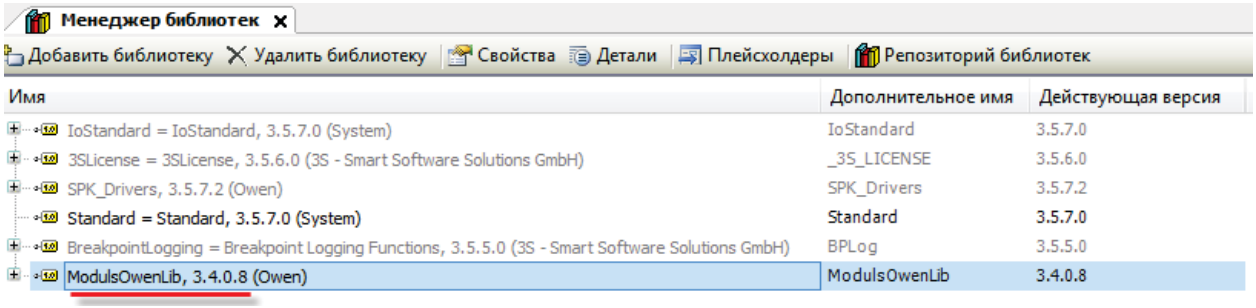


Рис. 5.5. Список библиотек проекта

5.3. Описание ФБ библиотеки

5.3.1. Блок ComConn

Функциональный блок **ComConn** используется для открытия/закрытия COM-порта, а также его настройки.

Имя переменной	Тип	Описание
Входные переменные		
Enable	BOOL	Переменная открытия порта. По ее переднему фронту COM-порт номер **PortNum** открывается с заданными настройками. Если переменная принимает значение **FALSE**, то порт не закрывается; для закрытия порта используется переменная **Port_CloseOn**.
PortNum	SysCom.COM_Ports	Номер порта СПК (см. п. 2.3.1).
PortBaudrate	SysCom.COM_Baudrate	Скорость обмена, бод. Возможные значения: **4800/9600/19200/38400/57600/115200**.
PortParity	SysCom.COM_Parity	Режим контроля паритета. Возможные значения: **0** – отсутствие контроля паритета (**NONE**); **1** – четный (**EVEN**); **2** – нечетный (**ODD**).
PortStopBits	SysCom.COM_StopBits	Количество стоп-бит. Возможные значения: **1** – 1 бит; **2** – 1.5 бит; **3** – 2 бит.
PortByteSize	BYTE	Количество информационных бит в передаваемых/принимаемых байтах. Обычно выбирается значение **8** для **Modbus RTU** и **7** для **Modbus ASCII**.
Port_Mode	WORD	Режим работы COM-порта. Возможные значения: **0** – RS-232; **2** – RS-485.
Port_ModeOn	BOOL	Переменная переключения режима работы COM-порта. Если она имеет значение **TRUE**, то при открытии порта будет выбран режим работы, указанный в переменной **Port_Mode**.
Port_CloseOn	BOOL	Переменная закрытия порта. По ее переднему фронту COM-порт номер **PortNum** закрывается. Во время закрытия порта переменная **Enable** должна иметь значение **FALSE**.
Выходные переменные		
Handle	SysCom.RTS_IEC_HANDLE	Идентификатор открытого порта, используется для обращения к ФБ опроса модулей.
Done	BOOL	Переменная состояния порта. Принимает значение **TRUE**, если порт открыт. Принимает значение **FALSE**, если порт закрыт.
ErrCode	SysCom.RTS_IEC_RESULT	Код ошибки. Расшифровка кодов приведена в системной библиотеке **CmpErrors.library**.

5.3.2. Блоки входов и выходов модулей

В библиотеке **ModulsOwenLib** входы и выходы каждого модуля **Mx110** представляют собой отдельные функциональные блоки. Все эти блоки содержат ряд типичных переменных (описаны ниже) и уникальных переменных, использующихся для непосредственного обращения к входам/выходам (приведены в приложении Б). ***Обратите внимание***, что опрос модулей реализован через групповые запросы (т.е. при использовании ФБ опроса на всех его выходах появятся считанные с модуля данные).

Имя переменной	Тип	Описание
Входные переменные		
Handle	SysCom.RTS_IEC_HANDLE	Идентификатор порта, поступающий с выхода блока ComConn.
Enabl	BOOL	Переменная опроса модуля. Опрос происходит по переднему фронту переменной.
Modd	MB_MODE	Режим работы протокола. Возможные значения: **MB_RTU** – Modbus RTU; **MB_ASCII** – Modbus ASCII.
Addr	BYTE	Адрес модуля.
TimeOut	TIME	Таймаут ответа модуля. Если в течение этого времени модуль не отвечает, то СПК совершает число перезапросов, равное **ErCl,** и в случае отсутствия ответа переходит к опросу следующего модуля.
ErCl	BYTE	Число перезапросов при отсутствии ответа.
Выходные переменные		
wErrCode	WORD	Код ошибки. Значение **0** соответствует отсутствию ошибок. Код ошибки **16#FFFF (255**) характеризует отсутствие ответа от слэйва по истечению таймаута опроса. Список кодов остальных ошибок приведен в приложении В, табл. В2.
Done	BOOL	Флаг опроса модуля. Принимает значения **TRUE** после завершения опроса модуля, на следующем цикле сбрасывается в **FALSE**.

5.3.3. Блоки UniRead/UniWrite

Блоки **UniRead/UniWrite** используются для чтения/записи данных с произвольных slave-устройств (отличных от модулей **Mx110**). Используемая при этом функция зависит от типа считываемых/записываемых данных (вход **VarType**).

Имя переменной	Тип	Описание
Входные переменные		
Handle	SysCom.RTS_IEC_HANDLE	Идентификатор порта, поступающий с выхода блока ComConn.
Enabl	BOOL	Переменная опроса slave-устройства. Опрос происходит по переднему фронту переменной.
Modd	MB_MODE	Режим работы протокола. Возможные значения: **MB_RTU** – Modbus RTU; **MB_ASCII** – Modbus ASCII.
Addr	BYTE	Адрес slave-устройства.
RegAddr	WORD	Адрес считываемого значения.
TimeOut	TIME	Таймаут ответа slave-устройства. Если в течение этого времени устройство не отвечает, то СПК совершает число перезапросов, равное **ErCl**, и в случае отсутствия ответа переходит к опросу следующего slave-устройства.
ErCl	BYTE	Число перезапросов при отсутствии ответа.
VarAdr	POINTER TO BYTE	Для **UniRead**: указатель на первый байт массива, в который записывается считанное значение. Для **UniWrite:** указатель на первый байт массива, в котором хранится записываемое значение.
VarType	WORD	Тип считываемых/записываемых данных. Возможные значения: **0** – BYTE; **1** – WORD; **2** – DWORD: **3** – REAL.
Выходные переменные		
wErrCode	WORD	Код ошибки. Значение **0** соответствует отсутствию ошибок. Код ошибки **16#FFFF (255)** характеризует отсутствие ответа от слэйва по истечению таймаута опроса. Список кодов остальных ошибок приведен в приложении В, табл. В2.
Done	BOOL	Флаг опроса модуля. Принимает значения **TRUE** после завершения опроса модуля, на следующем цикле сбрасывается в **FALSE**.

5.4. Пример: СПК207 + модули Mx110 (MB110-8A, MB110-16Д, МУ110-16Р)

Рассмотрим пример настройки обмена с **модулями Mx110** с использованием **библиотеки ModulsOwenLib**. В нем мы наладим связь между контроллером **СПК207.03** (master) и тремя модулями (slave-устройствами).

Реализуемый алгоритм: если значение 1-го аналогового входа модуля **MB110-8A** превышает 30 и при этом 1-ый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то произвести запись в 1-й дискретный выход модуля **МУ110-16Р** значения **TRUE** (замкнут). Во всех остальных случаях присвоить дискретному выходу значение **FALSE** (разомкнут).

Структурная схема примера приведена на рис. 5.6:

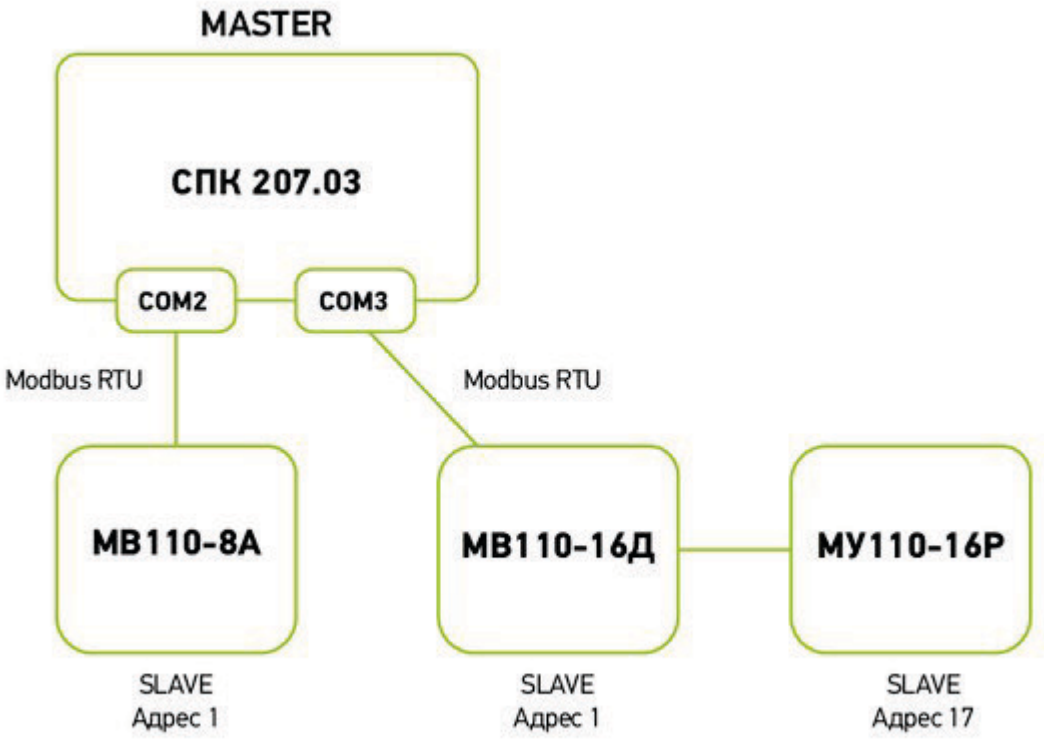


Рис. 5.6. Структурная схема примера **ModulsOwenLib (СПК+модули Mx110)**

Пример создан в среде **CODESYS 3.5 SP7 Patch4** и подразумевает запуск на **СПК207.03.CS(-WEB)**.

Пример доступен для скачивания: Example_ModulsOwenLib.projectarchive

Сетевые параметры модулей приведены в табл. 5.1.

Табл. 5.1. Сетевые параметры модулей Mx110

Параметр	**MB110-8A**	**MB110-16Д**	**МУ110-16Р**
COM-порт СПК, к которому подключен модуль	*COM2*	*COM3*	
Адрес модуля	1	1	17
Скорость обмена	115200		
Количества бит данных	8		
Контроль четности	отсутствует		
Количество стоп-бит	1		

Пример содержит два проекта – **Device_CFC** и **Device_ST**. Каждый проект содержит библиотеку **ModulsOwenLib**, в каждом из них реализован описанный алгоритм на соответствующем языке программирования.

Каждый проект содержит три программы:

- **COM2** (реализация обмена через COM-порт COM2 с модулем **MB110-8A**);

- **COM3** (реализация обмена через COM-порт COM3 с модулями **MB110-16Д** и **МУ110-16Р**);

- **PLC_PRG** (основная программа, реализующая пользовательский алгоритм).

Каждая программа привязана к отдельной задаче, время цикла каждой из задач – **t#20ms**.

5.4.1. Описание реализации на языке CFC

1. Переменные программы **COM2**:

COM2

```
VAR
    ComConn_COM2:              ComConn;          // ФБ настройки и открытия порта COM2
    My8A:                      MV110_8A_inputs;  // ФБ опроса модуля MB110-8A

    xModule1:                  BOOL;             // Переменная завершения цикла опроса

    rMV110_8A_input1:          REAL;             // Показание входа 1 модуля MB110-8A
    wsMV110_8A_input1_status:  WSTRING;          // Статус измерения входа 1
END_VAR
```

Рис. 5.8. Объявление переменных программы **COM2 (CFC)**

Код программы **COM2** на языке **CFC** (*рисунок хорошо масштабируется*):

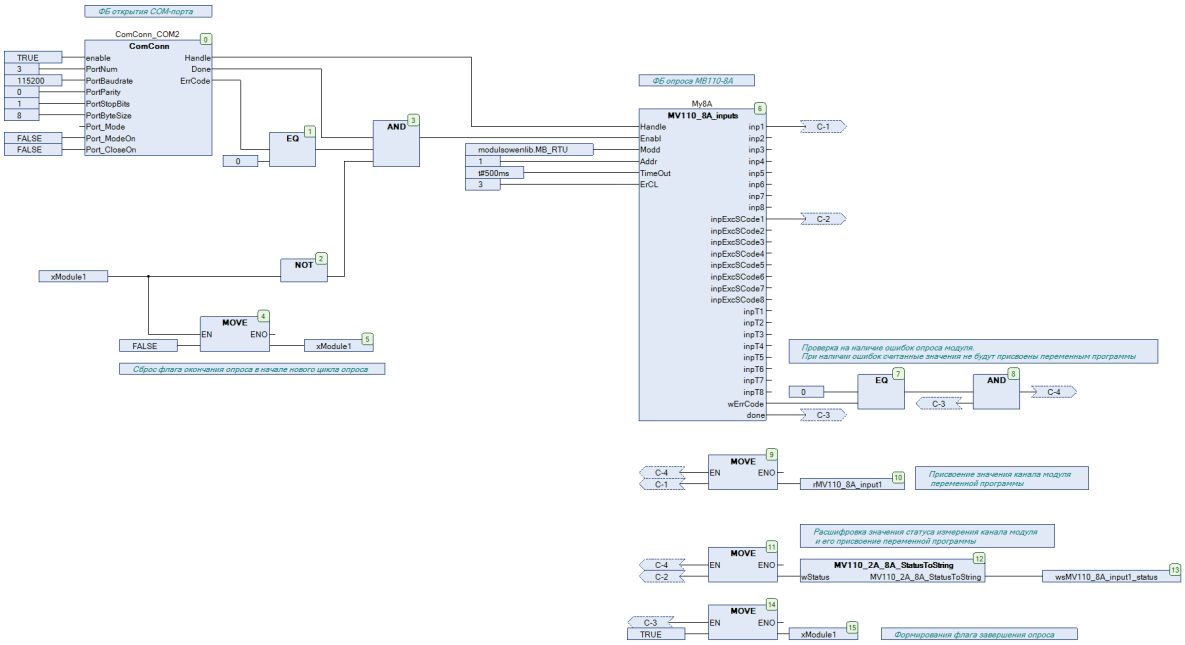


Рис. 5.9. Код программы **COM2 (CFC)**

Программа работает следующим образом:

Блок 0. При первом запуске программы с помощью ФБ ComConn открывается COM-порт **COM2** с сетевыми настройками, идентичными настройкам модулей (см. табл. 5.1). ***Обратите внимание***, что COM-порту **COM2** соответствует номер **3** (см. п. 2.3.1).

Блоки 1-3. Формируется сигнал инициализации опроса модуля. Опрос будет начат только в том случае, если:

- ФБ ComConn завершил работу (**Done=TRUE**) без ошибок (**ErrCode=0**);

- предыдущий цикл опроса завершен. ***Обратите внимание***, что флаг завершения предыдущего цикла опроса инвертируется. Это необходимо, так как опрос модуля начинается по переднему фронту входа **Enabl** – соответственно, перед началом нового цикла опроса он должен принять значение **FALSE**.

Блоки 4-5. Сброс флага завершения предыдущего цикла опроса.

Блок 6. Инициализация цикла опроса модуля **MB110-8A** с помощью ФБ MV110_8A_inputs. На вход **Handle** данного ФБ поступает значение выхода **Handle** ФБ ComConn. На вход **Addr** поступает адрес модуля – согласно табл. 5.1, данный модуль имеет адрес **1**.

Блоки 7-13. Если цикл опроса модуля завершен (об этом сигнализирует импульс по переднему фронту на выходе **Done**) без ошибок (**wErrCode=0**), тогда считанное с модуля значение 1-го входа присваивается переменной программы **rMV110_8A_input1**, а значение статуса измерения 1-го входа декодируется с помощью функции **MV110_2A_8A_StatusToString** (включена в target-файл СПК) и присваивается переменной **wsMV110_8A_input1_status**.

Блоки 14-15. Формируется флаг окончания данного цикла опроса.

Обратите внимание, что **цикл опроса модуля** происходит в течение нескольких **циклов программы**.

2. Переменные программы **COM3**:

COM3

```
PROGRAM COM3
VAR
    ComConn_COM3:           ComConn;          // ФБ настройки и открытия порта COM3

    My16D:                  MV110_16D_inputs; // ФБ опроса модуля МВ110-16Д
    My16R:                  MY110_16R_outs;   // ФБ опроса модуля МУ110-16Р

    xMV110_16D_input1:      BOOL;             // Состояние 1-го входа модуля МВ110-16Д

    xModule1, xModule2:     BOOL;             // Переменные завершения цикла опроса (1 - МВ110-16Д, 2 - МУ110-16Р)
END_VAR

VAR_INPUT
    xMY110_16R_output1:     BOOL;             // Переменная для записи состояния 1-го выхода модуля МУ110-16Р
END_VAR
```

Рис. 5.10. Объявление переменных программы **COM3 (CFC)**

Код программы **COM3** на языке **CFC** приведен на рис. 5.11 (*рисунок хорошо масштабируется*).

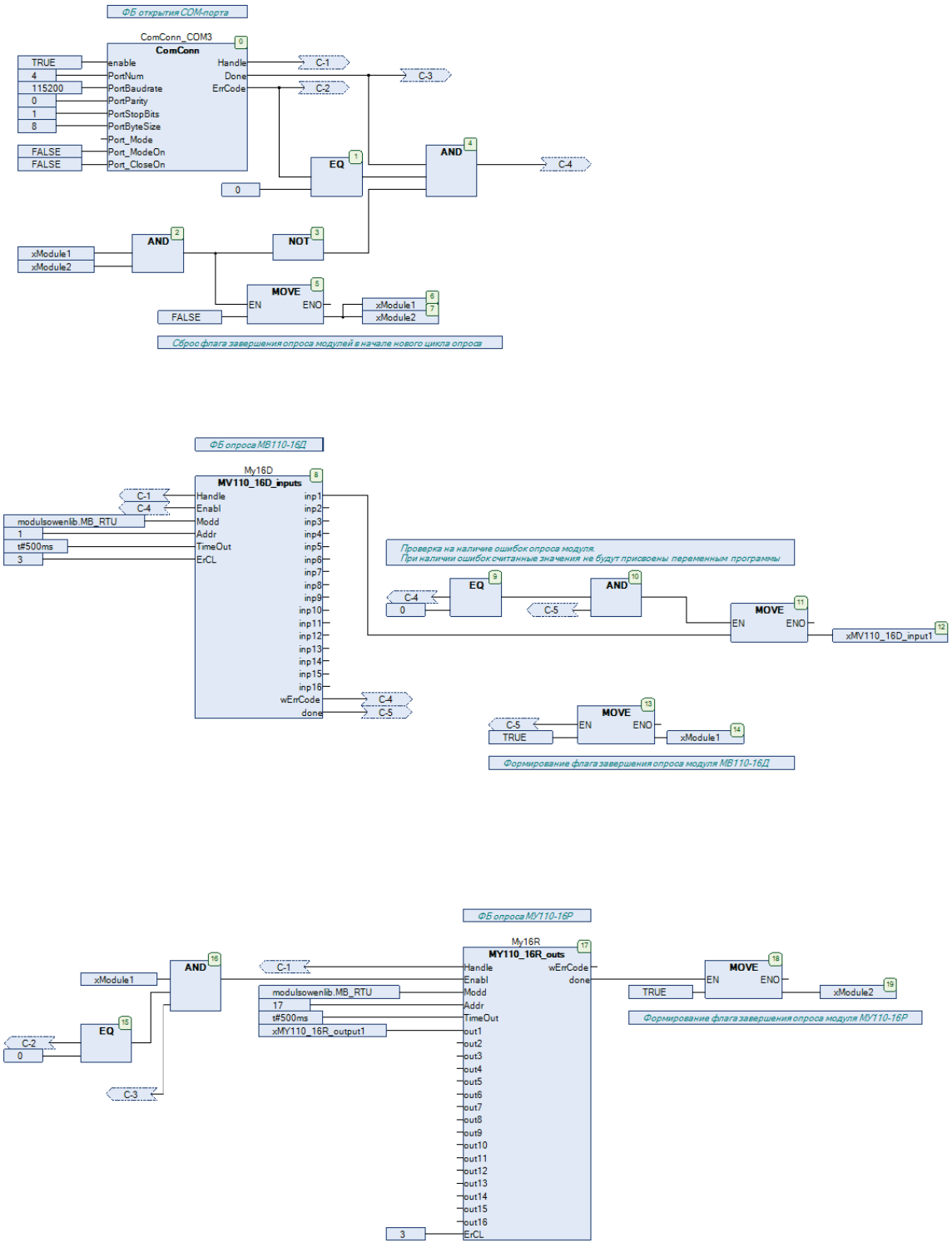


Рис. 5.11. Код программы **COM3 (CFC)**

Программа работает следующим образом:

Блок 0. При первом запуске программы с помощью ФБ **ComConn** открывается COM-порт **COM3** с сетевыми настройками, идентичными настройкам модулей (см. табл. 5.1). ***Обратите внимание***, что COM-порту **COM3** соответствует номер **4** (см. п. 2.3.1).

Блоки 1-4. Формируется сигнал инициализации опроса модуля **МВ110-16Д**. Опрос будет начат только в том случае, если:

- ФБ ComConn завершил работу (**Done=TRUE**) без ошибок (**ErrCode=0**);

- предыдущий цикл опроса завершен. ***Обратите внимание***, что флаги завершения предыдущего цикла опроса инвертируются. Это необходимо, так как опрос модулей начинается по переднему фронту входа **Enabl** – соответственно, перед началом нового цикла опроса они должны принять значение **FALSE**.

Блоки 5-7. Сброс флагов завершения предыдущего цикла опроса.

Блок 8. Инициализация цикла опроса модуля **МВ110-16Д** с помощью ФБ MV110_16D_inputs. На вход **Handle** данного ФБ поступает значения выхода **Handle** ФБ ComConn. На вход **Addr** поступает адрес модуля – согласно табл. 5.1, данный модуль имеет адрес **1**.

Блоки 9-12. Если цикл опроса модуля завершен (об этом сигнализирует импульс по переднему фронту на выходе **Done**) без ошибок (**wErrCode=0**), тогда считанное с модуля значение 1-го входа присваивается переменной программы **xMV110_16D_input1**.

Блоки 13-14. Формируется флаг окончания данного цикла опроса модуля **МВ110-16Д**.

Блоки 15-16. Формируется сигнал инициализации опроса модуля **МУ110-16Р**. Опрос будет начат только в том случае, если:

- ФБ ComConn завершил работу (**Done=TRUE**) без ошибок (ErrCode=0);

- текущий цикл опроса модуля **МВ110-16Д** завершен.

Блок 17. Инициализация цикла опроса модуля **МУ110-16Р** с помощью ФБ MY110_16R_outs. На вход **Handle** данного ФБ поступает значения выхода **Handle** ФБ ComConn. На вход **Addr** поступает адрес модуля – согласно табл. 5.1, данный модуль имеет адрес **17**. Значение переменной программы **xMY110_16R_output1** записывается на 1-й выход модуля.

Блоки 18-19. Формируется флаг окончания данного цикла опроса модуля **МУ110-16Р**.

Обратите внимание, что **цикл опроса модуля** происходит в течение нескольких **циклов программы**.

При необходимости количество опрашиваемых модулей можно увеличить – для этого, соответственно, потребуется увеличить число переменных **xModule№.** ***Обратите внимание***, что приступать к опросу следующего модуля можно только после завершения опроса предыдущего (т.е. после того, когда переменная **xModule№** предыдущего модуля принимает значение **TRUE)**.

3. Код программы **PLC_PRG:**

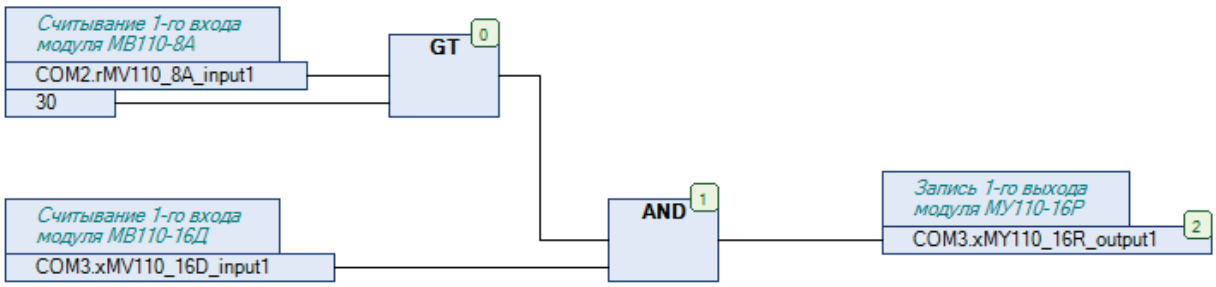


Рис. 5.12. Код программы **PLC_PRG (CFC)**

Программа **PLC_PRG** оперирует переменными программ **COM2** и **COM3**, и реализует следующий алгоритм: если значение 1-го аналогового входа модуля **MB110-8A** превышает 30 и при этом 1-ый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то произвести запись в 1-й дискретный выход модуля **MУ110-16P** значения **TRUE** (замкнут). Во всех остальных случаях присвоить дискретному выходу значение **FALSE** (разомкнут).

5.4.2. Описание реализации на языке ST

1. Переменные программы **COM2**:

```
PROGRAM COM2
VAR
    ComConn_COM2:               ComConn;          // ФБ настройки и открытия порта COM2
    My8A:                       MV110_8A_inputs;  // ФБ опроса модуля MB110-8A

    rMV110_8A_input1:           REAL;             // Показание входа 1 модуля MB110-8A
    wsMV110_8A_input1_status:   WSTRING;          // Статус измерения входа 1
END_VAR
```

Рис. 5.13. Объявление переменных программы **COM2 (ST)**

Код программы **COM2** на языке **ST** (*листинг приведен в Приложении Д.1*):

```
// [1] открываем COM-порт
ComConn_COM2
(
    enable:=TRUE,
    PortNum:=3,
    PortBaudrate:=115200,
    PortParity:=0,
    PortStopBits:=1,
    PortByteSize:=8,
    Port_ModeOn:=FALSE,
    Port_CloseOn:=FALSE,
);


// [2] запускаем ФБ опроса модуля
My8A
(
    Handle:=ComConn_COM2.Handle,
    // если COM-порт открыт и отсутствуют ошибки...
    // ...то начинаем новый цикл опроса
    Enabl:=(ComConn_COM2.Done AND ComConn_COM2.ErrCode=0),
    Modd:=modulsowenlib.MB_MODE.MB_RTU,
    Addr:=1,
    TimeOut:=T#500MS,
    ErCl:=3,
);


// [3] если ФБ опроса модуля завершил работу...
IF My8A.Done THEN
    // ...и ошибки отсутствуют, то забираем значения модуля...
    IF My8A.wErrCode=0 THEN
        rMV110_8A_input1:=My8A.inp1;
        wsMV110_8A_input1_status:=MV110_2A_8A_StatusToString(My8A.inpExcSCode1);
    END_IF

    // ...завершаем опрос модуля MB110-8A
    My8A(Enabl:=FALSE);

END_IF
```

Рис. 5.14. Код программы **COM2 (ST)**

Программа работает следующим образом:

Блок [1]. При первом запуске программы с помощью ФБ ComConn открывается COM-порт **COM2** с сетевыми настройками, идентичными настройкам модулей (см. табл. 5.1). ***Обратите внимание***, что COM-порту **COM2** соответствует номер **3** (см. п. 2.3.1).

Блок [2]. Инициализация цикла опроса модуля **MB110-8A** с помощью ФБ MV110_8A_inputs. На вход **Handle** данного ФБ поступает значение выхода **Handle** ФБ ComConn. На вход **Addr** поступает адрес модуля – согласно табл. 5.1, данный модуль имеет адрес **1**.

Опрос модуля будет начат только в том случае, если ФБ ComConn завершил работу (**Done=TRUE**) без ошибок (**ErrCode=0**).

Блок [3]. Если цикл опроса модуля завершен (об этом сигнализирует импульс по переднему фронту на выходе **Done**) без ошибок (**wErrCode=0**), тогда считанное с модуля значение 1-го входа присваивается переменной программы **rMV110_8A_input1**, а значение статуса измерения 1-го входа декодируется с помощью функции **MV110_2A_8A_StatusToString** (включена в target-файл СПК) и присваивается переменной **wsMV110_8A_input1_status**. Независимо от наличия ошибок завершаем данный цикл опроса.

Обратите внимание, что **цикл опроса модуля** происходит в течение нескольких **циклов программы**.

2. Переменные программы **COM3**:

```
PROGRAM COM3
VAR
    ComConn_COM3:           ComConn;          // ФБ настройки и открытия порта COM3

    My16D:                  MV110_16D_inputs; // ФБ опроса модуля MB110-16Д
    My16R:                  MY110_16R_outs;   // ФБ опроса модуля MV110-16P

    xMV110_16D_input1:      BOOL;             // Состояние 1-го входа модуля MB110-16Д

    xModule:INT; // Переменные начала цикла опроса (0 - MB110-16Д, 1 - MV110-16P)
END_VAR

VAR_INPUT
    xMY110_16R_output1:     BOOL;             // Переменная для записи состояния 1-го выхода модуля MB110-16P
END_VAR
```

Рис. 5.15. Объявление переменных программы **COM3 (ST)**

Код программы **COM3** на языке **ST** приведен на рис. 5.16 (*рисунок хорошо масштабируется, листинг приведен в Приложении Д.2*):

```
// [1] открываем COM-порт
ComConn_COM3
(
    enable:=TRUE,
    PortNum:=4,
    PortBaudrate:=115200,
    PortParity:=0,
    PortStopBits:=1,
    PortByteSize:=8,
    Port_ModeOn:=FALSE,
    Port_CloseOn:=FALSE,
);


// [2] xModule определяет опрашиваемый модуль: 0 - MB110-16Д, 1 - MУ110-16Р
CASE xModule OF

    0:

        // [2.0.1] запускаем ФБ опроса модуля MB110-16Д
        My16D
        (
            Handle:=ComConn_COM3.Handle,
            // если COM-порт открыт и ошибки отсутствуют, то начинаем опрос
            Enabl:=(ComConn_COM3.Done AND ComConn_COM3.ErrCode=0),
            Modd:=modulsowenlib.MB_MODE.MB_RTU,
            Addr:=1,
            TimeOut:=T#500MS,
            ErCl:=3,
        );


        // [2.0.2] если ФБ опроса модуля завершил работу...
        IF My16D.Done THEN
            // ...и ошибки отсутствуют, то забираем значения модуля
            IF My16D.wErrCode=0 THEN
                xMV110_16D_input1:=My16D.inp1;
            END_IF

            // завершаем опрос модуля MB110-16Д
            My16D(Enabl:=FALSE);

            // переходим к опросу модуля MУ110-16Р
            xModule:=1;

        END_IF

    1:

        // [2.1.1] запускаем ФБ опроса модуля MУ110-16Р
        My16R
        (
            Handle:=ComConn_COM3.Handle,
            // если COM-порт открыт и ошибки отсутствуют, то начинаем опрос
            Enabl:=(ComConn_COM3.Done AND ComConn_COM3.ErrCode=0),
            Modd:=modulsowenlib.MB_MODE.MB_RTU,
            Addr:=17,
            TimeOut:=T#500MS,
            ErCl:=3,
            // записываем значения в модуль
            out1:=xMY110_16R_output1,
        );


        /// [2.1.2] если ФБ опроса модуля завершил работу...
        IF My16R.Done THEN

            // ... то завершаем опрос модуля MУ110-16Р...
            My16R(Enabl:=FALSE);

            // ...и начинаем новый цикл опроса
            xModule:=0;
        END_IF

END CASE
```

Рис. 5.16. Код программы **COM3 (ST)**

Программа работает следующим образом:

Блок [1]. При первом запуске программы с помощью ФБ ComConn открывается COM-порт **COM3** с сетевыми настройками, идентичными настройкам модулей (см. табл. 5.1). ***Обратите внимание***, что COM-порту **COM3** соответствует номер **4** (см. п. 2.3.1).

Блок [2]. Реализация последовательного опроса модулей через оператор **CASE**:

- в случае **xModule=0** происходит опрос модуля **МВ110-16Д**;

- в случае **xModule=1** происходит опрос модуля **МУ110-16Р**.

Блок [2.0.1]. Инициализация цикла опроса модуля **МВ110-16Д** с помощью ФБ MV110_16D_inputs. На вход **Handle** данного ФБ поступает значение выхода **Handle** ФБ ComConn. На вход **Addr** поступает адрес модуля – согласно табл. 5.1, данный модуль имеет адрес **1**.

Опрос модуля будет начат только в том случае, если ФБ ComConn завершил работу (**Done=TRUE**) без ошибок (**ErrCode=0**).

Блок [2.0.2]. Если цикл опроса модуля завершен (об этом сигнализирует импульс по переднему фронту на выходе **Done**) без ошибок (**wErrCode=0**), тогда считанное с модуля значение 1-го входа присваивается переменной программы **xMV110_16D_input1**. Независимо от наличия ошибок завершаем данный цикл опроса модуля **МВ110-16Д** и начинаем цикл опроса модуля **МУ110-16Р**.

Блок [2.1.1]. Инициализация цикла опроса модуля **МУ110-16Р** с помощью ФБ MY110_16R_outs. На вход **Handle** данного ФБ поступает значение выхода **Handle** ФБ ComConn. На вход **Addr** поступает адрес модуля – согласно табл. 5.1, данный модуль имеет адрес **17**.

Опрос модуля будет начат только в том случае, если ФБ ComConn завершил работу (**Done=TRUE**) без ошибок (**ErrCode=0**).

Запись значения переменной **xMY110_16R_output1** на 1-й выход модуля.

Блок [2.0.2]. Если цикл опроса модуля завершен (об этом сигнализирует импульс по переднему фронту на выходе **Done**), то завершаем данный цикл опроса модуля **МУ110-16Р** и начинаем цикл опроса модуля **МВ110-16Д**.

Обратите внимание, что **цикл опроса модуля** происходит в течение нескольких **циклов программы**.

При необходимости количество опрашиваемых модулей можно увеличить – для этого, соответственно, потребуется увеличить количество действий в операторе **CASE.** ***Обратите внимание***, что перед переходом к опросу следующего модуля, необходимо завершить опрос предыдущего, вызвав ФБ опроса с **Enable=FALSE**.

3. Код программы **PLC_PRG**:

```
1  COM3.xMY110_16R_output1:=(COM2.rMV110_8A_input1>30 AND COM3.xMV110_16D_input1);
```

Рис. 5.17. Код программы **PLC_PRG (ST)**

Программа **PLC_PRG** оперирует переменными программ **COM2** и **COM3**, и реализует следующий алгоритм: если значение 1-го аналогового входа модуля **МВ110-8А** превышает 30 и при этом 1-ый дискретный вход модуля **МВ110-16Д** имеет значение **TRUE** (замкнут), то произвести запись в 1-й дискретный выход модуля **МУ110-16Р** значения **TRUE** (замкнут). Во всех остальных случаях присвоить дискретному выходу значение **FALSE** (разомкнут).

5.5. Рекомендации по использованию библиотеки

1. ***Необходимо понимать***, любой из ФБ библиотеки не может выполниться в течение одного цикла контроллера. При опросе значительного количества регистров, полный опрос одного модуля может занять несколько десятков циклов.

2. Начинать опрос модулей следует только в том случае, когда порт открыт без ошибок (т.е. у ФБ ComConn выход **Done**=**TRUE**, выход **ErrCode**=**0**). Если на выходе **ErrCode** возникают ошибки, то рекомендуется закрыть порт и спустя 100 мс открыть его снова.

3. Опрос модулей должно производиться строго последовательно – т.е. опрос каждого следующего модуля должен начинаться после окончания предыдущего. Об окончании опроса сигнализирует импульс по переднему фронту на выходе **Done** ФБ опроса модуля.

4. При опросе модулей следует анализировать коды возникающих ошибок (выход **wErrCode**). Если с модуля считываются данные, то в случае ошибки полученные значения могут быть некорректны. Если в модуль записываются данные, то в случае ошибки запись может быть не произведена.

5. Значение таймаута опроса должно выбираться индивидуально в каждом конкретном случае в зависимости от скорости обмена, числа опрашиваемых регистров и особенностей slave-устройства (например, некоторые устройства могут удерживать линию после ответа).

6. Библиотека Modbus

6.1. Установка библиотеки

Библиотека **Modbus** доступна на диске с ПО, входящем в комплект поставки, а также на сайте компании ОВЕН в разделе **CODESYS V3/Библиотеки**. Она распространяется как отдельно, так и в составе пакета библиотек Овен **LibInstall.**

Для установки пакета в **CODESYS** в меню **Инструменты** выберите пункт **Менеджер пакетов**, после чего укажите путь к файлу пакета и нажмите **Установить**:

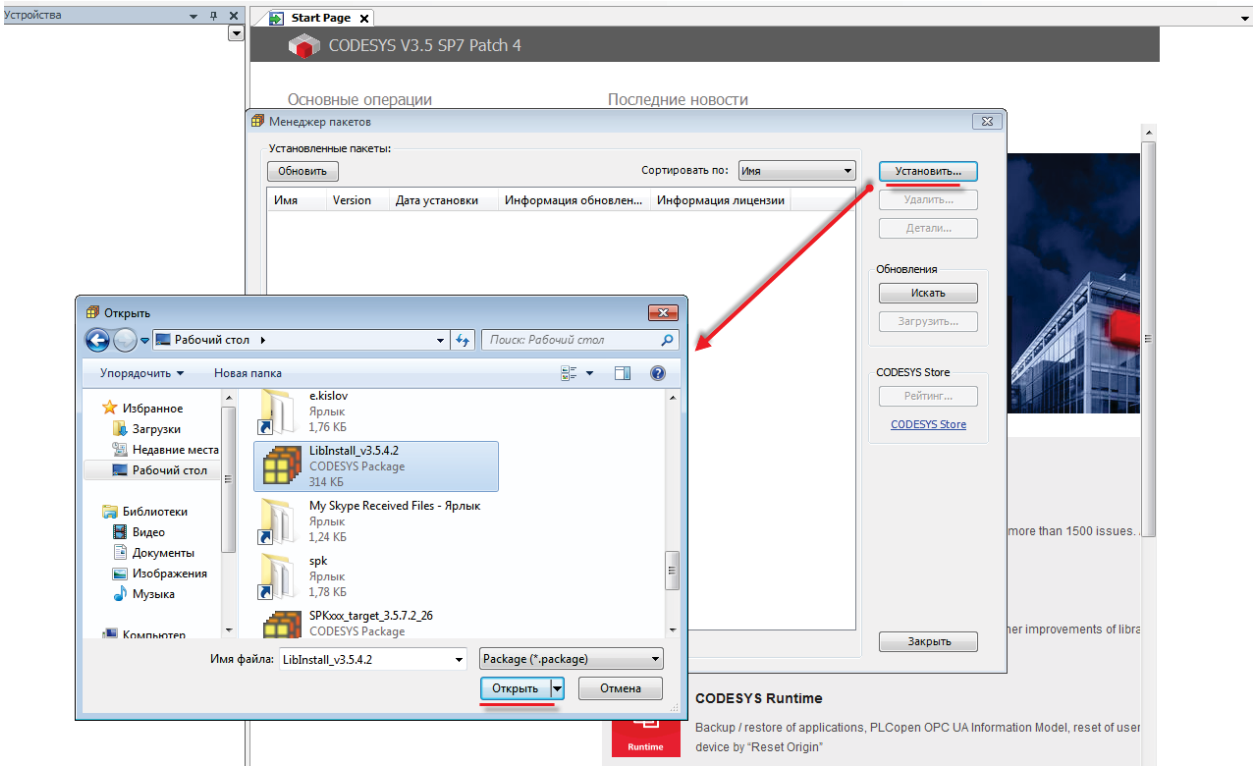


Рис. 6.1. Установка пакета библиотек Овен в среду **CODESYS**

В появившемся диалоговом окне выберите пункт **Полная установка**, после чего нажмите кнопку **Next**:

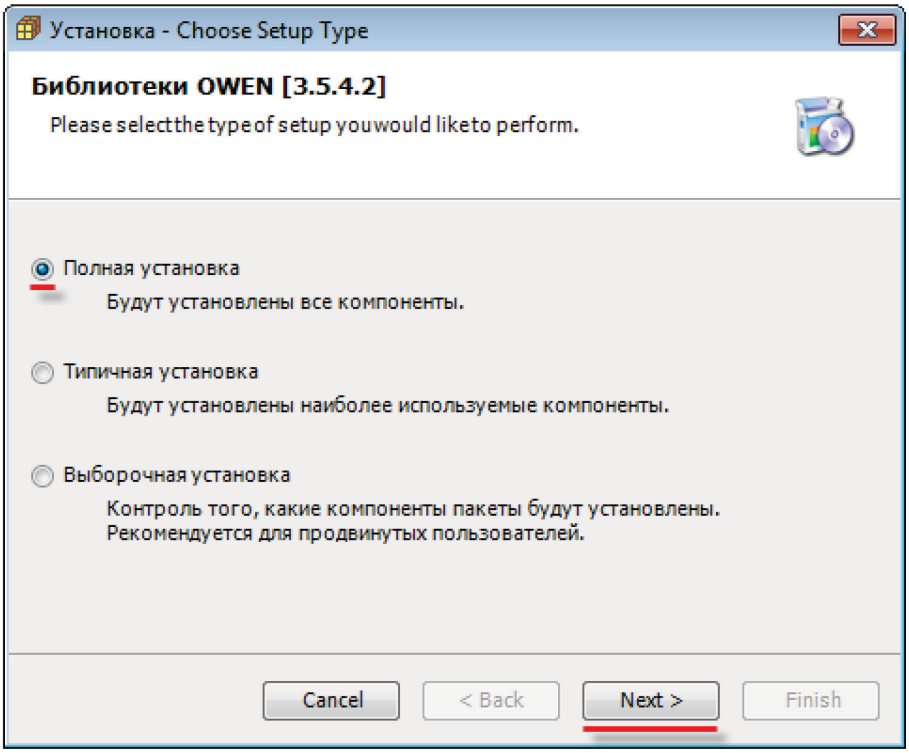


Рис. 6.2. Начало установки пакета библиотек

После завершения установки закройте диалоговое окно с помощью кнопки **Finish**:

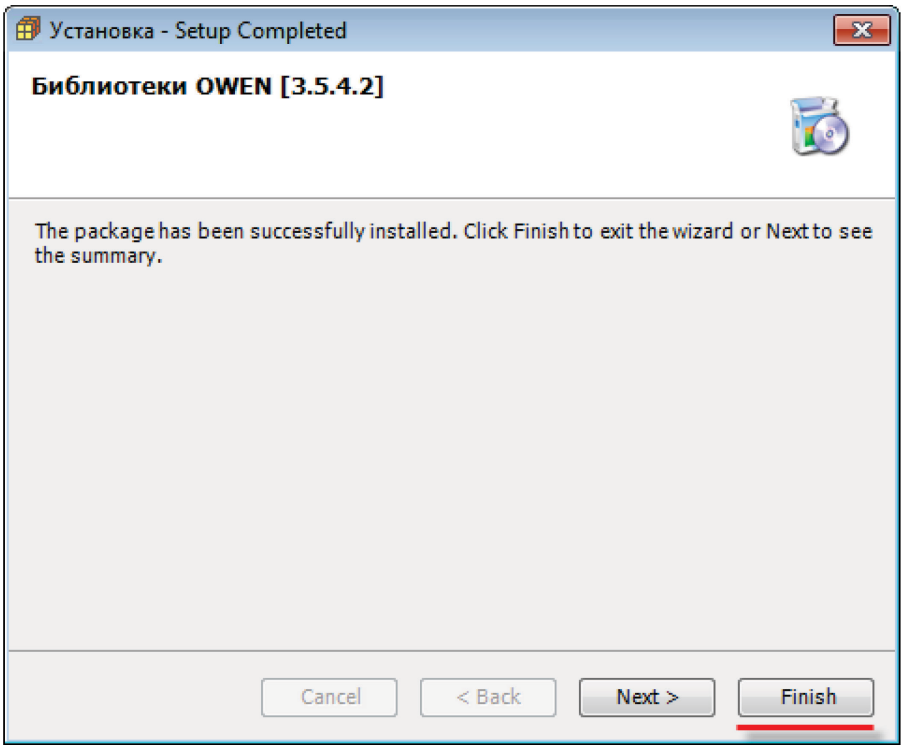


Рис. 6.3. Завершение установки пакета библиотек

6.2. Добавление библиотеки в проект CODESYS

Для добавления библиотеки **Modbus** в проект **CODESYS**, в **Менеджере библиотек** нажмите кнопку **Добавить библиотеку** и в строке поиска введите **modbus**, после чего выберите из списка нужную библиотеку и нажмите **ОК**.

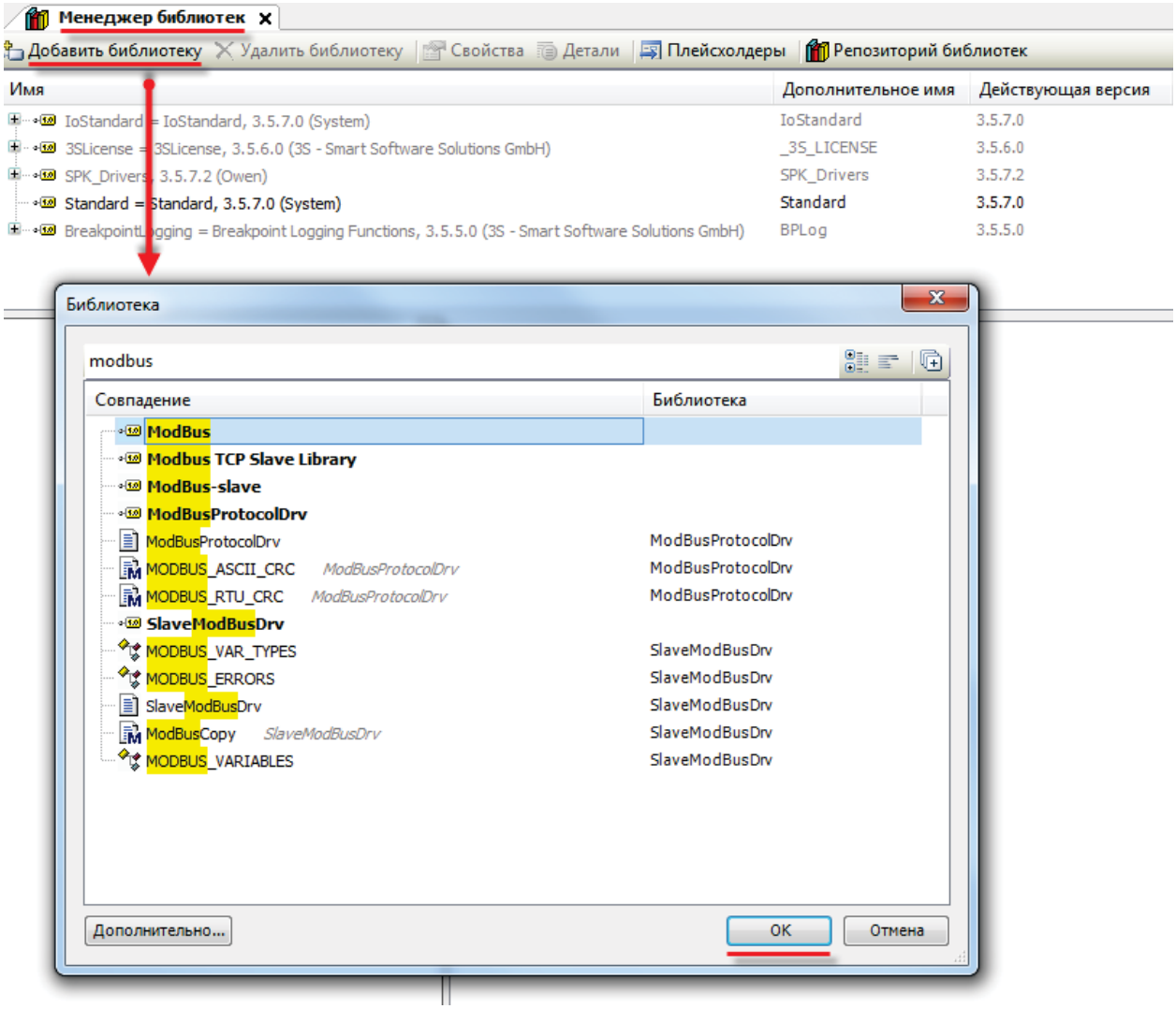


Рис. 6.4. Добавление библиотеки **Modbus**

В библиотеку **Modbus** не входит ФБ открытия COM-порта, поэтому для этой цели необходимо воспользоваться другой библиотекой. Рекомендуется использовать библиотеку **ComService**.

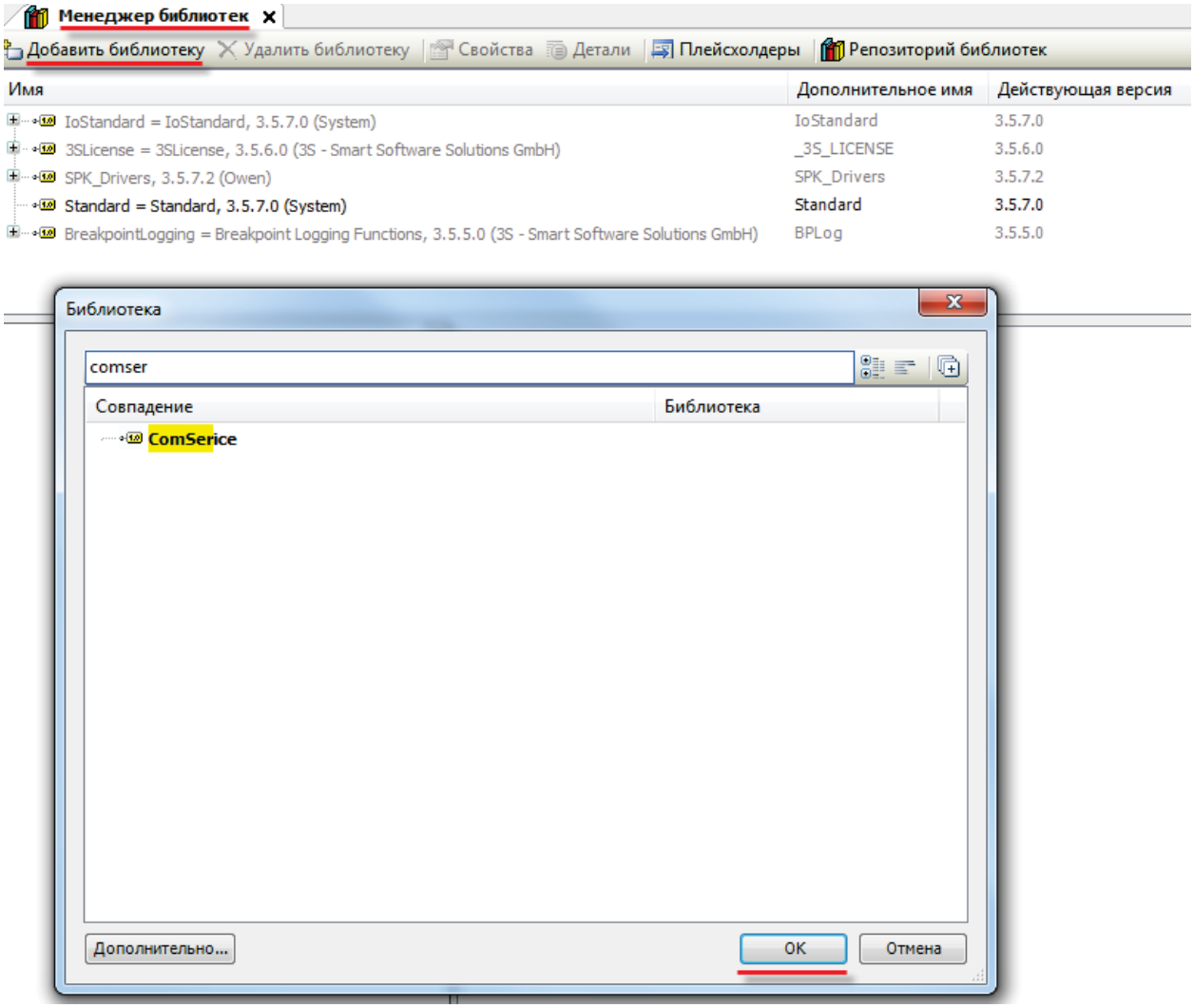


Рис. 6.5. Добавление библиотеки **ComService**

После добавления библиотеки появятся в списке **Менеджера библиотек**:

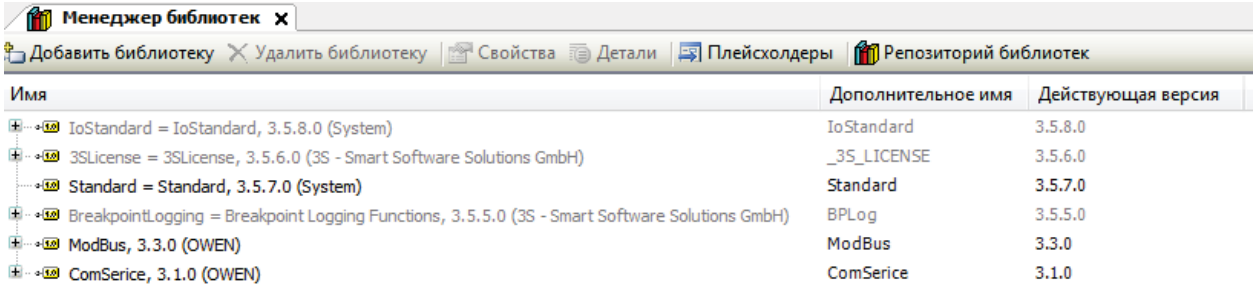


Рис. 6.6. Список библиотек проекта

6.3. Описание ФБ библиотеки

6.3.1. Блок ComService (библиотека ComService)

Функциональный блок **ComService**, входящий в библиотеку **ComService**, используется для открытия/закрытия COM-порта, а также его настройки.

Имя переменной	Тип	Описание
Входные переменные		
Enable	BOOL	Переменная действия с портом. По ее переднему фронту с COM-портом номер **sPort** совершается действие, определяемое переменной **Task**.
Settings	SysCom.Com_Settings	Структура настроек COM-порта.
sPort	SysCom.COM_Ports	Номер порта СПК (см. п. 2.3.1).
byStopBits	SysCom.COM_StopBits	Количество стоп-бит. Возможные значения: **1** – 1 бит; **2** – 1.5 бит; **3** – 2 бит.
byParity	SysCom.COM_Parity	Режим контроля паритета. Возможные значения: **0** – отсутствие контроля паритета (**NONE**); **1** – четный (**EVEN**); **2** – нечетный (**ODD**).
uiBaudrate	SysCom.COM_Baudrate	Скорость обмена, бод. Возможные значения: **4800/9600/19200/38400/57600/115200**.
uiTimeout	SysCom.COM_Timeout	Время ожидания следующего символа. В СПК параметр не используется.
uiBufferSize	UDINT	Размер FIFO буфера COM-порта. В СПК параметр не используется.
sets_ex	SysCom.Com_SettingsEx	Структура расширенных настроек COM-порта. Описание всех переменных доступно в документации библиотеки **SysCom**.
byByteSize	BYTE	Количество информационных бит в передаваемых/принимаемых байтах. Обычно выбирается значение **8** для **Modbus RTU** и **7** для **Modbus ASCII**.
Task	COM_TSK	Действие, совершаемое с портом. Возможные значения: **OPEN_TSK** – открытие порта; **RESET_TSK** – перезапуск порта; **CLOSE_TSK** – закрытие порта.
Выходные переменные		
Ready	BOOL	Переменная состояния порта. Принимает значение **TRUE**, если порт открыт. Принимает значение **FALSE**, если порт закрыт.
Handle	SysCom.RTS_IEC_HANDLE	Идентификатор открытого порта, используется для обращения к ФБ опроса модулей.

6.3.2. Блоки функций Modbus

Каждая из функций Modbus представлена в библиотеке в виде функционального блока. В табл. 6.1 приведено соответствие функций и названий ФБ. Входы/Выходы ФБ описаны ниже.

Табл. 6.1. Соответствие функций **Modbus** и названий ФБ библиотеки **Modbus**

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

Описание входов/выходов ФБ **MB_RD_COILS**, **MB_RD_INPUTS**, **MB_RD_INP_REG**, **MB_RD_HOLD_REGS**

Имя переменной	Тип	Описание
Входные переменные		
Enable	BOOL	Переменная опроса модуля. Опрос происходит по переднему фронту переменной.
Mode	MB_MODE	Режим работы протокола. Возможные значения: **MB_RTU** – Modbus RTU; **MB_ASCII** – Modbus ASCII.
DevAddr	BYTE	Адрес slave-устройства.
FirstAddr	WORD	Адрес первого бита/регистра, считываемого со slave-устройства.
Quantity	WORD (для битов) BYTE (для регистров)	Количество бит/регистров, считываемых со slave-устройства.
ComHandle	SysCom.RTS_IEC_HANDLE	Идентификатор порта, поступающий с выхода блока ComService (или аналогичного).
TimeOut	TIME	Таймаут ответа slave-устройства. Если в течение этого времени устройство не отвечает, то СПК переходит к опросу следующего slave-устройства.
Buffer	ARRAY [0..255] OF BYTE	Буфер ФБ. После поднятия флага **Complete** из него можно забрать считанные данные.
Выходные переменные		
Complete	BOOL	Флаг опроса slave-устройства. Принимает значения **TRUE** после завершения опроса, на следующем цикле сбрасывается в **FALSE**.
Exception	BYTE	Код ошибки. Значение **0** соответствует отсутствию ошибок. Код ошибки **16#FFFF (255)** характеризует отсутствие ответа от слэйва по истечению таймаута опроса. Список кодов остальных ошибок приведен в приложении В, табл. В2.
ByteCnt	BYTE	Размер считанных данных в байтах.

Описание входов/выходов ФБ **MB_WR_SNG_COIL**, **MB_WR_SNG_REG**

Имя переменной	Тип	Описание
Входные переменные		
Enable	BOOL	Переменная опроса модуля. Опрос происходит по переднему фронту переменной.
Mode	MB_MODE	Режим работы протокола. Возможные значения: **MB_RTU** – Modbus RTU; **MB_ASCII** – Modbus ASCII.
DevAddr	BYTE	Адрес slave-устройства.
CoilAddr/RedAddr	WORD	Адрес бита/регистра, записываемого в slave-устройство.
Value	BOOL(для битов) WORD (для регистров)	Значение, записываемое в бит/регистр slave-устройства.
ComHandle	SysCom.RTS_IEC_HANDLE	Идентификатор порта, поступающий с выхода блока ComService (или аналогичного).
TimeOut	TIME	Таймаут ответа slave-устройства. Если в течение этого времени устройство не отвечает, то СПК переходит к опросу следующего slave-устройства.
Выходные переменные		
Complete	BOOL	Флаг опроса slave-устройства. Принимает значения **TRUE** после завершения опроса, на следующем цикле сбрасывается в **FALSE**.
Exception	BYTE	Код ошибки. Значение **0** соответствует отсутствию ошибок. Код ошибки **16#FFFF (255)** характеризует отсутствие ответа от слэйва по истечению таймаута опроса. Список кодов остальных ошибок приведен в приложении В, табл. В2.

Описание входов/выходов ФБ **MB_WR_COILS**, **MB_WR_REGS**

Имя переменной	Тип	Описание
Входные переменные		
Enable	BOOL	Переменная опроса модуля. Опрос происходит по переднему фронту переменной.
Mode	MB_MODE	Режим работы протокола. Возможные значения: **MB_RTU** – Modbus RTU; **MB_ASCII** – Modbus ASCII.
DevAddr	BYTE	Адрес slave-устройства.
FirstAddr	WORD	Адрес первого бита/регистра, записываемого в slave-устройство.
Quantity	WORD (для битов) BYTE (для регистров)	Количество битов/регистров, записываемых в slave-устройство.
ComHandle	SysCom.RTS_IEC_HANDLE	Идентификатор порта, поступающий с выхода блока ComService (или аналогичного).
TimeOut	TIME	Таймаут ответа slave-устройства. Если в течение этого времени устройство не отвечает, то СПК переходит к опросу следующего slave-устройства.
Buffer	ARRAY [0..255] OF BYTE	Массив данных, записываемых в slave-устройство.
Выходные переменные		
Complete	BOOL	Флаг опроса slave-устройства. Принимает значения **TRUE** после завершения опроса, на следующем цикле сбрасывается в **FALSE.**
Exception	BYTE	Код ошибки. Значение **0** соответствует отсутствию ошибок. Код ошибки **16#FFFF (255)** характеризует отсутствие ответа от слэйва по истечению таймаута опроса. Список кодов остальных ошибок приведен в приложении В, табл. В2.
CoilCnt/RegCnt	WORD/BYTE	Количество записанных бит/регистров.

6.4. Пример: СПК207 + модули Mx110 (MB110-8A, MB110-16Д, МУ110-16Р)

Рассмотрим пример настройки обмена с **модулями Mx110** с использованием **библиотеки Modbus**. В нем мы наладим связь между контроллером **СПК207.03** (master) и тремя модулями (slave-устройствами).

Реализуемый алгоритм: если значение 1-го аналогового входа модуля **MB110-8A** превышает 30 и при этом 1-ый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то произвести запись в 1-й дискретный выход модуля **МУ110-16Р** значения **TRUE** (замкнут). Во всех остальных случаях присвоить дискретному выходу значение **FALSE** (разомкнут).

Структурная схема примера приведена на рис. 6.7:

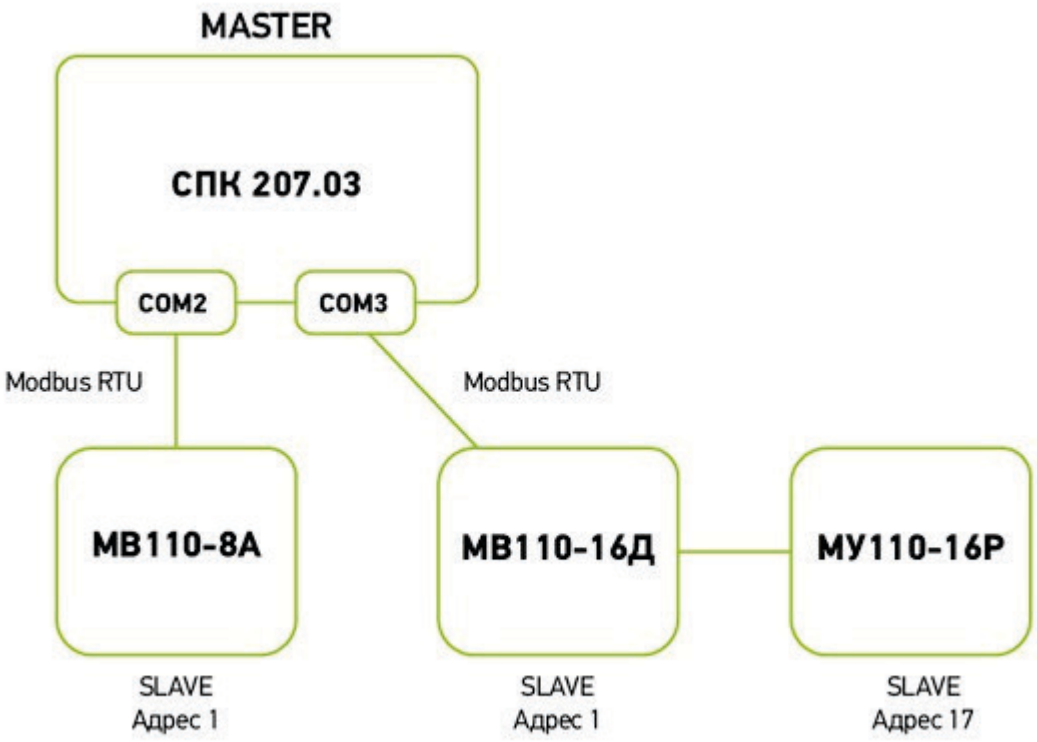


Рис. 6.7. Структурная схема примера **Библиотека Modbus (СПК+модули Mx110)**

Пример создан в среде **CODESYS 3.5 SP7 Patch4** и подразумевает запуск на **СПК207.03.CS(-WEB)**.

Пример доступен для скачивания: Example_ModbusMasterLibrary.projectarchive

Сетевые параметры модулей приведены в табл. 6.2.

Табл. 6.2. Сетевые параметры модулей **Mx110**

Параметр	**MB110-8A**	**MB110-16Д**	**МУ110-16Р**
COM-порт СПК, к которому подключен модуль	*COM2*	*COM3*	
Адрес модуля	1	1	17
Скорость обмена	115200		
Количества бит данных	8		
Контроль четности	отсутствует		
Количество стоп-бит	1		

Пример содержит два проекта – **Device_CFC** и **Device_ST**. Каждый проект содержит библиотеку **Modbus**, в каждом из них реализован описанный алгоритм на соответствующем языке программирования.

Каждый проект содержит три программы:

- **COM2** (реализация обмена через COM-порт COM2 с модулем **MB110-8A**);

- **COM3** (реализация обмена через COM-порт COM3 с модулями **MB110-16Д** и **МУ110-16Р**);

- **PLC_PRG** (основная программа, реализующая пользовательский алгоритм).

Каждая программа привязана к отдельной задаче, время цикла каждой из задач – **t#20ms**.

6.4.1. Описание реализации на языке CFC

1. Переменные объединения **Word_Real** (используется для представления считанного с модуля **MB110-8A** значения в формате **REAL**):

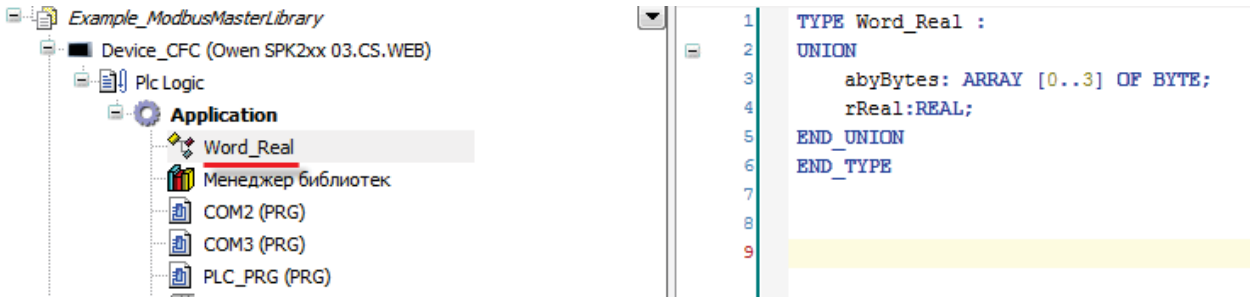


Рис. 6.8. Переменные объединения **Word_Real**

2. Переменные программы **COM2**:

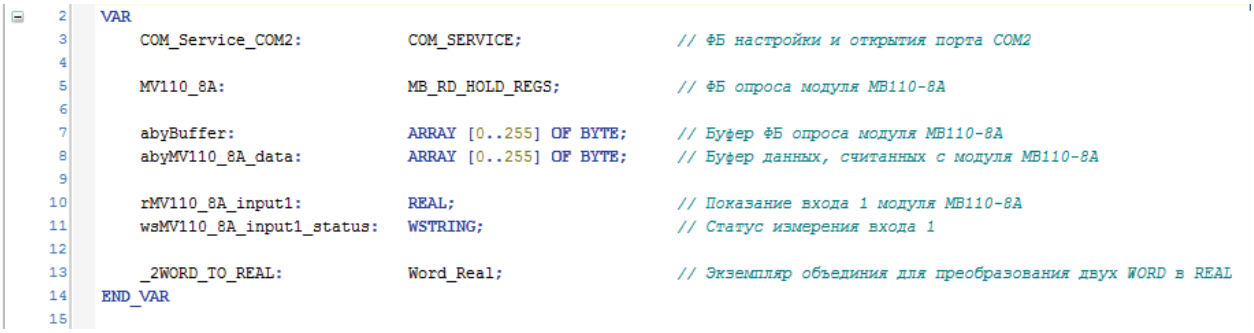


Рис. 6.9. Объявление переменных программы **COM2 (CFC)**

Код программы **COM2** на языке **CFC** (*рисунок хорошо масштабируется*):

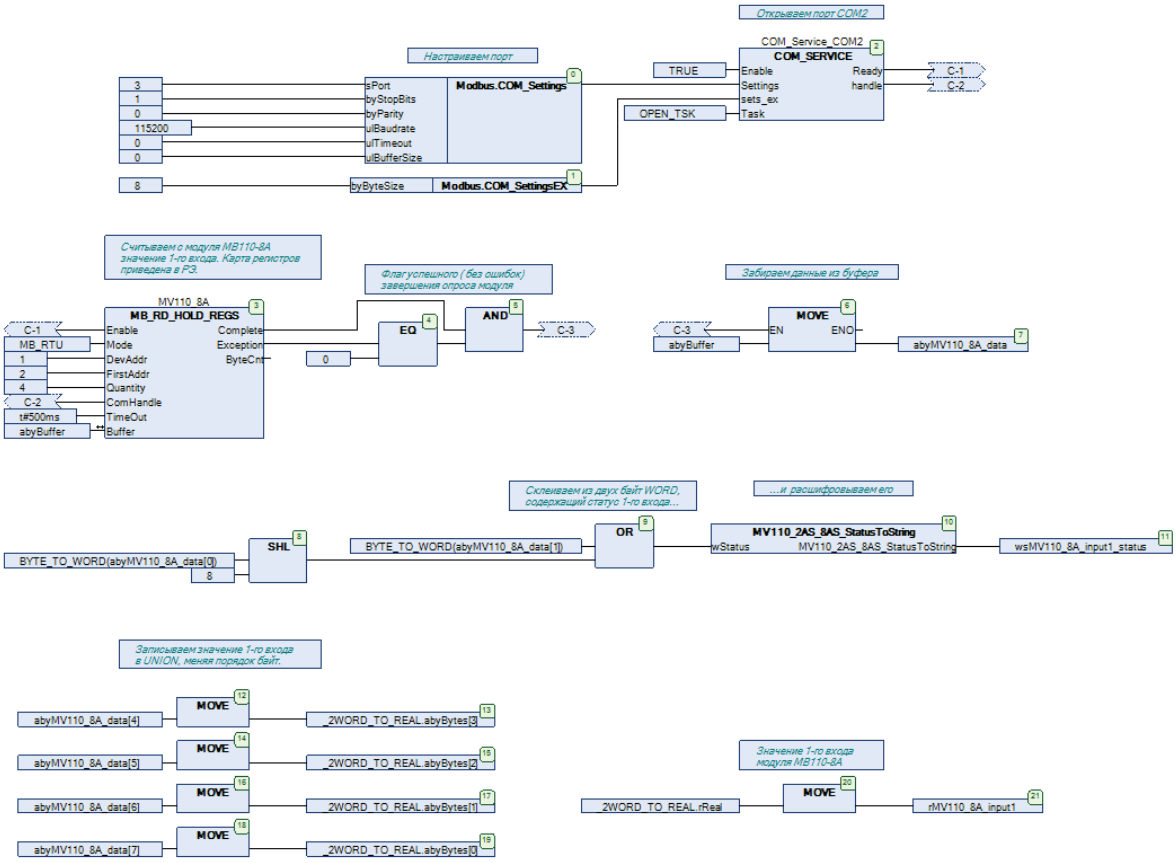


Рис. 6.10. Код программы **COM2 (CFC)**

Программа работает следующим образом:

Блоки 0-1. Настройка COM-порта в соответствии с табл. 6.2. ***Обратите внимание***, что COM-порту **COM2** соответствует номер **3** (см. п. 2.3.1).

Блок 2. При первом запуске программы с помощью ФБ ComService открывается COM-порт **COM2** с сетевыми настройками, заданными в блоках 0-1.

Блок 3. Инициализация цикла опроса модуля **MB110-8A** с помощью ФБ MB_RD_HOLD_REGS. Опрос будет начат только в том случае, если ФБ ComService завершил работу (выход **Ready**=**TRUE**). На вход **ComHandle** поступает значение выхода **Handle** ФБ ComService. На вход **DevAddr** поступает адрес модуля – согласно табл. 6.2, данный модуль имеет адрес **1**. На входе **FirstAddr** указан первый опрашиваемый регистр модуля (**2**), на входе **Quantity** – число считываемых регистров (**4**). Таким образом, будут считаны регистры 2-5. В регистре 2 хранится статус измерения 1-го входа модуля, в регистрах 4-5 – значение 1-го входа модуля в представлении с плавающей точкой. Карта регистров и поддерживаемые функции Modbus приведены в РЭ на модуль.

Блоки 4-6. Если цикл опроса модуля завершен (об этом сигнализирует импульс по переднему фронту на выходе **Complete**) без ошибок (**Exception=0**), то формируется флаг завершения опроса модуля (**C-3**) и считанные данные из буфера ФБ копируются в пользовательский буфер (массив **abyMV110_8A_data**). ***Обратите внимание***, что обращаться к буферу ФБ можно только по флагу **Complete** и крайне рекомендуется предварительно копировать значения из буфера, после чего работать с ними в программе.

Блоки 8-11. Байты 0-1 пользовательского буфера содержат статус измерения 1-го входа модуля. Байты меняются местами (это связано с различием представления данных в модуле и СПК), после чего код статуса декодируется с помощью функции **MV110_2A_8A_StatusToString** (включена в target-файл СПК) и присваивается переменной **wsMV110_8A_input1_status**.

Блоки 12-19. Байты 4-7 пользовательского буфера содержат значение 1-го входа модуля в представлении с плавающей точкой. Порядок байт меняется (это связано с различием представления данных в модуле и СПК), после чего они записываются в байты объединения **Word_Real**.

Блоки 20-21. Значение 1-го входа модуля в представлении с плавающей точкой копируется из переменной объединения в переменную **rMV110_8A_input1**.

Обратите внимание, что **цикл опроса модуля** происходит в течение нескольких **циклов программы**.

3. Переменные программы **COM3**:

```
PROGRAM COM3
VAR
    COM_SERVICE_COM3:       COM_SERVICE;            // ФБ настройки и открытия порта COM3

    MV110_16D:              MB_RD_HOLD_REGS;        // ФБ опроса модуля МВ110-16Д
    MY110_16R:              MB_WR_REGS;             // ФБ опроса модуля МУ110-16Р

    abyMV110_16D_buffer:    ARRAY [0..255] OF BYTE; // Буфер ФБ опроса модуля МВ110-16Д
    abyMV110_16D_data:      ARRAY [0..255] OF BYTE; // Буфер данных, считанных с модуля МВ110-16Д

    xMV110_16D_input1:      BOOL;                   // Состояние 1-го входа модуля МВ110-16Д
    abyMY110_16R_buffer:    ARRAY [0..255] OF BYTE; // Буфер ФБ опроса модуля МУ110-16Р

    xModule1, xModule2:     BOOL;                   // Переменные завершения цикла опроса (1 - МВ110-16Д, 2 - МУ110-16Р)
END_VAR

VAR_INPUT
    xMY110_16R_output1:     BOOL;                   // Переменная для записи состояния 1-го выхода модуля МУ110-16Р
END_VAR
```

Рис. 6.11. Объявление переменных программы **COM3 (CFC)**

Код программы **COM3** на языке **CFC** (*рисунок хорошо масштабируется*):

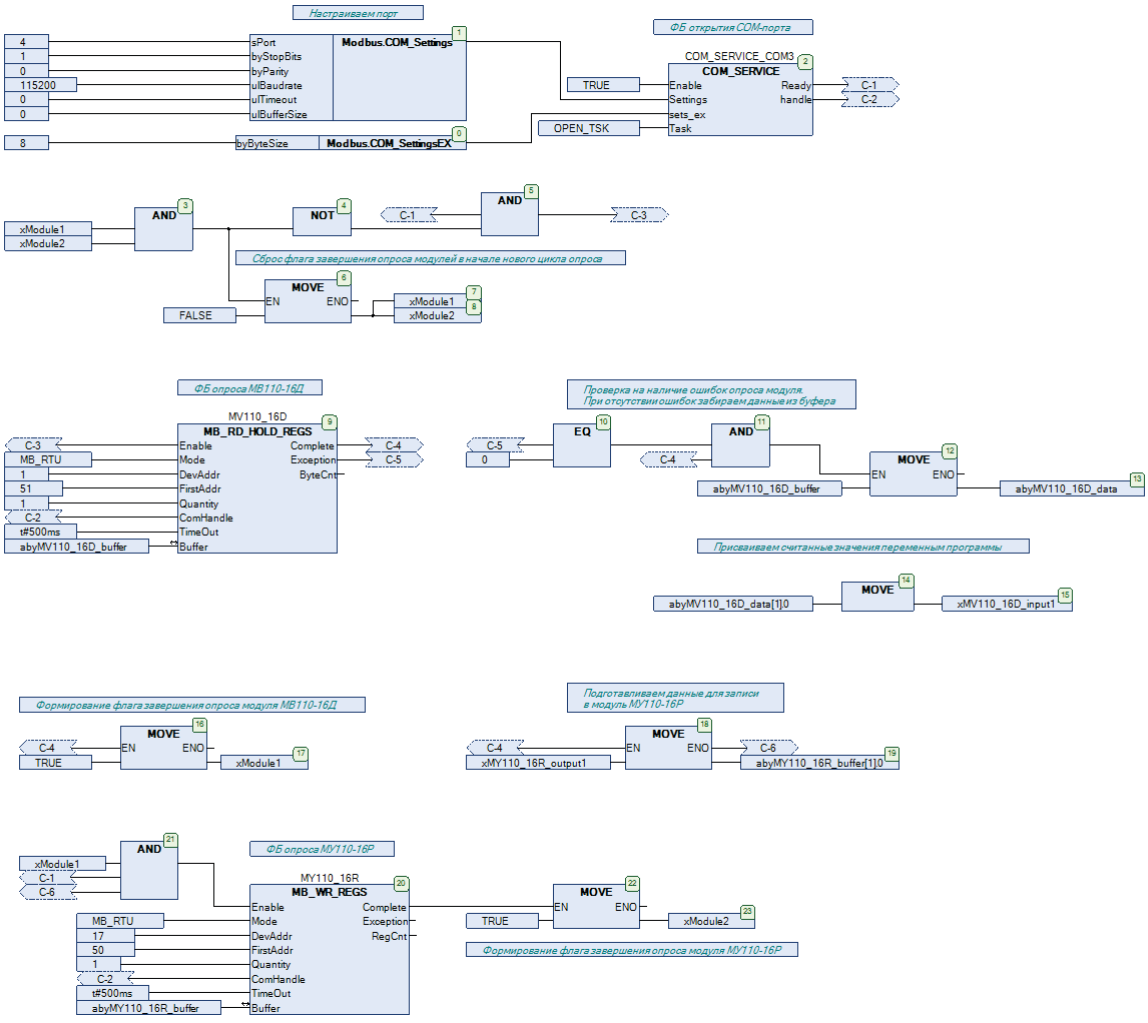


Рис. 6.12. Код программы **COM3 (CFC)**

Программа работает следующим образом:

Блоки 0-1. Настройка COM-порта в соответствии с табл. 6.2. ***Обратите внимание***, что COM-порту **COM3** соответствует номер **4** (см. п. 2.3.1).

Блок 2. При первом запуске программы с помощью ФБ ComService открывается COM-порт **COM3** с сетевыми настройками, заданными в блоках 0-1.

Блоки 3-5. Формируется сигнал инициализации опроса модуля **MB110-16Д**. Опрос будет начат только в том случае, если:

- ФБ ComService завершил работу (**Ready=TRUE**);

- предыдущий цикл опроса завершен. ***Обратите внимание***, что флаги завершения предыдущего цикла опроса инвертируются. Это необходимо, так как опрос модулей начинается по переднему фронту входа **Enable** – соответственно, перед началом нового цикла опроса они должны принять значение **FALSE**.

Блоки 6-8. Сброс флага завершения предыдущего цикла опроса.

Блок 9. Инициализация цикла опроса модуля **MB110-16Д** с помощью ФБ MB_RD_HOLD_REGS. На вход **ComHandle** данного ФБ поступает значение выхода **Handle** ФБ ComService. На вход **DevAddr** поступает адрес модуля – согласно табл. 6.2, данный модуль имеет адрес **1**. На входе **FirstAddr** указан первый опрашиваемый регистр модуля (**51**), на входе **Quantity** – число считываемых регистров (**1**). Таким образом, будет считан регистр 51, в котором хранится битовая маска состояний входов модуля. Карта регистров и поддерживаемые функции Modbus приведены в РЭ на модуль.

Блоки 10-13. Если цикл опроса модуля **MB110-16Д** завершен (об этом сигнализирует импульс по переднему фронту на выходе **Complete**) без ошибок (**Exception=0**), то формируется флаг завершения опроса модуля (**C-4**) и считанные данные из буфера ФБ копируются в пользовательский буфер (массив **abyMV110_16D_data**). ***Обратите внимание***, что обращаться к буферу ФБ можно только по флагу **Complete** и крайне рекомендуется предварительно копировать значения из буфера, после чего работать с ними в программе.

Блоки 14-15. Значение 1-го входа модуля копируется из пользовательского буфера в переменную **xMV110_16D_input1**.

Блоки 16-17. Формируется флаг окончания данного цикла опроса модуля **MB110-16Д**.

Блоки 18-19. Значение переменной **xMY110_16R_output1** записывается в буфер ФБ MB_WR_REGS, который будет отправлен в модуль **МУ110-16Р**. ***Обратите внимание***, что значение записывается в первый, а не нулевой байт буфера – это связано с различием представления данных в модуле и СПК. В нулевом байте буфера содержаться значения, которые будут записаны на 9-16 выходы модуля.

Блок 20. Формируется сигнал инициализации опроса модуля **МУ110-16Р**. Опрос будет начат только в том случае, если:

- ФБ ComService завершил работу (**Ready=TRUE**) без ошибок;

- текущий цикл опроса модуля **МВ110-16Д** завершен;

- значение, которое должно быть записано в модуль, помещено в буфер ФБ.

Блок 21. Инициализация цикла опроса модуля **МУ110-16Р** с помощью ФБ MB_WR_REGS. На вход **ComHandle** данного ФБ поступает значения выхода **Handle** ФБ ComService. На вход **DevAddr** поступает адрес модуля – согласно табл. 6.2, данный модуль имеет адрес **17**. На входе **FirstAddr** указан первый записываемый регистр модуля (**50**), на входе **Quantity** – число записываемых регистров (**1**). Таким образом, будет записан регистр 50, в котором хранится битовая маска состояний выходов модуля. Карта регистров и поддерживаемые функции Modbus приведены в РЭ на модуль.

Блоки 22-23. Формируется флаг окончания данного цикла опроса модуля **МУ110-16Р**.

Обратите внимание, что **цикл опроса модуля** происходит в течение нескольких **циклов программы**.

При необходимости количество опрашиваемых модулей можно увеличить – для этого, соответственно, потребуется увеличить число переменных **xModule№. *Обратите внимание***, что приступать к опросу следующего модуля можно только после завершения опроса предыдущего (т.е. после того, когда переменная **xModule№** предыдущего модуля принимает значение **TRUE)**.

4. Код программы **PLC_PRG**:

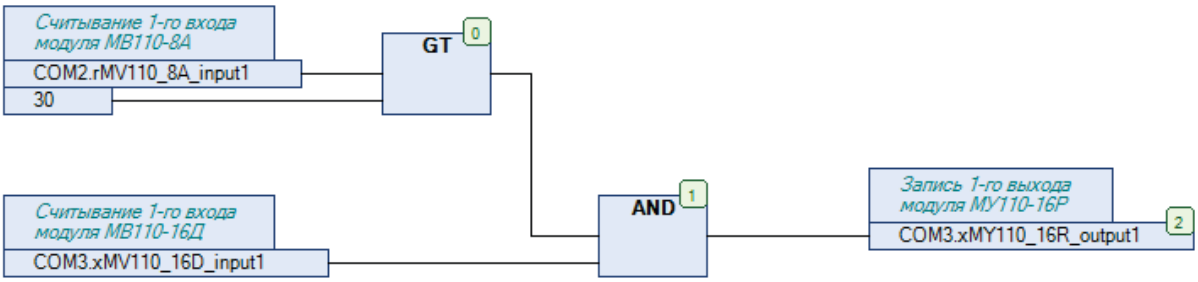


Рис. 6.13. Объявление переменных программы **PLC_PRG (CFC)**

Программа **PLC_PRG** оперирует переменными программ **COM2** и **COM3**, и реализует следующий алгоритм: если значение 1-го аналогового входа модуля **MB110-8A** превышает 30 и при этом 1-ый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то произвести запись в 1-й дискретный выход модуля **MУ110-16P** значения **TRUE** (замкнут). Во всех остальных случаях присвоить дискретному выходу значение **FALSE** (разомкнут).

6.4.2. Описание реализации на языке ST

1. Переменные объединения **Word_Real** (используется для представления считанного с модуля **MB110-8A** значения в формате **REAL**):

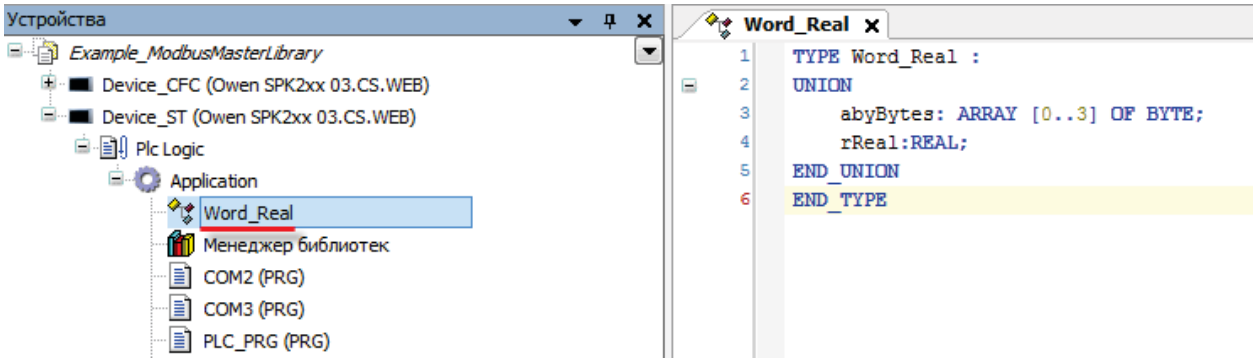


Рис. 6.14. Переменные объединения **Word_Real**

2. Переменные программы **COM2**:

```
PROGRAM COM2
VAR
    COM_Service_COM2:          COM_SERVICE;              // ФБ настройки и открытия порта COM2
    Settings_COM2:             COM_SETTINGS;             // Структура настроек порта COM2
    SettingsEX_COM2:           COM_SETTINGSex;           // Структура расширенных настроек порта COM2

    MV110_8A:                  MB_RD_HOLD_REGS;          // ФБ опроса модуля MB110-8A

    rMV110_8A_input1:          REAL;                     // Показание входа 1 модуля MB110-8A
    wMV110_8A_input1_status:   WORD;                     // Код ошибки измерения входа 1
    wsMV110_8A_input1_status:  WSTRING;                  // Статус измерения входа 1

    abyBuffer:                 ARRAY [0..255] OF BYTE;   // Буфер ФБ опроса модуля MB110-8A
    abyMV110_8A_data:          ARRAY [0..255] OF BYTE;   // Буфер данных, считанных с модуля MB110-8A

    _2WORD_TO_REAL:            Word_Real;                // Экземпляр объединия для преобразования двух WORD в REAL
END_VAR
```

Рис. 6.15. Объявление переменных программы **COM2 (ST)**

Код программы **COM2** на языке **ST** (*рисунок хорошо масштабируется, листинг приведен в Приложении Д.3*):

```
// [1] настраиваем COM-порт
Settings_COM2.sPort:=3;
Settings_COM2.byStopBits:=1;
Settings_COM2.byParity:=0;
Settings_COM2.ulBaudrate:=115200;
Settings_COM2.ulTimeout:=0;
Settings_COM2.ulBufferSize:=0;

SettingsEX_COM2.byByteSize:=8;


// [2] открываем COM-порт
COM_Service_COM2
(
    Enable:=TRUE,
    Settings:=Settings_COM2,
    Sets_Ex:=SettingsEX_COM2,
    Task:=OPEN_TSK,
);


// [3] запускаем ФБ опроса модуля MB110-8A
MV110_8A
(
    Enable:=COM_Service_COM2.Ready,
    Mode:=MB_RTU,
    DevAddr:=1,
    FirstAddr:=2,
    Quantity:=4,
    ComHandle:=COM_Service_COM2.handle,
    TimeOut:=T#500MS,
    Buffer:=abyBuffer,
);


// [4] если ФБ опроса модуля завершил работу...
IF MV110_8A.Complete THEN
    // ...и ошибки отсутствуют, то забираем буфер модуля
    IF MV110_8A.Exception=0 THEN
        abyMV110_8A_data:=abyBuffer;

        // байты 0 и 1 содержат статус измерения входа 1
        // склеиваем их в WORD и декодируем в текстовое сообщение
        wMV110_8A_input1_status:=BYTE_TO_WORD(abyMV110_8A_data[1]) OR SHL(BYTE_TO_WORD(abyMV110_8A_data[0]),8);
        wsMV110_8A_input1_status:=MV110_2AS_8AS_StatusToString(wMV110_8A_input1_status);

        // байты 4-7 содержат значение на входе 1 с плавающей точкой.
        // переставляем байты местами и преобразуем в REAL
        _2WORD_TO_REAL.abyBytes[3]:=abyMV110_8A_data[4];
        _2WORD_TO_REAL.abyBytes[2]:=abyMV110_8A_data[5];
        _2WORD_TO_REAL.abyBytes[1]:=abyMV110_8A_data[6];
        _2WORD_TO_REAL.abyBytes[0]:=abyMV110_8A_data[7];

        rMV110_8A_input1:=_2WORD_TO_REAL.rReal;
    END_IF

    // завершаем опрос модуля MB110-8A
    MV110_8A(Enable:=FALSE, Buffer:=abyBuffer);

END_IF
```

Рис. 6.16. Объявление переменных программы **COM2 (ST)**

Программа работает следующим образом:

Блок [1]. Настройка COM-порта в соответствии с табл. 6.2. ***Обратите внимание***, что COM-порту **COM2** соответствует номер **3** (см. п. 2.3.1).

Блок [2]. При первом запуске программы с помощью ФБ ComService открывается COM-порт **COM2** с сетевыми настройками, заданными в блоках 0-1.

Блок [3]. Инициализация цикла опроса модуля **MB110-8A** с помощью ФБ MB_RD_HOLD_REGS. Опрос будет начат только в том случае, если ФБ ComService завершил работу (выход **Ready=TRUE**). На вход **ComHandle** поступает значение выхода **Handle** ФБ ComService. На вход **DevAddr** поступает адрес модуля – согласно табл. 6.2, данный модуль имеет адрес **1**. На входе **FirstAddr** указан первый опрашиваемый регистр модуля (**2**), на входе **Quantity** – число считываемых регистров (**4**). Таким образом, будут считаны регистры 2-5. В регистре 2 хранится статус измерения 1-го входа модуля, в регистрах 4-5 – значение 1-го входа модуля в представлении с плавающей точкой. Карта регистров и поддерживаемые функции Modbus приведены в РЭ на модуль.

Блоки [4]. Если цикл опроса модуля завершен (об этом сигнализирует импульс по переднему фронту на выходе **Complete**) без ошибок (**Exception=0**), то считанные данные из буфера ФБ копируются в пользовательский буфер (массив **abyMV110_8A_data**). ***Обратите внимание***, что обращаться к буферу ФБ можно только по флагу **Complete** и крайне рекомендуется предварительно копировать значения из буфера, после чего работать с ними в программе.

Байты 0-1 пользовательского буфера содержат статус измерения 1-го входа модуля. Байты меняются местами (это связано с различием представления данных в модуле и СПК), после чего код статуса декодируется помощью функции **MV110_2A_8A_StatusToString** (включена в target-файл СПК) и присваивается переменной **wsMV110_8A_input1_status**.

Байты 4-7 пользовательского буфера содержат значение 1-го входа модуля в представлении с плавающей точкой. Порядок байт меняется (это связано с различием представления данных в модуле и СПК), после чего они записываются в байты объединения **Word_Real**. Значение 1-го входа модуля в представлении с плавающей точкой копируется из переменной объединения в переменную **rMV110_8A_input1**.

Обратите внимание, что **цикл опроса модуля** происходит в течение нескольких **циклов программы**.

3. Переменные программы **COM3**:

```
PROGRAM COM3
VAR
    COM_SERVICE_COM3:       COM_SERVICE;            // ФБ настройки и открытия порта COM3
    Settings_COM3:          COM_SETTINGS;           // Структура настроек порта COM3
    SettingsEX_COM3:        COM_SETTINGSex;         // Структура расширенных настроек порта COM3

    xModule:INT;                                    // Переменные начала цикла опроса (0 - MB110-16Д, 1 - MY110-16P)

    MV110_16D:              MB_RD_HOLD_REGS;        // ФБ опроса модуля MB110-16Д
    MY110_16R:              MB_WR_REGS;             // ФБ опроса модуля MY110-16P

    abyMV110_16D_buffer:    ARRAY [0..255] OF BYTE; // Буфер ФБ опроса модуля MB110-16Д
    abyMV110_16D_data:      ARRAY [0..255] OF BYTE; // Буфер данных, считанных с модуля MB110-16Д

    xMV110_16D_input1:      BOOL;                   // Состояние 1-го входа модуля MB110-16Д
    abyMY110_16R_buffer:    ARRAY [0..255] OF BYTE; // Буфер ФБ опроса модуля MY110-16P
END_VAR

VAR_INPUT
    xMY110_16R_output1:     BOOL;           // Переменная для записи состояния 1-го выхода модуля MB110-16P
END_VAR
```

Рис. 6.17. Объявление переменных программы **COM3 (ST)**

Код программы **COM3** на языке **ST** (*рисунок хорошо масштабируется, листинг приведен в Приложении Д.4*):

```
// [1] настраиваем COM-порт
Settings_COM3.sPort:=4;
Settings_COM3.byStopBits:=1;
Settings_COM3.byParity:=0;
Settings_COM3.ulBaudrate:=115200;
Settings_COM3.ulTimeout:=0;
Settings_COM3.ulBufferSize:=0;

SettingsEX_COM3.byByteSize:=8;


// [2] открываем COM-порт
COM_Service_COM3
(
    Enable:=TRUE,
    Settings:=Settings_COM3,
    Sets_Ex:=SettingsEX_COM3,
    Task:=OPEN_TSK,
);

// [3] xModule определяет опрашиваемый модуль: 0 - MB110-16Д, 1 - MY110-16P
CASE xModule OF

    0:

        // [3.0.1] запускаем ФБ опроса модуля MB110-16Д
        MV110_16D
        (
            Enable:=COM_Service_COM3.Ready,
            Mode:=MB_RTU,
            DevAddr:=1,
            FirstAddr:=51,
            Quantity:=1,
            ComHandle:=COM_Service_COM3.handle,
            TimeOut:=T#500MS,
            Buffer:=abyMV110_16D_buffer,
        );

        // [3.0.2] если ФБ опроса модуля завершил работу...
        IF MV110_16D.Complete THEN
            // ...и ошибки отсутствуют, то забираем значения модуля
            IF MV110_16D.Exception=0 THEN
                abyMV110_16D_data:=abyMV110_16D_buffer;

                xMV110_16D_input1:=abyMV110_16D_data[1].0;
            END_IF

            // завершаем опрос модуля MB110-16Д
            MV110_16D(Enable:=FALSE, Buffer:=abyMV110_16D_buffer);

            // переходим к опросу модуля MY110-16P
            xModule:=1;


        END_IF

    1:

        // [3.1.1] копируем записываемое значение в буфер ФБ

        abyMY110_16R_buffer[1].0:=xMY110_16R_output1;

        // [3.1.2] запускаем ФБ опроса модуля MY110-16P
        MY110_16R
        (
            Enable:=COM_SERVICE_COM3.Ready,
            Mode:=MB_RTU,
            DevAddr:=17,
            FirstAddr:=50,
            Quantity:=1,
            ComHandle:=COM_SERVICE_COM3.handle,
            TimeOut:=T#500MS,
            Buffer:=abyMY110_16R_buffer,
        );

        // [3.1.3] если ФБ опроса модуля завершил работу...
        IF MY110_16R.Complete THEN

            // ...то завершаем опрос модуля MY110-16P...
            MY110_16R(Enable:=FALSE, Buffer:=abyMY110_16R_buffer);

            // ...и начинаем новый цикл опроса
            xModule:=0;
        END_IF
END_CASE
```

Рис. 6.18. Код программы **COM3 (ST)**

Программа работает следующим образом:

Блок [1]. Настройка COM-порта в соответствии с табл. 6.2. ***Обратите внимание***, что COM-порту **COM3** соответствует номер **4** (см. п. 2.3.1).

Блок [2]. При первом запуске программы с помощью ФБ ComService открывается COM-порт **COM3** с сетевыми настройками, заданными в блоках 0-1.

Блок [3]. Реализация последовательного опроса модулей через оператор **CASE**:

- в случае **xModule=0** происходит опрос модуля **МВ110-16Д**;

- в случае **xModule=1** происходит опрос модуля **МУ110-16Р**.

Блок [3.0.1]. Инициализация цикла опроса модуля **МВ110-16Д**. Опрос будет начат только в том случае, если ФБ ComService завершил работу (**Ready=TRUE**) без ошибок. На вход **ComHandle** данного поступает значение выхода **Handle** ФБ ComService. На вход **DevAddr** поступает адрес модуля – согласно табл. 6.2, данный модуль имеет адрес **1**. На входе **FirstAddr** указан первый опрашиваемый регистр модуля (**51**), на входе **Quantity** – число считываемых регистров (**1**). Таким образом, будет считан регистр 51, в котором хранится битовая маска состояний входов модуля. Карта регистров и поддерживаемые функции Modbus приведены в РЭ на модуль.

Блок [3.0.2]. Если цикл опроса модуля завершен (об этом сигнализирует импульс по переднему фронту на выходе **Complete**) без ошибок (**Exception=0**), то считанные данные из буфера ФБ копируются в пользовательский буфер (массив **abyMV110_16D_data**). ***Обратите внимание***, что обращаться к буферу ФБ можно только по флагу **Complete** и крайне рекомендуется предварительно копировать значения из буфера, после чего работать с ними в программе. Значение 1-го входа модуля в представлении копируется из пользовательского буфера в переменную **xMV110_16D_input1**.

Независимо от наличия ошибок завершаем данный цикл опроса модуля **МВ110-16Д** и начинаем цикл опроса модуля **МУ110-16Р**.

Блок [3.1.1]. Значение переменной **xMY110_16R_output1** записывается в буфер ФБ MB_WR_REGS, который будет отправлен в модуль **МВ110-16Р**. ***Обратите внимание***, что значение записывается в первый, а не нулевой байт буфера – это связано с различием представления данных в модуле и СПК. В нулевом байте буфера содержаться значения, которые будут записаны на 9-16 выходы модуля.

Блок [3.1.2]. Инициализация цикла опроса модуля **МУ110-16Р** с помощью ФБ MB_WR_REGS. Опрос будет начат только в том случае, если ФБ ComService завершил работу (**Ready=TRUE**) без ошибок. На вход **ComHandle** данного ФБ поступает значения выхода **Handle** ФБ ComService. На вход **DevAddr** поступает адрес модуля – согласно табл. 6.2, данный модуль имеет адрес **17**. На входе **FirstAddr** указан первый записываемый регистр модуля (**50**), на входе **Quantity** – число записываемых регистров (**1**). Таким образом, будет записан регистр 50, в котором хранится битовая маска состояний выходов модуля. Карта регистров и поддерживаемые функции Modbus приведены в РЭ на модуль.

Блок [3.1.3]. Если цикл опроса модуля завершен (об этом сигнализирует импульс по переднему фронту на выходе **Complete**) , то завершаем данный цикл опроса модуля **МУ110-16Р** и начинаем следующий цикл опроса модуля **МВ110-16Д**.

Обратите внимание, что **цикл опроса модуля** происходит в течение нескольких **циклов программы**.

При необходимости количество опрашиваемых модулей можно увеличить – для этого, соответственно, потребуется увеличить количество действий в операторе **CASE.** ***Обратите внимание***, что перед переходом к опросу следующего модуля, необходимо завершить опрос предыдущего, вызвав ФБ опроса с **Enable=FALSE**.

4. Код программы **PLC_PRG**:

```
1  COM3.xMY110_16R_output1:=(COM2.rMV110_8A_input1>30 AND COM3.xMV110_16D_input1);
```

Рис. 6.19. Код **PLC_PRG (ST)**

Программа **PLC_PRG** оперирует переменными программ **COM2** и **COM3**, и реализует следующий алгоритм: если значение 1-го аналогового входа модуля **МВ110-8А** превышает 30 и при этом 1-ый дискретный вход модуля **МВ110-16Д** имеет значение **TRUE** (замкнут), то произвести запись в 1-й дискретный выход модуля **МУ110-16Р** значения **TRUE** (замкнут). Во всех остальных случаях присвоить дискретному выходу значение **FALSE** (разомкнут).

6.5. Рекомендации по использованию библиотеки

1. ***Необходимо понимать***, любой из ФБ библиотеки не может выполниться в течение одного цикла контроллера. При опросе значительного количества регистров, полный опрос одного устройства может занять несколько десятков циклов.

2. Начинать опрос модулей следует только в том случае, когда порт открыт (т.е. у ФБ ComService выход **ready**=**TRUE**).

3. Опрос модулей должно производиться строго последовательно – т.е. опрос каждого следующего модуля должен начинаться после окончания предыдущего. Об окончании опроса сигнализирует импульс по переднему фронту на выходе **Complete** ФБ опроса модуля.

4. При опросе модулей следует анализировать код возникающих ошибок (выход **Exception**). Если с модуля считываются данные, то в случае ошибки полученные значения могут быть некорректны. Если в модуль записываются данные, то в случае ошибки запись может быть не произведена.

5. Значение таймаута опроса должно выбираться индивидуально в каждом конкретном случае в зависимости от скорости обмена, числа опрашиваемых регистров и особенностей slave-устройства (например, некоторые устройства могут удерживать линию после ответа).

6. Изменение параметров ФБ опроса (адрес устройства, номера регистров и т.д.) должно производиться только в случае значения **FALSE** на входе **Enable** блока.

7. При чтении данных рекомендуется по флагу **Complete** на выходе ФБ копировать содержимое буфера в пользовательский массив, после чего производить над данными нужные операции и присваивать их в переменные. При записи данных в slave-устройства перед инициализацией ФБ необходимо заполнить буфер ФБ соответствующими значениями.

7. Библиотека Modbus Slave

7.1. Установка библиотеки

Библиотека **ModbusSlave** доступна на диске с ПО, входящем в комплект поставки, а также на сайте компании ОВЕН в разделе **CODESYS V3/Библиотеки**. Она распространяется как отдельно, так и в составе пакета библиотек Овен **LibInstall.**

Для установки пакета в **CODESYS** в меню **Инструменты** выберите пункт **Менеджер пакетов**, после чего укажите путь к файлу пакета и нажмите **Установить**:

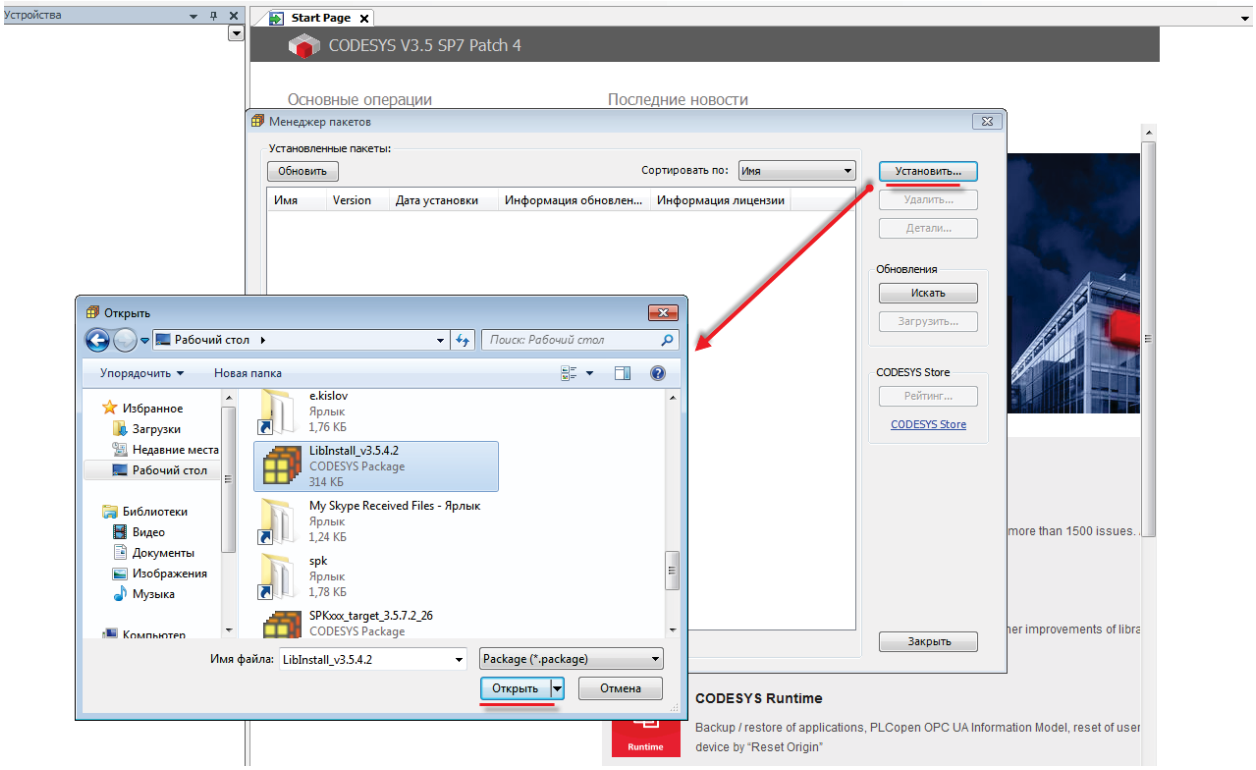


Рис. 7.1. Установка пакета библиотек Овен в среду **CODESYS**

В появившемся диалоговом окне выберите пункт **Полная установка**, после чего нажмите кнопку **Next**:

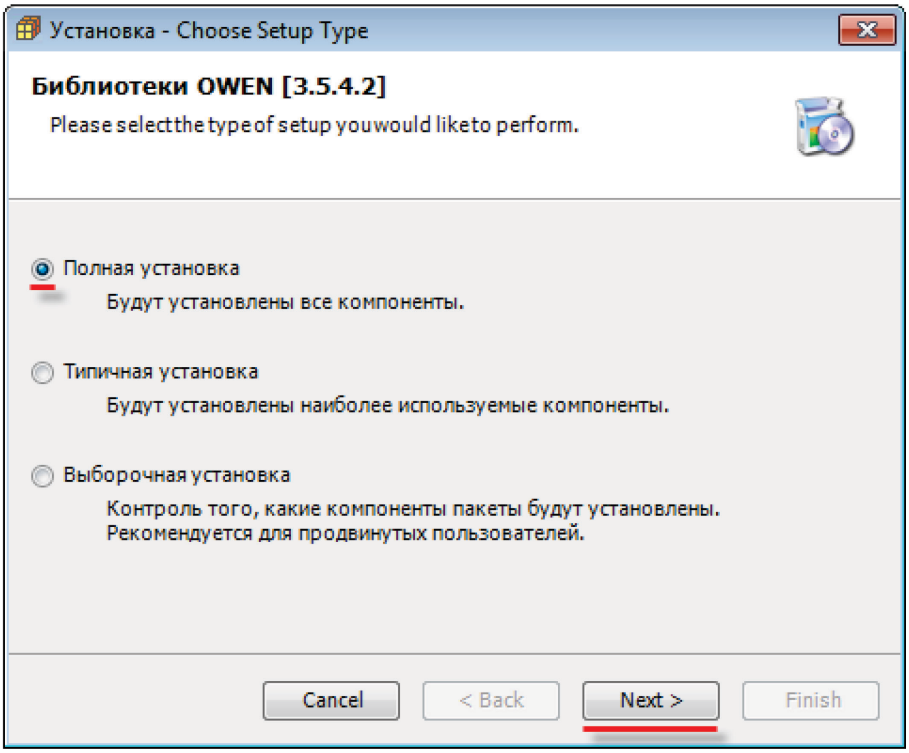


Рис. 7.2. Начало установки пакета библиотек

После завершения установки закройте диалоговое окно с помощью кнопки **Finish**:

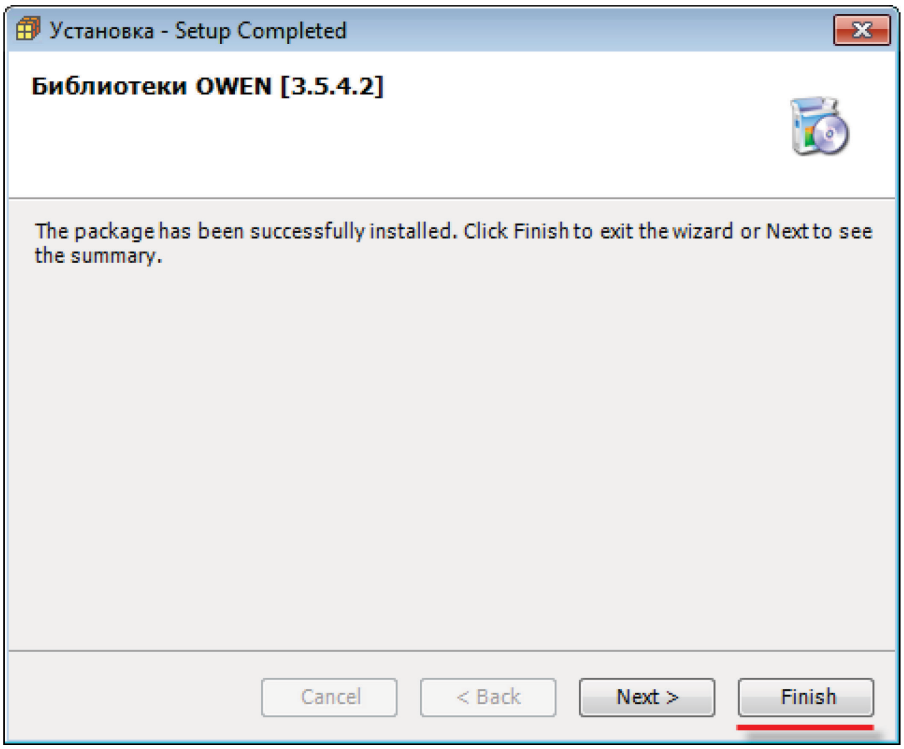


Рис. 7.3. Завершение установки пакета библиотек

7.2. Добавление библиотеки в проект CODESYS

Для добавления библиотеки **ModbusSlave** в проект **CODESYS**, в **Менеджере библиотек** нажмите кнопку **Добавить библиотеку** и в строке поиска введите **modbus**, после чего выберите из списка нужную библиотеку и нажмите **ОК**.

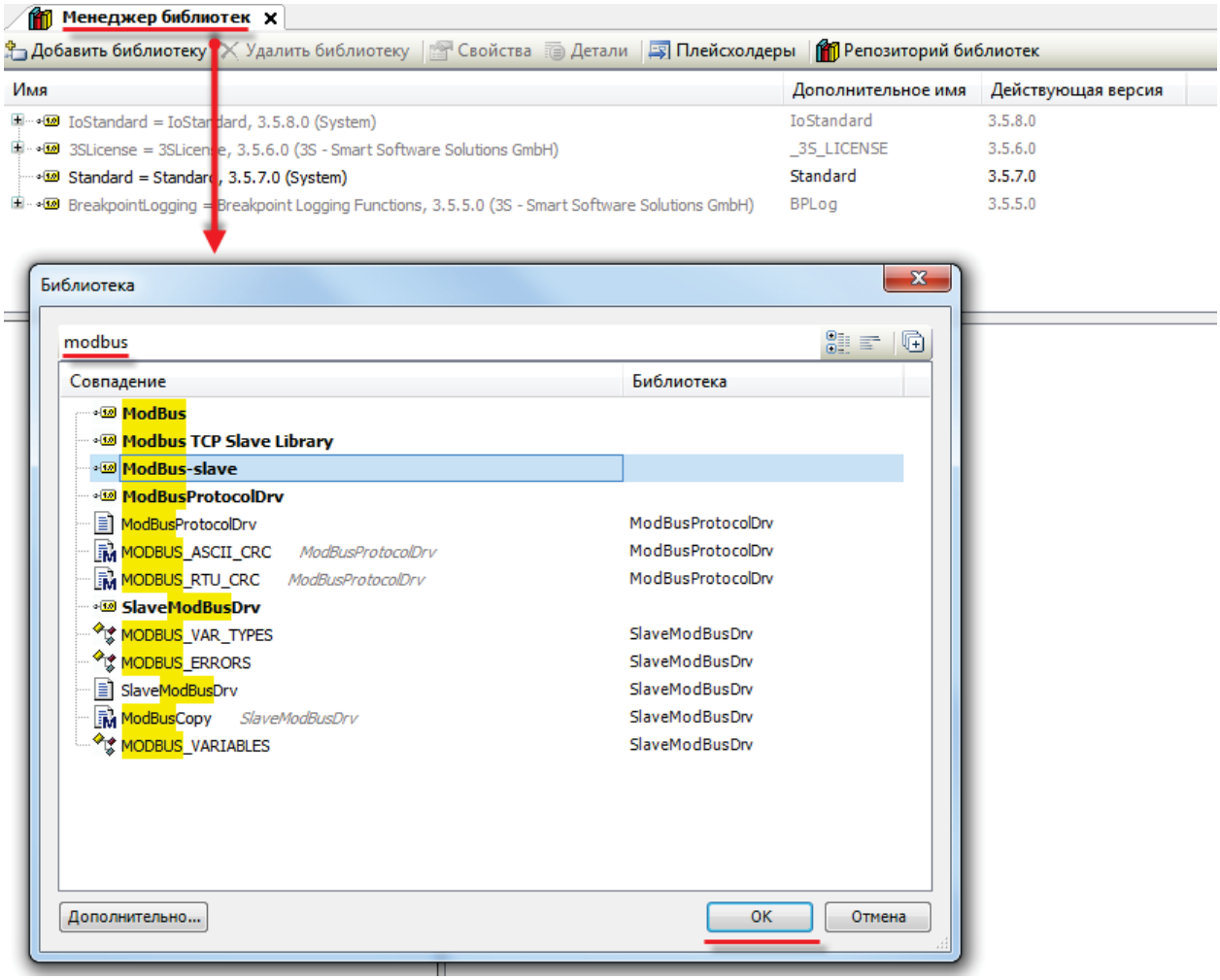


Рис. 7.4. Добавление библиотеки **ModbusSlave**

В библиотеку **ModbusSlave** не входит ФБ открытия COM-порта, поэтому для этой цели необходимо воспользоваться другой библиотекой. Рекомендуется использовать библиотеку **ComService**.

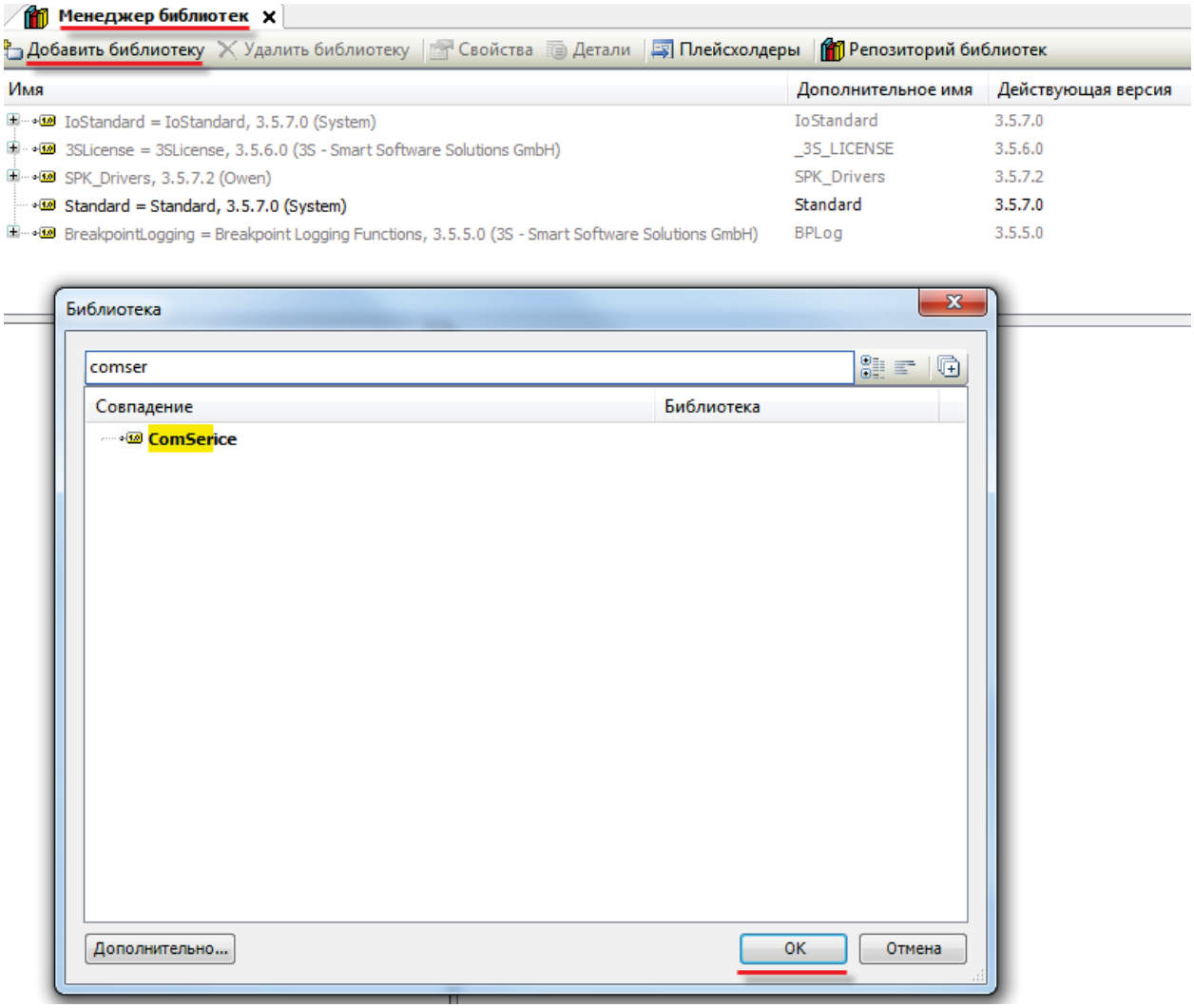


Рис. 7.5. Добавление библиотеки **ComService**

После добавления библиотеки появятся в списке **Менеджера библиотек**:

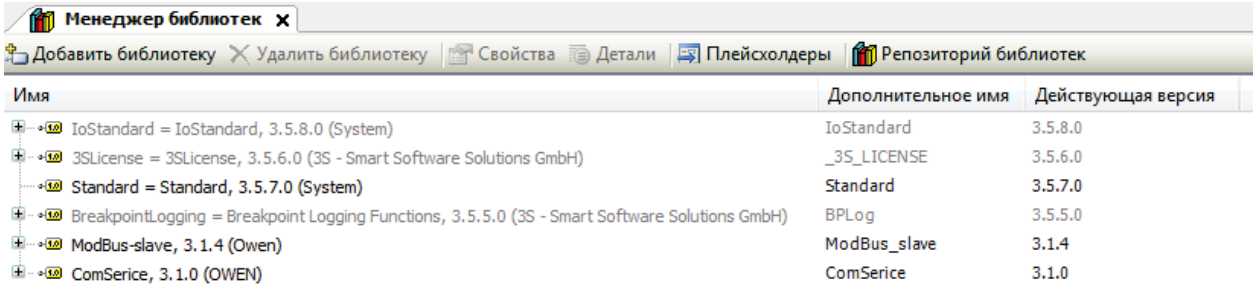


Рис. 7.6. Список библиотек проекта

7.3. Описание ФБ библиотеки

7.3.1. Блок ComService (библиотека ComService)

Функциональный блок **ComService**, входящий в библиотеку **ComService**, используется для открытия/закрытия COM-порта, а также его настройки. Его описание приведено в п. 6.3.1.

7.3.2. Блок MB_SLAVE

Функциональный блок **MB_SLAVE** реализует slave-устройство Modbus RTU. Все области памяти наложены друг на друга в буфере ФБ, и обратиться к данным буфера можно с помощью любой из функций (см. табл. 2.1).

Имя переменной	Тип	Описание
Входные переменные		
ComHandle	DWORD	Идентификатор порта, поступающий с выхода блока ComService (или аналогичного).
DevAddr	BYTE	Адрес данного slave-устройства.
pBuffer	POINTER TO BYTE	Указатель на первый байт буфера ФБ.
BufSize	WORD	Размер буфера ФБ в байтах.
Выходные переменные		
NewData	BOOL	Флаг наличия в буфере новых данных от мастера.
Error	MB_ERROR_TYPE	Код ошибки. **0** – ошибок нет; **16#1** – осуществлен запрос неподдерживаемой функцией; **16#2** – осуществлен запрос с некорректным адресом регистра; **16#3** – осуществлена попытка записи некорректного значения.

7.4. Пример: СПК207 (master) + СПК207 (slave)

Рассмотрим пример настройки обмена между двумя контроллерами **СПК207**, один из которых выполняется функцию **master** (с помощью стандартных средств конфигурирования CODESYS), а другой – **slave** (с помощью библиотеки **ModbusSlave**).

Формулировка задачи: с помощью СПК207 (master) реализовать чтение переменных различных типов из СПК207 (slave). Предусмотреть возможность записи этих данных по триггеру. Slave-устройство также должно иметь возможность по триггеру изменять значения своих переменных.

Структурная схема примера приведена на рис. 7.7:

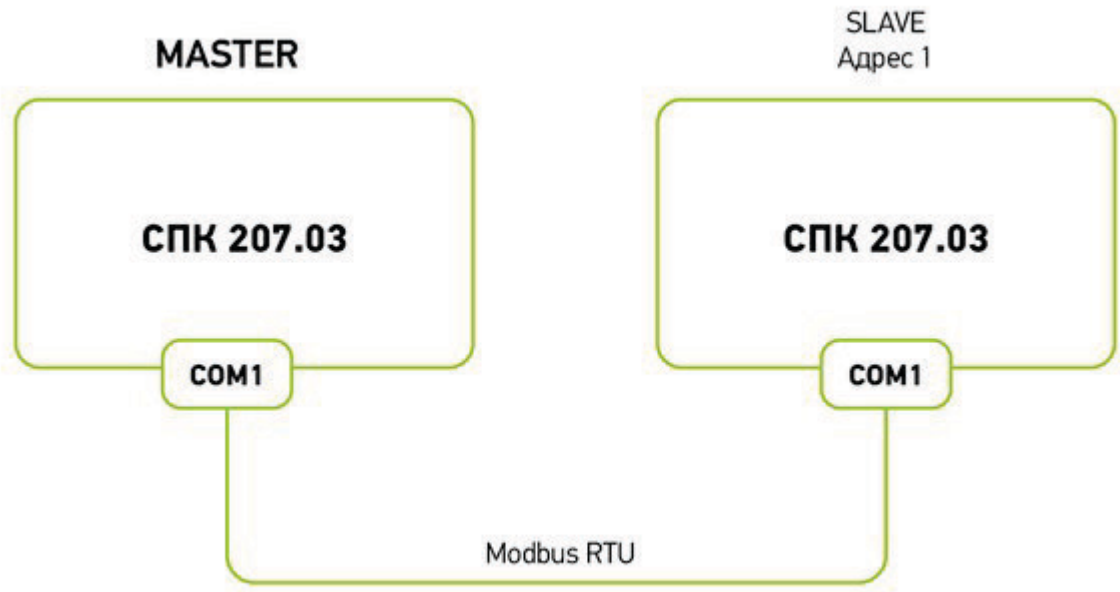


Рис. 7.7. Структурная схема примера **Библиотека ModbusSlave (СПК+СПК)**

Пример создан в среде **CODESYS 3.5 SP7 Patch4** и подразумевает запуск на **СПК207.03.CS(-WEB)**.

Пример доступен для скачивания: Example_ModbusSlaveLibrary.projectarchive

Сетевые параметры модулей приведены в табл. 7.1.

Список переменных, используемых в примере, приведен в табл. 7.2.

Табл. 7.1. Сетевые параметры СПК

Параметр	**СПК207 (master)**	**СПК207 (slave)**
Используемый COM-порт	*COM1*	*COM1*
Адрес (Slave ID)	-	1
Скорость обмена	115200	
Количества бит данных	8	
Контроль четности	отсутствует	
Количество стоп-бит	1	

Табл. 7.2. Список переменных программы

		СПК207 (master)	**СПК207 (slave)**	
Переменная	**Функция**	**Переменная**	**Переменная**	**Регистры**
Мастер считывает				
BOOL	Read Discrete Inputs	xVarRead	xVarWrite	0
WORD	Read Input Registers	wVarRead	wVarWrite	1
REAL	Read Input Registers	_2WORD_TO_REAL.rRealValue	_REAL_TO_2WORD.rRealValue	2-3
STRING	Read Input Registers	_3WORD_TO_STRING.sStringValue	_STRING_TO_3WORD.sStringValue	4-6
Мастер записывает				
BOOL	Write Single Coil	xVarWrite	xVarRead	0
WORD	Write Single Register	wVarWrite	wVarRead	1
REAL	Write Multiple Registers	_REAL_TO_2WORD.rRealValue	_2WORD_TO_REAL.rRealValue	2-3
STRING	Write Multiple Registers	_STRING_TO_3WORD.sStringValue	_3WORD_TO_STRING.sStringValue	4-6

Еще раз обратите внимание, что мастер читает и записывает одни и те же регистры slave-устройства. Slave-устройство получает данные, записанные мастером, и при этом может изменять их – соответственно, мастер получит их при следующей операции чтения.

7.4.1. Настройка СПК207 (master)

1. Создайте новый проект **CODESYS** для **СПК207** с программой **PLC_PRG** на языке **CFC**.

2. Добавьте в проект объединение с именем **Real_Word**:

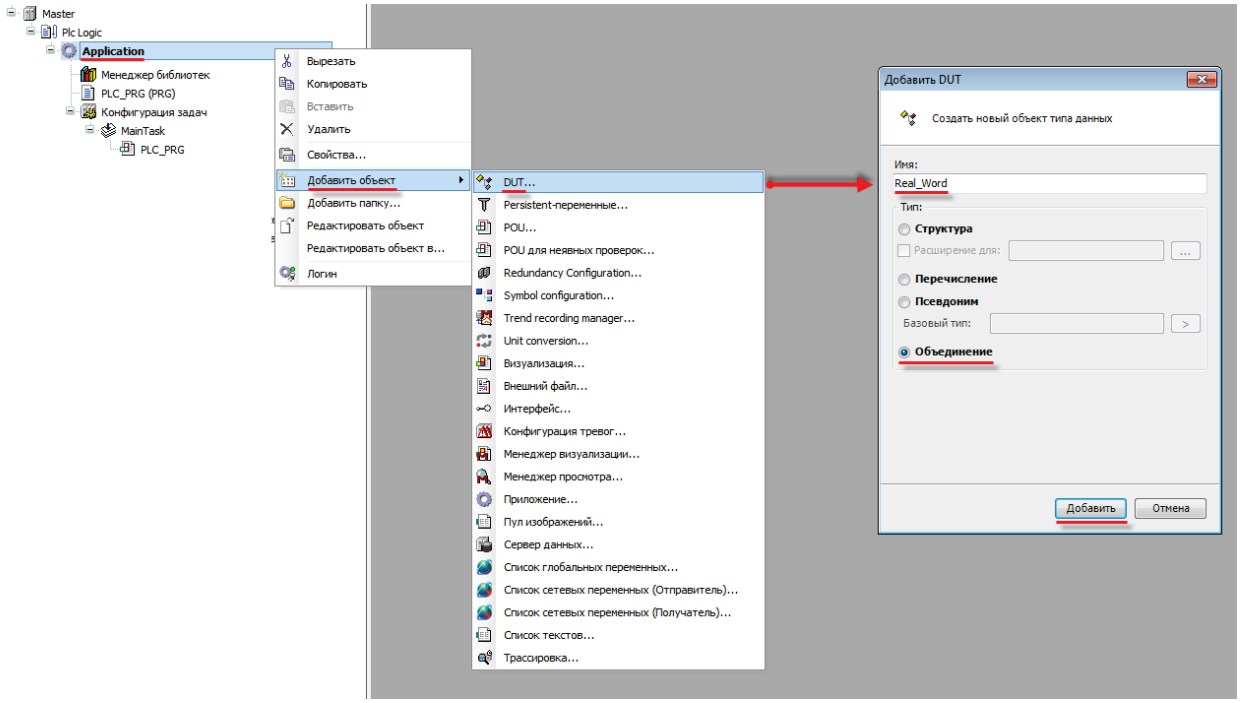


Рис. 7.8. Добавление в проект объединения

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

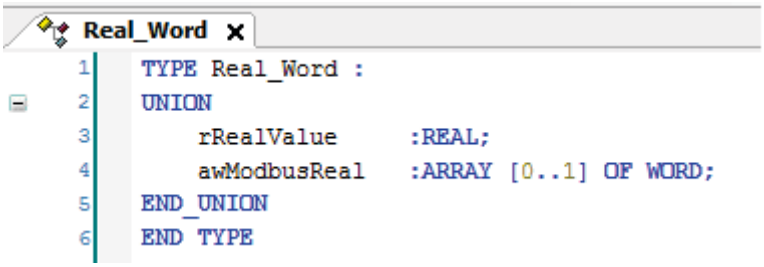


Рис. 7.9. Объявление переменных объединения

3. Добавьте в проект объединение с именем **String_Word**.

В объединении объявите переменную **sStringValue** типа **STRING** (с ограничением на размер в 6 символов) и массив **awModbusString** типа **WORD**, содержащий три элемента (**STRING** сможет содержать до 6 символов, поскольку каждый **WORD** может содержать два символа):

```
TYPE String_Word :
UNION
    awModbusString      :ARRAY [0..2] OF WORD;
    sStringValue        :STRING;
END_UNION
END_TYPE
```

Рис. 7.10. Объявление переменных объединения

4. Объявите в программе нужные переменные:

```
PROGRAM PLC_PRG
VAR
    (*считываемые значения*)
    xVarRead            :BOOL;
    wVarRead            :WORD;
    _2WORD_TO_REAL      :Real_Word;
    _3WORD_TO_STRING    :String_Word;

    (*записываемые значения*)
    xVarWrite           :BOOL;
    wVarWrite           :WORD;
    _REAL_TO_2WORD      :Real_Word;
    _STRING_TO_3WORD    :String_Word;

    xTrigger            :BOOL;          // триггер записи в slave
END_VAR
```

Рис. 7.11. Объявление переменных программы

5. Код программы будет выглядеть следующим образом:

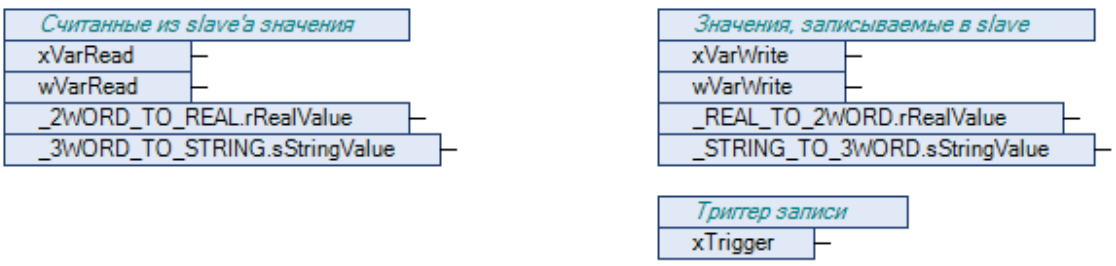


Рис. 7.12. Код программы на языке **CFC**

6. Добавьте в проект устройство **Modbus COM**:

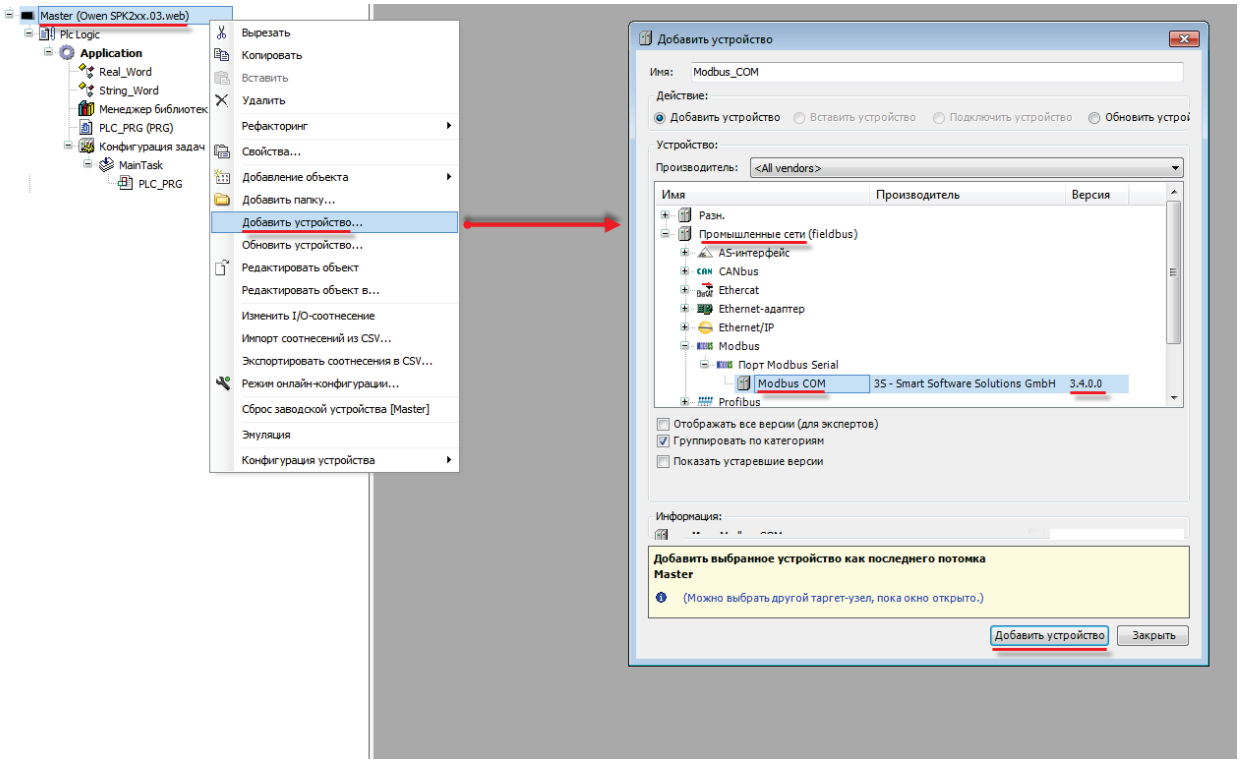


Рис. 7.13. Добавление устройства **Modbus COM**

В конфигурации COM-порта укажите сетевые настройки в соответствии с табл. 7.1, а также номер порта. COM-порту **1** будет соответствовать номер **2** (см. п. 2.3.1).

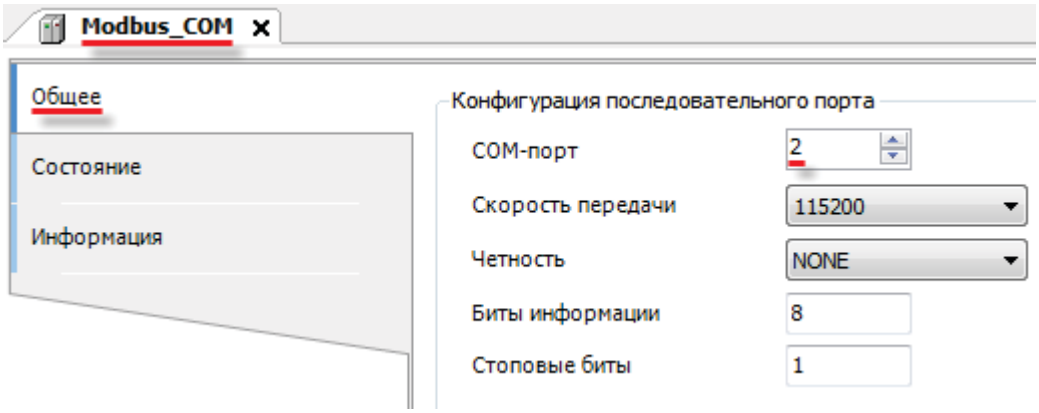


Рис. 7.14. Настройки COM-порта **COM1**

7. В COM-порт добавьте компонент **Modbus Master**:

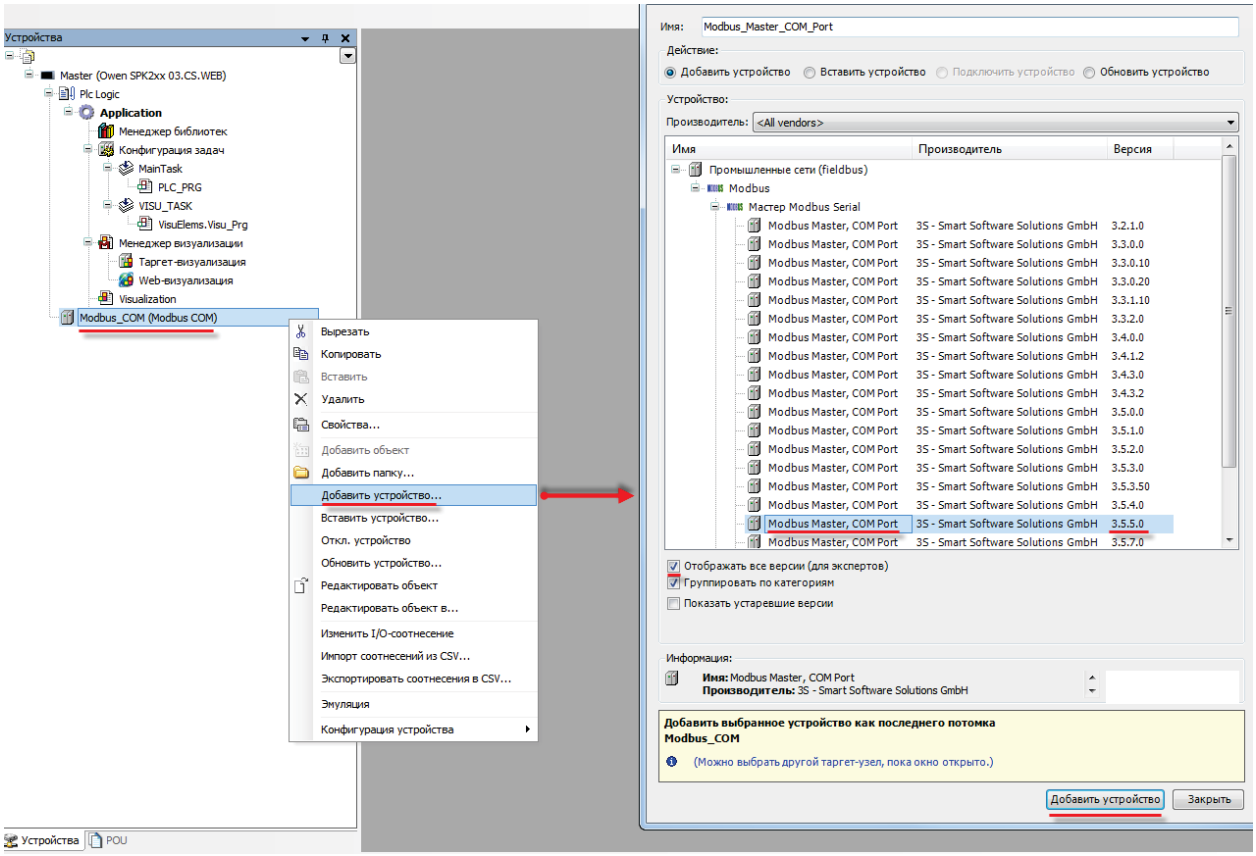


Рис. 7.15. Добавление компонента **Modbus Master**

В настройках компонента поставьте галочку **Автоперезапуск соединения**.

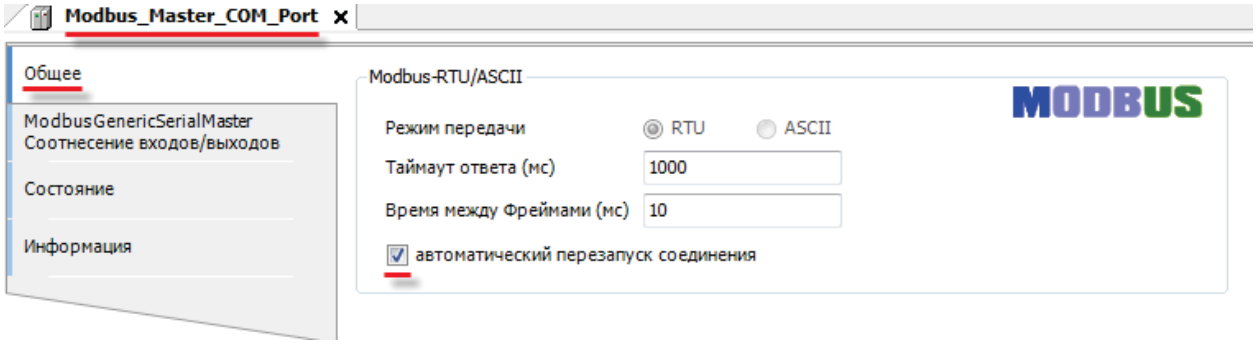


Рис. 7.16. Настройка компонентов **Modbus Master**

8. В **Modbus Master** добавьте компонент **Modbus Slave**:

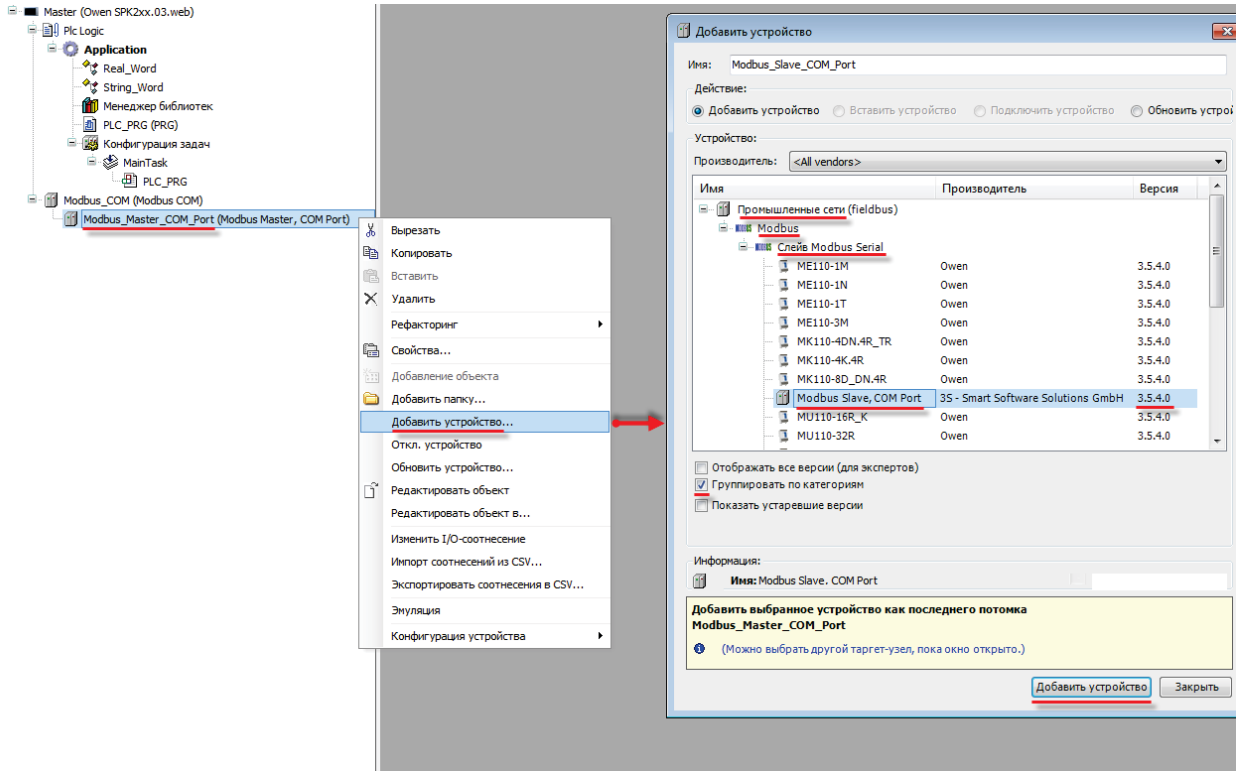


Рис. 7.17. Добавление компонента **Modbus Slave** в проект

В настройках компонента на вкладке **Общее** укажите адрес slave-устройства в соответствии с табл. 7.1.

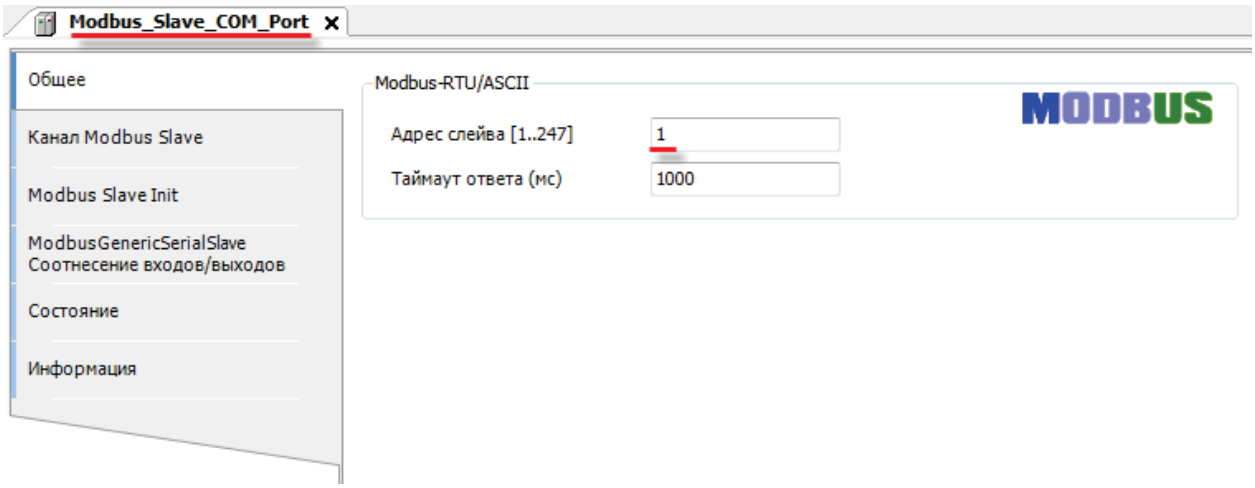


Рис. 7.18. Добавление компонента **Modbus Slave** в проект

На вкладке **Канал Modbus Slave** создайте 8 каналов – четыре для чтения данных и четыре для записи. ***Обратите внимание***, что считываемые и записываемые данные расположены в одной области памяти (т.е. происходит чтение и запись одних и тех же регистров). Это видно по соответствию значений столбцов **Сдвиг** и **Длина**.

Имя	Тип доступа	Триггер	Сдвиг READ	Длина	Обработка ошибок	Сдвиг WRITE	Длина
Read_BOOL	Read Discrete Inputs (Код функции 02)	CYCLIC, t#100ms	16#0000	1	Сохранить последнее значение		
Read_WORD	Read Input Registers (Код функции 04)	CYCLIC, t#100ms	16#0001	1	Сохранить последнее значение		
Read_REAL	Read Input Registers (Код функции 04)	CYCLIC, t#100ms	16#0002	2	Сохранить последнее значение		
Read_STRING	Read Input Registers (Код функции 04)	CYCLIC, t#100ms	16#0004	3	Сохранить последнее значение		
Write_BOOL	Write Single Coil (Код функции 05)	RISING_EDGE				16#0000	1
Write_WORD	Write Single Register (Код функции 06)	RISING_EDGE				16#0001	1
Write_REAL	Write Multiple Registers (Код функции 16)	RISING_EDGE				16#0002	2
Write_STRING	Write Multiple Registers (Код функции 16)	RISING_EDGE				16#0004	3

Рис. 7.19. Настройка каналов **Modbus Slave**

На вкладке **ModbusGenericSerialSlave Соотнесение входов/выходов привяжите** к каналам переменные программы в соответствии с табл. 7.2. Не забудьте у параметра **Всегда обновлять переменные** выставить значение **Включено 2**.

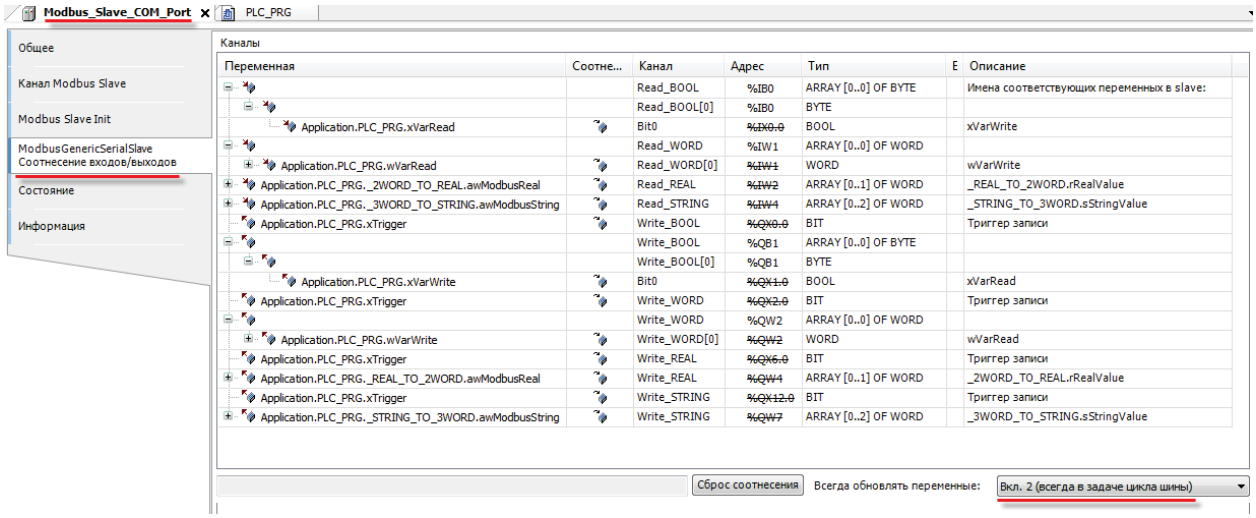


Рис. 7.20. Привязка переменных к каналу

На этом настройка **СПК (master)** завершена.

7.4.2. Настройка СПК207 (slave)

1. В проект CODESYS, содержащий СПК (master), добавьте контроллер СПК (slave).

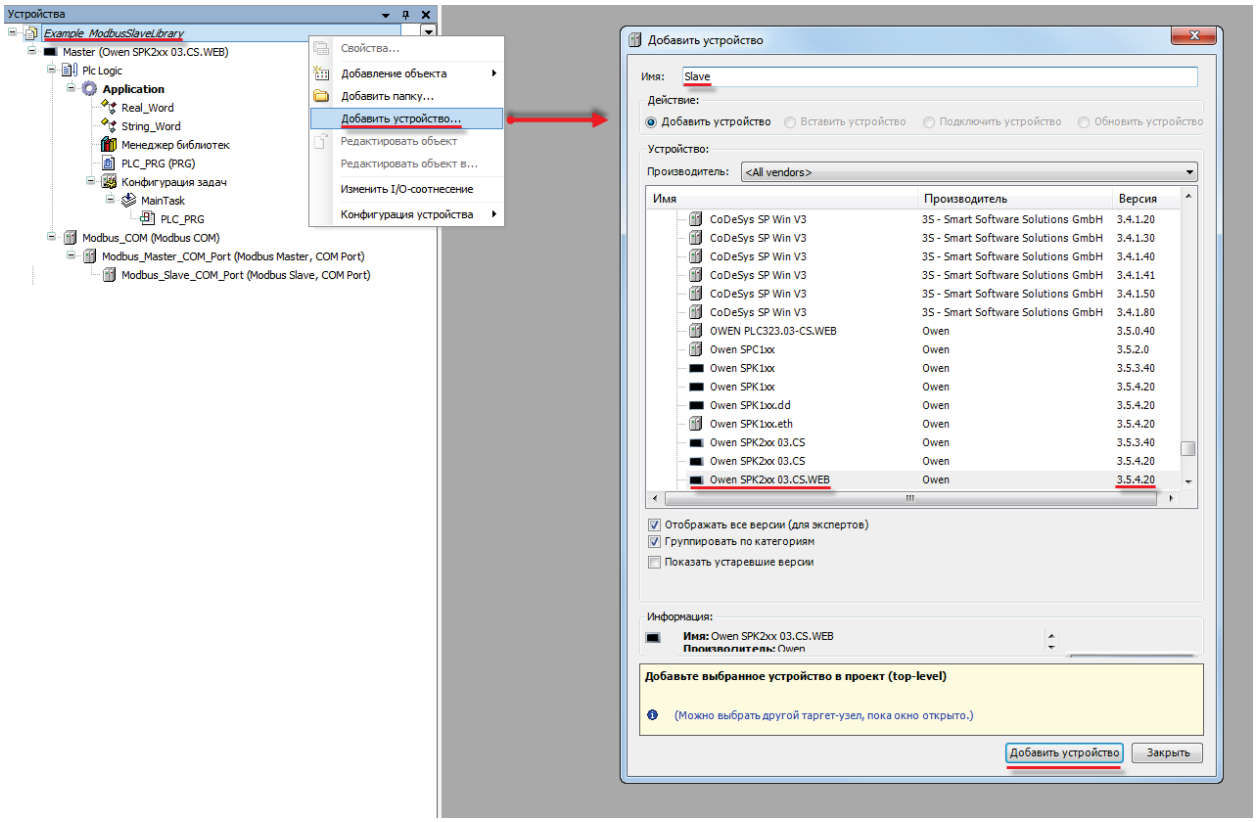


Рис. 7.21. Добавление СПК (slave) в проект **CODESYS**

2. Добавьте в проект объединение с именем **Real_Word**.

В объединении объявите переменную **rRealValue** типа **REAL**, массив **awModbusReal** типа **WORD**, содержащий два элемента, и массив **abyModbusReal** типа **BYTE**, содержащий четыре элемента:

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

Рис. 7.22. Объявление переменных объединения

3. Добавьте в проект объединение с именем **String_Word**.

В объединении объявите переменную **sStringValue** типа **STRING** (с ограничением на размер в 6 символов), массив **awModbusString** типа **WORD**, содержащий три элемента, и массив **abyModbusString** типа **BYTE**, содержащий шесть элементов (**STRING** сможет содержать до 6 символов, поскольку каждый **WORD** может содержать два символа):

String_Word

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

Рис. 7.23. Объявление переменных объединения

4. Объявите в программе нужные переменные:

```
1  PROGRAM PLC_PRG
2  VAR
3      ModbusSlave         :MB_SLAVE;              // ФБ, реализующий Modbus Slave
4      COM_Service_COM1    :COM_SERVICE;           // ФБ настройки и открытия COM-порта
5
6      pBuffer             :POINTER TO BYTE;       // указатель на первый байт буфера slave'а
7      abyBuffer           :ARRAY [0..15] OF BYTE; // буфер slave'а
8
9      (*значения, которые мастер записывает в slave*)
10     xVarRead            :BOOL;
11     wVarRead            :WORD;
12     _2WORD_TO_REAL      :Real_Word;
13     _3WORD_TO_STRING    :String_Word;
14
15     (*значения, которые slave подготавливает для чтения мастером*)
16     xVarWrite           :BOOL;
17     wVarWrite           :WORD;
18     _REAL_TO_2WORD      :Real_Word;
19     _STRING_TO_3WORD    :String_Word;
20
21     xTrigger            :BOOL;       // триггер записи подготовленных значений в буфер
22 END_VAR
```

Рис. 7.24. Объявление переменных программы

5. Код программы будет выглядеть следующим образом (*рисунок хорошо масштабируется*). **Обратите внимание**, что программа содержит 4 действия: **BUFFER_TO_REAL, BUFFER_TO_STRING, REAL_TO_BUFFER и STRING_TO_BUFFER**. Их код приведен ниже.

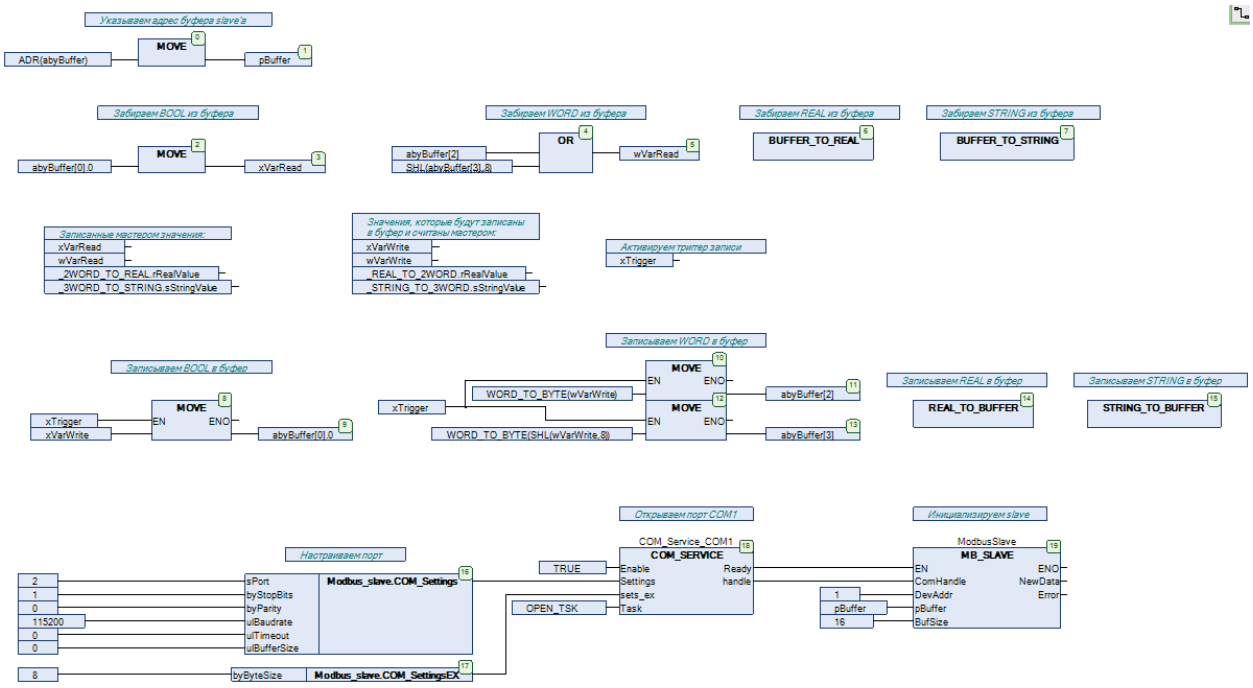


Рис. 7.25. Код программы на языке CFC

Действия были использованы исключительно с целью сократить код основной программы.

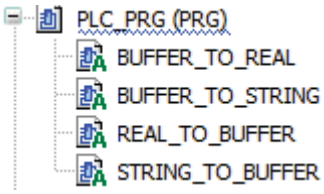


Рис. 7.26. Действия программы **PLC_PRG**

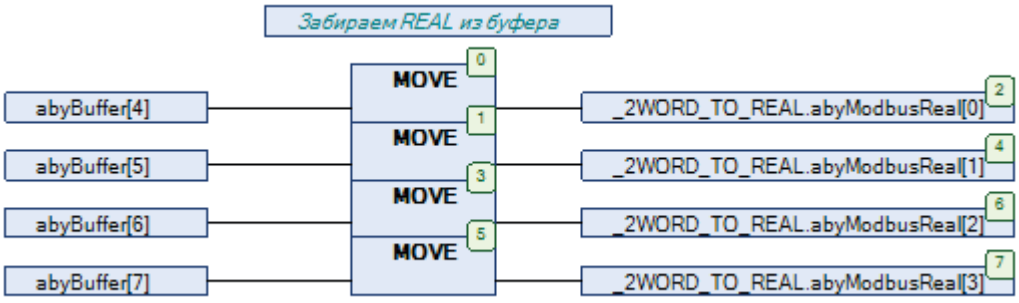


Рис. 7.27. Код действия **Buffer_TO_REAL**

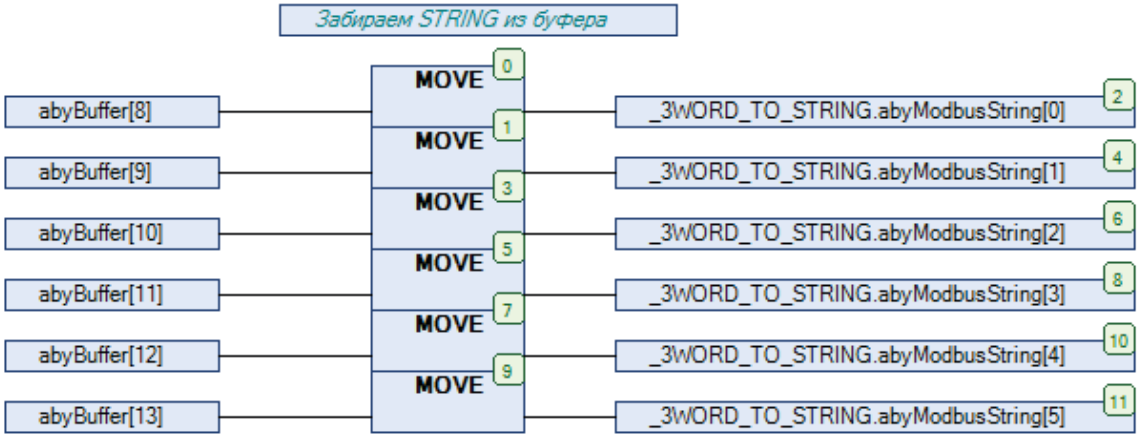


Рис. 7.28. Код действия **Buffer_TO_STRING**

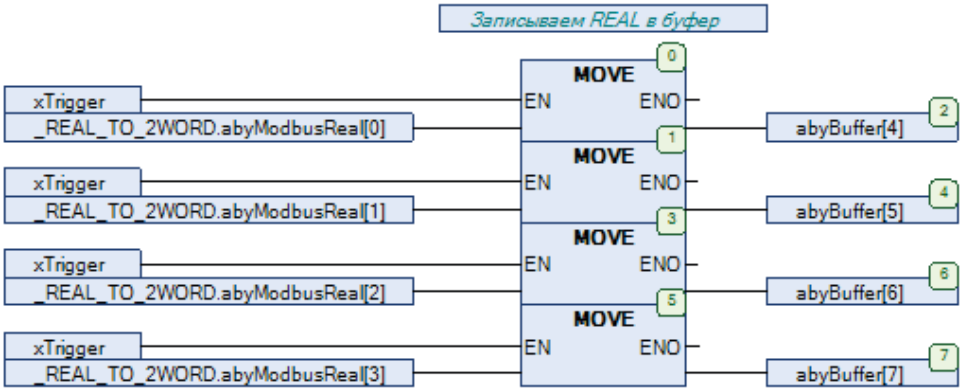


Рис. 7.29. Код действия **REAL_TO_BUFFER**

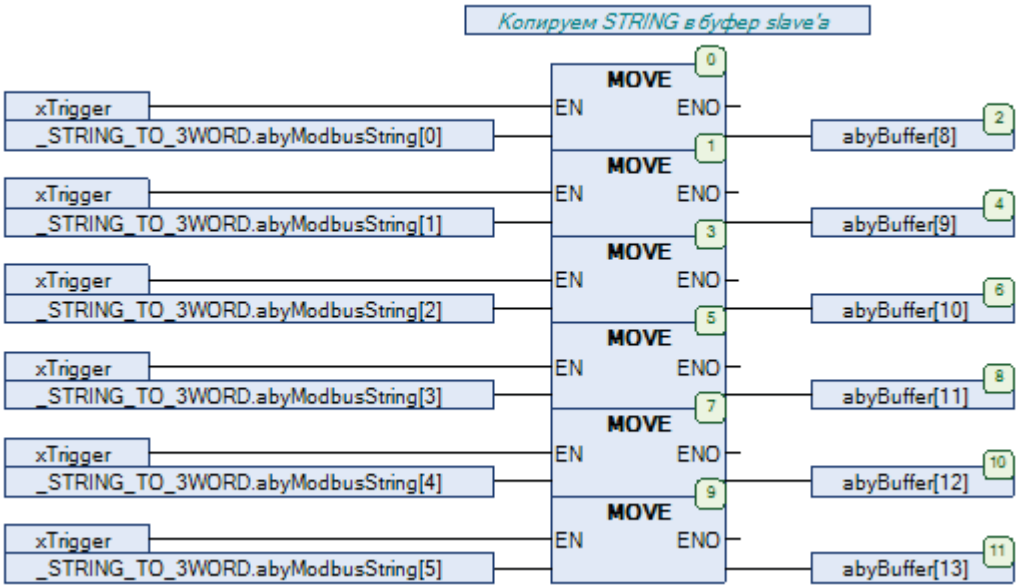


Рис. 7.30. Код действия **STRING_TO_BUFFER**

6. Программа, приведенная на рис. 7.25, работает следующим образом:

Блоки 0-1. В переменную **pBuffer** записывается адрес массива, который будет использоваться в качестве буфера slave-устройства.

Блоки 2-3. Значение нулевого бита нулевого байта буфера записывается в переменную **xVarRead**. ***Обратите внимание***, что рекомендуется выравнивать данные, занимающие один и более регистров, по кратным адресам байтов – т.е. не стоит, например, записывать переменную типа **WORD** в байты 1-2 – поэтому первый байт буфера в данном примере использоваться не будет. В то же время необходимо понимать, что при необходимости можно было бы использовать его для работы с переменными типа **BOOL.**

Блоки 4-5. Значения второго и третьего байта буфера склеиваются в **WORD**, значение которого записывается в переменную **wVarRead**.

Блок 6. Значения четвертого, пятого, шестого и седьмого байта буфера записываются в байты объединения **_2WORD_TO_REAL** (см. рис. 7.28). Это приведет к появлению в переменной **rRealValue** (типа **REAL**) данного объединения значения с плавающей точкой.

Блок 7. Значения с восьмого по тринадцатый байта буфера записывается в байты объединения **_3WORD_TO_STRING** (см. рис. 7.29). Это приведет к появлению в переменной **sStringValue** (типа **STRING**) данного объединения текстового значения.

Блоки 8-9. По переднему фронту переменной **xTrigger** значение переменной **xVarWrite** записывается в нулевой бит нулевого байта буфера.

Блоки 10-13. По переднему фронту переменной **xTrigger** значение переменной **wVarWrite** записывается во второй-третий байты буфера.

Блок 14. По переднему фронту переменной **xTrigger** значение переменной **rRealValue** объединения **_REAL_TO_2WORD** записывается в четвертый…седьмой байты буфера.

Блок 15. По переднему фронту переменной **xTrigger** значение переменной **rRealValue** объединения **_STRING_TO_3WORD** записывается в восьмой…пятнадцатый байты буфера.

Блоки 16-17. Настройка COM-порта в соответствии с табл. 7.1. ***Обратите внимание***, что COM-порту **COM1** соответствует номер **2** (см. п. 2.3.1).

Блок 18. При первом запуске программы с помощью ФБ ComService открывается COM-порт **COM2** с сетевыми настройками, заданными в блоках 0-1.

Блок 19. Инициализация ФБ, реализующего slave-устройство. ФБ будет инициализирован только в том случае, если ФБ ComService завершил работу (выход **Ready**=**TRUE**). На вход **ComHandle** поступает значение выхода **Handle** ФБ ComService. На входе **DevAddr** задается адрес slave-устройства – согласно табл. 7.2, он будет равен **1**. На вход **pBuffer** подается указатель на первый байт буфера – см. блоки 0-1. На входе **BufSize** указывается размер буфера в байтах. В данном примере задействовано 14 байт буфера, округлим до **16-ти**.

Работа с примером описана в п. 7.4.3.

7.4.3. Запуск примера

1. Соедините первый COM-порт СПК (master) с первым COM-портом СПК (slave).

2. Загрузите проекты в каждый из СПК и запустите их.

3. Измените в программе СПК (slave) значения переменных **xVarWrite, wVarWrite, _REAL_TO_2WORD.rRealValue** и **_STRING_TO_3WORD.sStringValue**.

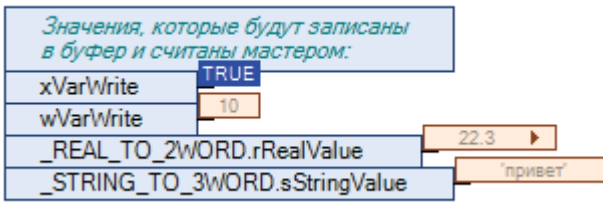


Рис. 7.31. Изменение значений переменных в программе СПК (slave)

Подайте единичный импульс в переменную **xTrigger** – это приведет к записи введенных значений в буфер.

Рис. 7.32. Активация триггера записи значений в буфер

Наблюдайте соответствующие изменения в программе СПК (master):

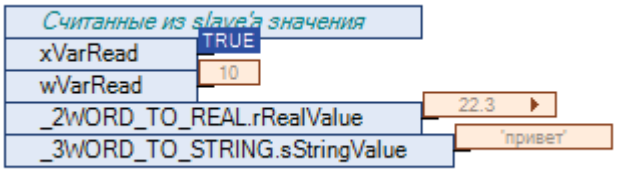


Рис. 7.33. СПК (master) считал значения из СПК (slave)

4. Измените в программе СПК (master) значения переменных **xVarWrite, wVarWrite, _REAL_TO_2WORD.rRealValue** и **_STRING_TO_3WORD.sStringValue**.

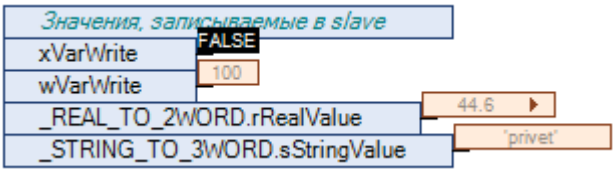


Рис. 7.34. Изменение значений переменных в программе СПК (master)

Подайте единичный импульс в переменную **xTrigger** – это приведет к записи введенных значений в slave-устройство.

Рис. 7.35. Активация триггера записи в slave-устройство

Наблюдайте соответствующие изменения в программе СПК (slave):

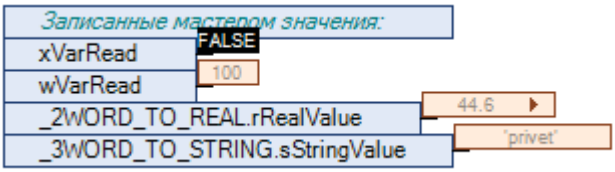


Рис. 7.36. В СПК (slave) были записаны данные из СПК (master)

8. Modbus TCP (стандартные средства конфигурирования

8.1. Особенности работы с Modbus TCP

В целом, настройка обмена с помощью **Modbus TCP** стандартными средствами конфигурирования **CODESYS** аналогично настройке **Modbus RTU**.

Общая методика конфигурирования интерфейсов приведена в п. 4.1.

Способы преобразования данных описаны в п. 4.4.

Диагностика обмена описана в п. 4.5.

Ниже описаны особенности конфигурации **Modbus TCP** по сравнению **Modbus RTU** в режимах **master** и **slave**.

8.2. Настройка СПК в режиме Modbus TCP Master

1. Нажмите **ПКМ** на компонент **Device** и добавьте компонент **Ethernet,** расположенный во вкладке **Промышленные сети/Ethernet-адаптер**. ***Обратите внимание***, что версия компонента не должна превышать версию **target-файла** СПК. Для отображения предыдущих версий компонента поставьте галочку **Отображать все версии**. См. рекомендации в приложении Г.

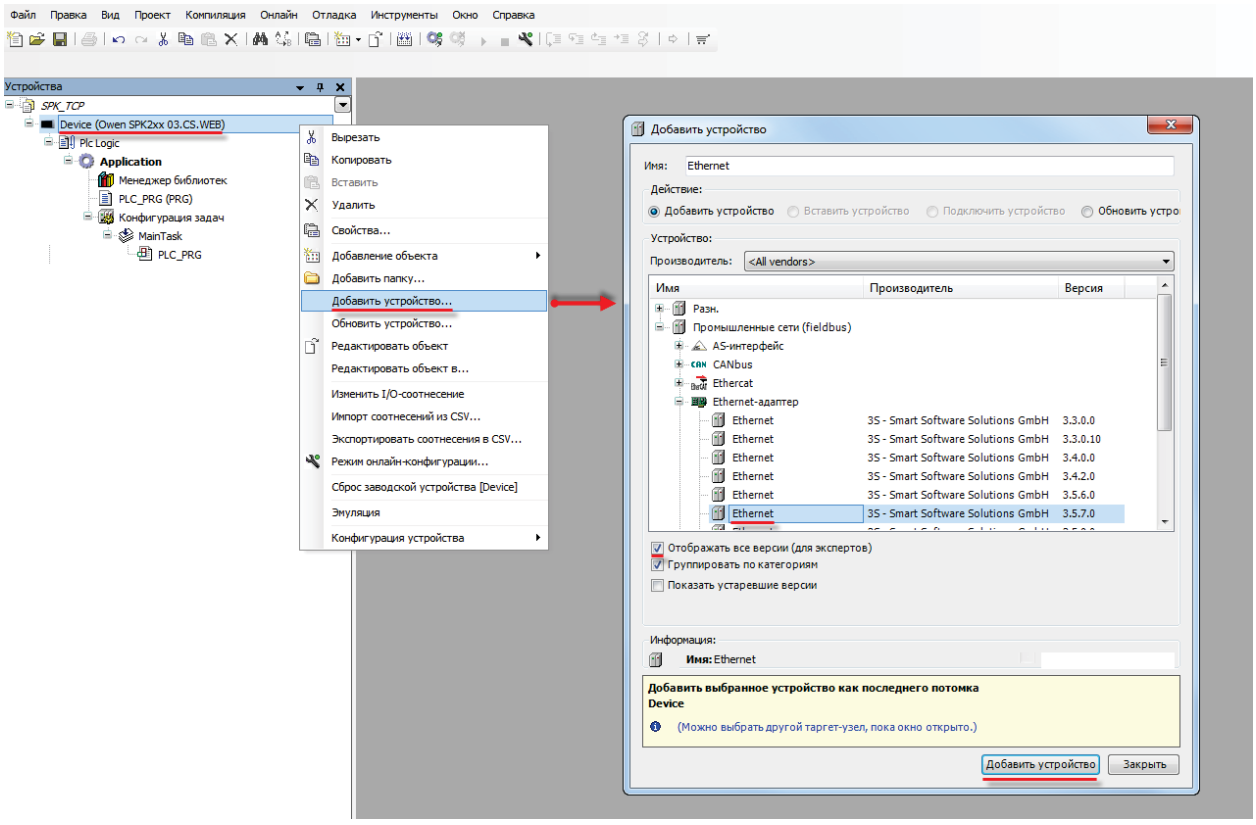


Рис. 8.1. Добавление компонента **Ethernet**

В настройках компонента на вкладке **Конфигурация Ethernet** необходимо указать IP-адрес СПК, маску подсети и IP-адрес шлюза. ***Обратите внимание***, что эти настройки должны соответствовать сетевым параметрам, указанным в **Конфигураторе СПК**.

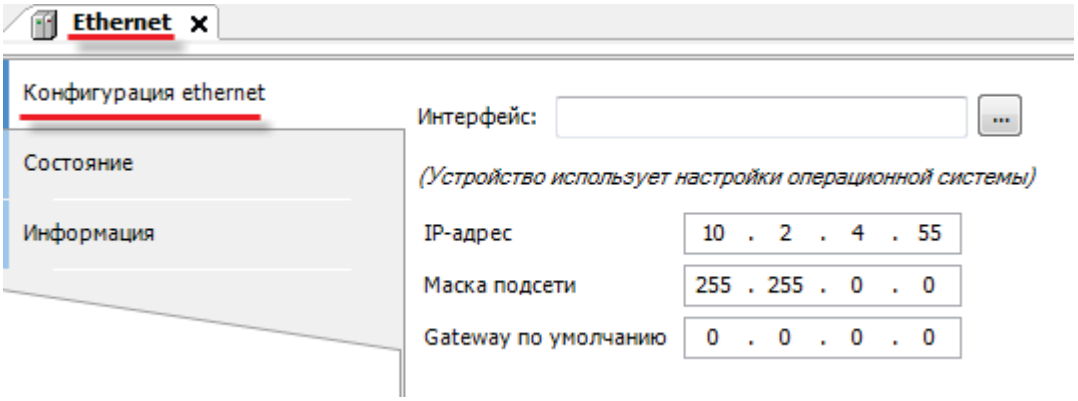


Рис. 8.2. Настройки компонента **Ethernet**

2. Нажмите **ПКМ** на компоненте **Modbus COM** и добавьте компонент **Modbus Master**, расположенный во вкладке **Промышленные сети/Modbus/Мастер Modbus TCP.**

Обратите внимание, что версия компонента не должна превышать версию **target-файла** СПК. Для отображения предыдущих версий компонента поставьте галочку **Отображать все версии**. См. рекомендации в приложении Г.

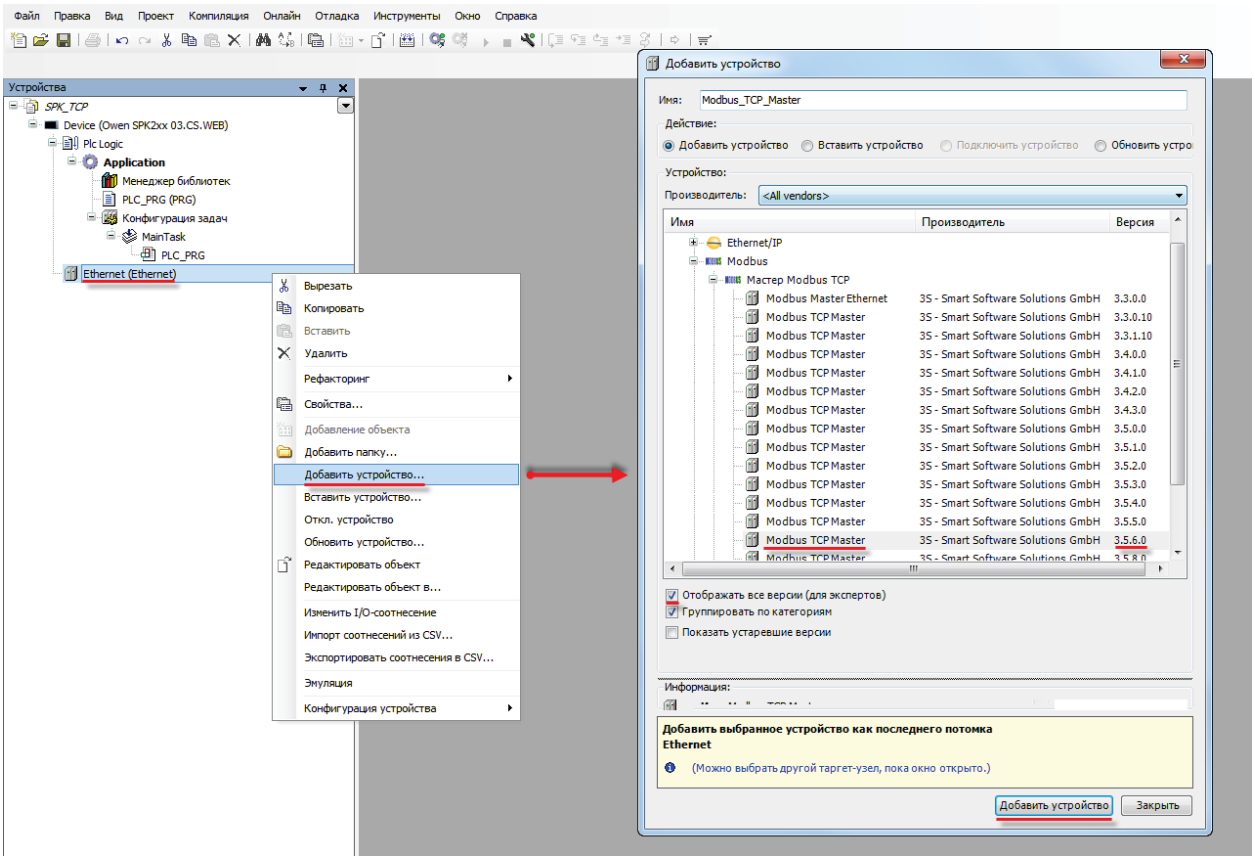


Рис. 8.3. Добавление компонента **Modbus TCP Master**

В настройках компонента на вкладке **Общее** необходимо задать сетевые настройки мастера.

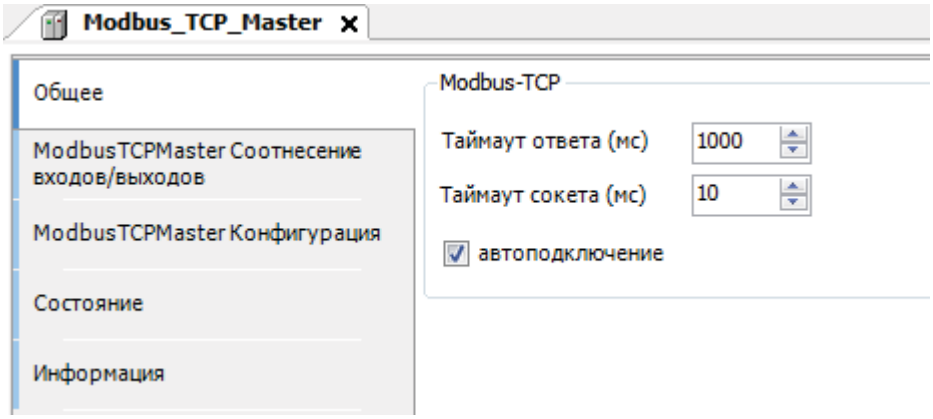


Рис. 8.4. Настройки компонента **Modbus TCP Master**

Таймаут ответа – время, которое master дает slave-устройству на ответ. По истечению этого времени, master переходит к опросу следующего slave-устройства. Значение, введенное здесь, будет по умолчанию использоваться для всех slave-устройств. На вкладке **Конфигурация Modbus Slave** (см. рис. 8.6) для каждого устройства можно задать индивидуальный таймаут ответа;

Таймаут сокета – время ожидания сокетом входящих пакетов TCP/IP. При превышении таймаута происходит реинициализация сокета. Значение параметра может изменяться в зависимости от пропускной способности сети, но в большинстве случаев рекомендуется оставлять значение по умолчанию;

Автоподключение – при **отсутствии** галочки, не ответившее slave-устройство исключается из дальнейшего опроса***. Настоятельно рекомендуется*** всегда включать эту опцию.

3. Нажмите **ПКМ** на компоненте **Modbus TCP Master** и добавьте компонент **Modbus TCP Slave**, расположенный во вкладке **Промышленные сети/Modbus/Слейв Modbus TCP**. Число компонентов должно соответствовать числу slave-устройств. ***Обратите внимание***, что версия компонента не должна превышать версию **target-файла** СПК. Для отображения предыдущих версий компонента поставьте галочку **Отображать все версии**. См. рекомендации в приложении Г.

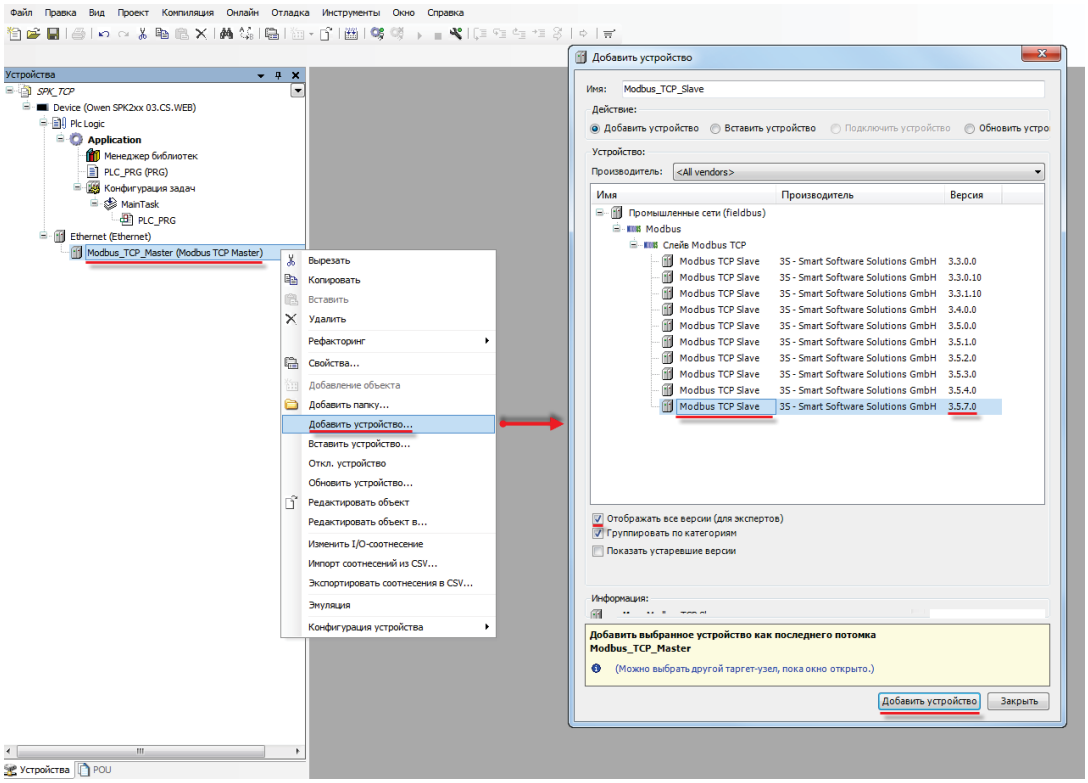


Рис. 8.5. Добавление компонента **Modbus TCP Slave**

В настройках компонента на вкладке **Общее** укажите **IP-адрес** slave-устройства, используемый **порт** (обычно выбирается порт **502**) и **адрес устройства** в сети Modbus (**Slave ID**). При необходимости можно указать индивидуальный **таймаут ответа** – он будет иметь приоритет по сравнению с установленным в настройках **Modbus TCP Master** (см. рис. 8.4).

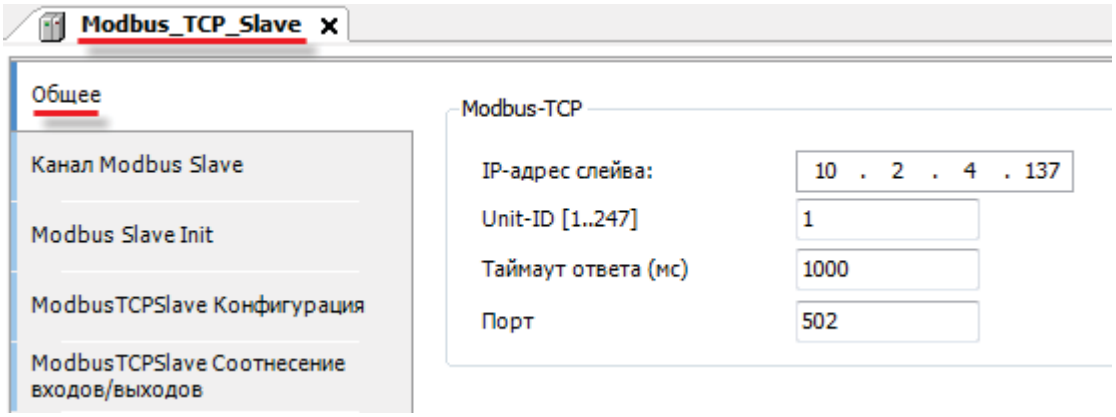


Рис. 8.6. Настройки компонента **Modbus TCP Slave**

Остальные вкладки (**Канал Modbus Slave** и **Соотнесение входов/выходов**) описаны в п. 4.2, пп. 3.

8.3. Настройка СПК в режиме Modbus TCP Slave

1. Нажмите **ПКМ** на компонент **Device** и добавьте компонент **Ethernet**, расположенный во вкладке **Промышленные сети/Ethernet-адаптер**. ***Обратите внимание***, что версия компонента не должна превышать версию **target-файла** СПК. Для отображения предыдущих версий компонента поставьте галочку **Отображать все версии**. См. рекомендации в приложении Г.

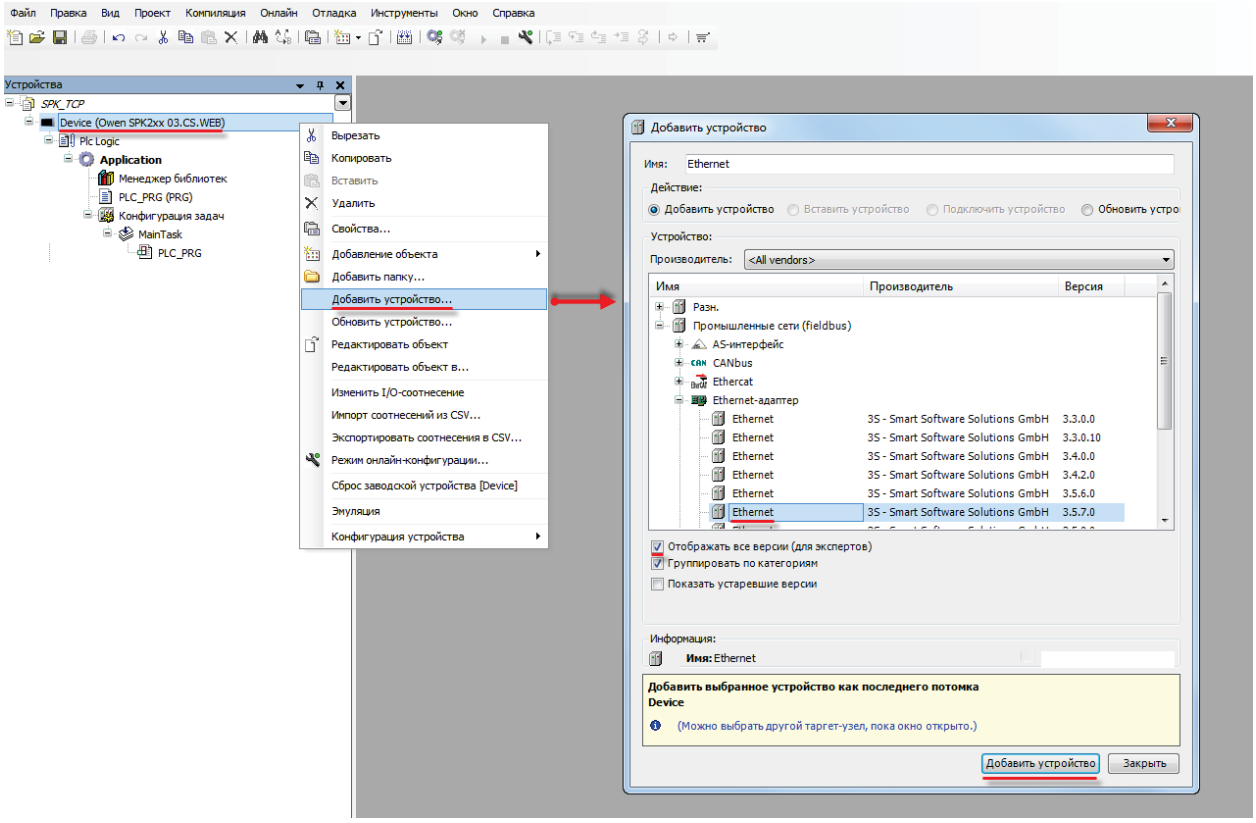


Рис. 8.7. Добавление компонента **Ethernet**

В настройках компонента на вкладке **Конфигурация Ethernet** необходимо указать IP-адрес СПК, маску подсети и IP-адрес шлюза. Обратите внимание, что эти настройки должны соответствовать сетевым параметрам, указанным в Конфигураторе СПК.

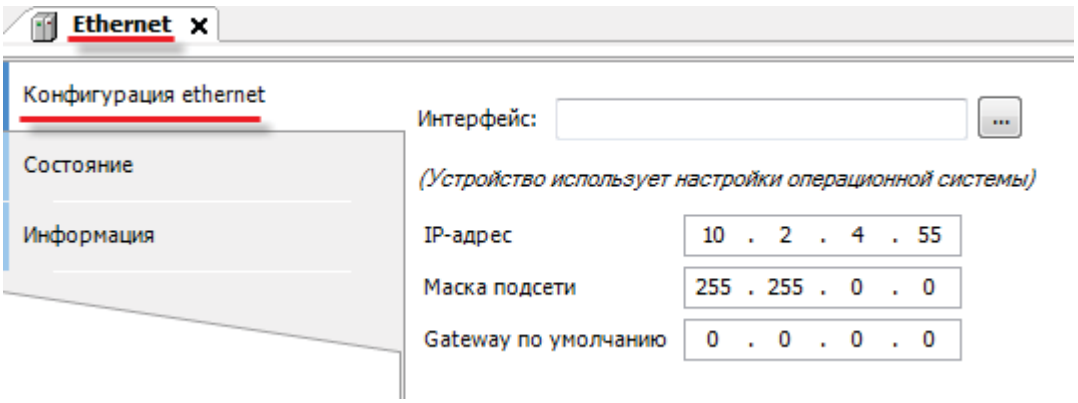


Рис. 8.8. Настройки компонента **Ethernet**

2. Нажмите **ПКМ** на компонент **Ethernet** и добавьте компонент **Modbus TCP Slave Device**, расположенный во вкладке **Промышленные сети/Modbus/Слэйв-устройство Modbus TCP**.

Обратите внимание, что версия компонента не должна превышать версию **target-файла** СПК. Для отображения предыдущих версий компонента поставьте галочку **Отображать все версии**. См. рекомендации в приложении Г.

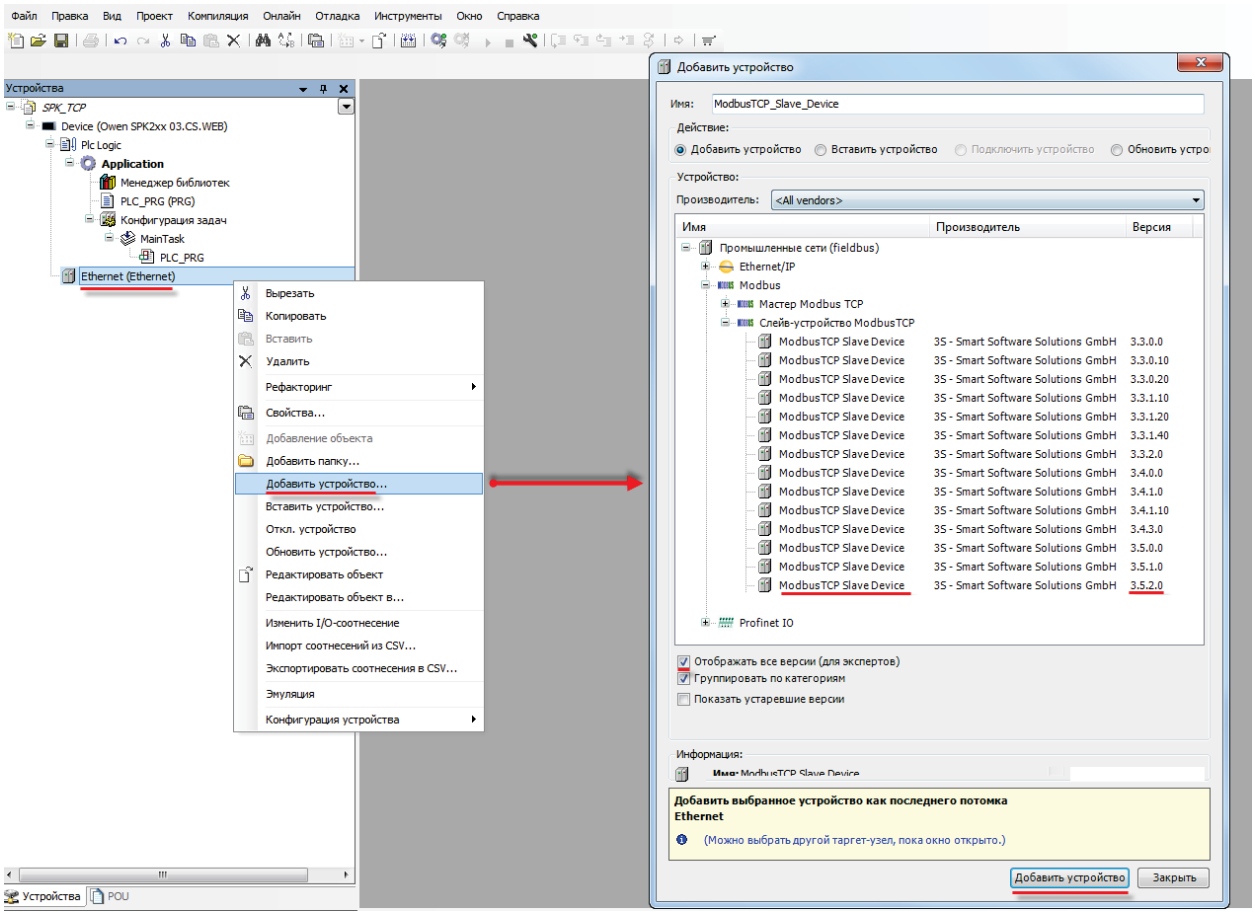


Рис. 8.9. Добавление компонента **Modbus TCP Slave Device**

На вкладке **Страница конфигурации** укажите настройки slave-устройства:

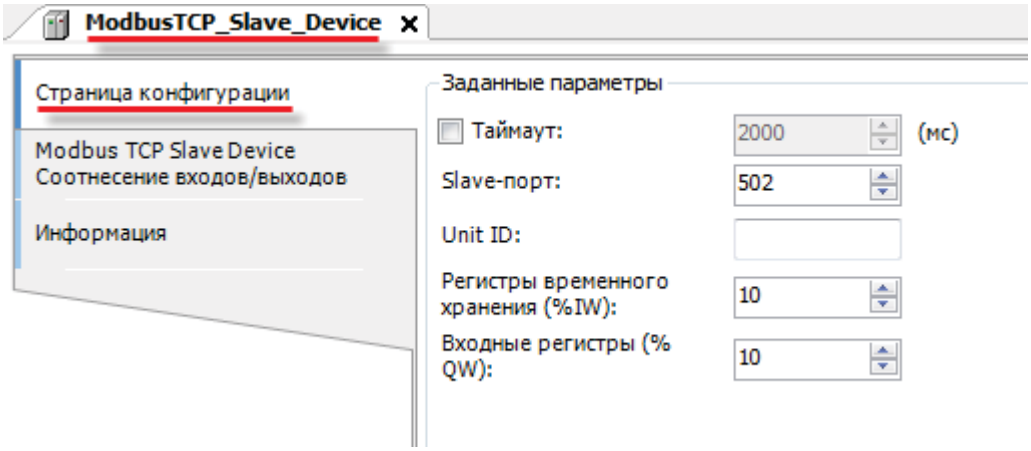


Рис. 8.10. Настройки компонента **Modbus TCP Slave Device**

Таймаут – время ожидания (в мс) запроса от master-устройства. Если за это время запроса не происходит, то данные в регистрах обнуляются. ***При отсутствии галочки*** обнуления данных ***не происходит***;

Slave-порт – используемый для связи порт. Обычно выбирается порт **502**;

ID – адрес, который будет назначен данному slave-устройству;

Регистры хранения – число регистров хранения (**holding registers**) для данного slave-устройства;

Входные регистры – число входных регистров (**input registers**) для данного slave-устройства.

Обратите внимание на различия входных регистров и регистров хранения (см. табл. 2.1).

Область данных	Обозначение	Тип данных	Тип доступа	Изменение из программы
Holding Registers (Регистры хранения)	4x (%IW)	WORD	чтение/запись	невозможно
Input Registers (Регистры ввода)	3x (%QW)	WORD	только чтение	возможно

Обратите внимание, что область памяти **1x** (**Discrete Inputs**) наложена на **3x** (**Input Registers**), а область **0x** (**Coils**) – на **4x** (**Holding Registers**). Поддержаны все функции работы с битами.

Настройки вкладки **Соотнесение входов/выходов** описаны в п. 4.3, пп. 2.

8.4. Пример: СПК207 (master) + СПК207 (slave)

Рассмотрим пример настройки обмена по протоколу **Modbus TCP** между двумя контроллерами **СПК207**, один из которых выполняется функцию **master**, а другой – **slave**.

Формулировка задачи: с помощью СПК207 (master) выполнить следующие действия:

1. считать из СПК207 (slave) переменную типа **BOOL**;
2. считать из СПК207 (slave) переменную типа **WORD**;
3. записать в СПК207 (slave) переменную типа **REAL**;
4. записать в СПК207 (slave) переменную типа **STRING**.

Структурная схема примера приведена на рис. 8.11:

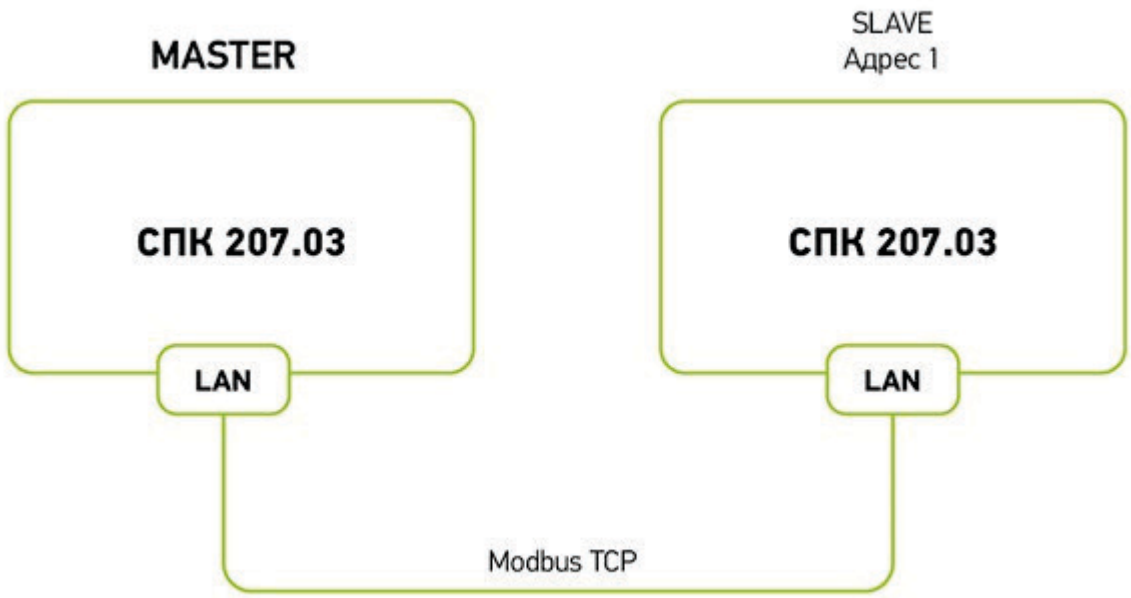


Рис. 8.11. Структурная схема примера **Стандартные функции конфигурирования Modbus TCP (СПК+СПК)**

Пример создан в среде **CODESYS 3.5 SP7 Patch4** и подразумевает запуск на **СПК207.03.CS(-WEB)**.

Пример доступен для скачивания: Example_StandardConfiguration_TCP.projectarchive

Сетевые параметры модулей приведены в табл. 8.1.

Список переменных, используемых в примере, приведен в табл. 8.2.

Табл. 8.1. Сетевые параметры СПК

Параметр	**СПК207 (master)**	**СПК207 (slave)**
IP-адрес	10.2.5.206	10.2.4.137
Адрес (Slave ID)	-	1
Маска подсети	255.255.0.0	
Шлюз	0.0.0.0	

Табл. 8.2. Список переменных программы

		СПК207 (master)	**СПК207 (slave)**	
Переменная	**Тип**	**Функция**	**Область памяти**	**Регистры**
xVar	BOOL	Read Discrete Inputs	Input Registers	0
wVar	WORD	Read Input Registers	Input Registers	1
rRealValue	REAL	Write Multiple Registers	Holding Registers	0-1
sStringValue	STRING	Write Multiple Registers	Holding Registers	2-4

8.4.1. Настройка СПК207 (master)

1. Создайте новый проект **CODESYS** для **СПК207** с программой **PLC_PRG** на языке **CFC**.

2. Добавьте в проект объединение с именем **Real_Word**:

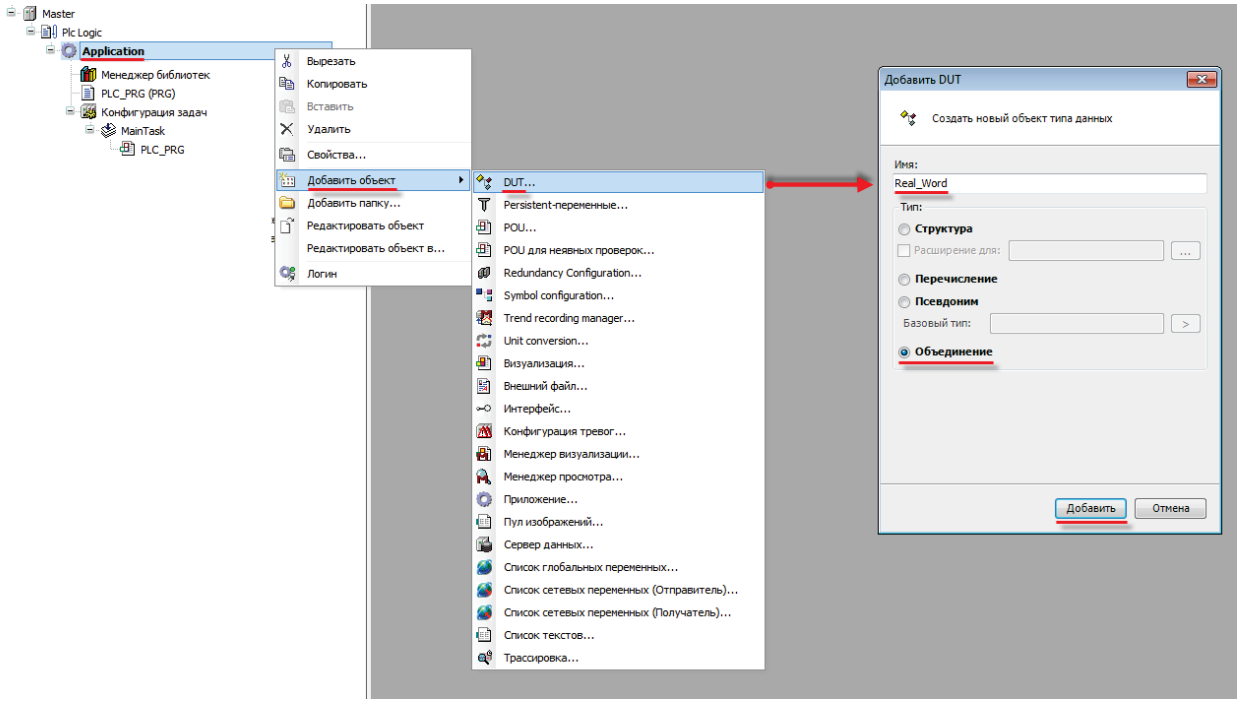


Рис. 8.12. Добавление в проект объединения

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

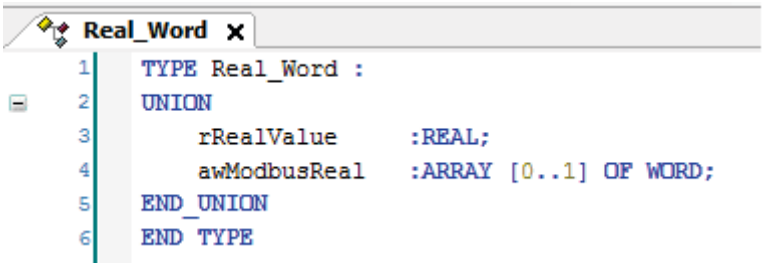


Рис. 8.13. Объявление переменных объединения

3. Добавьте в проект объединение с именем **String_Word**.

В объединении объявите переменную **sStringValue** типа **STRING** (с ограничением на размер в 6 символов) и массив **awModbusString** типа **WORD**, содержащий три элемента (**STRING** сможет содержать до 6 символов, поскольку каждый **WORD** может содержать два символа):

```
TYPE String_Word :
UNION
    awModbusString      :ARRAY [0..2] OF WORD;
    sStringValue        :STRING;
END_UNION
END_TYPE
```

Рис. 8.14. Объявление переменных объединения

4. Объявите в программе переменную **xVar** типа **BOOL** и **wVar** типа **WORD**, а также экземпляр объединения **Real_Word** с именем **_REAL_TO_2WORD** и экземпляр объединения **String_Word** с именем **_STRING_TO_3WORD**.

PLC_PRG

```
PROGRAM PLC_PRG
VAR
    xVar                :BOOL;
    wVar                :WORD;
    _REAL_TO_2WORD      :Real_Word;
    _STRING_TO_3WORD    :String_Word;
END_VAR
```

Рис. 8.15. Объявление переменных программы

5. Код программы будет выглядеть следующим образом:

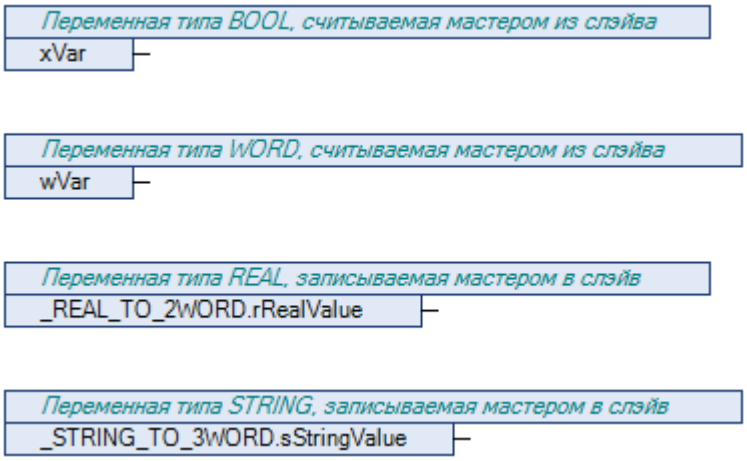


Рис. 8.16. Код программы на языке **CFC**

6. Добавьте в проект компонент **Ethernet**:

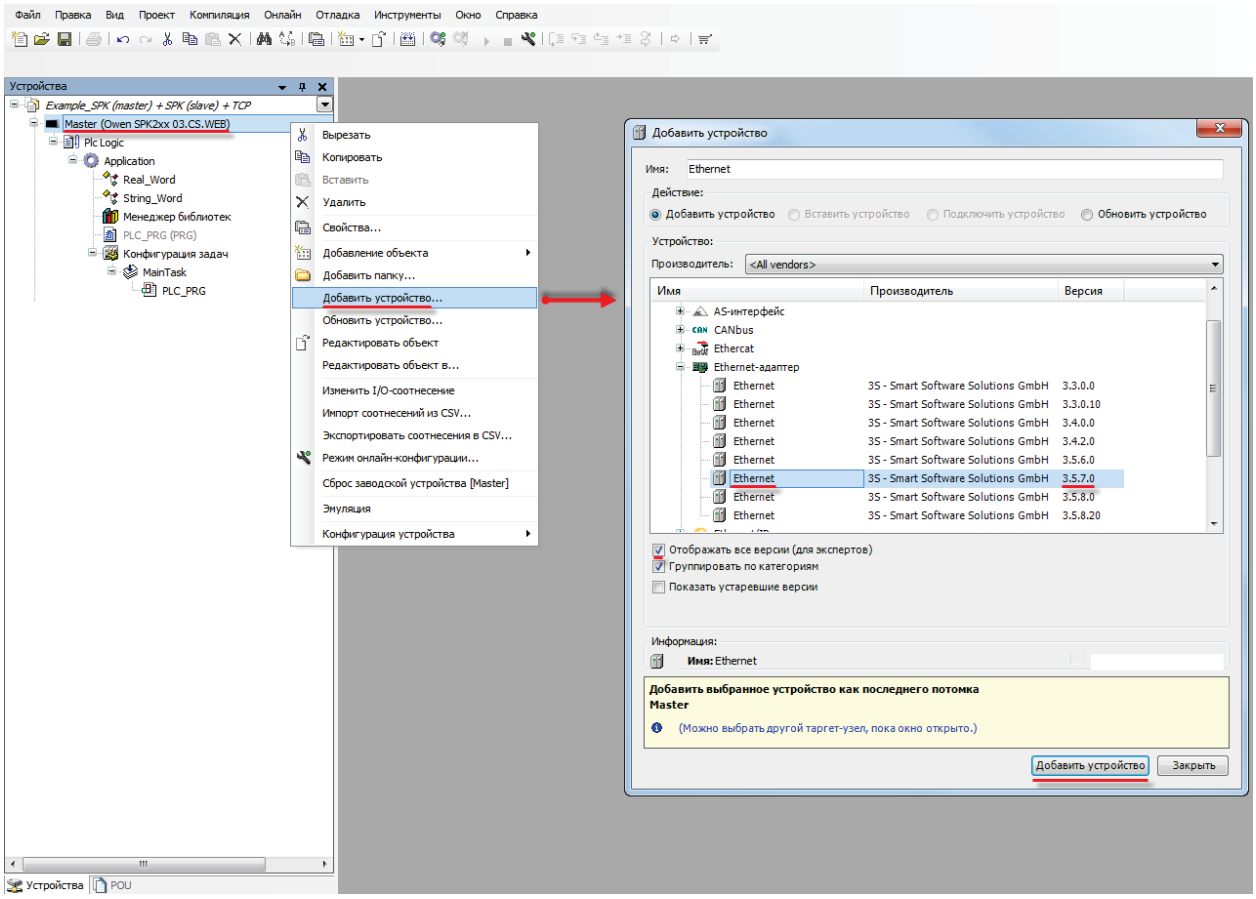


Рис. 8.17. Добавление устройства **Modbus COM**

Во вкладке **Конфигурация Ethernet** укажите сетевые настройки в соответствии с табл. 8.1.

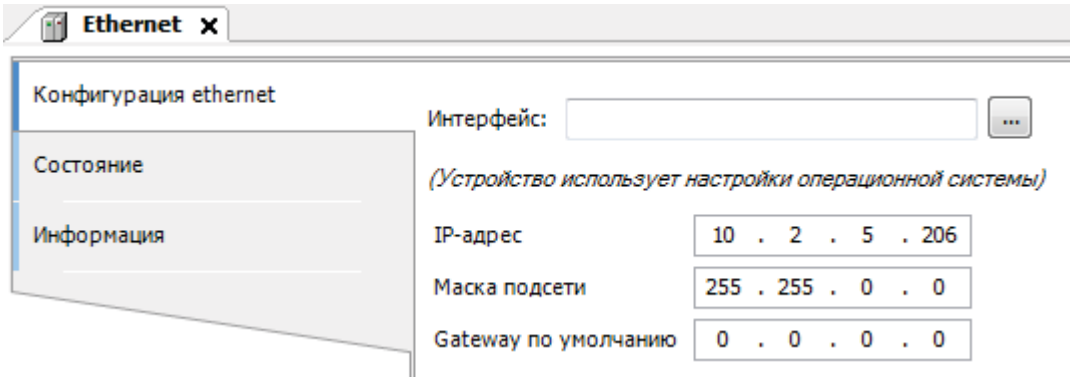


Рис. 8.18. Настройки **Ethernet**

7. В компонент **Ethernet** добавьте компонент **Modbus TCP Master**:

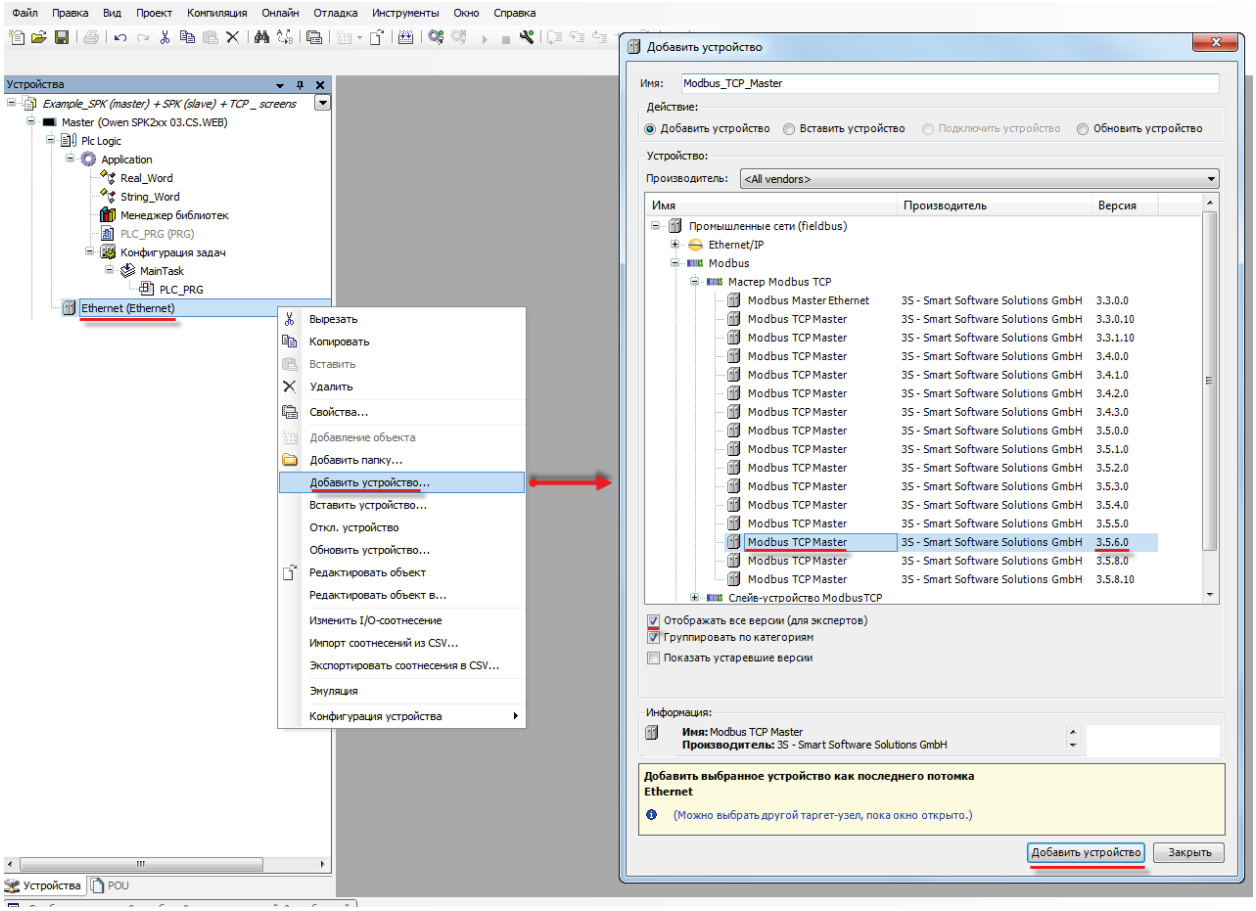


Рис. 8.19. Добавление компонента **Modbus TCP Master**

В настройках компонента поставьте галочку **Автоподключение**.

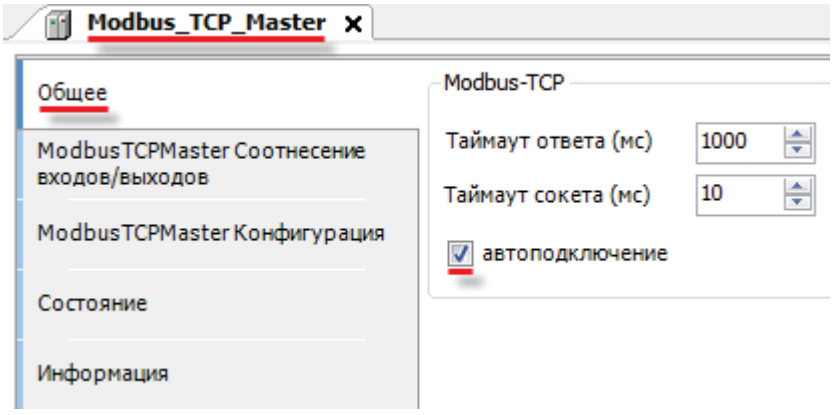


Рис. 8.20. Настройка компонента **Modbus TCP Master**

8. В **Modbus TCP Master** добавьте компонент **Modbus TCP Slave**:

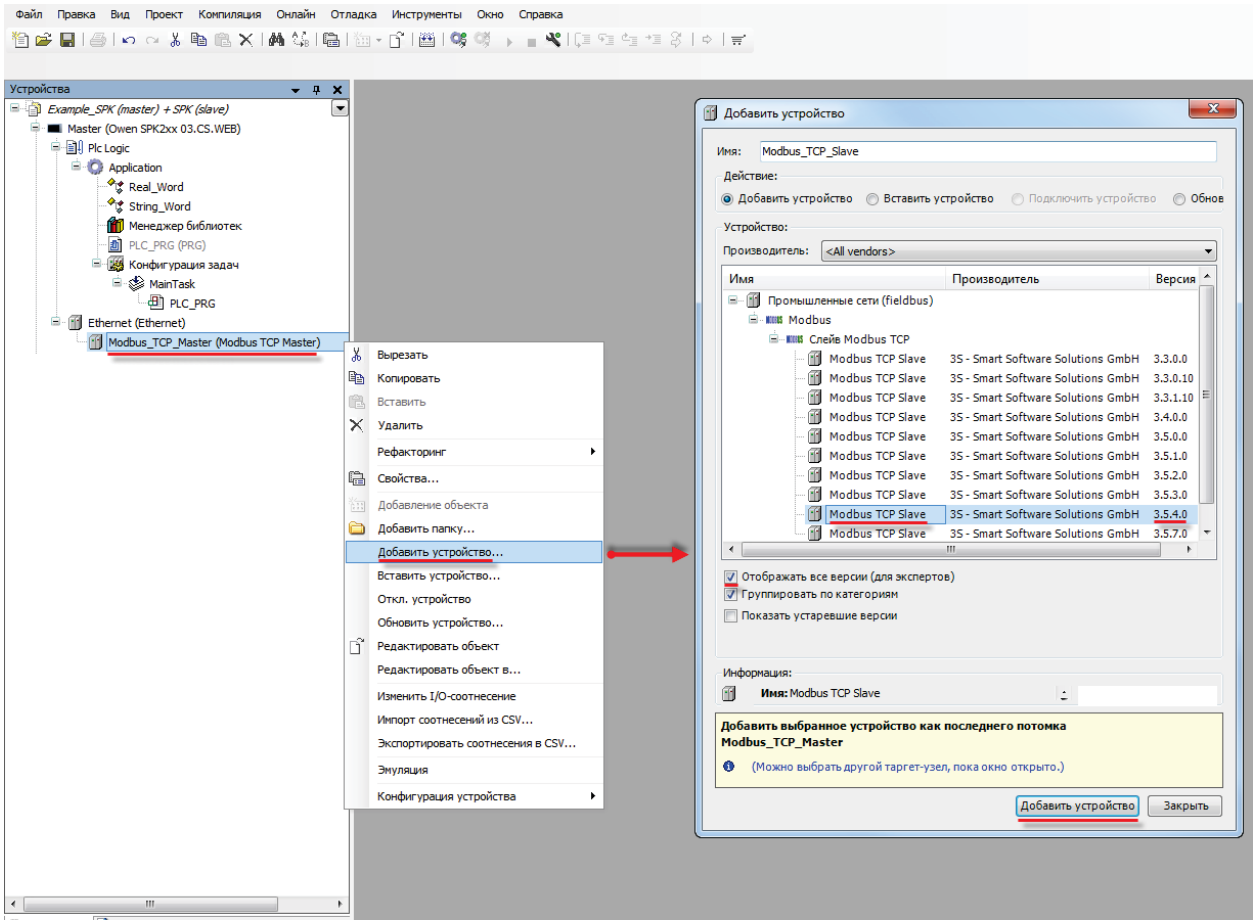


Рис. 8.21. Добавление компонента **Modbus TCP Slave** в проект

В настройках компонента на вкладке **Общее** укажите IP-адрес slave-устройства и его адрес в сети Modbus (**Slave ID**) в соответствии с табл. 8.1.

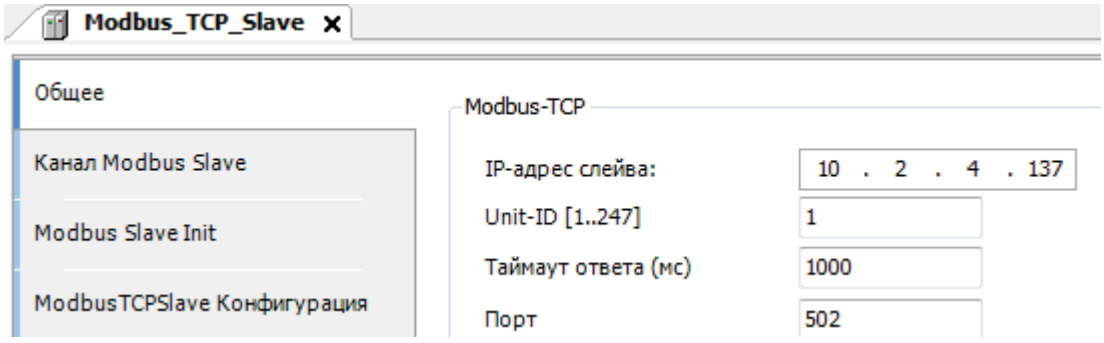


Рис. 8.22.Настройка компонента **Modbus TCP Slave**

На вкладке **Канал Modbus Slave** создайте 4 канала – по одному на каждую из переменных. Переменные типа **BOOL** и **WORD** СПК (master) будет считывать с помощью функций **Read Discrete Inputs** и **Read Inputs Registers** из ***Input*** регистров СПК (slave). Переменные типа **REAL** и **STRING** СПК (master) будет записывать с помощью функции **Write Multiple Registers** в ***Holding*** регистры СПК (slave). Обратите внимание, что ***Input*** и ***Holding*** регистры – разные области памяти СПК (slave).

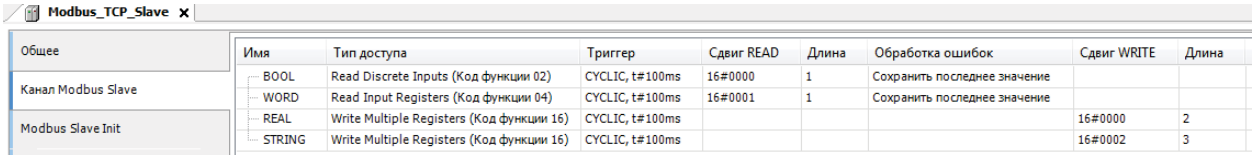


Рис. 8.23. Настройка каналов **Modbus Slave**

На вкладке **ModbusTCPslave Соотнесение входов/выходов привяжите** к каналам переменные программы в соответствии с табл. 8.2. Не забудьте у параметра **Всегда обновлять переменные** выставить значение **Включено 2**.

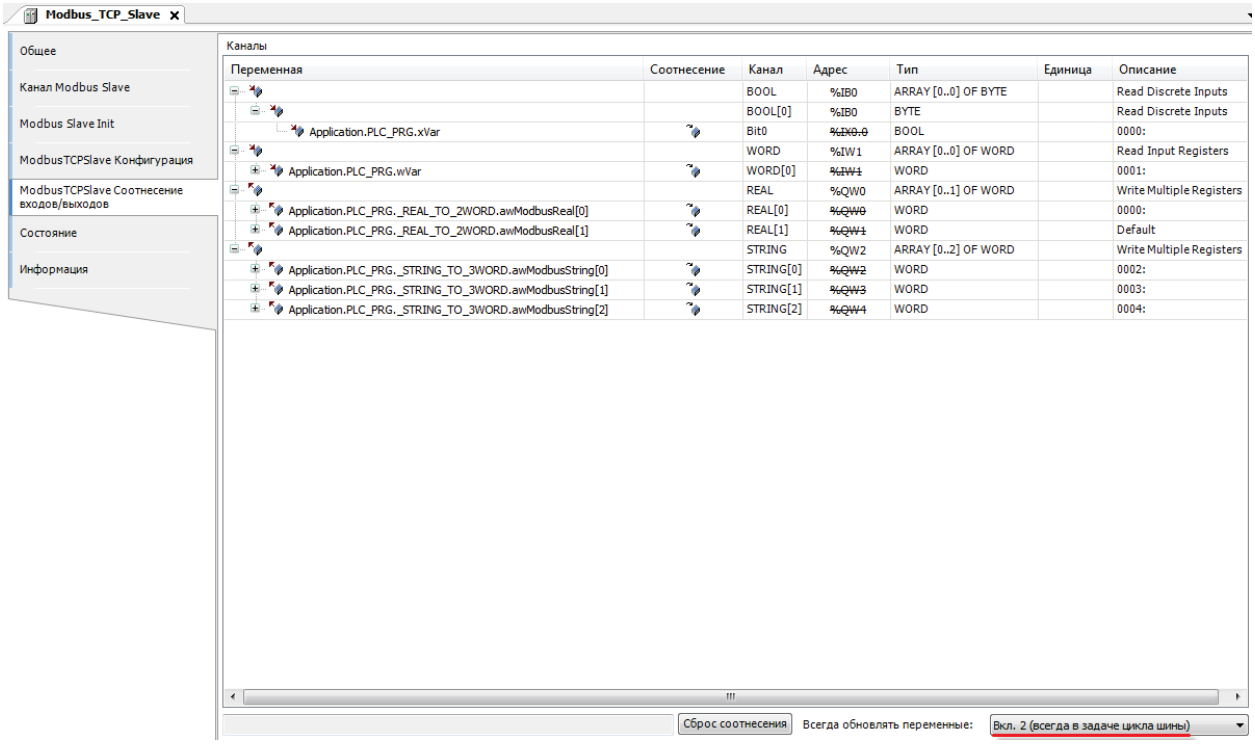


Рис. 8.24. Привязка переменных к каналу

На этом настройка **СПК (master)** завершена.

8.4.2. Настройка СПК207 (slave)

1. В проект CODESYS, содержащий СПК (master), добавьте контроллер СПК (slave).

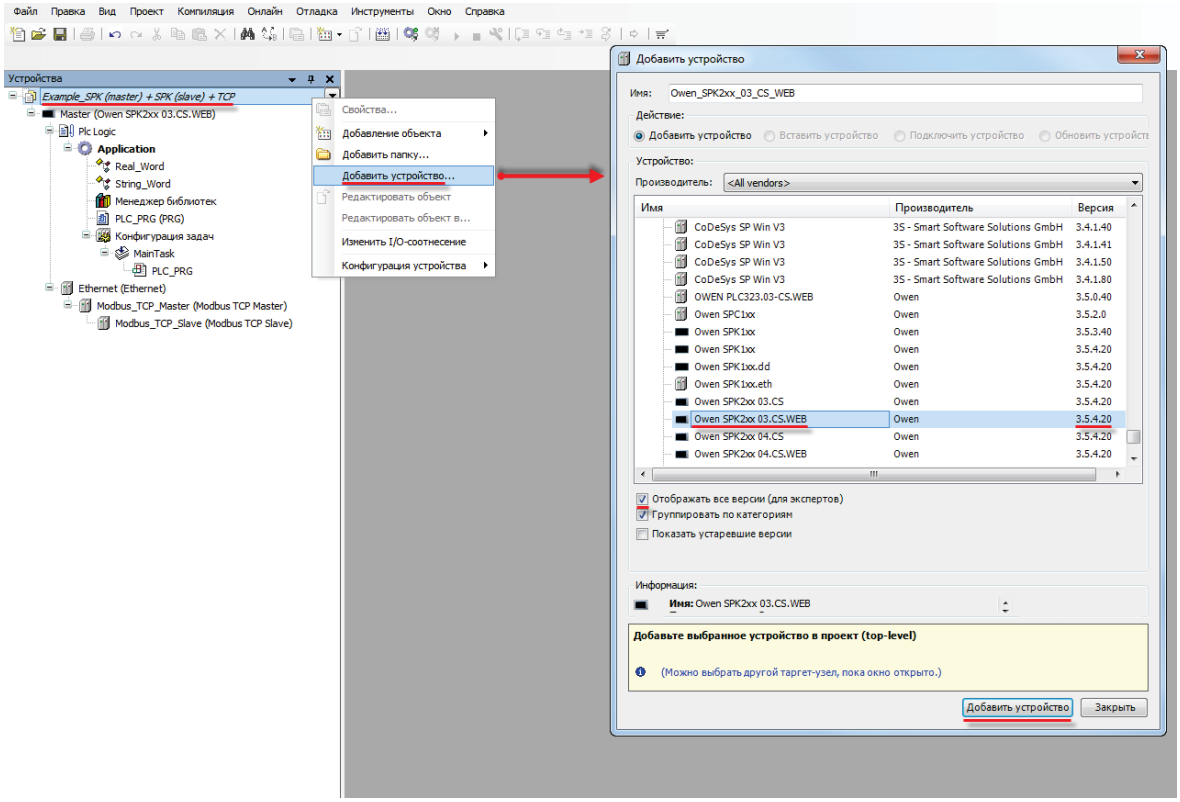


Рис. 8.25. Добавление СПК (slave) в проект **CODESYS**

В приложение **Application** устройства СПК (slave) добавьте программу с именем **PLC_PRG** на языке **CFC** и компонент **Конфигурация задач**. Привяжите программу **PLC_PRG** к задаче **MainTask**. После этого дерево проекта будет выглядеть следующим образом:

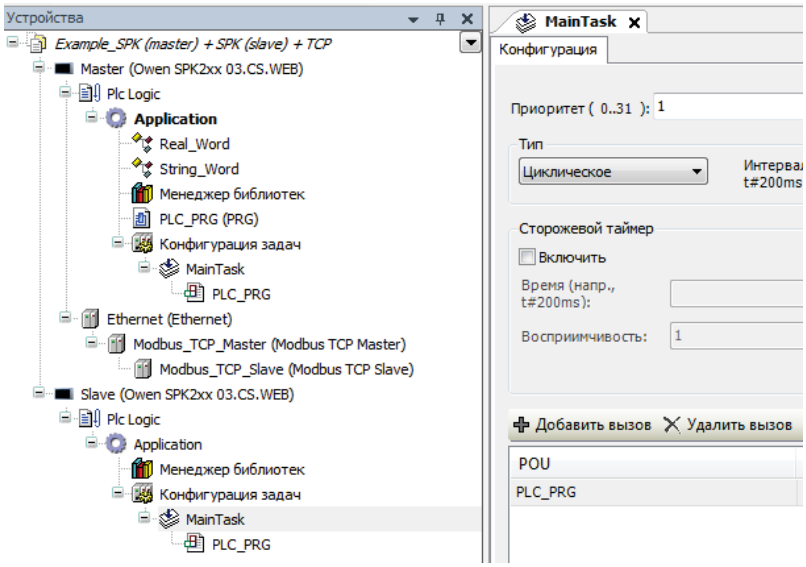


Рис. 8.26. Внешний вид дерева проекта после привязки программы к задаче

2. Добавьте в проект объединение с именем **Real_Word**:

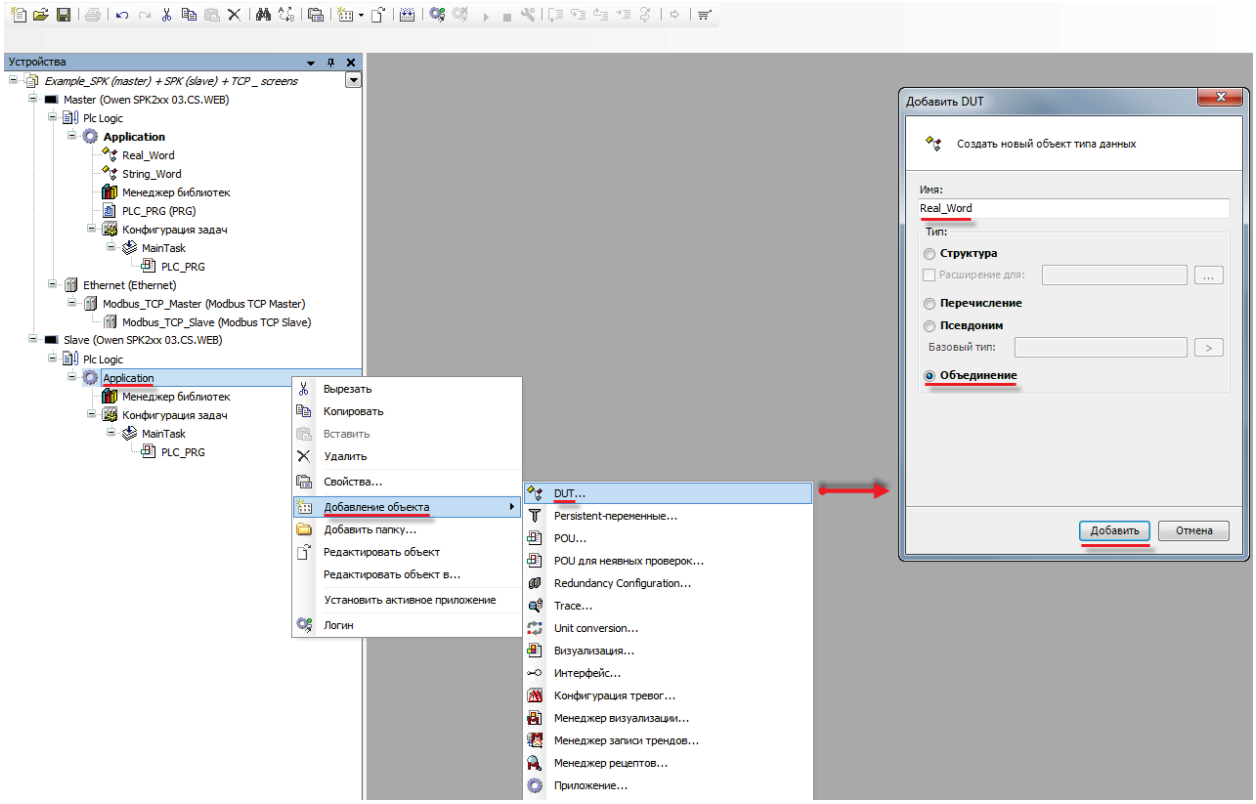


Рис. 8.27. Добавление объединения в проект

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

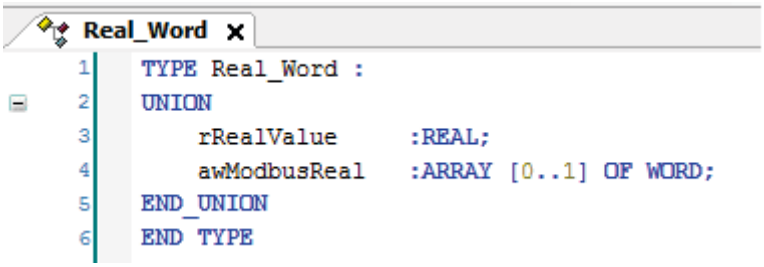


Рис. 8.28. Объявление переменных объединения

3. Добавьте в проект объединение с именем **String_Word**.

В объединении объявите переменную **sStringValue** типа **STRING** (с ограничением на размер в 6 символов) и массив **awModbusString** типа **WORD**, содержащий три элемента (**STRING** сможет содержать до 6 символов, поскольку каждый **WORD** может содержать два символа):

```
TYPE String_Word :
UNION
    awModbusString      :ARRAY [0..2] OF WORD;
    sStringValue        :STRING;
END_UNION
END_TYPE
```

Рис. 8.29. Объявление переменных объединения

4. Объявите в программе переменную **xVar** типа **BOOL** и **wVar** типа **WORD**, а также экземпляр объединения **Real_Word** с именем **_2WORD_TO_REAL** и экземпляр объединения **String_Word** с именем _ **3WORD_TO_STRING**.

```
PROGRAM PLC_PRG
VAR
    xVar                :BOOL;
    wVar                :WORD;
    _2WORD_TO_REAL      :Real_Word;
    _3WORD_TO_STRING    :String_Word;
END VAR
```

Рис. 8.30. Объявление переменных программ

5. Код программы будет выглядеть следующим образом:

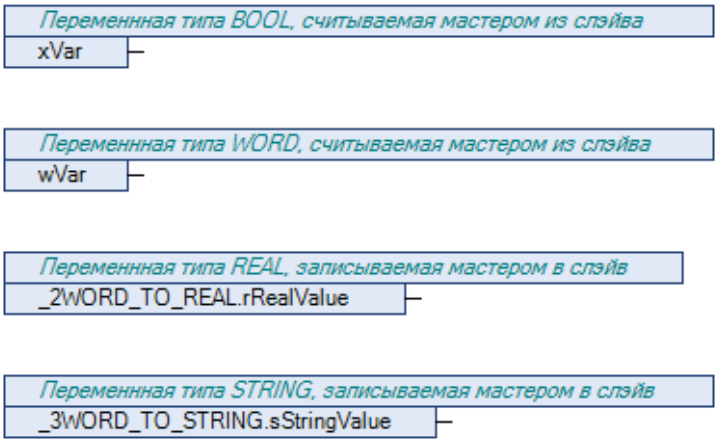


Рис. 8.31. Код программы на языке **CFC**

6. Добавьте в проект компонент **Ethernet**:

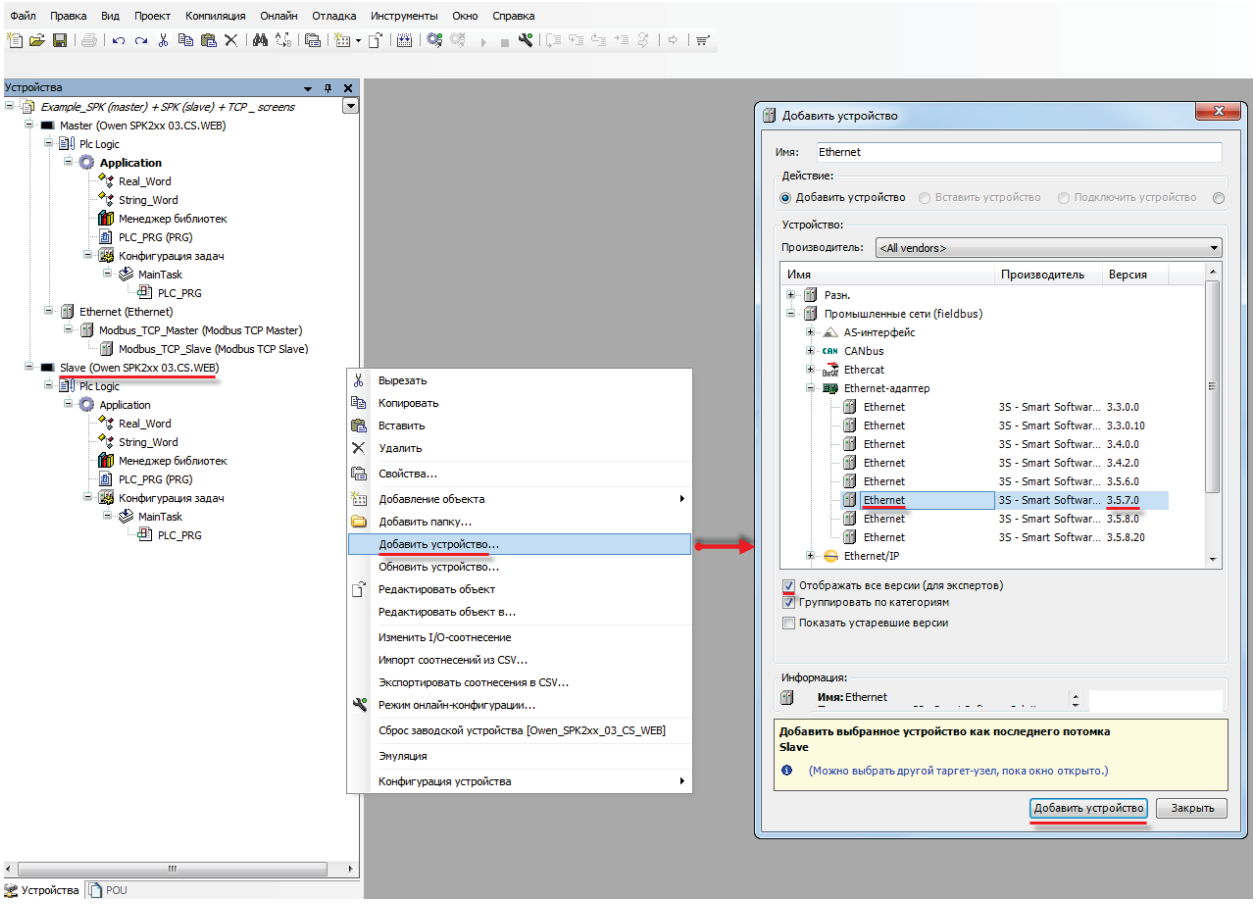


Рис. 8.32. Добавление в проект устройства **Ethernet**

Во вкладке **Конфигурация Ethernet** укажите сетевые настройки в соответствии с табл. 8.1.

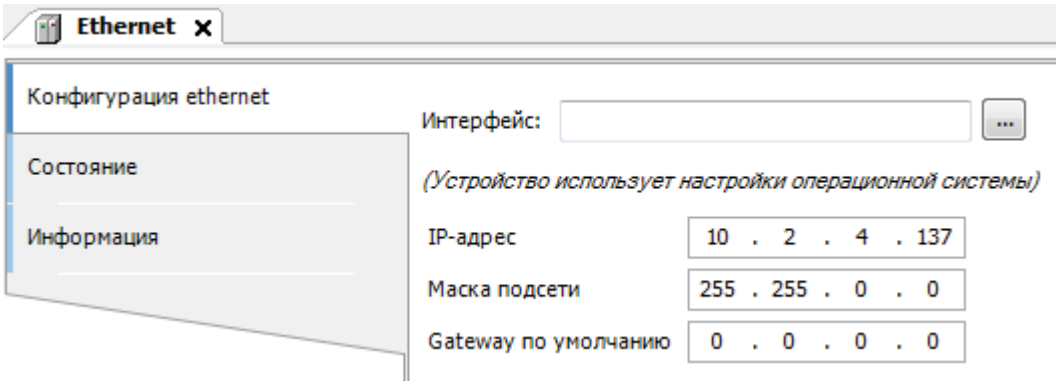


Рис. 8.33. Настройки компонента **Ethernet**

7. В компонент **Ethernet** добавьте компонент **Modbus TCP Slave Device**:

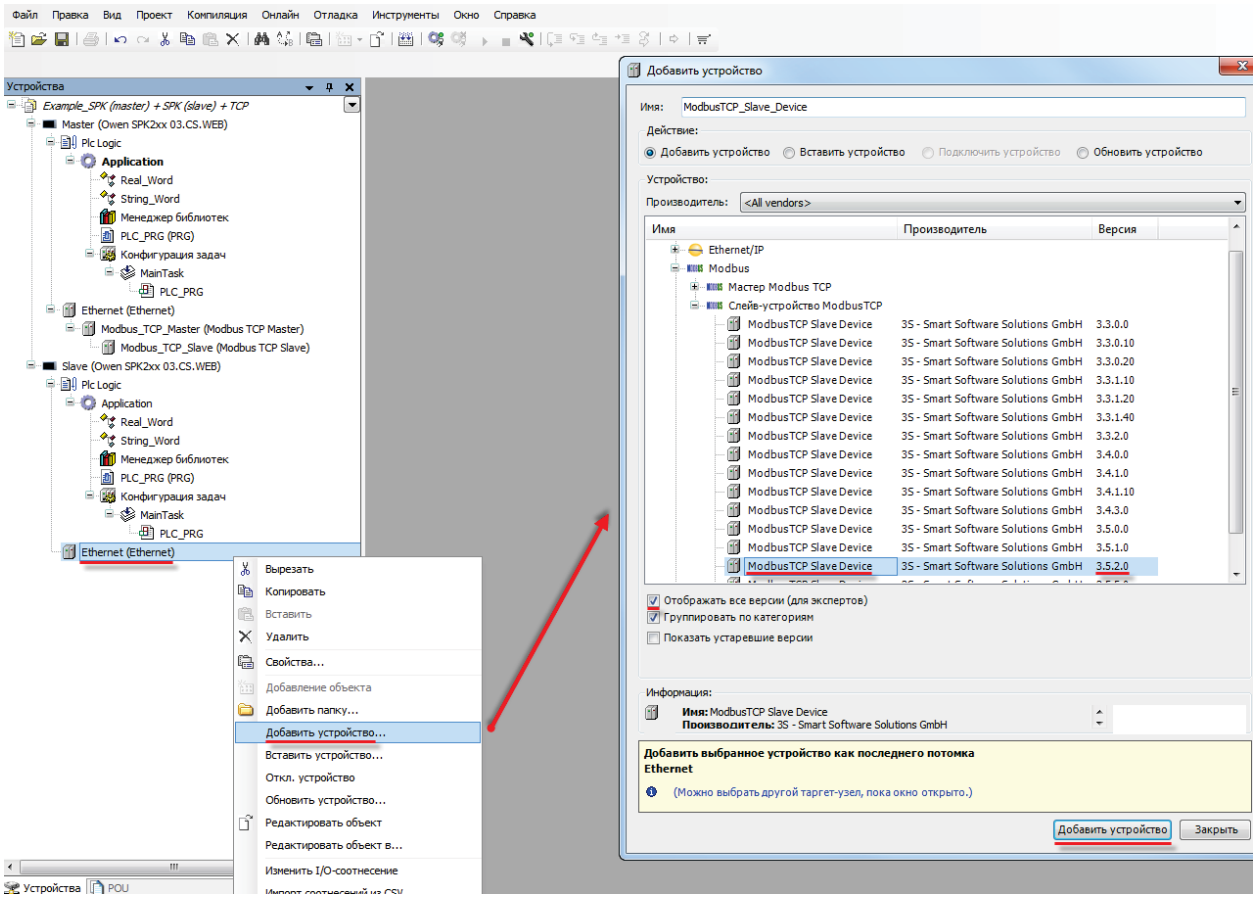


Рис. 8.34. Добавление компонента **Modbus TCP Slave Device**

В настройках компонента на вкладке **Страница конфигурации** укажите адрес slave-устройства (**1** в соответствии с табл. 8.1) и используемый порт.

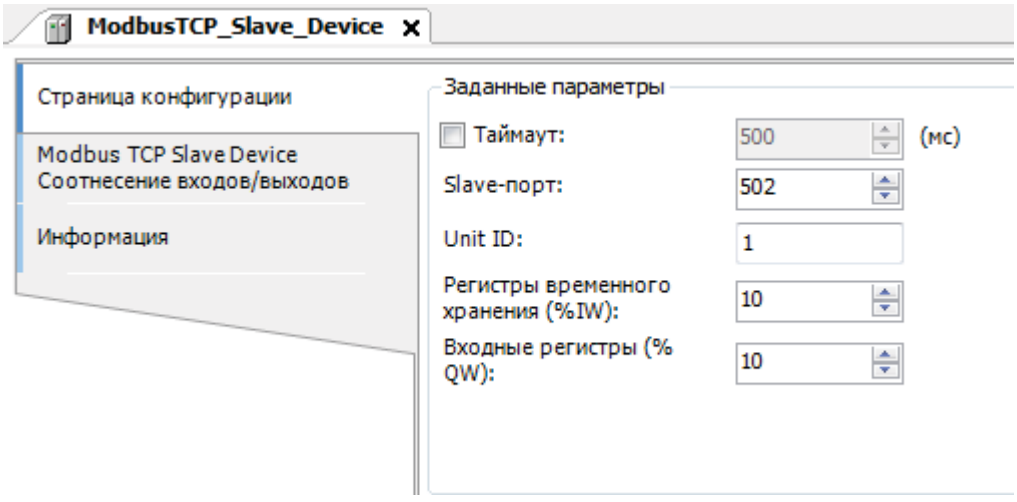


Рис. 8.35. Настройки компонента **Modbus TCP Slave Device**

На вкладке **Modbus TCP Slave Device Соотнесение входов/выходов** привяжем к регистрам переменные программы в соответствии с табл. 8.2. Не забудьте у параметра **Всегда обновлять переменные** выставить значение **Включено 2**.

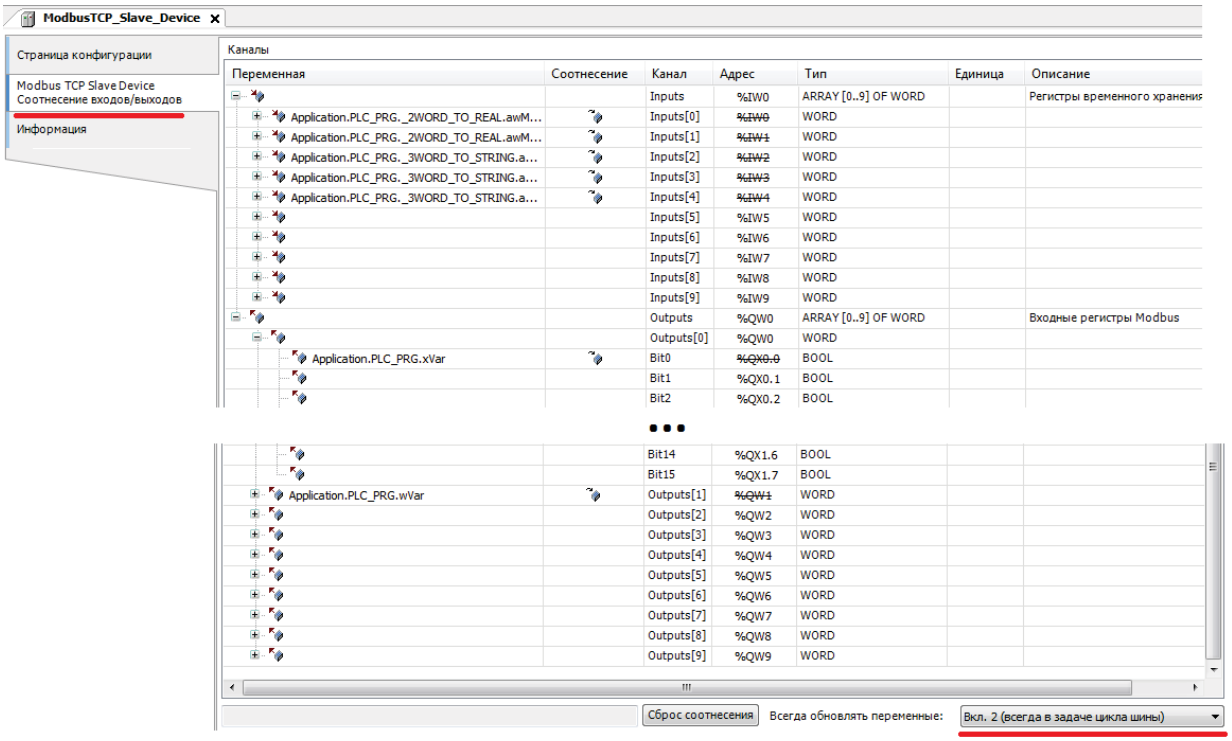


Рис. 8.36. Привязка переменных к регистрам slave-устройства

На этом настройка **СПК (slave)** завершена.

8.4.3. Запуск примера

1. Задайте обоим СПК сетевые настройки в соответствии с табл. 8.1.

2. Подключите оба СПК к одной локальной сети (или напрямую друг к другу).

3. Загрузите проекты в каждый из СПК и запустите их.

4. Изменяйте в программе СПК (slave) значения переменных **xVar** и **wVar** и наблюдайте соответствующие изменения в программе СПК (master).

Изменяйте в программе СПК (master) значения переменных **rRealValue** и **sStringValue** и наблюдайте соответствующие изменения в программе СПК (slave).

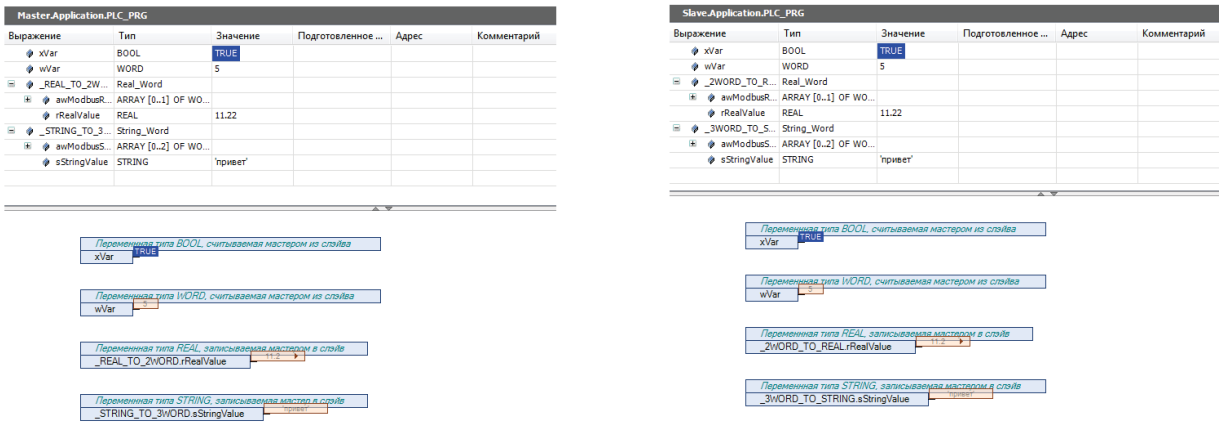


Рис. 8.37. Выполнение программ в режиме **Online**

9. FAQ

Данная глава содержит наиболее часто возникающие вопросы по настройке обмена по **Modbus** для СПК и ответы на них. При необходимости ответ представляет собой ссылку на соответствующий пункт документа.

9.1. Что делать, если не удается наладить обмен по Modbus?

1. Проверьте (прозвоните) линию связи. Проверьте распиновку кабеля (см. п. 2.3.2, распиновка остальных портов приведена в **РЭ**). Убедитесь, что контакты **A** и **B** (или **RXD** и **TXD**) не перепутаны местами.

2. Проверьте выбранный режим работы порта в конфигураторе СПК.

3. Проверьте используемый номер COM-порта в **CODESYS** – ***обратите внимание,*** что нумерация в **CODESYS** не соответствует нумерации на корпусе СПК (см. п. 2.3.1).

4. Проверьте соответствие сетевых настроек СПК и подключаемых приборов (скорость обмена, количество стоп-бит и т.д.).

5. Проверьте адресацию регистров и их тип. ***Обратите внимание***, в какой системе указываются номера регистров slave-устройства (в десятичной или шестнадцатеричной).

6. Проверьте, нет ли разрывов в карте регистров slave-устройства (при использовании группового опроса).

7. Убедитесь, что в сети находится только одно мастер-устройство (для **Modbus RTU/ASCII**).

8. Убедитесь, что в сети нет slave-устройств с одинаковыми адресами.

9. При использовании модулей **Mx110** – убедитесь, что произведена настройка модулей с помощью программы Конфигуратор Mx110.

10. При использовании стандартных средств конфигурирования – убедитесь, что на вкладке привязки переменных для параметра **Всегда обновлять переменные** установлено значение **Вкл. 2 (Всегда в задаче цикла шины)**.

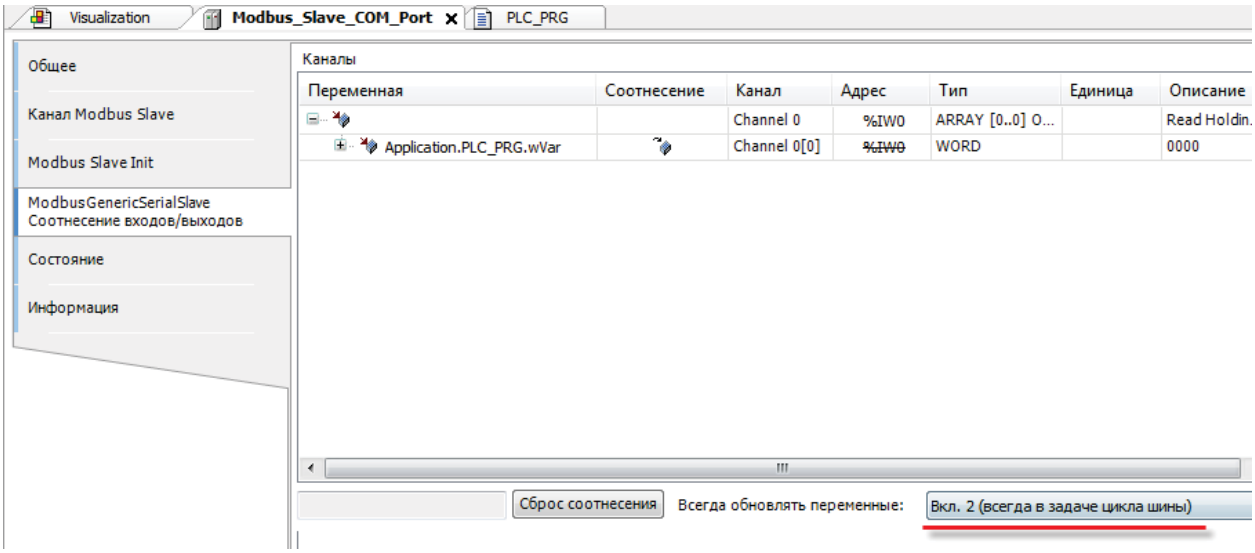


Рис. 9.1. Настройка параметра **Всегда обновлять переменные**

Если успешное выполнение всех вышеперечисленных пунктов не приведет к наладке обмена, то вам необходимо обратиться в техподдержку компании Овен. При обращении, пожалуйста, предоставьте следующую информацию и материалы:

- Модель и модификацию СПК, 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, 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).

9.2. Каким образом считать/передать значение с плавающей точкой (REAL)?

См. п. 4.4.

9.3. Каким образом считать/передать отрицательное значение (INT)?

Если необходимо считать отрицательное число, то после получения соответствующего **WORD** преобразуйте его в **INT** с помощью стандартной функции **WORD_TO_INT**.

Если необходимо передать отрицательное число, то преобразуйте его в **WORD** с помощью стандартной функции **INT_TO_WORD**. На устройстве, которое примет эти данные, необходимо произвести обратную операцию.

9.4. Как работать с примерами?

Примеры созданы в среде **CODESYS 3.5 SP7 Patch4** и подразумевают запуск на **СПК207.03**.

Некоторые примеры содержат проекты сразу для двух СПК. Чтобы выбрать, с каким из проектов работать в данный момент, нажмите **ПКМ** на **Application и** выберите команду **Установить активное приложение**.

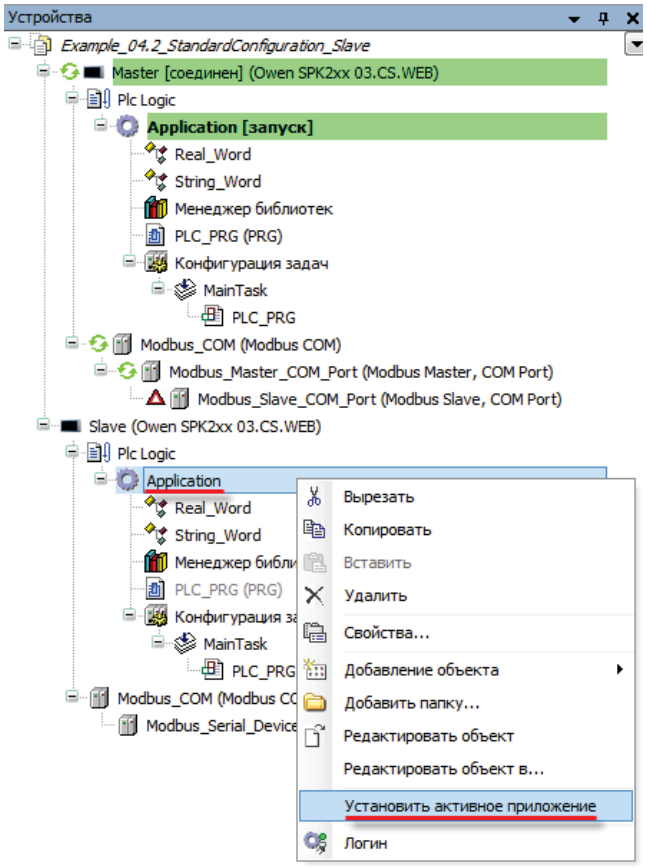


Рис. 9.2. Выбор активного приложения

9.5. Стандартные средства конфигурирования

9.5.1. Какие версии Modbus компонентов рекомендуются к использованию?

Таблица рекомендуемых версий компонентов Modbus приведена в приложении Г.

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

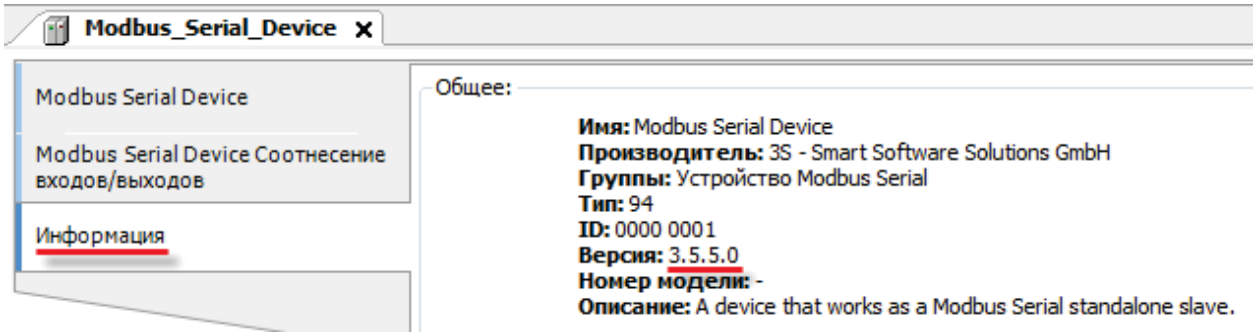


Рис. 9.2. Версия компонента **Modbus Slave Serial Device**

Чтобы изменить версию компонента, нажмите на нем **ПКМ** и выберите команду **Обновить устройство**.

9.5.2. [СПК - master] Как реализовать чтение/запись по триггеру?

При настройке канала для параметра **Trigger** выберите значение **Rising Edge**.

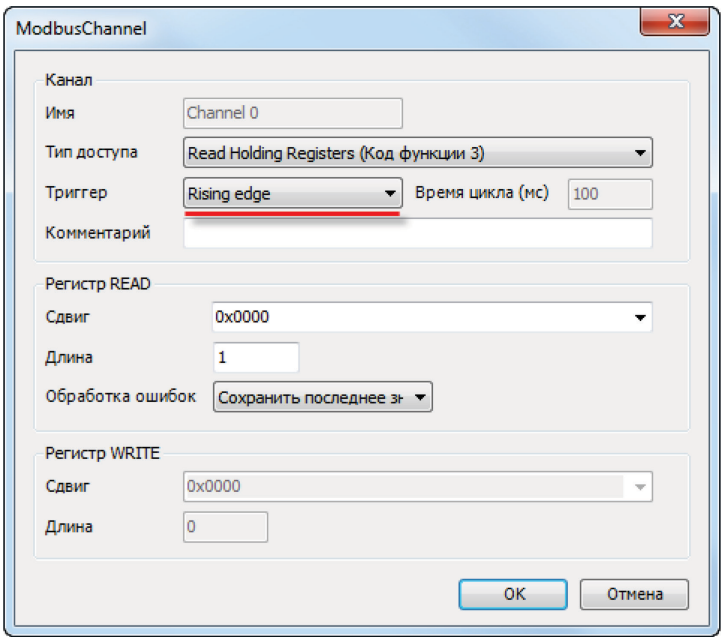


Рис. 9.3. Настройка опроса канала по триггеру

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

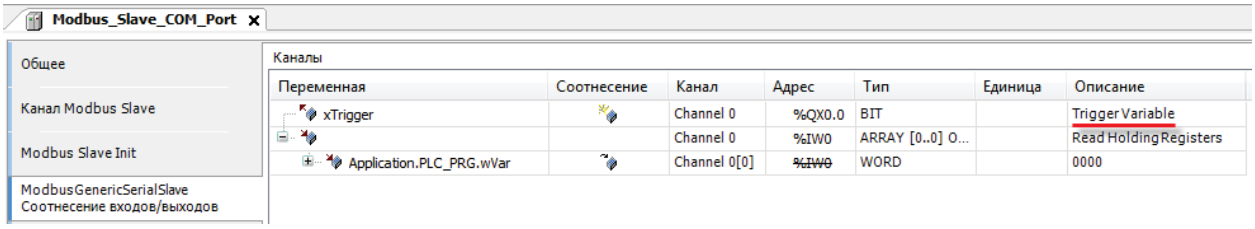


Рис. 9.4. Привязка триггерной переменной

9.5.3. [СПК – slave] Почему принятые значения сбрасываются в 0?

Если СПК используется в режиме **Slave**, а мастером является устройство, производящее запись по триггеру (например, панель оператора записывает введенное значение однократно, а не циклически), то необходимо отключить галочку **Задержка**:

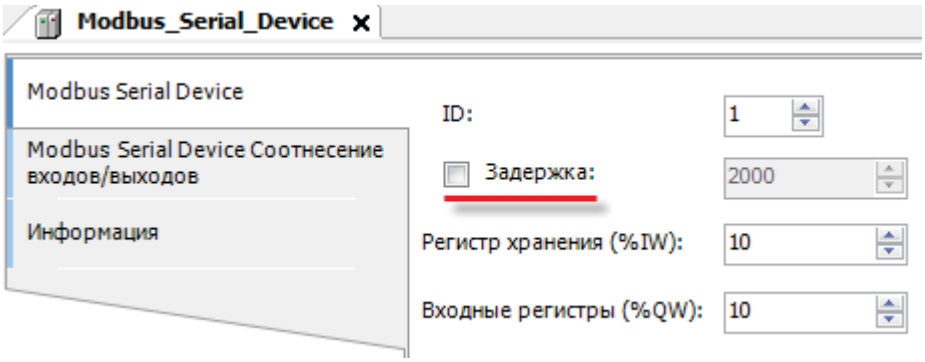


Рис. 9.5. Настройки СПК в режиме **Modbus Serial Device**

9.5.4. [СПК – slave] Как работать с Coils/Discrete Inputs?

В компоненте **Modbus Serial Device** битовые и регистровые области памяти наложены друг на друга: Coils – на Holding Registers, а Discrete Inputs – на Input Registers.

На рис. 9.5 показана настройка **Modbus Serial Device**. Пользователь может обратиться к нулевому holding регистру (к которому привязана переменная **wVar**), а может – к его конкретной ячейке (coil) [например, к нулевой – к ней привязана переменная **xVar**]. Соответственно, в каждом из случаев мастер-устройство должно использовать подходящие функции Modbus***. Обратите внимание***, что нельзя одновременно привязывать переменные и к регистрам, и к битам.

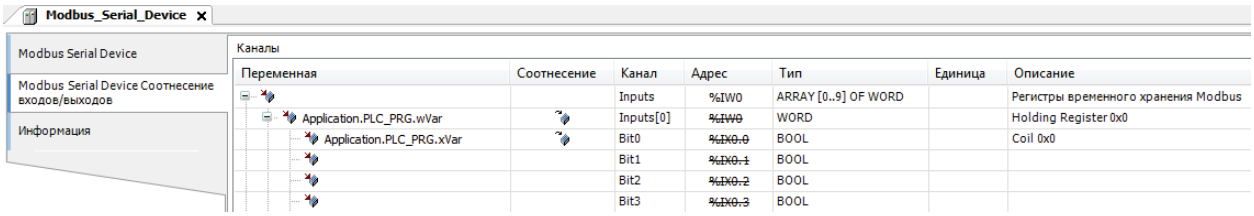


Рис. 9.6. Наложение областей памяти в **Modbus Serial Device**

9.5.5. [СПК – slave] 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 воспользуйтесь библиотекой ModbusSlave.

9.5.6. Как произвести диагностику обмена в программе?

См. п. 4.5. В случае использования шаблонов модулей Mx110 см. п. 3.3.

9.5.7. Как расшифровываются пиктограммы статуса обмена?

 - связь установлена;

 - ожидание ответа;

 - связь установлена, но в процессе обмена происходят ошибки (например, попытка обратиться к slave-устройству не поддерживаемой им функцией;

 - отсутствие связи.

Приложение

А. Список переменных шаблонов модулей Mx110

Список переменных всех шаблонов модулей приведен в табл. А. Подробное описание переменных приведено в **РЭ** на соответствующие модули.

Табл. А. Список переменных шаблонов модулей **Mx110**

Имя переменной	Описание	Тип данных	Тип доступа
1. Модули аналогового ввода MB110			
1.1. Модуль ввода аналоговых сигналов MB110-2A			
C_Time_1 (C_Time_2)	Циклическое время измерения входа 1 (2)	WORD	только чтение
Read_1 (Read_2)	Показание входа 1 (2) в представлении с плавающей точкой	REAL	только чтение
Stat_1 (Stat_2)	Статус измерения входа 1 (2) [код ошибки]	WORD	только чтение
1.2. Модуль скоростного ввода аналоговых сигналов MB110-2AC			
C_Time_1 (C_Time_2)	Циклическое время измерения входа 1 (2)	WORD	только чтение
Read_1 (Read_2)	Показание входа 1 (2) в представлении с плавающей точкой	REAL	только чтение
SRD_1 (SRD_2)	Статус измерения входа 1 (2) [код ошибки]	WORD	только чтение
1.3. Модуль ввода аналоговых сигналов MB110-8A			
C_Time_1…C_Time_8	Циклическое время измерения входов 1…8	WORD	только чтение
Read_1...Read_8	Показание входов 1…8 в представлении с плавающей точкой	REAL	только чтение
Stat_1…Stat_8	Статус измерения входов 1…8 (код ошибки)	WORD	только чтение
1.4. Модуль скоростного ввода аналоговых сигналов MB110-8AC			
C_Time_1 … C_Time_8	Циклическое время измерения входов 1…8	WORD	только чтение
Read_1...Read_8	Показание входов 1…8 в представлении с плавающей точкой	REAL	только чтение
SRD_1…SRD_8	Статус измерения входов 1…8 (код ошибки)	WORD	только чтение
1.5. Модуль ввода сигналов тензодатчиков MB110-224.1TД			
RdfF1	Измеренное значение физической величины, ед.	REAL	только чтение
RdSt	Статус измерения	WORD	только чтение

1.6. Модуль ввода сигналов тензодатчиков МВ110-224.4ТД			
RdfF1…RdfF4	Измеренное значение физической величины входа 1...4, ед.	REAL	только чтение
RdSt	Статус измерения	WORD	только чтение
1.7. Модуль аналогового ввода МВ110-224.pH			
RdRs	Результаты измерений канала pH/ОВП	REAL	только чтение
RdSt	Статус измерения	WORD	только чтение
RdTm	Измеренное значение температуры	REAL	только чтение
2. Модули аналогового вывода МУ110			
2.1. Модуль аналогового вывода МУ110-8И			
rOE_1…rOE_8	Значение на выходе 1 - 8 (диапазон 0-1000 ед. соответствует сигналу на выходе 0-100%)	REAL	чтение/запись
2.2. Модуль аналогового вывода МУ110-6У			
rOE_1…rOE_6	Значение на выходе 1 - 6 (диапазон 0-1000 ед. соответствует сигналу на выходе 0-100%)	REAL	чтение/запись
3. Модули дискретного ввода МВ110			
3.1. Модуль ввода дискретных сигналов МВ110-16ДН(Д)			
rCounter1…rCounter16	Счетчик входа 1...16	WORD	только чтение
rInput1…rInput16	Состояние входа 1…16	BOOL	только чтение
wCounter1…wCounter16	Счетчик входа 1...16	WORD	только запись
3.2. Модуль ввода дискретных сигналов МВ110-8ДФ			
rCounter1…rCounter8	Счетчик входа 1...8	WORD	только чтение
rInput1…rInput8	Состояние входа 1…8	BOOL	только чтение
wCounter1…wCounter8	Счетчик входа 1...8	WORD	только запись
3.3. Модуль ввода дискретных сигналов МВ110-32ДН			
rCounter1…rCounter32	Счетчик входа 1...32	WORD	только чтение
rInput1…rInput32	Состояние входа 1…32	BOOL	только чтение
wCounter1…wCounter32	Счетчик входа 1...32	WORD	только запись

4. Модули дискретного вывода МУ110			
4.1. Модуль дискретного вывода МУ110-8Р(К)			
rOut1…8	Состояние выхода 1...8	BOOL	только чтение
wOut1…8	Состояние выхода 1...8	BOOL	только запись
4.2. Модуль дискретного вывода МУ110-16Р(К)			
rOut1…16	Состояние выхода 1...16	BOOL	только чтение
wOut1…16	Состояние выхода 1...16	BOOL	только запись
4.3. Модуль дискретного вывода МУ110-32Р			
rOut1…32	Состояние выхода 1...32	BOOL	только чтение
wOut1…32	Состояние выхода 1...32	BOOL	только запись
5. Модули дискретного ввода-вывода МК110			
5.1. Модуль ввода-вывода дискретных сигналов МК110-8ДН(Д).4Р			
rCounter1…rCounter8	Счетчик входа 1…8	WORD	только чтение
rInput1…rInput8	Состояние входа 1…8	BOOL	только чтение
rOut1… rOut4	Состояние выхода 1…4	BOOL	только чтение
wCounter1…wCounter8	Счетчик входов 1…8	WORD	только запись
wOut1… wOut4	Состояние выхода 1…4	BOOL	только запись
5.2. Модуль ввода-вывода дискретных сигналов МК110-4К.4Р			
rInput1…4	Состояние входа 1…4	BOOL	только чтение
rOut1…4	Состояние выхода 1…4	BOOL	только чтение
wOut1...4	Состояние выхода 1…4	BOOL	только запись
5.3. Модуль ввода-вывода дискретных сигналов МК110-4ДН.4ТР(Р)			
rCounter1…rCounter4	Счетчик входа 1…4	WORD	только чтение
rInput1…rInput4	Состояние входа 1…4	BOOL	только чтение
rOut1… rOut4	Состояние выхода 1…4	BOOL	только чтение
wCounter1…wCounter4	Счетчик входов 1…4	WORD	только запись
wOut1… wOut4	Состояние выхода 1…4	BOOL	только запись

6. Модули ввода параметров электрической сети			
6.1. Модуль ввода параметров электрической сети МЭ110-220.3М			
CosFi_a, CosFi_b, CosFi_c	Измеренное значение коэффициента мощности на соответствующем входе	REAL	только чтение
F	Измеренное значение частоты сети	REAL	только чтение
Ia, Ib, Ic	Измеренное значение тока на соответствующем входе	REAL	только чтение
In	Измеренное значение тока нейтрали	REAL	только чтение
powerPa, powerPb, powerPc	Измеренное значение активной мощности на соответствующем входе	REAL	только чтение
powerQa, powerQb, powerQc	Измеренное значение реактивной мощности на соответствующем входе	REAL	только чтение
powerSa, powerSb, powerSc	Измеренное значение полной мощности на соответствующем входе	REAL	только чтение
Ua, Ub, Uc	Измеренное значение напряжение на соответствующем входе	REAL	только чтение
Uab, Ubc, Uca	Измеренное значение межфазного напряжения между соответствующими входами	REAL	только чтение
Vab, Vbc, Vca	Измеренное значение фазового угла между соответствующими входами	REAL	только чтение
6.2. Однофазный мультиметр МЭ110-1М			
cosFi	Измеренное значение коэффициента мощности	REAL	только чтение
F	Измеренное значение частоты сети	REAL	только чтение
Ia	Измеренное значение тока	REAL	только чтение
powerP	Измеренное значение активной мощности	REAL	только чтение
powerQ	Измеренное значение реактивной мощности	REAL	только чтение
PowerS	Измеренное значение полной мощности	REAL	только чтение
Ua	Измеренное значение напряжения	REAL	только чтение
6.3. Однофазный вольтметр МЭ110-1Н			
Ua	Измеренное значение напряжения	REAL	только чтение
F	Измеренное значение частоты сети	REAL	только чтение
6.4. Однофазный амперметр МЭ110-1Н			
Ia	Измеренное значение тока	REAL	только чтение

Б. Список переменных входов/выходов модулей в ФБ библиотеки ModulsOwenLib

Список переменных входов/выходов ФБ модулей библиотеки ModulsOwenLib приведен в табл. Б. Подробное описание переменных приведено в **РЭ** на соответствующие модули.

Описание переменных опроса ФБ приведено в п. 5.3.2.

Табл. Б. Список переменных входов/выходов ФБ библиотеки **ModulsOwenLib**

Имя переменной	Описание	Тип данных	Тип доступа
1. Модули аналогового ввода MB110			
1.1. Модуль ввода аналоговых сигналов MB110-2A (ФБ MV110_2A_inputs)			
InpT1 (InpT2)	Циклическое время измерения входа 1 (2)	WORD	только чтение
Inp1 (Inp2)	Показание входа 1 (2) в представлении с плавающей точкой	REAL	только чтение
InpExcSCode1 (InpExcSCode2)	Статус измерения входа 1 (2) [код ошибки]	WORD	только чтение
1.2. Модуль скоростного ввода аналоговых сигналов MB110-2AC (ФБ MV110_2AS_inputs)			
InpT1 (InpT2)	Циклическое время измерения входа 1 (2)	WORD	только чтение
Inp1 (Inp2)	Показание входа 1 (2) в представлении с плавающей точкой	REAL	только чтение
InpExcSCode1 (InpExcSCode2)	Статус измерения входа 1 (2) [код ошибки]	WORD	только чтение
1.3. Модуль ввода аналоговых сигналов MB110-8A (ФБ MV110_8A_inputs)			
InpT1… InpT8	Циклическое время измерения входов 1…8	WORD	только чтение
Inp1...Inp8	Показание входов 1…8 в представлении с плавающей точкой	REAL	только чтение
InpExcSCode1… InpExcSCode8	Статус измерения входов 1…8 (код ошибки)	WORD	только чтение
1.4. Модуль скоростного ввода аналоговых сигналов MB110-8AC (ФБ MV110_8AS_inputs)			
InpT1… InpT8	Циклическое время измерения входов 1…8	WORD	только чтение
Inp1...Inp8	Показание входов 1…8 в представлении с плавающей точкой	REAL	только чтение
InpExcSCode1… InpExcSCode8	Статус измерения входов 1…8 (код ошибки)	WORD	только чтение

1.5. Модуль ввода сигналов тензодатчиков MB110-224.1ТД (ФБ MV110_1TD_inputs)			
Inp1	Измеренное значение физической величины, ед.	REAL	только чтение
InpExcSCode1	Статус измерения	WORD	только чтение
1.6. Модуль ввода сигналов тензодатчиков MB110-224.4ТД (ФБ MV110_4TD_inputs)			
Inp1…Inp4	Измеренное значение физической величины, ед.	REAL	только чтение
InpExcSCode1	Статус измерения	WORD	только чтение
1.7. Модуль аналогового ввода MB110-224.pH (ФБ MV110_pH_inputs)			
InpPH	Результаты измерений канала pH/ОВП	REAL	только чтение
InpExcSCode	Статус измерения	WORD	только чтение
InpTemp	Измеренное значение температуры	REAL	только чтение
2. Модули аналогового вывода МУ110			
2.1. Модуль аналогового вывода МУ110-8И (ФБ MY110_8I_outs)			
Out1…Out8	Значение на выходе 1 - 8 (диапазон 0-1000 ед. соответствует сигналу на выходе 0-100%)	REAL	чтение/запись
2.2. Модуль аналогового вывода МУ110-6У (ФБ MY110_6Y_outs)			
Out1…Out6	Значение на выходе 1 - 6 (диапазон 0-1000 ед. соответствует сигналу на выходе 0-100%)	REAL	чтение/запись
3. Модули дискретного ввода MB110			
3.1. Счетчики входов модуля ввода дискретных сигналов MB110-16ДН(Д) (ФБ MV110_16D_counter)			
Cnt1… Cnt16	Счетчик входа 1...16	WORD	чтение/запись
3.2. Входы модуля ввода дискретных сигналов MB110-16ДН(Д) (ФБ MV110_16D_inputs)			
Inp1… Inp16	Состояние входа 1…16	BOOL	только чтение

3.3. Счетчики входов модуля ввода дискретных сигналов МВ110-8ДФ (ФБ MV110_8D_counter)			
Cnt1… Cnt8	Счетчик входа 1...8	WORD	чтение/запись
3.4. Входы модуля ввода дискретных сигналов МВ110-8ДФ (ФБ MV110_8D_inputs)			
Inp1… Inp8	Состояние входа 1…8	BOOL	только чтение
4. Модули дискретного вывода МУ110			
4.1. Модуль дискретного вывода МУ110-8Р(К) (ФБ MY110_8R_outs)			
Out1…Out8	Состояние выхода 1...8	BOOL	чтение/запись
4.2. Модуль дискретного вывода МУ110-16Р(К) (ФБ MY110_16R_outs)			
Out1…Out16	Состояние выхода 1...16	BOOL	чтение/запись
5. Модули дискретного ввода-вывода МК110			
5.1. Счетчики входов модуля ввода-вывода дискретных сигналов МК110-8ДН(Д).4Р (ФБ MK110_8dn_4rR_counter)			
Cnt1… Cnt8	Счетчик входа 1...8	WORD	чтение/запись
5.2. Входы модуля ввода-вывода дискретных сигналов МК110-8ДН(Д).4Р (ФБ MK110_8dn_4R_inputs)			
Inp1… Inp8	Состояние входа 1…8	BOOL	только чтение
5.3. Выходы модуля ввода-вывода дискретных сигналов МК110-8ДН(Д).4Р (ФБ MK110_8dn_4R_outs)			
Out1…Out8	Состояние выхода 1...8	BOOL	чтение/запись
5.4. Входы модуля ввода-вывода дискретных сигналов МК110-4К.4Р (ФБ MK110_4K_4R_inputs)			
Inp1… Inp4	Состояние входа 1…4	BOOL	только чтение
5.5. Выходы модуля ввода-вывода дискретных сигналов МК110-4К.4Р (ФБ MK110_4K_4R_outs)			
Out1…Out4	Состояние выхода 1...4	BOOL	чтение/запись

5.6. Счетчики входов модуль ввода-вывода дискретных сигналов МК110-4ДН.4Р(Р) (ФБ МК110_4dn_4r_counter)			
Cnt1… Cnt4	Счетчик входа 1...4	WORD	чтение/запись
5.7. Счетчики входов модуль ввода-вывода дискретных сигналов МК110-4ДН.4Р(Р) (ФБ МК110_4dn_4r_inputs)			
Inp1… Inp4	Состояние входа 1…4	BOOL	только чтение
5.8. Счетчики входов модуль ввода-вывода дискретных сигналов МК110-4ДН.4Р(Р) (ФБ МК110_4dn_4r_outs)			
Out1…Out4	Состояние выхода 1...4	BOOL	чтение/запись
6. Модули ввода/вывода			
6.1. Модуль ввода аналоговый МВА8 (ФБ MVA8_inputs)			
Inp1...Inp8	Показание входов 1…8 в представлении с плавающей точкой	REAL	только чтение
InpExcSCode1… InpExcSCode8	Статус измерения входов 1…8 (код ошибки)	WORD	только чтение
6.2. Входы модуля дискретного ввода-вывода МДВВ (ФБ MDVV_inputs)			
Inp1… Inp12	Состояние входа 1…12	BOOL	только чтение
6.3. Входы модуля дискретного ввода-вывода МДВВ (ФБ MDVV_outs)			
Out1…Out8	Состояние выхода 1...8	BOOL	чтение/запись

B. Список переменных диагностики Modbus

Табл. B1. Список переменных диагностики

Имя переменной	Описание	Тип переменной	Тип доступа
Modbus Master, COM Port			
uiNumberOfCommunicatingSlaves	Число подключенных slave-устройств.	UINT	только чтение
xAllSlavesOk	Принимает значение **FALSE** при возникновении ошибки хотя бы с одним из slave-устройств, при отсутствии ошибок обмена имеет значение **TRUE**.	BOOL	только чтение
xResetComPort	Перезапускает COM-порт по переднему фронту логической переменной.	BOOL	чтение/запись
xStop	При значении **TRUE** обмен со всеми slave-устройствами прекращается. При значении **FALSE** – восстанавливается.	BOOL	чтение/запись
Modbus TCP Master			
uiNumberOfCommunicatingSlaves	Число подключенных slave-устройств.	UINT	только чтение
xSlaveError	Принимает значение **TRUE** при возникновении ошибки хотя бы с одним из slave-устройств.	BOOL	только чтение
xStop	При значении **TRUE** обмен со всеми slave-устройствами прекращается. При значении **FALSE** – восстанавливается.	BOOL	чтение/запись
Modbus Slave, COM Port / Modbus TCP Slave			
byModbusError	Код ошибки обмена. См. табл. B2.	BYTE	только чтение
xInitDone	Принимает **TRUE**, если все команды инициализации были отправлены.	BOOL	только чтение
xError	Принимает **TRUE** в случае возникновения ошибки обмена. Связь с устройством прекращается.	BOOL	только чтение
xAcknowledge (xConfirmErrox)	При значении **TRUE** происходит попытка восстановления связи с устройством, при этом значения переменных **byModbusError** и **xError** сохраняют свои последние значения.	BOOL	чтение/запись
xReset	При значении **TRUE** происходит попытка	BOOL	чтение/запись

	восстановления связи с устройством, при этом значения переменных **byModbusError** и **xError** сбрасываются.		
xTrigger	Производит опрос устройства по переднему фронту.	BOOL	чтение/запись
Modbus Serial Device			
xInternalError	Значение **TRUE** характеризует неустранимую ошибку (например, отсутствие лицензии на компонент или ошибку выделения памяти).	BOOL	только чтение
Modbus TCP Slave Device			
uiClientConnections	Число установленных в данный момент клиентских соединений.	BOOL	только чтение
xInternalError	Значение **TRUE** характеризует неустранимую ошибку (например, отсутствие лицензии на компонент или ошибку выделения памяти).	UINT	только чтение

Табл. B2. Коды ошибок обмена

Код ошибки	Имя ошибки	Описание
16#0	RESPONSE SUCCESS	Отсутствие ошибок обмена.
16#1	ILLEGAL FUNCTION	Принятый код функции не может быть обработан
16#2	ILLEGAL DATA ADDRESS	Адрес данных, указанный в запросе, недоступен.
16#3	ILLEGAL DATA VALUE	Значение, содержащееся в поле данных запроса, является недопустимой величиной.
16#4	SLAVE DEVICE FAILURE	Пока slave-устройство пыталось выполнить затребованное действие, произошла ошибка.
16#5	ACKNOWLEDGE	Ведомое устройство приняло запрос и обрабатывает его, но это потребует много времени. Этот ответ предохраняет master-устройство от генерации ошибки таймаута.
16#6	SLAVE DEVICE BUSY	Slave-устройство занято обработкой команды. Master-устройство должно повторить сообщение позже, когда ведомое освободится.
16#8	MEMORY PARITY ERROR	Ошибка при использовании функций 20 и 21.
16#A	GATEWAY PATH UNAVALIABLE	Ошибка конфигурации сетевого шлюза.
16#B	GATEWAY DEVICE FAILED TO RESPONSE	Отправленный запрос не был получен сетевым шлюзом.
16#A1	RESPONSE TIMEOUT	Отсутствие ответа от slave-устройства.
16#A2	RESPONSE CRC FAIL	Контрольная сумма ответа некорректна.
16#A3	RESPONSE WRONG SLAVE	Получен ответ от неверного устройства.
16#A4	RESPONSE WRONG FUNCTION CODE	Получен ответ с неверным кодом функции.
16#A5	REQUEST FAILED TO SEND	Ошибка COM-порта master-устройства. Запрос не был отправлен.

Г. Рекомендуемые версии компонентов Modbus

Различные версии **CODESYS** включают в себя разные версии Modbus-компонентов. В табл. Г приведены рекомендуемые версии компонентов в зависимости от используемой прошивки СПК:

Табл. Г. Рекомендуемые версии компонентов Modbus

Прошивка СПК	**4.000***	**3.9xx**	**3.3xx – 3.4xx**	**3.1xx-3.2xx**
Таргет-файл	3.5.7.0	3.5.4.20 (23)	3.5.3.40 (13)	3.5.2.0
CODESYS	3.5 SP7 Patch4	3.5. SP5 Patch5	3.5 SP3 Patch5	3.5. SP2 Patch3
Modbus COM	3.4.0.0	3.4.0.0	3.4.0.0	3.4.0.0
Modbus RTU Master	3.5.5.0	3.5.5.0	3.5.3.50	3.5.2.0
Modbus RTU Slave	3.5.4.0	3.5.4.0	3.5.3.0	3.5.2.0
Modbus RTU Serial Device	3.5.5.0	3.5.5.0	3.5.3.50	3.5.2.0
Ethernet-адаптер	3.5.7.0	3.4.2.0	3.4.2.0	3.4.2.0
Modbus TCP Master	3.5.6.0	3.5.5.0	3.5.3.0	3.5.2.0
Modbus TCP Slave	3.5.7.0	3.5.4.0	3.5.3.0	3.5.2.0
Modbus TCP Device	3.5.5.0	3.5.2.0	3.5.2.0	3.5.2.0

* *экспериментальная прошивка, использовать только для тестирования*

Д. Листинги программ

Д.1. Листинг программы COM2 из п. 5.4.2

```
PROGRAM COM2
VAR
	ComConn_COM2:			ComConn;		// ФБ настройки и открытия порта COM2
	My8A:				MV110_8A_inputs;	// ФБ опроса модуля MB110-8A

	rMV110_8A_input1:		REAL;			// Показание входа 1 модуля MB110-8A
	wsMV110_8A_input1_status:	WSTRING;		// Статус измерения входа 1
END_VAR



// [1] открываем COM-порт
ComConn_COM2
(
	enable:=TRUE,
	PortNum:=3,
	PortBaudrate:=115200,
	PortParity:=0,
	PortStopBits:=1,
	PortByteSize:=8,
	Port_ModeOn:=FALSE,
	Port_CloseOn:=FALSE,
);


// [2] запускаем ФБ опроса модуля MB110-8A
My8A
(
	Handle:=ComConn_COM2.Handle,
	// если COM-порт открыт и отсутствуют ошибки...
	// ...то начинаем новый цикл опроса
	Enabl:=(ComConn_COM2.Done AND ComConn_COM2.ErrCode=0),
	Modd:=modulsowenlib.MB_RTU,
	Addr:=1,
	TimeOut:=T#500MS,
	ErCl:=3,
);


// [3] если ФБ опроса модуля завершил работу...
IF My8A.Done THEN
	// ...и ошибки отсутствуют, то забираем значения модуля...
	IF My8A.wErrCode=0 THEN
		rMV110_8A_input1:=My8A.inp1;
		wsMV110_8A_input1_status:=MV110_2A_8A_StatusToString(My8A.inpExcSCode1);
	END_IF

	// ...завершаем опрос модуля MB110-8A
	My8A(Enabl:=FALSE);

END_IF
```

Д.2. Листинг программы COM3 из п. 5.4.2

```
PROGRAM COM3
VAR
	ComConn_COM3:	ComConn;		// ФБ настройки и открытия порта COM3

	My16D:		MV110_16D_inputs; // ФБ опроса модуля MB110-16Д
	My16R:		MY110_16R_outs;	// ФБ опроса модуля MУ110-16Р

	xMV110_16D_input1: BOOL;		// Состояние 1-го входа модуля MB110-16Д

	xModule:	INT; // Переменные начала цикла опроса (0 - MB110-16Д, 1 - MУ110-16Р)
END_VAR

VAR_INPUT
	xMY110_16R_output1: BOOL;  // Переменная для записи состояния 1-го выхода модуля MB110-16Р
END_VAR



// [1] открываем COM-порт
ComConn_COM3
(
	enable:=TRUE,
	PortNum:=4,
	PortBaudrate:=115200,
	PortParity:=0,
	PortStopBits:=1,
	PortByteSize:=8,
	Port_ModeOn:=FALSE,
	Port_CloseOn:=FALSE,
);


// [2] xModule определяет опрашиваемый модуль: 0 - MB110-16Д, 1 - MУ110-16Р
CASE xModule OF

	0:

		// [2.0.1] запускаем ФБ опроса модуля MB110-16Д
		My16D
		(
			Handle:=ComConn_COM3.Handle,
			// если COM-порт открыт и ошибки отсутствуют, то начинаем опрос
			Enabl:=(ComConn_COM3.Done AND ComConn_COM3.ErrCode=0),
			Modd:=modulsowenlib.MB_RTU,
			Addr:=1,
			TimeOut:=T#500MS,
			ErCl:=3,
		);


		// [2.0.2] если ФБ опроса модуля завершил работу...
		IF My16D.Done THEN
			// ...и ошибки отсутствуют, то забираем значения модуля
			IF My16D.wErrCode=0 THEN
				xMV110_16D_input1:=My16D.inp1;
			END_IF

			// завершаем опрос модуля MB110-16Д
			My16D(Enabl:=FALSE);

			// переходим к опросу модуля MУ110-16Р
			xModule:=1;

		END_IF
```

```
        1:

                // [2.1.1] запускаем ФБ опроса модуля МУ110-16Р
                My16R
                (
                        Handle:=ComConn_COM3.Handle,
                        // если COM-порт открыт и ошибки отсутствуют, то начинаем опрос
                        Enabl:=(ComConn_COM3.Done AND ComConn_COM3.ErrCode=0),
                        Modd:=modulsowenlib.MB_RTU,
                        Addr:=17,
                        TimeOut:=T#500MS,
                        ErCl:=3,
                        // записываем значения в модуль
                        out1:=xMY110_16R_output1,
                );


                // [2.1.2] если ФБ опроса модуля завершил работу...
                IF My16R.Done THEN

                        // ...то завершаем опрос модуля МУ110-16Р...
                        My16R(Enabl:=FALSE);

                        // ...и начинаем новый цикл опроса
                        xModule:=0;
                END_IF

END_CASE
```

Д.3. Листинг программы COM2 из п. 6.4.2

```
PROGRAM COM2
VAR
	COM_Service_COM2:		COM_SERVICE;		// ФБ настройки и открытия порта COM2
	Settings_COM2:			COM_SETTINGS;		// Структура настроек порта COM2
	SettingsEX_COM2:		COM_SETTINGSex;		// Структура расширенных настроек порта COM2

	MV110_8A:			 MB_RD_HOLD_REGS;	// ФБ опроса модуля MB110-8A

	rMV110_8A_input1:		 REAL;			// Показание входа 1 модуля MB110-8A
	wMV110_8A_input1_status:  WORD;			// Код ошибки измерения входа 1
	wsMV110_8A_input1_status: WSTRING;		// Статус измерения входа 1

	abyBuffer:		ARRAY [0..255] OF BYTE;	// Буфер ФБ опроса модуля MB110-8A
	abyMV110_8A_data: ARRAY [0..255] OF BYTE;	// Буфер данных, считанных с модуля MB110-8A

	_2WORD_TO_REAL:	Word_Real;	// Экземпляр объединия для преобразования двух WORD в REAL
END_VAR



// [1] настраиваем COM-порт
Settings_COM2.sPort:=3;
Settings_COM2.byStopBits:=1;
Settings_COM2.byParity:=0;
Settings_COM2.ulBaudrate:=115200;
Settings_COM2.ulTimeout:=0;
Settings_COM2.ulBufferSize:=0;

SettingsEX_COM2.byByteSize:=8;


// [2] открываем COM-порт
COM_Service_COM2
(
	Enable:=TRUE,
	Settings:=Settings_COM2,
	Sets_Ex:=SettingsEX_COM2,
	Task:=OPEN_TSK,
);


// [3] запускаем ФБ опроса модуля MB110-8A
MV110_8A
(
	Enable:=COM_Service_COM2.Ready,
	Mode:=MB_RTU,
	DevAddr:=1,
	FirstAddr:=2,
	Quantity:=4,
	ComHandle:=COM_Service_COM2.handle,
	TimeOut:=T#500MS,
	Buffer:=abyBuffer,
);
```

```
// [4] если ФБ опроса модуля завершил работу...
IF MV110_8A.Complete THEN
	// ...и ошибки отсутствуют, то забираем буфер модуля
	IF MV110_8A.Exception=0 THEN
		abyMV110_8A_data:=abyBuffer;

		// байты 0 и 1 содержат статус измерения входа 1
		// склеиваем их в WORD и декодируем в текстовое сообщение
		wMV110_8A_input1_status:=BYTE_TO_WORD(abyMV110_8A_data[1]) OR
			SHL(BYTE_TO_WORD(abyMV110_8A_data[0]),8);
		wsMV110_8A_input1_status:=MV110_2A_8A_StatusToString(wMV110_8A_input1_status);

		// байты 4-7 содержат значение на входе 1 с плавающей точкой.
		// переставляем байты местами и преобразуем в REAL
		_2WORD_TO_REAL.abyBytes[3]:=abyMV110_8A_data[4];
		_2WORD_TO_REAL.abyBytes[2]:=abyMV110_8A_data[5];
		_2WORD_TO_REAL.abyBytes[1]:=abyMV110_8A_data[6];
		_2WORD_TO_REAL.abyBytes[0]:=abyMV110_8A_data[7];

		rMV110_8A_input1:=_2WORD_TO_REAL.rReal;
	END_IF

	// завершаем опрос модуля MB110-8A
	MV110_8A(Enable:=FALSE, Buffer:=abyBuffer);

END_IF
```

Д.4. Листинг программы COM3 из п. 6.4.2

```
PROGRAM COM3
VAR
	COM_SERVICE_COM3: COM_SERVICE;		// ФБ настройки и открытия порта COM3
	Settings_COM3:	COM_SETTINGS;		// Структура настроек порта COM3
	SettingsEX_COM3:	COM_SETTINGSex;	// Структура расширенных настроек порта COM3

	xModule:		INT;	// Переменные начала цикла опроса (0 - MB110-16Д, 1 - MУ110-16Р)

	MV110_16D:		MB_RD_HOLD_REGS;		// ФБ опроса модуля MB110-16Д
	MY110_16R:		MB_WR_REGS;			// ФБ опроса модуля MУ110-16Р

	abyMV110_16D_buffer: ARRAY [0..255] OF BYTE;	// Буфер ФБ опроса модуля MB110-16Д
	abyMV110_16D_data:	ARRAY [0..255] OF BYTE;	// Буфер данных, считанных с модуля MB110-16Д

	xMV110_16D_input1:	BOOL;			// Состояние 1-го входа модуля MB110-16Д
	abyMY110_16R_buffer: ARRAY [0..255] OF BYTE;	// Буфер ФБ опроса модуля MУ110-16Р
	testMY110_16R_buffer:ARRAY [0..255] OF BYTE;	// Буфер ФБ опроса модуля MУ110-16Р
END_VAR

VAR_INPUT
	xMY110_16R_output1:	BOOL;	// Переменная для записи состояния 1-го выхода модуля MB110-16Р
END_VAR



// [1] настраиваем COM-порт
Settings_COM3.sPort:=4;
Settings_COM3.byStopBits:=1;
Settings_COM3.byParity:=0;
Settings_COM3.ulBaudrate:=115200;
Settings_COM3.ulTimeout:=0;
Settings_COM3.ulBufferSize:=0;

SettingsEX_COM3.byByteSize:=8;


// [2] открываем COM-порт
COM_Service_COM3
(
	Enable:=TRUE,
	Settings:=Settings_COM3,
	Sets_Ex:=SettingsEX_COM3,
	Task:=OPEN_TSK,
);

// [3] xModule определяет опрашиваемый модуль: 0 - MB110-16Д, 1 - MУ110-16Р
CASE xModule OF

	0:

		// [3.0.1] запускаем ФБ опроса модуля MB110-16Д
		MV110_16D
		(
			Enable:=COM_Service_COM3.Ready,
			Mode:=MB_RTU,
			DevAddr:=1,
			FirstAddr:=51,
			Quantity:=1,
			ComHandle:=COM_Service_COM3.handle,
			TimeOut:=T#500MS,
			Buffer:=abyMV110_16D_buffer,
		);
```

```
		// [3.0.2] если ФБ опроса модуля завершил работу...
		IF MV110_16D.Complete THEN
			// ...и ошибки отсутствуют, то забираем значения модуля
			IF MV110_16D.Exception=0 THEN
				abyMV110_16D_data:=abyMV110_16D_buffer;

				xMV110_16D_input1:=abyMV110_16D_data[1].0;
			END_IF

			// завершаем опрос модуля МВ110-16Д
			MV110_16D(Enable:=FALSE, Buffer:=abyMV110_16D_buffer);

			// переходим к опросу модуля МУ110-16Р
			xModule:=1;


		END_IF

	1:

		// [3.1.1] копируем записываемое значение в буфер ФБ

		abyMY110_16R_buffer[0].0:=xMY110_16R_output1;

		// [3.1.2] запускаем ФБ опроса модуля МУ110-16Р
		MY110_16R
		(
			Enable:=COM_SERVICE_COM3.Ready,
			Mode:=MB_RTU,
			DevAddr:=17,
			FirstAddr:=50,
			Quantity:=1,
			ComHandle:=COM_SERVICE_COM3.handle,
			TimeOut:=T#500MS,
			Buffer:=abyMY110_16R_buffer,
		);

		// [3.1.3] если ФБ опроса модуля завершил работу...
		IF MY110_16R.Complete THEN

			// ...то завершаем опрос модуля МУ110-16Р...
			MY110_16R(Enable:=FALSE, Buffer:=abyMY110_16R_buffer);

			// ...и начинаем новый цикл опроса
			xModule:=0;
		END_IF
END_CASE
```