

CODESYS V3.5

Настройка обмена по протоколу Modbus



Руководство пользователя

03.09.2019 версия 2.0

Оглавление

1	Цель документа. Способы работы с Modbus в CODESYS V3.5	6
2	Общие сведения	7
2.1	Основные сведения об интерфейсе RS-485	7
2.2	Основные сведения о протоколе Modbus	7
2.3	Нумерация COM-портов в CODESYS	10
2.4	Особенности работы с модулями Mx110	11
3	Шаблоны модулей Mx110 и Mx210	13
3.1	Установка шаблонов модулей в среду CODESYS	13
3.2	Пример: СПК1хх [М01] + модули Мх110	15
3.3	Пример: СПК1xx [M01] + модули Mx210	23
3.4	Диагностика и управление обменом	30
3.5	Библиотеки Mx Assistant	32
4	Стандартные средства конфигурирования	34
4.1	Общая методика конфигурирования интерфейсов	34
4.2	Настройка контроллера в режиме Modbus Serial Master	35
4.3	Настройка контроллера в режиме Modbus RTU Slave	42
4.4	Настройка контроллера в режиме Modbus TCP Master	46
4.5	Настройка контроллера в режиме Modbus TCP Slave	51
4.6	Диагностика и управление обменом	54
4.7	Компоненты Modbus и конфигурация задач	60
4.8	Преобразование данных для передачи по Modbus	61
	4.8.1 Использование объединений (UNION)	61
	4.8.2 ИСПОЛЬЗОВАНИЕ УКАЗАТЕЛЕИ	65

4.9	Прим	lep: СПК1xx [M01] (Modbus RTU Master) + модули Mx110	68
4.10	Прим	ıер: СПК1xx [M01] (Modbus RTU Slave) + MasterOPC Universal Modbus Servei	80
4.11	Прим	iep: СПК1xx [M01] (Modbus TCP Master) + модули Mx210	91
4.12	2 Прим	ıер: СПК1xx [M01] (Modbus TCP Slave) + MasterOPC Universal Modbus Server	[.] 102
5	Библ	иотека OwenCommunication	113
F 4	0.000		440
5.1	Осно	вная информация	
5.2	Уста	ювка библиотеки	113
5.3	Доба	вление библиотеки в проект CODESYS	114
5.4	Стру	ктуры и перечисления	115
	5.4.1	Перечисление ERROR	115
	5.4.2	Перечисление COM_PARITY	116
	5.4.3	Перечисление СОМ_STOPBIT	116
	5.4.4	Перечисление MB_FC	116
	5.4.5	Структура MB_REQ_INFO	117
5.5	ФБн	астройки интерфейсов	118
	5.5.1	ФБ COM_Control	118
	5.5.2	ФБ TCP_Client	119
5.6	ФБп	ротокола Modbus	120
	5.6.1	∙ ФБ MB_SerialRequest	
	5.6.2	ФБ MB_SerialSlave	
	5.6.3	_ ФБ MB_TcpRequest	
	5.6.4	 ФБ MB_TcpSlave	
57	ΦБн		128
•	571	ΦΕ UNM SerialRequest	128
	572	Φ5 UNM_Containtequest	130
	573	ФБ UNM_UdpRequest	132
F 0	•		404
5.8	Функ		134
	5.0.1		
	5.0.2		
	5.0.3		130
	0.0.4		120
	5.6.5		130
5.9	Прим	еры	139
	5.9.1	СПК1xx [M01] (Modbus RTU Master) + модули Mx110	139
	5.9.2	СПК1xx [M01] (Modbus RTU Slave) + MasterOPC Universal Modbus Server	149
	5.9.3	СПК1xx [M01] (Modbus TCP Master) + модули Mx210	157
	5.9.4	СПК1xx [M01] (Modbus TCP Slave) + MasterOPC Universal Modbus Server	167
	5.9.5	СПК1хх [M01] (Modbus TCP Slave) – чтение файлов с помощью 20 функции 175	Modbus

6	FAQ		2
6.1	Что де	елать, если не удается наладить обмен по Modbus?18	2
6.2	Каким	і образом считать/передать значение с плавающей точкой (REAL)?	3
6.3	Каким	і образом считать/передать отрицательное значение (INT)?	3
6.4	Вопро	осы по стандартным средствам конфигурирования	3
	6.4.1	Какие версии компонентов рекомендуются к использованию?	3
	6.4.2	Modbus Serial Master: как реализовать чтение/запись по триггеру?	4
	6.4.3	Modbus RTU Slave: почему принятые значения сбрасываются в 0?	5
	6.4.4	Можно ли менять данные holding регистров из программы? 18	5
	6.4.5	Как произвести диагностику обмена в программе?	5
	6.4.6	Как расшифровываются пиктограммы статуса обмена? 18	5
6.5	Вопро	осы по библиотеке OwenCommunication18	6
	6.5.1 проект?	В примерах работы с библиотекой используются действия. Как добавить их 186	в
	6.5.2 сравнені	Позволяет ли библиотека организовать опрос с более высокой частотой п ию со стандартными средствами конфигурирования?	0 6
Пр	иложен	ние А. Рекомендуемые версии компонентов Modbus18	7
Пр	иложен	ние Б. Листинги примеров188	8
Б1 I	Пистинг г	1римера из п. 5.9.1 18	8
5	Б11Кол	апрограммы PLC PRG ST 18	8
	Б 1 2 Кол	д программия г 20_1 г 00_01 18 1 лействия СОМ1 18	9
	Б.1.3 Код	действия СОМ2	0
Б2 J	Пистинг г	19имера из п. 5.9.2	2
Б3 J	Пистинг г	19имера из п. 5.9.3	4
	Б.3.1 Код	а программы PLC PRG ST	4
	Б.3.2 Код	действия MV210_101	5
	Б.З.З Код	_ д действия MK210_301	6
Б4 J	Пистинг г	іримера из п. 5.9.4	8
Б5 J	Пистинг г	аримера из п. 5.9.5	0

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

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

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

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

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

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

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

3. Для профессионалов – библиотека OwenCommunication.

Библиотека позволяет настроить обмен с любым устройством, но, в отличие от стандартных средств (пп. 2), лишена их ограничений и предоставляет дополнительный функционал.

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

2 Общие сведения

2.1 Основные сведения об интерфейсе RS-485

- **1.** Интерфейс <u>RS-485</u> подразумевает использование исключительно топологии типа «шина» (топологии типа «звезда» и «кольцо» не описаны в стандарте).
- 2. В сети может присутствовать только одно master-устройство, которое отсылает запросы и принимает ответы от подчиненных slave-устройств. Slave-устройства не могут являться инициаторами обмена.
- Число slave-устройств в сегменте сети не должно превышать 32. Сегменты могут быть соединены повторителями (например, <u>OBEH AC5</u>), но следует учитывать что так как опрос всех устройств происходит последовательно, то время одного полного цикла опроса может значительно увеличиться. Общее ограничение числа slave-устройств в сети для протокола Modbus – 247.
- **4.** На первом и последнем устройстве в сети рекомендуется устанавливать согласующий резистор (терминатор) с сопротивлением **120 Ом**.
- 5. Для линий связи RS-485 необходимо использовать экранированный кабель с витой парой, предназначенный для промышленного интерфейса RS-485 с волновым сопротивлением 120 Ом (например, КИПЭВ). Экран кабеля должен быть соединен с функциональной землей только в одной точке.
- 6. Некоторые устройства имеют встроенные резисторы подтяжки интерфейса RS-485. Информация и рекомендации по их использованию приведены в руководстве по эксплуатации на соответствующие приборы.

2.2 Основные сведения о протоколе Modbus

<u>Modbus</u> – открытый коммуникационный протокол, основанный на архитектуре **Master-Slave** (ведущийведомый). Спецификация протокола доступна на сайте <u>Modbus Organization</u>.

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

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

Существуют две основные реализации протокола:

- 1. **Modbus Serial** для передачи данных с использованием последовательных интерфейсов <u>RS-</u> <u>232/RS-485;</u>
- 2. Modbus TCP для передачи данных через сети <u>TCP/IP</u>.

Modbus Serial имеет два режима передачи данных:

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

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

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

Запрос master-устройства к slave-устройству содержит:

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

Ответ slave-устройства имеет схожую структуру.

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

Таблица 2.1 -	Области данных	протокола І	Modbus
---------------	----------------	-------------	--------

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

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



ПРИМЕЧАНИЕ

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

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

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

Таблица 2.2 – Основные функции протокола Modbus



ПРИМЕЧАНИЕ

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

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

Парамотр	Two	Адрес регистра		
Параметр	1911	(hex)	(dec)	
Положение десятичной точки в целом значении для входа 2 (знач. DP)	int16	0006	6	
Целое значение измерение входа 2 со смещением точки	int16	0007	7	
Статус измерения входа 2 (код исключительной ситуации)	int16	0008	8	
Циклическое время измерения входа 2	int16	0009	9	
Измерение входа 2 в представлении с плавающей точкой	Float32	000A,000B	10,11	
•••				
Положение десятичной точки в целом значении для входа 8 (знач. DP)	int16	002A	42	
Целое значение измерение входа 8 со смещением точки	int16	002B	43	
Статус измерения входа 8 (код исключительной ситуации)	int16	002C	44	
Циклическое время измерения входа 8	int16	002D	45	
Измерение входа 8 в представлении с плавающей точкой	Float32	002E,002F	46,47	

2 При передаче 4-х байтных значений (тип Float 32) старшее слово передается в регистре с меньшим номером.

Рисунок 2.1 – Фрагмент карты регистров модуля МВ110-8А

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

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

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

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

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

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

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

2.3 Нумерация СОМ-портов в CODESYS

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



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

2.4 Особенности работы с модулями Мх110

Перед тем, как подключать модули Mx110 к контроллеру, их следует сконфигурировать с помощью программы Конфигуратор Mx110. Программа доступна на сайте <u>OBEH</u> на странице любого из модулей.

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

Установка связи с приб	бором	×
	Параметры порта	Значение
	Скорость	115200
	Длина слова данных	8
	Четность	Отсутствует
and the second se	Кол-во стоп-бит	1
M 110	Длина адреса	8
	Адрес	1
	Порт компьютера	COM9
ØIBIEH	Установить связь Сканирование сети Заводские сетевые настройки	Работать offline Выход

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

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

ПРИМЕЧАНИЕ

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

л Прибор Дерево Режимы программы Справ	ка				
🖻 😻 🧏 🎭 🎭 🛸 🎭 🥩 🥖	INIT 🖆 🎓 💐 😵				
исание	≜ Параметр	Значение	Атрибут редакт	Владелец	Ошибки обмена и несоответствия
📸 Конфигурация М110 (Имя не задано)					
🛓 🗀 Параметры прибора					
🛓 🛅 Общие параметры прибора					
	dev	MV110-16			
Версия прошивки	ver	1.11			
🖃 🖵 Сетевые параметры					
Корость обмена данными	bPS	115200	Редактируемый	Пользователь	
💱 Длина слова данных	LEn	8	Редактируемый	Пользователь	
💱 Тип контроля четности слова данных	PrtY	Отсутствует	Редактируемый	Пользователь	
Количество стоп-битов в посылке	Sbit	1	Редактируемый	Пользователь	
💱 Длина сетевого адреса	A.Len	8	Редактируемый	Пользователь	
Авс Базовый адрес прибора	Addr	1	Редактируемый	Пользователь	alena len
— Авс Максимальный сетевой тайм-аут (сек)	t.out	0	Редактируемый	Пользователь	
Авс Задержка ответа по RS-485.мс	Rs.dL	2	Редактируемый	Пользователь	

Рисунок 2.4 – Настройка модуля с помощью конфигуратора

Конфигурирование модулей происходит по протоколу **OBEH**. В связи с особенностями протокола во время конфигурирования каждый модуль занимает количество адресов, равное количеству его каналов. То есть в случае попытке настраивать модули, уже находящиеся в сети и имеющие последовательные адреса (1, 2, 3), могут возникнуть ошибки. Если предполагается, что в будущем может потребоваться перенастройка модулей, то следует изначально задавать адреса модулей с промежутками, равными числу каналов в модулях.

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

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

3 Шаблоны модулей Мх110 и Мх210

3.1 Установка шаблонов модулей в среду CODESYS

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

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

Пакеты доступны на сайте компании <u>OBEH</u> в разделе <u>CODESYS V3/Библиотеки и компоненты</u>.

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



Рисунок 3.1.1 – Установка пакета шаблонов в среду CODESYS

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

🗊 Установка - Choose Setup Type 💽
Модули ввода \вывода компании OBEH [Please select the type of setup you would like to perform.
Полная установка Будут установлены все компоненты.
 Типичная установка Будут установлены наиболее используемые компоненты.
Выборочная установка Контроль того, какие компоненты пакеты будут установлены. Рекомендуется для продвинутых пользователей.
Cancel < Back Next > Finish

Рисунок 3.1.2 – Начало установки шаблонов модулей

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

🗇 Установка - Setup Completed 🧾	
Модули ввода\вывода компании ОВЕН [
The package has been successfully installed. Click Finish to exit the wizard or Next to see the summary.	
Cancel < Back Next > Finish	

Рисунок 3.1.3 – Завершение установки шаблонов модулей

3.2 Пример: СПК1хх [M01] + модули Mx110

В качестве примера будет рассмотрена настройка обмена с модулями <u>Mx110</u> (MB110-8A, MB110-16Д, MУ110-16Р) с использованием **шаблонов**. В примере используются шаблоны версии **3.5.11.4**.

Реализуемый алгоритм: если значение первого аналогового входа модуля **MB110-8A** превышает **30** и при этом первый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **MУ110-16P** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).



Структурная схема примера приведена на рисунке ниже:

Рисунок 3.2.1 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.x. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания: <u>Example_TemplatesMx110_3511v1.projectarchive</u>

Сетевые параметры модулей приведены в таблице ниже:

Параметр	MB110-8A	МВ110-16Д	МУ110-16Р	
СОМ-порт контроллера, к которому	COM1 COM2		NM2	
подключен модуль	COMIT		////2	
ID COM-порта	1	2		
Адрес модуля	1	1 1 17		
Скорость обмена	115200			
Количество бит данных	8			
Контроль четности	Отсутствует			
Количество стоп-бит	1			

Для настройки обмена следует:

1. Настроить модули **Mx110** с помощью программы **Конфигуратор Mx110** в соответствии с таблицей 3.2.1. Подключить модули к СОМ-портам контроллера в соответствии с <u>рисунком3.2.1</u>.

- 2. Установить пакет шаблонов модулей Mx110 в CODESYS (см. п. 3.1.1).
- 3. Создать новый проект CODESYS с программой PLC_PRG на языке CFC:

Стандартны	ій проект		x
	Вы собирает следующие о - Одно прогр - Программа - Циклическа - Ссылка на н	есь создать новый стандартный проект. При этом будут созданы бъекты: аммируемое устройство, как задано ниже PLC_PRG на языке, выбранном ниже я задача, вызывающая PLC_PRG ювейшую установленную версию библиотеки Standard library.	
	Устройство: PLC_PRG на:	SPK1xx[M01] (Production association OWEN) Непрерывные функциональные схемы (CFC) ОК От	мена

Рисунок 3.2.2 – Создание проекта CODESYS

4. Объявить в программе следующие переменные:

1) PLC	C_PRG X		
	1	PROGRAM PLC_PRG		
	2	VAR		
	з	rAnalogInput1:	REAL;	// 1-й вход модуля MB110-8A
	4	xDiscreteInput1:	BOOL;	// 1-й вход модуля MB110-16Д
	5	xDiscreteOutput1:	BOOL;	// 1-й выход модуля MV110-16P
	6	END_VAR		
	7			

Рисунок 3.2.3 – Объявление переменных в программе PLC_PRG

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

ПРИМЕЧАНИЕ

i

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 3.2.4 – Добавление компонента Modbus COM

В конфигурации СОМ-портов следует указать <u>номера СОМ-портов</u> и сетевые настройки в соответствии с <u>таблицей 3.2.1</u>:

Сомі 🗙		
Общее	Конфигурация последоват	ельного порта
SerialPort Конфигурация	СОМ-порт	1
Состояние	Скорость передачи	115200 🔻
	Четность	NONE
Информация	Биты информации	8
	Стоповые биты	1

Рисунок 3.2.5 – Настройки СОМ-порта СОМ1

6. В каждый из СОМ-портов добавить компонент Modbus Master.



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 3.2.6 – Добавление компонента Modbus Master



Рисунок 3.2.7 – Внешний вид дерева проекта после добавления Modbus Master

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

Modbus_Master_COM_Port	×		
Общее	Modbus-RTU/ASCII		MODRUS
ModbusGenericSerialMaster Соотнесение входов/выходов	Режим передачи	RTU O ASCII	MUDDUJ
Состояние	Таймаут ответа (мс)	1000	
	Время между фреймами (мс)	20	
Информация	👿 Автоперезапуск соединен	ия	

Рисунок 3.2.8 – Настройки компонентов Modbus Master

7. В компонент Modbus Master порта COM1 следует добавить шаблон модуля MB110-8A, а в Modbus Master порта COM2 – MB110-16Д и MУ110-16P.

ПРИМЕЧАНИЕ

i

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 3.2.9 – Добавление шаблонов модулей в проект CODESYS

Устройства	-
Example_TemplatesMx110_3511v1	
E Device (SPK1xx[M01])	
🖻 🗐 Plc Logic	
🖹 🔘 Application	
🎬 🎁 Менеджер библиотек	
DIC_PRG (PRG)	
🖹 🎆 Конфигурация задач	
🗎 🕸 MainTask	
PLC_PRG	
🖻 🕸 OwenCloudTask	
OwenStorage.CLOUD_PRG	
🖻 💮 COM1 (Modbus COM)	
🖻 👘 Modbus_Master_COM_Port (Modbus Master, COM Port)	
MV110_8A (MV110-8A)	
🖃 💮 COM2 (Modbus COM)	
🖹 👘 Modbus_Master_COM_Port_1 (Modbus Master, COM Port)	
MV110_16D_DN (MV110-16D_DN)	
MU110_16R_K (MU110-16R_K)	
OwenRTC (OwenRTC)	
OwenCloud (OwenCloud)	
Buzzer (Buzzer)	
Drives (Drives)	
····몹 Network (Network)	
Screen (Screen)	
Debug (Debug)	
Info (Info)	
Watchdog (Watchdog)	

Рисунок 3.2.10 – Внешний вид дерева проекта после добавления шаблонов модулей

В настройках шаблонов следует указать адреса модулей согласно <u>таблице 3.2.1</u> (МВ110-8А – адрес 1, МУ110-16Р – адрес 17):

MV110_8A X			
Общее	Modbus-RTU/ASCII		MODDIIC
ModbusGenericSerialSlave Соотнесение входов/выходов	Адрес слейва [1247]	1	MUDDU3
Mx110_Driver Соотнесение входов/выходов	Таймаут ответа (мс)	1000	
Состояние			
Информация			

Рисунок 3.2.11 – Настройка шаблона модуля MV110_8A

8. На вкладке **Mx110_Driver Соотнесение входов/выходов** привязать переменные программы к каналам шаблонов в соответствии с данной таблицей:

Таблица 3.2.2 – Привязка переменных к каналам шаблонов

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

(MV110_8A X							
Общее	Найти переменную	Фильтр	Показать все		•		
ModbusGenericSerialSlave	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
Соотнесение входов/выходов	*		Отключить модуль	%QX0.0	BIT		Исключить модуль из опроса
Mx110_Driver Coothecenue	**		Флаг ошибки	%IX84.0	BIT		Признак ошибки опроса модуля
входов/выходов	🖻 🥅 Вход 1						
Состояние	- *		Код статуса	%IW43	Enumeration of UINT		Статус измерения входа
	*		Циклическое время	%IW51	UINT		Циклическое время входа
Информация	Application.PLC_PRG.rAnalogInput1	*	Измеренное значение	%ID30	REAL		Измеренное значение входа с плавающей точкой
	🕮 🗀 Вход 2						
	🕮 🚞 Вход 3						
	🕮 🧰 Вход 4						
	🕮 🚞 Вход 5						
	🕮 🚞 Вход б						
	🖷 - 🦢 Вход 7						



(MV110_16D_DN x							
Общее	Найти переменную		Показать все		•		
ModbusGenericSerialSlave	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
Соотнесение входов/выходов			Отключить модуль	%QX66.0	BIT		Исключить модуль из опроса
Mx110_Driver Соотнесение	*>		Флаг ошибки	%IX186.0	BIT		Признак ошибки опроса модуля
входов/выходов	*		Маска входов	%IW94	WORD		Чтение состояния входов
Состояние	Application.PLC_PRG.xDiscreteInput1	*	Вход 1	%IX188.0	BOOL		
			Вход 2	%IX188.1	BOOL		
Информация	1		Вход 3	%IX188.2	BOOL		
	· · · · · · · · · · · · · · · · · ·		Вход 4	%IX188.3	BOOL		
	*		Вход 5	%IX188.4	BOOL		



/ [MU110_16R_K X							
Общее	Найти переменную	Фильтр	Показать все	•			
ModbusGenericSerialSlave	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
Соотнесение входов/выходов	ferra 🍫		Отключить модуль	%QX168.0	BIT		Исключить модуль из опроса
Mx110_Driver Соотнесение			Флаг ошибки	%IX224.0	BIT		Признак ошибки опроса модуля
входов/выходов			Маска выходов (запись)	%QW85	WORD		Запись состояния выходов
Состояние	Application.PLC_PRG.xDiscreteOutput1	*	Выход 1	%QX170.0	BOOL		
			Выход 2	%QX170.1	BOOL		
Информация			Выход 3	%QX170.2	BOOL		
			Выход 4	%QX170.3	BOOL		
			Выход 5	%QX170.4	BOOL		
	- L I		Выход 6	%QX170.5	BOOL		

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

9. Код программы PLC_PRG будет выглядеть следующим образом:



Рисунок 3.2.15 – Код программы на языке CFC

10. Загрузить проект в контроллер и запустить его.

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

Если значение rAnalogInput1 превысит 30 и при этом значение xDiscreteInput1 будет равно TRUE, то в переменную xDiscreteOutput1 будет записано значение TRUE, что приведет к замыканию первого дискретного выхода модуля MY110-16P. Если одно из условий перестанет выполняться, то выход будет разомкнут.



Рисунок 3.2.16 – Выполнение программы в режиме Online



ПРИМЕЧАНИЕ

Более подробная информация о настройках компонентов **ModbusCom** и **Modbus Master** приведена в <u>п. 4.2</u>.

3.3 Пример: СПК1хх [M01] + модули Mx210

В качестве примера будет рассмотрена настройка обмена с модулями Mx210 (MB210-101 и MK210-301) с использованием **шаблонов**. В примере используются шаблоны версии **3.5.11.4**.

Реализуемый алгоритм: если значение первого аналогового входа модуля **MB210-101** превышает **30** и при этом первый дискретный вход модуля **MK210-301** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **MK210-301** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).

Структурная схема примера приведена на рисунке ниже:



Рисунок 3.3.1 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.x. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания: <u>Example_TemplatesMx210_3511v1.projectarchive</u> Видеоверсия примера доступна по <u>ссылке</u>.

Сетевые параметры устройств приведены в таблице ниже:

Таблица 3.3.1 – Сетевые параметры устройств

Параметр	СПК1хх [М01]	MK210-301	MB210-101		
ІР-адрес	10.2.11.170	10.2.11.180	10.2.11.181		
Маска подсети	255.255.0.0				
IP-адрес шлюза	10.2.1.1				

Для настройки обмена следует:

1. Настроить модули **Mx210** с помощью программы **OBEH Конфигуратор** в соответствии с таблицей 3.3.1 (см. руководство **Mx210. Примеры настройки обмена**). Подключить модули к контроллеру.

2. Установить пакет шаблонов модулей Mx210 в CODESYS (см. п. 3.1.1).

3. Создать новый проект CODESYS с программой PLC_PRG на языке CFC:

Стандартнь	ій проект	x			
	Вы собираетесь создать новый стандартный проект. При этом будут созданы следующие объекты: - Одно программируемое устройство, как задано ниже - Программа PLC_PRG на языке, выбранном ниже - Циклическая задача, вызывающая PLC_PRG				
	- Ссылка на новейшую установленную версию библиотеки Standard library. Устройство: SPK1xx[M01] (Production association OWEN)	•			
	PLC_PRG на: Henpepывные функциональные схемы (CFC)	•			
	ОК От	мена			

Рисунок 3.3.2 – Создание проекта CODESYS

4. Объявить в программе следующие переменные:

	PLC	_PRG X		
	1	PROGRAM PLC_PRG		
-	2	VAR		
	3	rAnalogInput1:	REAL;	// 1-й вход модуля MB210-101
	4	xDiscreteInput1:	BOOL;	// 1-й вход модуля МК201-301
	5	xDiscreteOutput1:	BOOL;	// 1-й выход модуля МК201-301
	6	END_VAR		

Рисунок 3.3.3 – Объявление переменных в программе PLC_PRG

5. Добавить в проект компонент Ethernet.



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 3.3.4 – Добавление компонента Ethernet

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

Device X			
Установки соединения	Сканировать сеть	Gateway 👻 Устройство 👻	
Приложения			
Резервное копирование и восстановление			
Файлы			
Журнал		Gateway	▼ [[0000.08AA] (актия.)
Установки ПЛК		IP-Address:	Имя устройства:
Оболочка ПЛК		Port:	ыз Адрес устройства:
Пользователи и группы		1217	0000.0BAA
Размещение задачи			1D Tapreta: 1628 0072
Состояние			Тип таргета: 4096
Информация			Производитель таргета: Owen
			Версия таргета: 3.5.11.50
	1		

Рисунок 3.3.5 – Подключение к контроллеру

Ethernet 🗙	
Конфигурация ethernet	Интерфейс: еф0
Состояние	IP-адрес 10 . 2 . 11 . 170
Ethernet Device Соотнесение входов/выходов	Маска подсети 255 . 255 . 0 . 0
Информация	Gateway по умолчанию 10 . 2 . 1 . 1 Adjust Operating System Settings
	Сетевые адаптеры
	Интерфейсы:
	имя Описание іг-адрес
	10 127.0.0.1
	etto 0000
	ush0 10.0.6.10
	10.0.0.10
	IP-адрес 10 . 2 . 11 . 170
	Маска подсети 255 . 255 . 0 . 0
	Gateway по умолчанию 10 . 2 . 1 . 1
	MAC-agpec 6A:77:00:DD:BB:AA
	ОК Отмена

Рисунок 3.3.6 – Выбор используемого интерфейса



ПРИМЕЧАНИЕ

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

6. В компонент Ethernet добавить компонент Modbus TCP Master.



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.

стройства		+ 4 X		I Deferre					
Example_TemplatesMx210_3511v1				Ш дооавил	вустроиство				
Device (SPK1xx[M01])				Имя: Мо	dbus TCP Master				
E Plc Logic				0					
Application				деяствие					
Менеджер библио	тек			• Добав	ить устроиство 💿 Вставить устр	О Подключить устроиство	О Обновить	устроиство	
PLC_PRG (PRG)				Вледите с	троку для полнотехстового понока	TO DOUGROANTERN: (All verdore)			•
🖃 🎦 Конфигурация зад	lau					Chi Vendu 22			
B S MainTask				Имя		Производитель	Версия	Описание:	<u> </u>
- D PLC_PRG				8- 🗂 Pi	ан.				
= S Overcloud as	к 	0.0.000		8- 🗐 n	ромышленные сети (fieldbus)				
Ethernet (Ethernet)	ige.c.				Ethernet/IP				
() OwenRTC (OwenRTC)	8	Вырезать			4 Modbus				
A OwenCloud (OwenCloud)	Re:	Копировать			Macrep Modbus TCP				
Buzzer (Buzzer)		Вставить			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.4.0.0	A device that works as a Modbus Master on Ethernet.	
Drives (Drives)	Y	Veanuts			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.4.1.0	A device that works as a Modbus Master on Ethernet.	
Retwork (Network)	1	Americ			Modbus TCP Master	35 - smart Sottware Solutions GmbH	3.4.2.0	A device that works as a modpus master on Ethernet.	
C Screen (Screen)		Обзор			Madeus TCP Master	35 - Smart Software Solutions GmbH	3.4.3.0	A device that works as a modeus Master on Ethernet.	8
Debug (Debug)		Рефакторинг •			Madhus TCP Master	25 - Smart Software Solutions GmbH	2 5 1 0	A device that works as a modeus Master on Ethernet.	
Info (Info)	e	Conferen			Modbus TCP Master	25 - Smart Software Solutions GmbH	2 5 2 0	A device that works as a Modbus Master on Ethernet	
A Watchdog (Watchdog)	-	Courteen			Modeur TCP Master	35 - Smart Software Solutions GmbH	3 5 3 0	A device that works as a Modeur Marter on Ethernet	
	111	Добавление объекта			- Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.4.0	A device that works as a Morbus Master on Ethernet.	
	8	Добавить папку			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.5.0	A device that works as a Modbus Master on Ethernet.	
		Добавить устройство	\rightarrow		Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.6.0	A device that works as a Modbus Master on Ethernet.	
		Вставить устройство			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.8.10	A device that works as a Modbus Master on Ethernet.	
		OTKA VCTOOUCTRO			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.9.0	A device that works as a Modbus Master on Ethernet.	
		Of an an american			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.10.0	A device that works as a Modbus Master on Ethernet.	
		обновить устроиство			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.11.0	A device that works as a Modbus Master on Ethernet.	
		Редактировать объект			- 🚮 Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.11.29	A device that works as a Modbus Master on Ethernet.	
		Редактировать объект в			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.11.30	A device that works as a Modbus Master on Ethernet.	
		Изменить І/О-соотнесение			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.12.0	A device that works as a Modbus Master on Ethernet.	-
		Импорт соотнесений из CSV							
		BYCDODTHOOBATH COOTHECOHING & CSV							
				-					
		Эмуляция		V i pynn	ировать по категориям	жать все верски (для экспертов)	казать устар	евшие версии	
				<u>іі</u> и	мя: Modbus TCP Master			A	
				1 FC	роизводитель: 35 - Smart Software руппы: Mactep Modbus TCP	Solutions GmbH			
				B	ерсия: 3.5.11.30			3	2
				H O	омер модели: - писание: A device that works as a Mo	dbus Master on Ethernet.			5
								•	
				Лобарит	ь выбранное устройство как рос	леднего потомка			
				Ethernet	yesponetoo kak ilot	in the second			
				0 (Mo	жно выбрать другой таргет-узео, о	жа окно, открыто.)			
				- (HO	white entry one approximation reprict-young in				
								Добавить устройство	Закрыть

Рисунок 3.3.7 – Добавление компонента Modbus TCP Master

В настройках компонента вкладке Общее следует установить галочку Автоподключение.

Modbus_TCP_Master 🗙	
Общее	Modbus-TCP
ModbusTCPMaster Соотнесение входов/выходов	Таймаут ответа (мс) 1000 🚔 Таймаут сокета (мс) 10 🚔
ModbusTCPMaster Конфигурация	и автоподключение
Состояние	
Информация	

Рисунок 3.3.8 – Настройки компонента Modbus TCP Master

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



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.

			Huge MC	210 301			
Менержер Библиотек			Press Press	10_301			
PLC PRG (PRG)			Действие				
🗏 🌃 Конфигурация задач			 Добави 	ть устройство 🛛 💿 Вставит	ъ устройство 🛛 🔘 Подключить устро	йство 🔘 🕻	бновить устройство
🗏 🕄 MainTask					D		
B PLC_PRG			ведите ст	року для полнотекстового п	CALVENDO	\$>	•
😑 🎲 OwenCloudTask			Имя		Производитель	Версия	Описание:
OwenStorage.CLOUD_PRG			😑 🕤 No	омышленные сети (fieldbus)			
😑 🚮 Ethernet (Ethernet)			B- K1	Modbus			
Modbus_TCP_Master (Modbus TCP Master)				Слейв Modbus TCP			
() OwenRTC (OwenRTC)	ð	Вырезать		MK210-301	Production association OWEN	3.5.11.4	Модуль дискретного ввода/вывода МК210-301 Руководство по экс
- 🐣 OwenCloud (OwenCloud)	6	Копировать		- MK210-302	Production association OWEN	3.5.11.4	Модуль дискретного ввода/вывода МК210-302 Руководство по экс
- Buzzer (Buzzer)	12	Вставить		MK210-311	Production association OWEN	3.5.11.4	Модуль дискретного ввода/вывода МК210-311 Руководство по экс
Drives (Drives)	×	Удалить		- 🗍 MK210-312	Production association OWEN	3.5.11.4	Модуль дискретного ввода/вывода МК210-312 Руководство по эки
- H Network (Network)		Ofine		Modbus TCP Slave	3S - Smart Software Solutions GmbH	3.5.12.0	A generic Modbus device that is configured as Slave for a Modbus TCP Mas
- QF Screen (Screen)		ousep v		MU210-401	Production association OWEN	3.5.11.4	Модуль дискретного вывода МУ210-401 Руководство по эксплуата
Debug (Debug)		Рефакторинг 🕨		MU210-410	Production association OWEN	3.5.11.4	Модуль дискретного вывода МУ210-410 Руководство по эксплуата
1 Info (Info)	E	Свойства		- D MU210-501	Production association OWEN	3.5.11.4	Модуль аналогового вывода МУ210-501 Руководство по эксплуата
Watchdog (Watchdog)	1	Roberneure observe		MV210-101	Production association OWEN	3.5.11.4	Модуль аналогового вывода мв210-101 Руководство по эксплуата
				B MV210-202	Production association OWEN	3.5.11.4	Модуль дискретного ввода мв210-202 Руководство по эксплуатац
	-	дооавить папку		MV210-221	Production association OWEN	3 5 11 4	Модуль дискретного ввода М5210-201 Руководство по эксплуатац
		Добавить устройство	→	[] (W210-221	Froudcaurrassociation Official	3.3.11.4	модуль дискретного ввода мо210-221 — Руководство по акстиуатас
		Вставить устройство	<				•
		Откл. устройство					
		Обновить устройство					
	D°.	Редактировать объект				_	
		Press and a set of the set of	V I pynnu	ровать по категориям 👘 О	тооражать все версии (для экспертов)	Показа	ть устаревшие версии
		Редактировать обвект в					
		Изменить I/О-соотнесение	() UM	я: MK210-301			*
		Импорт соотнесений из CSV	fp	оизводитель: Production ass уппы: Слейв Modbus TCP	Inclation OWEN		
		Экспортировать соотнесения в CSV	Be	рсия: 3.5.11.4			
		Bety day and	On	исание:			Re .
		2117101201	Ma		005 MK 210-301		n /
			Лобавить	выбранное устройство ка			
			Modbus_1	CP_Master	in the stephene in the rolling		
			• (Mox	но выбрать другой тарсет-ч	зел, пока окно отконто.)		
			• (110)	and compare approximation reprict-y.	Service of the only birdly		
							Лобавить устройство Законть

Рисунок 3.3.9 – Добавление шаблонов модулей в проект CODESYS

В настройках модулей следует указать их IP-адреса согласно <u>таблице 3.3.1</u> (МК210-301 – **10.2.11.180**, MB210-101 – **10.2.11.181**).

🗿 МК210_301 🗙		
Общее	Modbus-TCP	
Мх210 Конфигурация	IP-адрес слейва:	10 . 2 . 11 . 180
ModbusTCPSlave Соотнесение	Unit-ID [1247]	1
входов/выходов Мх210 Соотнесение	Таймаут ответа (мс)	1000
входов/выходов	Порт	502
Состояние		
Информация		

Рисунок 3.3.10 – Настройка шаблона модуля МК210_301

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

Таблица 3.3.2 – Привязка переменных к каналам шаблонов

Переменная программы	Модуль	Канал				
rAnalogInput1	MV210_101	Входы/Вход 1/Измеренное значение				
xDiscreteInput1	MK210 201	Входы/Битовая маска входов/Вход 1				
xDiscreteOutput1	WIX210_301	Выходы/Битовая маска выходов (запись)/Выход 1				

Общее	Найти переменную	Фильтр По	казать все	•	
лх210 Конфигурация	Переменная	Соотнесение	Канал	Адрес	Тип
incere reality in pages			Исключить модуль из опроса	%QX40.0	BIT
lodbusTCPSlave Соотнесение			Флаг ошибки	%IX60.0	BIT
ходов/выходов	настройки				
1x210 Соотнесение	😑 📴 Входы				
входов/выходов	*		Вход 1	%ID56	
остояние	Application.PLC_PRG.rAnalogInput1	۵۵	Значение	%ID56	REAL
			Циклическое время	%IW114	UINT
нформация			Код статуса	%IB230	Enumeration of USI
	🗎 🖷 📲		Вход 2	%ID58	
	🗰 - 🏘		Вход 3	%ID60	
	- · · · · · · · · · · · · · · · · · · ·		Вход 4	%ID62	
	🕮 - 🍫		Вход 5	%ID64	
	🗎 🍫		Вход 6	%ID66	
			Вход 7	%ID68	
	🚊 🦄		Вход 8	%ID70	

Рисунок 3.3.11 – Привязка переменной к шаблону модуля МВ210-101

ПРИМЕЧАНИЕ

i

Конфигурационные параметры модулей доступны для изменения на вкладке **Мх210 Конфигурация**. В частности, на этой вкладке задается минимальный период опроса модулей и режим записи выходов (для модулей с выходами) – циклически или по изменению.

Общее	Найти переменную	Фильтр Пока	зать все	•		
Му210 Конфигурация	v210 Конфитурация		Канал	Адрес	Тип	Единица
nix210 Konipin y pagini			Исключить модуль из опроса	%QX2.0	BIT	
ModbusTCPSlave Соотнесение			Флаг ошибки	%IX0.0	BIT	
входов/выходов	🗐 📴 Входы					
Мх210 Соотнесение	🖨 - 🍬		Битовая маска входов	%IB1	BYTE	
лодов/выходов	Application.PLC_PRG.xDiscreteInput1	٩	Вход 1	%IX1.0	BOOL	
Состояние	**		Вход 2	%IX1.1	BOOL	
	*		Вход 3	%IX1.2	BOOL	
Інформация	**		Вход 4	%IX1.3	BOOL	
	*		Вход 5	%IX1.4	BOOL	
	- *		Вход б	%IX1.5	BOOL	
	🗎 🗀 Счетчик входа 1					
	🗉 🚞 Счетчик входа 2					
	🗉 🗀 Счетчик входа 3					
	🗉 🚞 Счетчик входа 4					
	🕮 🛄 Счетчик входа 5					
	🗉 🚞 Счетчик входа б					
	🖻 🔤 Выходы					
	🛱 🍗		Битовая маска выходов (чтение)	%IB2	BYTE	
	*		Битовая маска выходов (запись)	%QB3	BYTE	
	Application.PLC_PRG.xDiscreteOutput1	¢	Выход 1	%QX3.0	BOOL	
	*		Выход 2	%QX3.1	BOOL	
	*		Выход 3	%QX3.2	BOOL	
			Выход 4	%QX3.3	BOOL	
	* @		Выход 5	%QX3.4	BOOL	
			Выход 6	%QX3.5	BOOL	
	*		Выход 7	%QX3.6	BOOL	
	L		Выход 8	%QX3.7	BOOL	

Рисунок 3.3.12 – Привязка переменной к шаблону модуля МК210-301

9. Код программы PLC_PRG будет выглядеть следующим образом:



Рисунок 3.3.13 – Код программы на языке CFC

10. Загрузить проект в контроллер и запустить его.

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

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

i

ПРИМЕЧАНИЕ

Более подробная информация о настройках компонентов **Ethernet** и **Modbus TCP Master** приведена в <u>п. 4.4</u>.

3.4 Диагностика и управление обменом

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

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

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

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



Рисунок 3.4.1 – Переменные диагностики модулей Мх210

Название	Тип	Описание		
ErrorsCount		Количество ошибок связи с модулем с		
Litorscount	OBINI	момента последней загрузки контроллера		
FactoryNumber	STRING	Заводской номер модуля		
lp	ARRAY [03] OF BYTE	IP-адрес модуля, заданный в шаблоне		
LastError	loDrvModbusTCP.MB_ErrorCodes	Код последней ошибки обмена		
Name	STRING	Имя модуля, заданное в его прошивке		
PollState	loDrvMx210.MODULE_STATE	Состояние обмена с модулем		
Port	UINT	Порт Modbus TCP, заданный в шаблоне		
ProjectName	STRING	Имя модуля в дереве проекта		
SoftwareVersion	STRING	Версия прошивки модуля		

Таблица 3 4 1 – Список пе	оменных пиагностики ша	блонов молулей Мх210
гаолица э. ч . г – описок пе	леменных диагностики ша	олопов модулей міхсто

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

PLC_PRG X						
Device.Application.PLC_PRG						
Выражение	Тип	Значение		Подготовленное	Адрес	Комментарий
ø sName	STRING	'MK210-312'				иня модуля, считанное с модуля
sProjectName	STRING	'MK210_312'				имя модуля в дереве проекта
abyProjectIp	ARRAY [03] OF BYTE					IP-адрес модуля в дереве проекта
abyProjectIp[0]	BYTE	10				
abyProjectIp[1]	BYTE	2				
abyProjectIp[2]	BYTE	25				
abyProjectIp[3]	BYTE	218				
SFactoryNumber	STRING	67613190332110395				заводской номер модуля
SoftwareVersion	STRING	0.15.9				версия прошивки модуля
duerrorscount	UDINI MD 500 00055					числа ошиоок связи с модулем с последней загрузки контроллера
ecasterior apoliState	MODULE STATE	OK				
· · · · · · · · · · · · · · · · · · ·	MODOLL_STATE	UK				
1 SName WK210	312'	•=	MK210 31	2 OwenDri	ver Nam	e:
	012	•	111210_01			
2 SProjectName	'MK210_312'	:=	MK210_31	2_OwenDri	ver.Pro	jectName;
3 🔵 abyProjectIp		:=	MK210_31	2_OwenDri	ver.Ip;	
4 🌑 sFactoryNumbe	er '6761319033	، ا	MK210_31	2_OwenDri	ver.Fac	toryNumber;
5 🔵 sSoftwareVers	sion '0.15.9	:=	MK210_31	2_OwenDri	ver.Sof	twareVersion;
6 🔵 udiErrorsCou	6 udiErrorsCount 1 := MK210 312 OwenDriver.ErrorsCount;				orsCount;	
7 🔵 eLastError	ELASTERIOR UNDEFINED := MK210 312 OwenDriver.LastError;					tError;
8 🔵 ePollState	8 ePollState OK := MK210_312_OwenDriver.PollState; METURN					lState; RETURN

Рисунок 3.4.2 – Использование переменных диагностики

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

MV210_101 X	
Общее	Общее
Мх210 Конфигурация	Производитеть: Production association OWEN Группы: Слейв Modbus TCP
ModbusTCPSlave Соотнесение входов/выходов	ID: 1628 0011 Bepcus: 3.5.11.4
Мх210 Соотнесение входов/выходов	Номер модели: - Описание: Модуль аналогового вывода МВ210-101
Состояние	Руководство по эксплуатации на нодуль, схемы подключения, конфигуратор и прочая информация доступны по ссылка: https://www.owen.ru/product/moduli_analogovogo_vvoda_s_universal_nimi_vhodami_ethernet_mv210/documentation_and_software_
Информация	При записи настроек в модуль: в параметр "Положение десятичной точки" ВСЕГДА записывается значение 0.0; если "Наклон характеристики" равен 0, в модуль записывается значение по умолчанию: 1.0; если параметры "Верхняя граница" и "Нижняя граница" одновременно равны 0.0, в модуль записываются значения 100.0 и 0.0 соответственно.
	Состояние модуля опрашивается запросами со следующими параметрами:
	Адрес Количество Частота Функция первого регистров Назначение опроса Модоця регистра в запросе
	Версия встроенного **** 16#03 16#F000 32 ПО и имя прибора
	Заводской номер **** 16#03 16#F084 16 прибора
	Настройки ** 16#03 16#1004 12 входа 1 (чтение)
	Настройки ** 16#03 16#1074 12 входа 8 (чтение)
	Состояние * 16#03 16#0FA0 24 аналоговых входов

Рисунок 3.4.3 – Информация об устройстве шаблона

Информация о переменных диагностики стандартных компонентов (**Modbus Master, Modbus Slave** и т. д.) приведена в <u>п. 4.6</u>.

3.5 Библиотеки Mx Assistant

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

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

👔 Менеджер библиотек 🗙							
🗖 Добавить библиотеку 🗙 Удалить библиотеку 📑 Свойства 📷 Детали	🛐 Плейсхолд	еры 🏾 🎁 Репозиторий	библиот	тек			
Имя		Дополнительное им	ія Де	йствующая версия			
		IoDrvMx210	3.5	.11.4			
IoStandard = IoStandard, 3.5.10.0 (System)		IoStandard	3.5	. 10.0			
Mx210 Assistant, 3.5.11.4 (Production association OWEN)		Mx210Assistant	3.5	.11.4			
OwenStorage, 3.5.11.30 (Production association OWEN)		OwenStorage	3.5	.11.30			
Screen, 3.5.11.30 (Production association OWEN)		Screen	3.5	.11.30			
🕬 Standard = Standard, 3.5.9.0 (System)		Standard	3.5	.9.0			
• Watchdog, * (Production association OWEN)		Watchdog	3.5	.11.5001			
ANALOG_SENSOR_ERRORS	Имя	1	Тип	Наследовано от	Адрес	Начальн.	Комментарий
	NO	_ERROR	USINT			0	Нет ошибок
	NO	DAC_CONNECTION	USINT			1	Нет связи с ЦАП
	NO	LOAD	USINT			2	Отсутствие нагрузки
ANALOG SENSOR FROM TO WSTRING	DA	C_OVERHEAT	USINT			3	Перегрев ЦАП
	CRO	C_ERROR	USINT			4	Ошибка CRC
ANALOG SENSOR TO WSTRING	CH/	ANNEL_DISABLED	USINT			5	Канал отключен
Structs	SHO	ORT_CIRCUIT	USINT			6	Короткое замыкание
ANALOG_SENSOR_SETTINGS							

Рисунок 3.5.1 – Библиотека Mx210 Assistant в Менеджере библиотек

Таблица	3.5.1 -	Содержимс	е библиотеки	Mx110	Assistant
таолица	0.0.1	оодоржиние			Assistant

Название	Тип	Описание
	Перечисление	Код статуса аналогового входа модуля МВ110-
		2А/2АС/8А/8АС. Если зафиксирована ошибка обмена
MV_SENSOR_ERROR		(канал Флаг ошибки имеет значение TRUE), то будет
		возвращено значение MODBUS_ERROR
MvStatusToString	Функция	Функция преобразует код статуса в строку типа STRING
MvStatusToWstring	Функция	Функция преобразует код статуса в строку типа WSTRING

PLC_PRG X		Каналы				
1 PROGRAM PLC_PRG		Переменная	Соотнесение	Канал	Адрес	Тип
2 VAR		- TV		Отключить модуль	%QX0.1	BIT
3 eStatus:	Mx110Assistant.MV_SENSOR_ERROR; STDING:	- *>		Флаг ошибки	%IX92.0	BIT
5 END VAR	STRING,	🖃 — 🧰 Вход 1				
6		Application.PLC_PRG.eStatus	*≱	Код статуса	%IW47	Enumeration of UINT
		-*		Циклическое время	%IW55	UINT
1 eStatue :- My110Ae	eistant MuStatueToString(aStatue):	- *		Измеренное значение	%ID32	REAL
2	Sistent involution of the colored of	1 · · ·				

Рисунок 3.5.2 – Пример работы с функциями библиотеки Mx110 Assistant

Таблица 3.5.2 – Содержимое	библиотеки М	Ix210 Assistant
----------------------------	--------------	-----------------

Название	Тип	Описание
ANALOG_SENSOR_VALUE	Структура	Структура параметров аналогового входа модуля MB210-101
ANALOG_SENSOR_SETTINGS	Структура	Структура настроек аналогового входа модуля МВ210-101
ANALOG_SENSORS	Перечисление	Код типа датчика для аналогового входа модуля MB210-101
ANALOG_SENSOR_ERRORS	Перечисление	Код статуса аналогового входа модуля MB210-101 . Если зафиксирована ошибка обмена (канал Флаг ошибки имеет значение TRUE), то будет возвращено значение FIELDBUS_ERROR
ANALOG_OUTPUT_ERRORS	Перечисление	Код статуса аналогового выхода модуля МУ210-501
ANALOG_SENSOR_TO_STRING	Функция	Функция преобразует код типа датчика в строку типа STRING
ANALOG_SENSOR_TO_WSTRING	Функция	Функция преобразует код типа датчика в строку типа WSTRING
ANALOG_SENSOR_ERROR_TO_STRING	Функция	Функция преобразует код статуса аналогового входа в строку типа STRING
ANALOG_SENSOR_TO_ERROR_WSTRING	Функция	Функция преобразует код статуса аналогового входа в строку типа WSTRING
ANALOG_OUTPUT_ERROR_TO_STRING	Функция	Функция преобразует код статуса аналогового выхода в строку типа STRING
ANALOG_OUTPUT_ERROR_TO_WSTRING	Функция	Функция преобразует код статуса аналогового выхода в строку типа WSTRING

4 Стандартные средства конфигурирования

4.1 Общая методика конфигурирования интерфейсов

Настройка обмена в СОДЕЗҮЗ состоит из следующих действий:



Рисунок 4.1.1 – Последовательность конфигурирования Modbus в CODESYS

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

4.2 Настройка контроллера в режиме Modbus Serial Master

Для настройки контроллера в режиме Modbus Serial Master следует:

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

ПРИМЕЧАНИЕ

i

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



Рисунок 4.2.1 – Добавление компонента Modbus COM

В настройках компонента на вкладке Общее следует указать <u>номер СОМ-порта</u> контроллера и его сетевые настройки.

💮 СОМ1 🗙				
Общее	Конфигурация последовательного порта			
SerialPort Конфигурация	СОМ-порт	1		
6	Скорость передачи	115200 ▼		
Состояние	Четность	NONE		
Информация	Биты информации	8		
	Стоповые биты	1		

Рисунок 4.2.2 – Настройки компонента Modbus COM

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

Таблица 4.2.1 – Настройки компонента Modbus COM

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

i

ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 4.2.3 – Добавление компонента Modbus Master

В настройках компонента на вкладке Общее следует задать настройки мастера.

Modbus_Master_COM_Port X				
Общее	Modbus-RTU/ASCII		MODBIIC	
ModbusGenericSerialMaster Соотнесение входов/выходов	Режим передачи		MUDBUS	
Состояние	Таймаут ответа (мс)	1000		
	Время между фреймами (мс)	10		
Информация	📝 Автоперезапуск соединения			

Рисунок 4.2.4 – Настройки компонента Modbus Master
Параметр	Описание					
Режим передачи	Тип протокола обмена: Modbus RTU или Modbus ASCII					
	Время (в мс), в течение которого master ожидает ответа slave-					
	устройства. В случае отсутствия ответа по истечению этого					
	времени master делает паузу на время между фреймами и					
Таймаут отрета	переходит к опросу следующего slave-устройства. Значение,					
	введенное здесь, будет по умолчанию использоваться для всех					
	slave-устройств. На вкладке Конфигурация Modbus Slave (см.					
	рисунок 4.2.6) для каждого устройства можно задать					
	индивидуальный таймаут отклика					
	Время (в мс) между получением ответа от slave-устройства и					
	началом опроса следующего. Чем выше скорость, тем меньшим					
	может быть это значение (на скорости 115200 бит/с – 5–20 мс).					
	Некоторые устройства в течение определенного времени					
фреймами	(например, <u>ОВЕН СМИ2</u> – на 50 мс) удерживают линию связи					
фреимами	после ответа, поэтому в данном случае не имеет смысла					
	выставлять время между фреймами меньше, чем это значение.					
	При работе с модулями Мх110 рекомендуется использовать					
	значение 20 мс					
	В случае отсутствия галочки не ответившее slave-устройство					
	исключается из дальнейшего опроса. Настоятельно					
соединения	рекомендуется всегда включать эту опцию					

Таблица 4.2.2 – Настройки компонента Modbus Master

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

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 4.2.5 – Добавление компонента Modbus Slave

Modbus_Slave_COM_Port >	×]		
Общее	Modbus-RTU/ASCII		MONDIIS
Канал Modbus Slave	Адрес слейва [1247]	1	MUDDUJ
Modbus Slave Init	Таймаут ответа (мс)	1000	
ModbusGenericSerialSlave Соотнесение входов/выходов			
Состояние			
Информация			

Рисунок 4.2.6 – Настройки компонента Modbus Slave, вкладка Общее

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



ПРИМЕЧАНИЕ

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

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

щее	Имя Тип доступа Триггер Сдвиг READ Длина Обработка ошибок Сдвиг WRITE Длина
нал Modbus Slave	
dbus Slave Init	
dbusGenericSerialSlave отнесение входов/выходов	Vauxa Madhua
стояние	
рормация	Kanan Champel a
popriodina	THIS REAL Holding Registers (Kon dynamics)
	Комменталий
	Peructo READ
	Сдви охооо
	Сохранить посл. значен 👻
	Peructp WRITE
	Сдвиг 0х0000 -
	Длина 1
	ОК Отмена
	· · · · · · · · · · · · · · · · · · ·

Рисунок 4.2.7 – Добавление канала Modbus Slave

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

Таблица 4.2.3 – Параметры канала Modbus Slave

Ниже приведен пример конфигурации двух каналов Modbus Slave:

Modbus_Slave_COM_Port 🗙										
Общее		Имя	Тип доступа	Триггер	Сдвиг READ	Длина	Обработка ошибок	Cдвиг WRITE	Длина	Комментарий
	0	Channel 0	Read Holding Registers (Код функции 03)	Цикл., t#100ms	16#0000	2	Сохранить посл. значение			
Kahan Moddus Slave	1	Channel 1	Write Multiple Registers (Код функции 16)	Передний фронт				16#000A	1	
Modbus Slave Init										
ModbusGenericSerialSlave Соотнесение входов/выходов										
Состояние										
Информация										

Рисунок 4.2.8 – Пример настройки каналов Modbus Slave

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

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

Modbus_Slave_COM_Port X						
Общее	Строка	Тип доступа	Cдвиг WRITE	Значение по умолчанию	Длина	Комментарий
Канал Modbus Slave	ſ	Начальное значен	ие		x	1
Modbus Slave Init		Тип доступа	Write Multi	ple Registers (Код функции 16)	•	
Соотнесение входов/выходов		Сдвиг регистра	0x0000		-	
Состояние		Длина	1			
Информация		Начальное значе	ние 100]		
		Комментарий				
				ОК Отм	іена	
			Huwa	Honuž		Изнанить
		пе	пиже	повыи	<u> </u>	здалите Измените

Рисунок 4.2.9 – Настройки вкладки Modbus Slave Init

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



ПРИМЕЧАНИЕ

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

еменная Ф	Соотнесение	Канал	Annec	Tur	-	-
•			, where	тип	Единица	Описание
1. M.		Channel 0	%IW0	ARRAY [01] OF WORD		Read Holding Registers
<u> 취</u> ~ 19		Channel 0[0]	%IW0	WORD		0x0000
È 🧤		Channel 0[1]	%IW1	WORD		0x0001
ø		Channel 1	%QX0.0	BIT		Триггерная переменная
\$		Channel 1	%QW1	ARRAY [00] OF WORD		Write Multiple Registers
È 🍫		Channel 1[0]	%QW1	WORD		0x000A
E	1 *0 0 0 *0	8-19 9 9-19 8-19	Channel 0[1] Channel 1 Channel 1 Channel 1 Channel 1 Channel 1 Channel 1	% Channel 0[1] %IW1 Channel 1 %QX0.0 Channel 1 %QW1 % Channel 1[0] %QW1	P Mp Channel 0[1] %IW1 WORD Image: Channel 1 %QX0.0 BIT Image: Channel 1 %QW1 ARRAY [00] OF WORD Image: Time 1 %QW1 ARRAY [00] OF WORD Image: Time 1 %QW1 WORD	

Рисунок 4.2.10 – Настройки вкладки ModbusGenericSerialSlave Соотнесение входов/выходов

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



Рисунок 4.2.11 – Привязка переменных программы к каналам Modbus Slave

Modbus_Slave_COM_Port	×						
Общее	Найти переменную	Фильтр Показать в	ce		•		
Kawan Modhus Slave	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
	₽-1		Channel 0	%IW0	ARRAY [01] OF WORD		Read Holding Registers
Aodbus Slave Init	Application.PLC_PRG.wRead1	۵	Channel 0[0]	%IWO	WORD		0x0000
	Application.PLC_PRG.wRead2	۵	Channel 0[1]	%IW1	WORD		0x0001
AodbusGenericSerialSlave	Application.PLC_PRG.xWriteTrigger	۵۵	Channel 1	%QX0.0	BIT		Триггерная переменная
	- · · · · · · · · · · · · · · · · · · ·		Channel 1	%QW1	ARRAY [00] OF WORD		Write Multiple Registers
Состояние	Application.PLC_PRG.wWrite	۵۵	Channel 1[0]	%QW1	WORD		0x000A
1нформация							

Рисунок 4.2.12 – Привязка переменных программы к каналам Modbus Slave

Пример настройки контроллера в режиме **Modbus Serial Master** для опроса модулей **Mx110** приведен в <u>п. 4.9</u>.

4.3 Настройка контроллера в режиме Modbus RTU Slave

Для настройки контроллера в режиме Modbus RTU Slave следует:

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



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 4.3.1 – Добавление компонента Modbus COM

Настройки компонента описаны в п. 4.2 (пп. 1).

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



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 4.3.2 – Добавление компонента Modbus Serial Device

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

Modbus_Serial_Device 🗙				
Modbus Serial Device	ID:	1	* *	
Modbus Serial Device Соотнесение входов/выходов	📄 Сторожевой таймер:	500	-	A. V
Информация	Регистр хранения (%IW):	10		*
	Входные регистры (%QW):	10		×
	Рагистры		0	
	Регистры:		U	
	Дискретные входы:		0	* *
	Регистр временного хран	ения:	0	* *
	Регистр ввода:		0	*

Рисунок 4.3.3 – Настройки компонента Modbus Serial Device

ПРИМЕЧАНИЕ

Компонент не поддерживает протокол **Modbus ASCII**. В случае необходимости работы по этому протоколу следует использовать ФБ <u>MB SerialSlave</u> из библиотеки **OwenCommunication**.

Параметр	Описание
ID	Адрес (Slave ID) контроллера в рамках выбранного СОМ-порта
	Время ожидания (в мс) запроса от master-устройства. Если за это
Сторожевой таймер	время запроса не приходит, то данные в регистрах обнуляются. В
	случае отсутствия галочки обнуления данных не происходит
Регистр хранения	Количество holding регистров для данного slave-устройства (2500)
Входные регистры	Количество input регистров для данного slave-устройства (2500)
	Начальный адрес для каждой <u>области памяти</u> Modbus. В случае
	получения запроса к регистру, адрес которого меньше, чем адрес
Пачальные адреса	начального регистра, контроллер вернет ошибку 02
	(ILLEGAL_DATA_ADDRESS)

Таблица 4.3.1 – Настройки компонента Modbus Serial Device

i

ПРИМЕЧАНИЕ

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



ПРИМЕЧАНИЕ

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



ПРИМЕЧАНИЕ

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



ПРИМЕЧАНИЕ

Holding регистры и биты (coils) могут быть изменены со стороны master-устройства и не могут быть изменены со стороны программы контроллера. В случае необходимости изменения этих данных из программы следует использовать ФБ <u>MB SerialSlave</u> из библиотеки **OwenCommunication**.



ПРИМЕЧАНИЕ

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

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



ПРИМЕЧАНИЕ

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

	паити переменную	Фильтр Показать все	2		•		
Modbus Serial Device Coothecenne	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
входов/выходов			Входы	%IW0	ARRAY [09] OF WORD		Регистры временного хранения Modbus
1нформация	B- 🍫		Входы[0]	%IW0	WORD		
	B- 10		Входы[1]	%IW1	WORD		
	B- 🍫		Входы[2]	%IW2	WORD		
			Входы[3]	%IW3	WORD	Hold	іпа пегистры
			Входы[4]	%IW4	WORD	HOIC	ing pervicipbi
	B 🍫		Входы[5]	%IW5	WORD		
	iii		Входы[6]	%IW6	WORD		
	B-10		Входы[7]	%IW7	WORD		
	🕮 - 🍫		Входы[8]	%IW8	WORD		
	<u><u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u></u></u>		Вхолы[9]	%IW9	WORD		
	÷.*•		Выходы	%QW0	ARRAY [09] OF WORD		Входные регистры Modbus
	÷*		Выходы[0]	%QW0	WORD		
	👜 - 🍫		Выходы[1]	%QW1	WORD		
	÷-*•		Выходы[2]	%QW2	WORD	Inn	
	· · · · ·		Выходы[3]	%QW3	WORD	inp	и регистры
	÷-*		Выходы[4]	%QW4	WORD		
	· · * •		Выходы[5]	%QW5	WORD		
	÷-*>		Выходы[6]	%QW6	WORD		
	B- 🍫		Выходы[7]	%QW7	WORD		
	÷-**		Выходы[8]	%QW8	WORD		
			Prove ex (10)	8(0)4/0	WORD		

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

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

Madou Sani Device Madou Sani Madou Sani Ma	Modbus_Serial_Device X						_			
Other ServiceSon Presentence Kalan Age: Train Expension Oncounter dop/mau/s Boogle Stock Stock WOOD Prescript Sponsescript Sponse	odbus Serial Device	Найти переменную		Фильтр Показать	sce		-			
dochestenziołe Biogasel % Wielewicze Biogasel Bioga	odbus Serial Device Соотнесение	Переменная		Соотнесени	е Канал	Адрес	Тип		Единица	Описание
doprese doprese the the second seco	одов/выходов	₽-₩			Входы	%IW0	ARRAY [0	9] OF WORD		Регистры временного хранения Modbus
Boacigi Mura Works Boacigi Mura Works Boacigi Mura Works Boacigi Mura Works Boacigi Mura Works Boacigi Mura Works Commentanteest Commentanteest Commentanteest Boacidest Commentanteest Boacidest Commentanteest Boacidest Boacidest Boacidest Commentanteest Boacidest Boacidest Boacidest Boacidest Commentanteest Boacidest Boacidest Boacidest Boacidest Boacidest Commentanteest Boacidest Bo	формация	🛞 · 🍫			Входы[0]	%IW0	WORD			
Booke(2) 10/12 UKOB Booke(2) 10/12 UKOB Booke(🛞 – 🍫			Входы[1]	%IW1	WORD			
Pacaci(3) % WOD Cockreter seagat Image: Comparison of the com		۰ 🏘			Входы[2]	%IW2	WORD			
Accordent soogs		1 B-10			Входы[3]	%IW3	WORD			
Tercenski nanov, Kareropski Popenerske		Ассистент ввода			0	A/ DAVA		×	N	
Image: Market in the		Текстовый пои	ск Категории							
Image: Contract of Contrecon of Contract of Contrel of Contract of Cont		Переменные		▲ Имя	Т	in .	Адрес	Ис		
Image: Crysc rps Processor recently and the recent of t				= 😳 Application	Прила	жение				
Image: Second				🖹 PLC_PRG	PRO	SRAM				Входные регистры Modbus
C BR.co				wHoldingRegister0	W	RD				
Cong. Globals Via. C. Colu Via. C				TargetVariables						
				BPLog	Библі	отека		Breakpoir		
C Typertypepocareusi au WoldingRegister0: WORD; (VAR)				# - 🎑 IoConfig_Globals	VAR_0	LOBAL				
C Visuellem Cambigalyer Biofavor record Visuellem Cambigalyer C Visuellem Cambigalyer Biofavor record Visuellem Cambigalyer C Vis				± U VisuDialogs	Библи	отека		VisuDialog		
Crpycryseponawewi au Crpycryseponawew				U VisuElemCamDisplayer	Библи	отека		VisuElemC		
Crpycrypecoareañ au Corpectano				U VisuElemsWinControls	Библи	отека		VisuElems		
Image: Cripiert puppedareadit aug Gwmarp: Her				±- C VisuElemTrace	Библі	отека		VisuElem i		
C CTPINTTYRIQOBARENIÀ BILA C CTPINTTYRIQOBARENIA C C CTPINTTYRIQO C CTPINTTYRIQOBARENIA C C CTPINTTYRIQO C C CTPINTTYRIQO C C CTPINTTYRIQO C C CTPINTTYRIQO C C C CTPINTTYRIQO C C C C C C C C C C C C C C C C C C C										
Image: Control of the control of t										
Image:										
Image: Comparison of the company o										
Croyerryspossaweski ska Qurlamp: Image: Croyerryspossaweski ska Qurlamp: Image: Croyerryspossaweski ska Chokabeleta gokymetrizavia Image: Croyerryspossaweski ska Image: Croyerryspossaweski ska Image: Croyerryspossaweski ska MelodingRegister0: WORD; Image: Croyerryspossaweski ska Image: Croyerryspossaweski ska VelodingRegister0: WORD; Image: Croyerryspossawesk Image: Croyerryspossawesk <td></td> <td></td> <td></td> <td>4</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>				4						
Crypertyperposesent sua										
Image: Contract and Contrege and Contread and Contract and Contract and Contract and Contra		Структурир	ованный вид			Фильтр: Н	ет	•		
Ackylemaus: WłoldingRegister0: WORD; (VAR) Bccr.ga ośwoansta neperiewskie: Bion. 2 (scer.ga B saaskei		Показывать	документацию	✓ B	ставка с аргумент	ами	Встави	ка с префиксом		
WłoldingRegister0: WORD; (VAR)		Документация:								
		wHoldingRed	gister0: WORD;							
		(VAR)							Всегда	обновлять переменные: Вкл. 2 (всегда в задаче цикл
		r								
		4						_		
OKOTHOUS								· · · · · · · · · · · · · · · · · · ·		
		1					UK	Отмена		

Рисунка 4.3.5 – Привязка переменных программы к регистрам Modbus RTU Slave

Пример настройки контроллера как Modbus RTU Slave приведен в <u>п. 4.10</u>.

4.4 Настройка контроллера в режиме Modbus TCP Master

Для настройки контроллера в режиме Modbus TCP Master следует:

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



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 4.4.1 – Добавление компонента Ethernet

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

становки соединения	Сканировать сеть Gatew	ау 🗸 Устройство 👻	
Триложения			
езервное копирование и			
восстановление			
Файлы		ur 🌔	•
Курнал		Gateway	<u> </u>
(yphan		Gateway-1	▼ [0000.0BAA] (актив.)
/становки ПЛК		IP-Address:	Имя устройства:
Concurs DBK		localhost	kis
		Port: 1217	Адрес устройства: 0000.08АА
Тользователи и группы			10
			1628 0072
азнещение задачи			Тип таргета:
Состояние			4096
tube susure			Производитель таргета:
лнформация			Owen
			Версия таргета:

Рисунок 4.4.2 – Подключение к контроллеру

Ethernet X							
Конфигурация ethernet	Интерфейс: еth0						
Состояние	IP-agpec 10 . 2 . 11 . 170						
Ethernet Device Соотнесение	Маска подсети 255 . 255 . 0 . 0						
Информация	Gateway по умолчанию 10 . 2 . 1 . 1						
	Adjust Operating System Settings						
	Сетевые адаптеры						
	Mutantañou:						
	Имя Описание IP-адрес						
	lo 127.0.1						
	eth0 10.2.11.170						
	sit0 0.0.0						
	usb0 10.0.6.10						
	IP-адрес 10 . 2 . 11 . 170						
	Маска подсети 255 . 255 . 0 . 0						
	Gateway по умолчанию 10 . 2 . 1 . 1						
	MAC-agpec 6A:77:00:DD:8B:AA						
	ОК Отмена						

Рисунок 4.4.3 – Выбор используемого интерфейса



ПРИМЕЧАНИЕ

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



ПРИМЕЧАНИЕ

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

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



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



ПРИМЕЧАНИЕ

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

/стройства		- + X		I Aof-					- ×
= 🎒 Example_TemplatesMx210_3511v1				ш доса	мпь устроиство				
Device (SPK1xx[M01])				Ина:	Modbus TCP Master				
😑 🗐 Pic Logic									
Application				Деист	вие				
👘 Менеджер библио	тек			• До	Бавить устройство 💿 Вставить устройс	тво 🔘 Подключить устройство	Обновить	устройство	
PLC_PRG (PRG)				Prese					_
= 🔛 Конфигурация зад	lau			росди	е строку для полнотехстового поиска вс	All Vendors>			
🗏 🐉 MainTask				Имя		Производитель	Версия	Описание:	<u>^</u>
del PLC_PRG				- E	Разн.				
i≡ - ∰ OwenCloudTas	k _			8-1	Пронышленные сети (fieldbus)				
OwenStora	ige.C	LOUD_PRG			👄 Ethernet/IP				
Ethernet (Ethernet)	X	Вырезать		9	Modbus				
Wenkic (Owenkic)	Ba	Konvogate			Macrep Modbus TCP				
Covencioud (Owencioud)					- 🛗 Modbus TCP Master	3S - Smart Software Solutions GmbH	3.4.0.0	A device that works as a Modbus Master on Ethernet.	
(Buzzer (Buzzer)	U.G	DETABNID			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.4.1.0	A device that works as a Modbus Master on Ethernet.	
Supportives (Drives)	X	Удалить			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.4.2.0	A device that works as a Modbus Master on Ethernet.	
TE Streen (Streen)		Обзор 🔸			Modbus TCP Master	3S - Smart Software Solutions GmbH	3.4.3.0	A device that works as a Modbus Master on Ethernet.	
Debug (Debug)		Perhastronier +			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.0.0	A device that works as a Modbus Master on Ethernet.	
Info (Info)					Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.1.0	A device that works as a Modbus Master on Ethernet.	
Watchdog (Watchdog)	1	Свойства			Modbus TCP Master	3S - Smart Software Solutions GmbH	3.5.2.0	A device that works as a Modbus Master on Ethernet.	
		Добавление объекта			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.3.0	A device that works as a Modbus Master on Ethernet.	
	6	Добавить папку			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.4.0	A device that works as a Modbus Master on Ethernet.	
		Лобавить устройство			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.5.0	A device that works as a Modbus Master on Ethernet.	
		Common and an and a second sec	· · · · ·		Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.0.0	A device that works as a Modbus Master on Ethernet.	
		вставить устроиство			Modbus TCP Master	35 - Smart Software Solutions Gribbl	3.5.0.10	A device that works as a Modbus Master on Ethernet.	
		Откл. устройство			Madeus TCP Master	25 - Smart Software Solutions Cable	2 5 10 0	A device that works as a Modbus Master on Ethernet.	
		Обновить устройство			Modbus TCP Master	25 - Smart Software Solutions Cable	2 5 11 0	A device that works as a Modbus Master on Ethernet.	
	Ľ	Редактировать объект			Modus TCP Master	35 - Smart Software Solutions GmbH	3 5 11 29	A device that works as a Modeus Master on Ethernet	
		Редактировать объект в			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.11.30	A device that works as a Modulus Master on Ethernet.	
		10 mm 10 mm 10 mm			11 Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.12.0	A device that works as a Modbus Master on Ethernet.	
		изменить 1/0-соотнесение							
		Импорт соотнесений из CSV							
		Экспортировать соотнесения в CSV							
		Энуляция		👽 Гру	ппировать по категориям 📝 Отображат	ь все версии (для экспертов) 📃 По	жазать устара	вшие версии	
	-								
					Muuru Madhura TCD Master				
					Производитель: 35 - Smart Software Solu	itions GmbH		Â	
					Группы: Macrep Modbus TCP			:	-
					Номер модели: -				5
					Onucanue: A device that works as a Modbus	Master on Ethernet.			-
				_					
				Добав	ить выбранное устройство как после	днего потомка			
				Ether	net				
				0 (Можно выбрать другой таргет-узел, пока	окно открыто.)			
								Добавить устройство	Закрыть

Рисунок 4.4.4 – Добавление компонента Modbus TCP Master

В настройках компонента на вкладке Общее следует задать настройки мастера.

Modbus_TCP_Master X		
Общее	Modbus-TCP	MODRUS
ModbusTCPMaster Соотнесение входов/выходов	Таймаут ответа (мс) 1000 🚽 Таймаут сокета (мс) 10 🚔	mobbod
ModbusTCPMaster Конфигурация	и автоподключение	
Состояние		
Информация		

Рисунок 4.4.5 – Настройки компонента Modbus TCP Master

Таблица 4.4.1 – Настройки компонента Modbus TCP Master

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

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



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.

		- I X	Иня: Modbus_TCP_Slave				
25			Лействие				
e (SPK tvv[M01])			A Deferre umotions	0			6
Pic Lonic			С дооавить устроиство	оставить у	строиство — — Подключить устро	MCIBO OU	оновить устроиство
Application			Введите строку для полнотекс	тового поио	ка вс Производитель: <А уелоо	s>	•
TargetVariables					-		
imagePool			Имя		Производитель	Версия	Описание:
👔 Менеджер библиотек			 Ш Промышленные сети (fi 	eldbus)			
PLC_PRG (PRG)			H Kold Modbus				
😑 🌃 Конфигурация задач			Cneuts Modbus I	CP			
🖻 🍪 MainTask			MK210-301		Production association OWEN	3.5.11.3	Модуль дискретного ввода/вывода МК 210-
PLC_PRG			MK210-301		Production association OWEN	3.5.11.4	Модуль дискретного ввода/вывода МК210-
🖶 🍪 OwenCloudTask			MK210-302		Production association OWEN	2 5 11 4	Модуль дискретного ввода/вывода мк210-
OwenStorage.CLOUD_PRG			MK210-302		Production association OWEN	3.5.11.9	Молуль дискратного ввода/вывода МС210-
E 🕼 VISU_TASK			MK210-311		Production association OWEN	3 5 11 4	MORYNE RWCYDETHOLO BEORRA/BUBORR MC210-
UsuElems.Visu_Prg			- MK210-312		Production association OWEN	3.5.11.3	Модуль дискретного ввода/вывода МК210-
Менеджер визуализации			MK210-312		Production association OWEN	3.5.11.4	Модуль дискретного ввода/вывода МК210-
Visualization			Modbus TCP	Slave	3S - Smart Software Solutions GmbH	3.5.2.0	A generic Modbus device that is configured as SI
Ethernet (Ethernet)			Modbus TCP	Slave	3S - Smart Software Solutions GmbH	3.5.3.0	A generic Modbus device that is configured as SI
Modbus_TCP_Master (Modbus TCP Master)	X	Вырезать	- 🕤 Modbus TCP	Slave	35 - Smart Software Solutions GmbH	3.5.4.0	A generic Modbus device that is configured as SI
OwenClevel (OwenClevel)	Ba	Копировать	📶 Modbus TCP	Slave	35 - Smart Software Solutions GmbH	3.5.7.0	A generic Modbus device that is configured as SI
OwenCloud (OwenCloud)			- 🗂 Modbus TCP	Slave	35 - Smart Software Solutions GmbH	3.5.10.0	A generic Modbus device that is configured as SI
Drives (Drives)	0.55	berdonito	- Modbus TCP	Slave	3S - Smart Software Solutions GmbH	3.5.11.0	A generic Modbus device that is configured as SI
Network (Network)	×	Удалить	Modbus TCP	Slave	3S - Smart Software Solutions GmbH	3.5.11.30	A generic Modbus device that is configured as SI
Screen (Screen)		Обзор +	1 Modbus TCP	Slave	35 - Smart Software Solutions GmbH	3.5.12.0	A generic Modbus device that is configured as Sl 👻
Debug (Debug)		Рефакторинг 🕨	<				•
Info (Info)	a	Свойства					
watchdog (watchdog)	853	Добавление объекта					
	-	Johanum many	💟 Группировать по категория	ам 🔽 Отоб	ражать все версии (для экспертов)	🥅 Показат	ъ устаревшие версии
	-	Access and the second					
		дооавить устроиство	Имя: Modbus TCP Slave				
		Вставить устройство	Производитель: 35 - 3	Smart Softwa	are Solutions GmbH		
		Откл. устройство	Версия: 3.5.11.30	i i CP			
		Обновить устройство	Номер модели: -	dhun dauira	that is configured as Slave for a Mediu	o TCD Master	
	Dî l	Редактировать объект	Officanie: A generic Ho	abus device	that is configured as playe for a mouble	is for Master.	1
	_	Редактировать объект в					
		Manager T/O seemingsource					
		Manager segment with CSV					
		Экспортировать соотнесения в CSV	Добавить выбранное устро	йство как г	последнего потомка		
		Эмуляция	Modbus_TCP_Master				
			• (Можно выбрать другой	таргет-узел	, пока окно открыто.)		
							Добавить устройство Зако
							Accounte / c.p.onc.30

Рисунок 4.4.6 – Добавление компонента Modbus TCP Slave

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

Modbus_TCP_Slave X			
Общее	Modbus-TCP		
Канал Modbus Slave	IP-адрес слейва:	192 . 168 . 0 . 1	MODBUS
Modbus Slave Init	Таймаут ответа (мс) Порт	1000	
ModbusTCPSlave Конфигурация			
ModbusTCPSlave Соотнесение входов/выходов			
Состояние			
Информация			

Рисунок 4.4.7 – Настройки компонента Modbus TCP Slave, вкладка Общее

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

Общее	Параметр	Тип	Значение	Значение по умолчанию	Единица	Описание
	NewChannelConfig	BOOL	true	true		Use the new Channel-Config format
анал Modbus Slave	🖤 🖗 Unit-ID	USINT	16#FF	16#FF		Unit-ID of the Device
Modbus Slave Init	ResponseTimeout	DWORD	1000	1000		Maximum time for a Slave to respond in m
	IPAddress	ARRAY[03] OF BYTE	[192, 168, 0, 1]	[192, 168, 0, 1]		Configure IP Address of TCP SLave.
ModbusTCPSlave Kondury Daug	🔷 🖗 Port	UINT	502	502		Port where the slave is listening
	ConfigVersion	UDINT	16#03050B00	16#03050B00		
odbusTCPSlave Соотнесение «одов/выходов						
остояние						

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

Настройки вкладок Канал Modbus Slave и Modbus Slave Init идентичны настройкам одноименных вкладок компонента Modbus Slave и описаны в <u>п. 4.2 (пп. 3)</u>.

4.5 Настройка контроллера в режиме Modbus TCP Slave

Для настройки контроллера в режиме Modbus TCP Slave следует:

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

ПРИМЕЧАНИЕ

i

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



Рисунок 4.5.1 – Добавление компонента Ethernet

Настройки компонента описаны в п. 4.4 (пп. 1).

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



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



ПРИМЕЧАНИЕ

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



Рисунок 4.5.2 – Добавление компонента Modbus TCP Slave Device

Страница конфигурации	Заданные параметры			
Modbus TCP Slave Device	🔲 Сторожевой таймер:	500	A. V	(мс)
Соотнесение входов/выходов	Slave-nopt:	502	*	
1нформация	Unit ID:			
	Регистры временного хранения (%IW):	10	*	
	Входные регистры (% ОW):	10	*	1
	Модель данных			
	—Модель данных Начальные адреса:			
	Модель данных Начальные адреса: Регистры:		0	
	-Модель данных Начальные адреса: Регистры: Дискретные входы:		0	
	Модель данных Начальные адреса: Регистры: Дискретные входы: Регистр временного хра	анения:	0	

На вкладке Страница конфигурации следует указать настройки slave-устройства:

Рисунок 4.5.3 – Настройки компонента Modbus TCP Slave Device

Таблица 4.5.1 – Настройки компонента Modbus TCP Slave Device

Параметр	Описание			
	Время ожидания (в мс) запроса от master-устройства. Если за это			
Сторожевой таймер	время запроса не приходит, то данные в регистрах обнуляются. В			
	случае отсутствия галочки обнуления данных не происходит			
Slave-порт	Порт, используемый для обмена (по умолчанию – 502)			
	Адрес контроллера в рамках выбранного интерфейса. Используется			
Unit ID	в случае интеграции контроллера с сетью Modbus Serial через			
	шлюзы Modbus TCP/Modbus Serial			
	Количество holding регистров для данного slave-устройства			
гегистр хранения	(24096)			
Входные регистры	Количество input регистров для данного slave-устройства (24096)			
	Начальный адрес для каждой <u>области памяти</u> Modbus. В случае			
	получения запроса к регистру, адрес которого меньше, чем адрес			
	начального регистра, контроллер вернет ошибку 02			
	(ILLEGAL_DATA_ADDRESS).			
Начальные аллеса				
пачальные адрееа	В случае установки галочки Наложение областей данных			
	регистров временного хранения и ввода при считывании master-			
	устройством holding регистров контроллера будут возвращаться			
	значения соответствующих (совпадающих по номерам) input			
	регистров			

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

4.6 Диагностика и управление обменом

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



Рисунок 4.6.1 – Использование переменных диагностики в программе

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

Таблица 4.6.1 – Системные переменные компонентов Modbus

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

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

Таблица 4.6.2 – Описание элементов перечисления МВ_Е	ErrorCodes
--	------------

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

Название	Значение	Описание
OFF	16#0	Никаких действий не выполняется (например, slave-устройство
	10#0	исключено из опроса)
CONNECTING	16#1	Устанавливается соединение со slave-устройством
CONNECTED	16#2	Соединение установлено
DISCONNECTING	16#3	Соединение разорвано (со стороны контроллера)
SOCKET EDDOD	16#4	Ошибка операций чтения/записи через сокет (например, из-за
SOCKET_ERROR	10#4	разрыва линии связи)

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

Modbus_COM (Modbus COM) Modbus_Matter, COM Port (Modbus Master, COM Port) Modbus_Ishee_COM Port (Modbus Save, COM Port) Modbus_Sleve_COM_Port_1 (Modbus Save, COM Port) Modbus_Sleve_COM_Port_1 (Modbus Save, COM Port)	Modbus_Slave_COM_Port xError OR 0 1 Modbus_Slave_COM_Port xError EN EN EN EN Modbus_Slave_COM_Port xError T#1s PT ET Modbus_Master_COM_Port xResetComPort Z
---	---

Рисунок 4.6.2 – Переинициализация СОМ-порта

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

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

Device X	
Установки соединения	Приложение для обработки 1/0 Application
Приложения Резервное копирование и восстановление	Установки ПЛК Обновлять IO при остановке Выходы при остановке: Сохранить текущие значения
Файлы	Всегда обновлять переменные: Отключено (обновление только в задаче)
Журнал	Лицензии
Установки ПЛК	Опции цикла шины Задача цикла шины: <pre></pre> <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>
Оболочка ПЛК	Доп. установки
Пользователи и группы	Генерировать переменные фиксации для соотнесения IO IO Казывать предупреждения IO как ошибки
Размещение задачи	
Состояние	

Рисунок 4.6.3 – Включение диагностики устройств

2. Для отключения устройства из опроса в коде программы присвоить переменной <имя_устройства_из_дерева_проекта>.Enable значение FALSE.

	5 6 7 Modbus_TCP_Slave.En 8 Modbus_Slave_COM_Po:	able := FALSE; rt.Enable := FALSE;
Modbus_Master_COM_Port (Modbus Master, COM Port)		

Рисунок 4.6.4 – Исключение slave-устройств из опроса

3. Для возобновления опроса присвоить переменной **<имя_устройства_из_дерева_проекта>.Enable** значение **TRUE**.

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



Рисунок 4.6.5 – Изменение заданного адреса опрашиваемого slave-устройства

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

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



Рисунок 4.6.6 – Изменение заданных сетевых настроек опрашиваемого slave-устройства

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

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



Рисунок 4.6.7 – Изменение номера порта в режиме Modbus TCP Slave

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

4.7 Компоненты Modbus и конфигурация задач

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

4.8 Преобразование данных для передачи по Modbus

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

В **CODESYS V3.5** есть два базовых способа для подобных преобразований: <u>объединения</u> и <u>указатели</u>. Кроме того, можно воспользоваться <u>функциями конвертации</u> из библиотеки **OwenCommunication**.

4.8.1 Использование объединений (UNION)

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

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

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



Рисунок 4.8.1 – Добавление объединения в проект CODESYS

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

_ ~	Rea	L_Word X
	1	TYPE Real_Word :
-	2	UNION
	3	rRealValue :REAL;
	4	awModbusReal :ARRAY [01] OF WORD;
	5	END_UNION
	6	END TYPE

Рисунок 4.8.2 – Объявление переменных объединения

 В программе объявить экземпляр объединения Real_Word с названием _2WORD_TO_REAL:



Рисунок 4.8.3 – Объявление экземпляра объединения в программе

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

	awModbusReal
	rRealValue
-	

Рисунок 4.8.4 – Работа с переменными объединения в программе

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

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

Modbus_Slave_COM_Port X											
Общее		Имя	Тип доступа	Триггер	Сдвиг READ	Длина	Обработка ошибок	Cдвиг WRITE	Длина	Комментарий	
Kausa Madhua Claus	0	Channel 0	Read Holding Registers (Код функции 03)	Цикл., t#100ms	16#000A	2	Сохранить посл. значение				
Канал моория Slave											
Modbus Slave Init											

Рисунок 4.8.5 – Настройка канала для считывания значения с плавающей точкой

Modbus_Slave_COM_Port X							
Общее	Найти переменную	Фильтр Показать в	ce		•		
Kawas Modbus Slave	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
Kahan Modbus Slave	=- * Application.PLC_PRG2WORD_TO_REAL.awModbusReal	~ ø	Channel 0	%IW0	ARRAY [01] OF WORD		Read Holding Registers
Modbus Slave Init			Channel 0[0]	%IW0	WORD		0x000A
ModbusGenericSerialSlave	iii−¥p		Channel 0[1]	%IW1	WORD		0x000B
Соотнесение входов/выходов							

Рисунок 4.8.6 – Привязка переменной объединения к каналу

Выражение	Тип	Значение						
2WORD_TO_REAL	REAL_WORD							
rRealValue	REAL	3.3						
🖃 < awModbusReal	ARRAY [01] OF WO							
awModbusReal[0]	WORD	16#3333						
awModbusReal[1]	WORD	16#4053						
•								
_2WORD_TO_REAL.rRealValue								

Рисунок 4.8.7 – Использование REAL переменной объединения в программе

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

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

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

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

Modbus_Slave_COM_Port X									
Общее	Найти переменную Фильтр Показать	все		•					
Kawan Modbus Slave	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание		
	B- * 9		Channel 0	%IW0	ARRAY [01] OF WORD		Read Holding Registers		
Modbus Slave Init	Application.PLC_PRG2WORD_TO_REAL.awModbusReal[0]	*	Channel 0[0]	%IW0	WORD		0x000A		
	Application.PLC_PRG2WORD_TO_REAL.awModbusReal[1]	~	Channel 0[1]	%IW1	WORD		0x000B		
ModbusGenericSerialSlave Соотнесение входов/выходов									

Рисунок 4.8.8 – Привязка элементов массива объединения к регистрам канала

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

<u>_</u>	🔶 Word_Bytes 🗙			Arrow Bytes_Real X				
	1	TYPE Word_Bytes :	1	1	TYPE Bytes_Real :			
	2	UNION		2	UNION			
	3	awModbusReal :ARRAY [01] OF WORD;	:	3	abyModbusReal	:ARRAY [03] OF BYTE;		
	4	abyModbusReal :ARRAY [03] OF BYTE;	4	4	rRealValue	:REAL;		
	5	END_UNION	Į	5	END_UNION			
	6	END_TYPE		6	END TYPE			

Рисунок 4.8.9 – Объявление двух объединений

Выражение	Тип	Значение	Подготовленное	Адрес	Комментарий	
2WORD_TO_4BYTES	Word_Bytes					
🖃 < awModbusReal	ARRAY [01] OF WO					
awModbusReal[0]	WORD	16#5340				
awModbusReal[1]	WORD	16#3333				
🖃 < abyModbusReal	ARRAY [03] OF BYTE					
ø abyModbusReal[0]	BYTE	16#40				
ø abyModbusReal[1]	BYTE	16#53				
ø abyModbusReal[2]	BYTE	16#33				
ø abyModbusReal[3]	BYTE	16#33				
#BYTES_TO_REAL	Bytes_Real					
🖃 < abyModbusReal	ARRAY [03] OF BYTE					
ø abyModbusReal[0]	BYTE	16#33				
ø abyModbusReal[1]	BYTE	16#33				
ø abyModbusReal[2]	BYTE	16#53				
ø abyModbusReal[3]	BYTE	16#40				
rRealValue	REAL	3.3				
		1	A V			
_2WORD_TO_4BYTE _2WORD_TO_4BYTE _2WORD_TO_4BYTE _2WORD_TO_4BYTE	S.abyModbusReal[0] S.abyModbusReal[1] S.abyModbusReal[2] S.abyModbusReal[3]	16#40 Mi 16#33 Mi 16#33 Mi 16#33 Mi	DVE 0 DVE 2 DVE 4 DVE 6	4BYTES_T 4BYTES_T 4BYTES_T 4BYTES_T	0_REAL abyModbusReal[3] 1	

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

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

4.8.2 Использование указателей

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



ПРИМЕЧАНИЕ

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

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

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

2	VAR	
3	rRealValue	:REAL;
4	awModbusReal	:ARRAY [01] OF WORD;
5	prModbusReal	: POINTER TO REAL;
6	END_VAR	

Рисунок 4.8.11 – Объявление указателя

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

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

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

Рисунок 4.8.12 – Настройка канала для считывания значения с плавающей точкой

Modbus_Slave_COM_Port X									
Общее	Найти переменную	Фильтр Показать	все		•				
Kawan Modhun Slava	Переменная		Соотнесение	Канал	Адрес	Тип	Единица	Описание	
Renar Houbus Drave	=- 🐐 Application.PLC_PRG.awModbusReal		۵.	Channel 0	%IW0	ARRAY [01] OF WORD		Read Holding Registers	
Modbus Slave Init	🖶 👘			Channel 0[0]	%IW0	WORD		0x000A	
				Channel 0[1]	%IW1	WORD		0x000B	
ModbusGenericSerialSlave Соотнесение входов/выходов									

Рисунок 4.8.13 – Привязка переменной к каналу

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

Вы	раж	кение	Тип	Значение	Подготовленное	Адрес
	(rRealValue	REAL	3.3		
	\$	awModbusReal	ARRAY [01] OF WO			
		awModbusReal[0]	WORD	16#3333		
		awModbusReal[1]	WORD	16#4053		
-	\$	prModbusReal	POINTER TO REAL	16#03AD1A40		
		prModbusReal^	REAL	3.3		
_						
		ADR(awModbusReal)	MOVE	prMoc	IbusReal	0
		prModbusReal^ 3.3	MOVE ²	rReal ¹	Value 3 3.3	

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

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

Вь	ipa	жение	Тип	Значение
	Ŷ	rRealValue	REAL	3.3
	Ŷ	awModbusReal	ARRAY [01] OF WORD	
		awModbusReal[0]	WORD	16#3333
		awModbusReal[1]	WORD	16#4053
	Ŷ	pawModbusReal	POINTER TO ARRAY [01] OF WORD	16#03AD1A3C
•				
		ADR(rRealValue)	MOVE	pawModbusReal
		pawModbusReal^	MOVE 2	awModbusReal 3

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

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

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

Порядок **WORD** можно менять на этапе привязки переменных к регистрам – например, можно сравнить рисунки <u>4.8.14</u> и 4.8.16:

Modbus_Slave_COM_Port x								
Общее	Найти переменную	Фильтр Показаты	sce		•			
Kawan Modhus Slave	Переменная		Соотнесение	Канал	Адрес	Тип	Единица	Описание
Randi Hoddas Slave				Channel 0	%IW0	ARRAY [01] OF WORD		Read Holding Registers
Modbus Slave Init	Application.PLC_PRG.awModbusReal[1]		۰	Channel 0[0]	%IW0	WORD		0x000A
	Application.PLC_PRG.awModbusReal[0]		۰	Channel 0[1]	%IW1	WORD		0x000B
ModbusGenericSerialSlave Соотнесение входов/выходов								

Рисунок 4.8.16 – Привязка элементов массива к регистрам канала

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

Выра	ажение	Тип	Значение
(👂 rRealValue	REAL	3.3
Ξ (awModbusReal	ARRAY [01] OF WORD	
	awModbusReal[0]	WORD	16#5340
	awModbusReal[1]	WORD	16#3333
± (pabyModbusReal	POINTER TO ARRAY [03] OF BYTE	16#03AD1A40
8	abySwapBytes	ARRAY [03] OF BYTE	
	abySwapBytes[0]	BYTE	16#33
	abySwapBytes[1]	BYTE	16#33
	abySwapBytes[2]	BYTE	16#53
	abySwapBytes[3]	BYTE	16#40
H	prSwapBytes	POINTER TO REAL	16#03AD4164
•			
		MOVE	
	pabyModbusREAL^[0]	16#40	abySwapBytes[3]
	pabyModbusREAL^[1]	16#53 MOVE 6	abySwapBytes[2] 5 16#53
	pabyModbusREAL^[2]	16#33 MOVE 8	abySwapBytes[1]
	pabyModbusREAL^[3]	16#33	abySwapBytes[0] 9 16#33
	ADR(abySwapBytes)	MOVE	prSwapBytes 11 16#03AD4164

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

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

В качестве примера будет рассмотрена настройка обмена с модулями <u>Mx110</u> (MB110-8A, MB110-16Д, MУ110-16P) с использованием **стандартных средств конфигурации**.

Реализуемый алгоритм: если значение первого аналогового входа модуля **MB110-8A** превышает **30** и при этом первый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **MУ110-16P** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).



Структурная схема примера приведена на рисунке ниже:

Рисунок 4.9.1 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.х. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания: <u>Example_CodesysModbusRtuMasterMx110_3511v1.projectarchive</u>

Сетевые параметры модулей приведены в таблице ниже:

Таблица 4.9.1 – Сетевые параметры модулей Мх110

Параметр	MB110-8A	МВ110-16Д	МУ110-16Р		
СОМ-порт контроллера, к которому	COM1	COM2			
подключен модуль	CONT				
ID COM-порта	1 2				
Адрес модуля	1	1	17		
Скорость обмена		115200			
Количество бит данных		8			
Контроль четности	Отсутствует				
Количество стоп-бит		1			

Переменные примера описаны в таблице ниже:

Таблица 4.9.2 – Список переменных примера

Модуль	Имя переменной	Тип	Описание
	awModbusReal	ARRAY [01]	Значение температуры в виде двух WORD ,
MB110-8A	awwwoubusivear	OF WORD	считываемое с модуля
	rRealValue	REAL	Значение температуры в виде числа с
			плавающей точкой для использования в
			программе
		WORD	Значение дискретных входов в виде битовой
			маски. При обращении к отдельным входам
MB110-16Л	WDI		указывается их номер, начиная с 0:
мвтютод	WDT		wDI.0 – состояние первого входа (TRUE/FALSE)
			wDI.1 – состояние второго входа
	wDO	WORD	Значение дискретных выходов в виде битовой
			маски. При обращении к отдельным выходам
			указывается их номер, начиная с 0:
МУ110-16Р			wDO.0 – состояние первого выхода
			(TRUE/FALSE)
			wDO.1 – состояние второго выхода
-		WORD	Значение дискретных выходов в виде битовой
	wPrevDO		маски из предыдущего цикла программы.
			Используется для отправки команды записи
			только в случае изменения значений выходов
			(иначе будет производиться циклическая запись
			последнего значения)
			Триггерная переменная, управляющая функцией
-	xTrigger	BOOL	записи дискретного выхода (запись происходит
			по переднему фронту переменной)

Для настройки обмена следует:

1. Настроить модули **Mx110** с помощью программы **Конфигуратор Mx110** в соответствии с <u>таблицей 4.9.1</u>. Подключить модули к СОМ-портам контроллера в соответствии с <u>рисунком 4.9.1</u>.

2. Создать новый проект CODESYS с программой PLC_PRG на языке CFC:

Стандартн	ый проект	<u>}</u>			
67	Вы собираетесь создать новый стандартный проект. При этом будут созданы следующие объекты:				
	- Одно программируемое устройство, как задано ниже - Программа PLC_PRG на языке, выбранном ниже - Циклическая задача, вызывающая PLC_PRG - Ссылка на новейшую установленную версию библиотеки Standard library.				
	Устройство: SPK1xx[M01] (Production association OWEN)	•			
	PLC_PRG на: Непрерывные функциональные схемы (CFC) 🗸	•			
	ОК Отмена]			

Рисунок 4.9.2 – Создание проекта CODESYS

3. Добавить в проект объединение с именем Real_Word:

		• • •					
SPK_Union							
Device (CODESYS Control V	Vin V3)			ſ		
Plc Logic						doorente por	
= Q Application	X	Вырезать	1			Создать новый объект типа данных	
PLC PRG (PRG		Копировать				•	
😑 🎆 Конфигураци	na.	Вставить				Имя:	
🖻 👹 MainTask	×	Удалить				Real_Word	
PLC_P	E.	Свойства				Тип:	
	4:=	Concretion and and			_	💿 Структура	
		Добавить парки		Dol		Расширение для:	
		дооавить папку	U Do	Persistent-nepemenhuie		Перечисление	
	ш			200		Псевдоним	
		Редактировать объект в	•	РОО для неявных проверок		Базовый тип:	
	0Ş	Логин	89	Redundancy Configuration			
				Symbol configuration		о объединение	
			1	I rend recording manager			
			1	Unit conversion			
				Визуализация			
			1	Внешний файл			
			~	Интерфейс			
				Конфигурация тревог			
				Менеджер визуализации		Добавить	тмена
			N	Менеджер просмотра			
			0	Приложение			
			Ľ	Пул изображений			
				Сервер данных			
			2	Список глобальных переменных			
			1	Список сетевых переменных (Отправитель)			
			2	Список сетевых переменных (Получатель)			
				Список текстов			
			e,	Трассировка			

Рисунок 4.9.3 – Добавление в проект объединения

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



Рисунок 4.9.4 – Объявление переменных объединения

5. В программе PLC_PRG объявить экземпляр объединения Real_Word с названием _2WORD_TO_REAL, переменные wDI, wDO и wPrevDO типа WORD и переменную xTrigger типа BOOL. Описание переменных приведено в таблице 4.9.2.

	PLC_PRG X								
	1	PROGRAM PLC_PRG							
-	2	VAR							
	3	_2WORD_TO_REAL:	Real_Word;	// значение 1-го входа МВ110-8А					
	4	wDI:	WORD;	// битовая маска входов МВ110-16Д					
	5	wDO:	WORD;	// битовая маска выходов MV110-16P					
	6	wPrevDO:	WORD;	// битовая маска предыдущей записи выходов МУ110-16Р					
	7	xTrigger:	BOOL;	// триггер записи выходов					
	8	END_VAR							

Рисунок 4.9.5 – Объявление переменных программы

Код программы будет выглядеть следующим образом:



Рисунок 4.9.6 – Код программы PLC_PRG

Программа работает следующим образом: если значение переменной **rRealValue** (связанной с первым аналоговым входом модуля **MB110-8A**) превышает 30 и при этом значение <u>нулевого бита</u> переменной **wDI** (связанной с <u>первым дискретным входом</u> модуля **MB110-16Д**) имеет значение **TRUE**, то нулевому биту переменной **wDO** присваивается значение **TRUE**. Если на предыдущем цикле значение <u>нулевого бита</u> **wDO** отличалось от текущего, то переменная **xTrigger** принимает значение **TRUE**, что приводит к <u>однократной</u> записи текущего значения бита в <u>первый дискретный выход</u> модуля **MY110-16P**.

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

-	
_	

ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 4.9.7 – Добавление компонента Modbus COM

В конфигурации СОМ-портов следует указать <u>номера СОМ-портов</u> и сетевые настройки в соответствии с <u>таблицей 4.9.1</u>:

💮 СОМ1 🗙		
Общее	Конфигурация последовате	ельного порта
SerialPort Конфигурация	СОМ-порт	1
Состояние	Скорость передачи	115200 ▼
COCTONNIC	Четность	NONE
Информация	Биты информации	8
	Стоповые биты	1

Рисунок 4.9.8 – Настройки СОМ-порта СОМ1
7. В каждый из СОМ-портов добавить компонент Modbus Master.



ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 4.9.9 – Добавление компонента Modbus Master



Рисунок 4.9.10 – Внешний вид дерева проекта после добавления Modbus Master

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

Modbus_Master_COM_Port	×		
Общее	Modbus-RTU/ASCII		MODDIIC
ModbusGenericSerialMaster Соотнесение входов/выходов	Режим передачи	RTU O ASCII	MUDBO2
	Таймаут ответа (мс)	1000	
Состояние	Время между фреймами (мс)	20	
Информация	📝 Автоперезапуск соединен	ия	
		_	

Рисунок 4.9.11 – Настройки компонентов Modbus Master

8. В компонент Modbus Master порта COM1 следует компонент Modbus Slave с именем MV110_8A, а в компонент Modbus Master порта COM2 – компоненты Modbus Slave с именами MV110_16D и MU110_16R.



ПРИМЕЧАНИЕ

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

ойства 🗸 🗸	×		Добавить устройство	×
Example_CodesysModbusMasterMx110_3511v1	•			
Device (SPK1xx[M01])			MR: MV110_8A	
🖷 🗐 Plc Logic	- 11		Действие	
G Application	- 11		Обавить устройство Вставить устройство Подключить устройство	💿 Обновить устро
Real_Word	- 11			· · ·
Менеджер библиотек	- 11		Введите строку для полнотекстового поиска вк Производитель: (All vendors>	-
PLC_PRG (PRG)	- 11		Ина	Poncus
🖃 🌃 Конфигурация задач	- 11		производитель	берсия
🗐 🍪 MainTask	- 11		 Ш Промышленные сети (heldbus) 	E
PLC_PRG	- 11		Modbus	
🖻 🍪 OwenCloudTask	- 11		Cheve Modbus Senal	
OwenStorage.CLOUD_PRG	- 11		Modbus Slave, COM Port 35 - Smart Software Solutions GmbH	3.2.1.0
GOM1 (Modbus COM)	- 11		Madius Slave, COM Port 35 - Smart Software Solutions GmbH	3.3.0.0
Modbus_Master_COM_Port (Modbus Master, COM Port)	Y	Bulletath	Modivie Slave, COM Port 35 - Smart Software Solutions GmbH	3 3 0 20
COM2 (Modbus COM)	00	Kennen	Modive Slave, COM Port 35 - Smart Software Solutions GmbH	3 3 1 10
Modbus_Master_COM_Port_1 (Modbus Master, COM Port) 백달	Копировать	Modeus Slave, COM Port 35 - Smart Software Solutions GmbH	2 2 2 0
- 🕔 OwenRTC (OwenRTC)	Ē	Вставить	Modbus Slave, COM Port 35 - Smart Software Solutions GmbH	2400
- 🐣 OwenCloud (OwenCloud)	×	Удалить	Modbus Slave, COM Port 35 - Smart Software Solutions GmbH	2 5 1 0
- 🏟 Buzzer (Buzzer)		06200	Moduus Slave, COM Port 35 - Smart Software Solutions GmbH	3.5.1.0
Drives (Drives)			Modbus Slave, COM Port 35 - Smart Software Solutions GmbH	3 5 2 0
一号 Network (Network)		Рефакторинг • 7	Modbus Slave, COM Port 35 - Smart Software Solutions GmbH	3540
🛱: Screen (Screen)	6	Свойства	Modbus Slave, COM Port 35 - Smart Software Solutions GmbH	3 5 10 30
🖓 Debug (Debug)	24	Reference of our	MK110 4DN 4D TD DIAM Deduction association OMEN	2 5 4 12
- 🎼 Info (Info)			MK110 4DN 4D TD DWM Production association OWEN	2 5 11 4
Watchdog (Watchdog)		Добавить папку		3.3.11.4
		Добавить устройство	•	•
		Откл. устройство		
		Обновить устройство		
	1.5	Perseturorate of ert	Приложать по категорион (Потображать все версии (дор экспертов)	lor as a the works of the second
		Pedakinpubarb objekt		юказать устаревши
		Редактировать объект в		
		Изменить I/О-соотнесение	MMR: Modbus Slave, COM Port	
		Импорт соотнесений из CSV	Производитель: 35 - Smart Sortware Solutions GmbH Группы: Слейв Modbus Serial	
		Экспортировать соотнесения в CSV	Версия: 3.3.1.10	
			помер модели: - Описание: A generic device that works as a Modbus Slave on a serial bus.	2
		Энуляция	-	
	- 11			
	- 11			
			Добавить выбранное устройство как последнего потомка	
			ModDus_master_COM_Port	
			(Можно выбрать другой таргет-узел, пока окно открыто.)	
	- 11		Добавить устрой	акоыть
			Hoopping Acidor	Junporto

Рисунок 4.9.12 – Добавление slave-устройств в проект CODESYS

Устройства 🗸 🗸 🕂	×
Example_CodesysModbusMasterMx110_3511v1	
Device (SPK1xx[M01])	
🖹 🗐 Plc Logic	
🖹 🚫 Application	
Real_Word	
🗂 📶 Менеджер библиотек	
PLC_PRG (PRG)	
🖻 饠 Конфигурация задач	
🗏 🍪 MainTask	
PLC_PRG	
🖻 🍪 OwenCloudTask	
OwenStorage.CLOUD_PRG	
🖃 🎬 COM1 (Modbus COM)	
🖹 🕤 Modbus_Master_COM_Port (Modbus Master, COM Port)	
MV110_8A (Modbus Slave, COM Port)	
🖶 🔟 COM2 (Modbus COM)	
🖻 🕤 Modbus_Master_COM_Port_1 (Modbus Master, COM Port)	
MV110_16D (Modbus Slave, COM Port)	
MU110_16R (Modbus Slave, COM Port)	
OwenRTC (OwenRTC)	
📲 Buzzer (Buzzer)	
Drives (Drives)	
品 Network (Network)	
⇒∰: Screen (Screen)	
🖞 Debug (Debug)	
📆 Info (Info)	

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

В настройках slave-устройств следует указать их адреса согласно <u>таблице 4.9.1</u> (МВ110-8А – адрес 1, МВ110-16Д – адрес 1, МУ110-16Р – адрес 17):

MV110_8A X			
Общее	Modbus-RTU/ASCII		MONBUS
Канал Modbus Slave	Адрес слейва [1247]	1	MUDDU3
Modbus Slave Init	Таймаут ответа (мс)	1000	
ModbusGenericSerialSlave Соотнесение входов/выходов			

Рисунок 4.9.14 – Настройка slave-устройства MV110_8A

9. В настройках компонента MV110_8A на вкладке Канал Modbus Slave следует добавить канал, в котором с помощью функции Read Holding Registers будет считываться значение четвертого и пятого регистров модуля. В данных регистрах содержится значение входа 1 в представлении с плавающей точкой. Таблица регистров модуля и поддерживаемые функции приведены в руководстве по эксплуатации.

1мя	Тип доступа	Триггер	Сдвиг READ	Длина	Обработка ошибок	Сдви
Channel 0	Read Holding Registers (Код функ	ии 03) CYCLIC, t#100ms	16#0004	2	Сохранить последнее значение	
	ModbusChannel				×	
	Канал					
	Имя	Channel 0				
	Тип доступа	Read Holding Registers (Ke	од функции 3)		•	
	Триггер	Cyclic	• Время цикла (мс) 100		
	Комментарий					
	Регистр READ					
	Сдвиг	0x0004			•	
	Длина	2				
	Обработка оши	Сохранить последне	83⊦▼			
	Peructp WRITE					
	Сдвиг	0x0000			·	
	Длина	0				
			OK		Отмена	
			_	_		
		1				
		· \				
		```				
			\			
			<b>\</b>			
		III				

Рисунок 4.9.15 – Добавление канала в конфигурацию slave-устройства MV110_8A

На вкладке ModbusGenericSerialSlave Соотнесение входов/выходов следует привязать к каналу элементы объединения _2WORD_TO_REAL. Первый считываемый регистр присваивается первому элементу массива, а второй – нулевому. Это связано с тем, что порядок WORD в REAL у модуля и контроллера отличается.

Для параметра Всегда обновлять переменные следует установить значение Включено 2.

MV110_8A X							
Общее	Каналы						
· · ·	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
Канал Modbus Slave	B-*		Channel 0	%IW0	ARRAY [01] OF WORD		Read Holding Registe
	Application.PLC_PRG2WORD_TO_REAL.awModbusReal[1]	۵.	Channel 0[0]	%IW0	WORD		0004
Modbus Slave Init	Application.PLC_PRG2WORD_TO_REAL.awModbusReal[0]	۵۵	Channel 0[1]	%IW1	WORD		0005
МодрызСелегісSerialSlave Соотнесение входов/выходов Состояние Информация	•	Ш	тнесения Все	гда обновлят	ъ переменные: Вкл. 2 (е	сегда в задачи	раниција с цикла шины)

Рисунок 4.9.16 – Привязка переменных к каналу

**10**. В настройках компонента **MV110_16D** на вкладке **Канал Modbus Slave** следует добавить канал, в котором с помощью функции **Read Holding Registers** будет считываться значение регистра **0x0033**. В данном регистре содержится битовая маска состояний дискретных входов.

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

Имя	Тип до	ступа		Триггер	Сдвиг READ	Длина	Обработка ошибок	Сдвиг V
····· Channel 0	Read Ho	olding Registers (Код	функции 03)	CYCLIC, t#100ms	16#0033	1	Сохранить последнее значение	
Channel U	Read Ho	МоdbusChannel Канал Имя Тип доступа Триггер Комментарий Регистр READ Сдвиг Длина Обработка ошиб Регистр WRITE Сдвиг Длина	функции 03) Сhannel 0 Read Holding Сусіс 0x0033 1 0x0000 0	Registers (Код функц v Вре	16#0033	1 00 Отмена		
•				III				+
					N (7	Добавить ка	анал Удалить Редакт	ировать

Рисунок 4.9.17 – Добавление канала в конфигурацию slave-устройства MV110_16D

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

Общее	Каналы						
	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
(анал Modbus Slave			Channel 0	%IW2	ARRAY [00] OF WORD		Read Holding Registe
	😟 🦘 Application.PLC_PRG.wDI	*	Channel 0[0]	%IW2	WORD		0033
odbusGenericSerialSlave							
отнесение входов/выходов							
отнесение входов/выходов							
отнесение входов/выходов							
отнесение входов/выходов стояние формация							
оотнесение входов/выходов остояние формация							
отнесение входов/выходов стояние формация							
отнесение входов/выходов стояние формация							
отнесение входов/выходов							
отнесение входов/выходов стояние формация							
отнесение входов/выходов							
отнесение входов/выходов							

Рисунок 4.9.18 – Привязка переменных к каналу

11. В настройках компонента MU110_16R на вкладке Канал Modbus Slave следует добавить канал, в котором с помощью функции Write Multiple Registers будет записываться значение в регистр 0x0032. В данном регистре содержатся значения выходов модуля в виде битовой маски. У параметра Триггер следует установить значение Передний фронт, чтобы иметь возможность управлять записью в модуль с помощью логической переменной.

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

MU110_16R X	
Общее	Имя Типдоступа Триггер Сдвиг READ Длина Обработка ошибок Сдвиг WRITE Длина Комментарий
Канал Modbus Slave	Kauae Modhus
Modbus Slave Init	
ModbusGenericSerialSlave Соотнесение входов/выходов	Канал Иня Channel 0
Состояние	Тип доступа Write Multiple Registers (Код функции 16) 🔻
Информация	Триггер Передний фронт Время цикла (мс) 100 Комментарий
	Peructp READ
	Сдвиг
	Длина 1
	Обработка ошибок Сохранить посл. значен 💌
	Регистр WRITE
	Сдвиг 0x0032 -
	Длина 1
	Добавить канал Удалить Изменить

Рисунок 4.9.19 – Добавление канала в конфигурацию slave-устройства MU110_16R

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

Общее	Каналы						
	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
(анал Modbus Slave	Application.PLC_PRG.xTrigger	<b>*</b>	Channel 0	%QX0.0	BIT		Trigger Variable
	ė- <b>*</b>		Channel 0	%QW1	ARRAY [00] OF WORD		Write Multiple Regis
fodbus Slave Init		<b>*</b>	Channel 0[0]	%QW1	WORD		0032
Состояние							
юстояние Інформация							

Рисунок 4.9.20 – Привязка переменных к каналу

12. Загрузить проект в контроллер и запустить его.



#### ПРИМЕЧАНИЕ

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



Рисунок 4.9.21 – Пиктограммы статусов связи

В переменной _2WORD_TO_REAL.rRealValue будет отображаться текущее значение первого аналогового входа модуля MB110-8A. В нулевом бите переменной wDI (wDI.0) будет отображаться текущее значение первого дискретного входа модуля MB110-16Д.

Если значение _2WORD_TO_REAL.rRealValue превысит 30 и при этом значение wDI.0 будет равно TRUE, то в нулевой бит переменной wDO (wDO.0) будет однократно (по триггеру) записано значение TRUE, что приведет к замыканию первого дискретного выхода модуля MУ110-16P. Если одно из условий перестанет выполняться, то выход будет разомкнут.



Рисунок 4.9.22 – Выполнение программы в режиме Online

# 4.10 Пример: СПК1xx [M01] (Modbus RTU Slave) + MasterOPC Universal Modbus Server

В качестве примера будет рассмотрена настройка обмена с OPC-сервером <u>Insat MasterOPC Universal</u> <u>Modbus Server</u>, который будет использоваться в режиме **Modbus RTU Master**.

Структурная схема примера приведена на рисунке ниже:



Рисунок 4.10.1 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.х. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания: Example CodesysModbusRtuSlave 3511v1.zip

Сетевые параметры устройств приведены в таблице ниже:

Таблица 4.10.1 – Сетевые параметры устройств

Параметр	СПК1хх [М01]	ПК с ОРС-сервером		
СОМ-порт контроллера, к которому	COM1	COM4		
подключен модуль	CONT	001014		
ID COM-порта	1	4		
Режим работы	slave	master		
Slave ID	1	-		
Скорость обмена	115200			
Количество бит данных	8			
Контроль четности	Отсутствует			
Количество стоп-бит		1		

Переменные примера описаны в таблице ниже:

### Таблица 4.10.2 – Список переменных примера

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

Для настройки обмена следует:

- 1. Подключить контроллер к ПК (например, с помощью конвертера <u>OBEH AC4</u>).
- 2. Создать новый проект CODESYS с программой PLC_PRG на языке CFC:

Стандартн	ый проект	×
	Вы собираетесь создать новый стандартный проект. При этом будут созданы следующие объекты: - Одно программируемое устройство, как задано ниже	
	- Программа PLC_PRG на языке, выбранном ниже - Циклическая задача, вызывающая PLC_PRG - Ссылка на новейшую установленную версию библиотеки Standard library.	
	Устройство: SPK1xx[M01] (Production association OWEN) PLC_PRG на: Непрерывные функциональные схемы (CFC)	•
	ОК От	1ена

Рисунок 4.10.2 – Создание проекта CODESYS

3. Добавить в проект объединения с именами Real_Word и String_Word:

Устройства		<b>→</b> ₽ X			
SPK_Union					
Device (CODESYS Control V	Vin V3	3)			
Plc Logic					дооавить рот
= Q Application	V	Buperath			
Менеджер би	00	bbpesarb			
PLC_PRG (PRG	93	копировать			
Конфигурация		Вставить			Имя:
- The main ask	X	Удалить			Real_Word
	Ē.	Свойства			Тип:
	*:::	Добавить объект	F 🖓	DUT	🔘 Структура
	6	Добавить папку	<b>_</b>	Persistent-geneweuuuse	Расширение для:
	E C	Редактировать объект		DOLL	Перечисление
		Радактировать объект в		2011	Псевдоним
		гедактировать объект В	<u>e</u>	РОО для неявных проверок	Базовый тип:
	OŞ.	Логин	00	Redundancy Configuration	
			•	Symbol configuration	Объединение
				Trend recording manager	
			- C	Unit conversion	
			- 8	Визуализация	
				Внешний файл	
			~	Интерфейс	
				Конфигурация тревог	
				Менеджер визуализации	Добавить Отмена
				Менеджер просмотра	
			0	Придожение	
				Сервер дапНых	
			2	Список глобальных переменных	
			- 3	Список сетевых переменных (Отправитель)	
			- <u>-</u>	Список сетевых переменных (Получатель)	
				Список текстов	
			e(	Трассировка	

Рисунок 4.10.3 – Добавление в проект объединения Real_Word

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

4. В объединениях объявить следующие переменные:



Рисунок 4.10.4 – Объявление переменных объединения Real_Word



Рисунок 4.10.5 – Объявление переменных объединения String_Word

- 5. В менеджере библиотек добавить библиотеку CAA Memory.
- 6. В программе PLC_PRG объявить переменные в соответствии с таблицей 4.10.2.

/ 🛅 PL	C_PRG X	
1	PROGRAM PLC_PRG	
<b>⊟</b> 2	VAR	
3	xVar_OpcRead:	BOOL;
4	wVar_OpcRead:	WORD;
5	rVar_OpcRead:	REAL;
6	sVar_OpcRead:	STRING(16);
7	uRealWord_OpcRead:	Real_Word;
8	uStringWord_OpcRead:	String_Word;
9		
10		
11	xVar_OpcWrite:	BOOL;
12	wVar_OpcWrite:	WORD;
13	rVar_OpcWrite:	REAL;
14	sVar_OpcWrite:	STRING(16);
15	uRealWord_OpcWrite:	Real_Word;
16	uStringWord_OpcWrite:	String_Word;
17	END VAR	

Рисунок 4.10.6 – Объявление переменных программы

xVar_OpcRead		
wVar_OpcRead -	0	
rVar_OpcRead uRealWord_Op	cRead.rRealValue	
sVar_OpcReaduStringWord_O	pcRead.sStringValue	
	2	
	MEM.ReverseBYTEsInWORD	3
uStringWord_OpcRead.awModbusString[0]	wInput ReverseBYTEsInWORD	uStringWord_OpcRead.awModbusString[0]
	MEM.ReverseBYTEsInWORD	
uStringWord_OpcRead.awModbusString[1]	wInput ReverseBYTEsInWORD	<ul> <li>uStringWord_OpcRead.awModbusString[1]</li> </ul>
	MEM.ReverseBYTEsInWORD	
uStringWord_OpcRead.awModbusString[2]	winput ReverseBYTEsInWORD	<ul> <li>uStringWord_OpcRead.awModbusString[2]</li> </ul>
	MEM.ReverseBYTEsInWORD	
uStringWord_OpcRead.awModbusString[3]	wInput ReverseBYTEsInWORD	uStringWord_OpcRead.awModbusString[3]
	MEM.ReverseBYTEsInWORD	
uStringWord_OpcRead.awModbusString[4]	wInput ReverseBYTEsInWORD	uStringWord_OpcRead.awModbusString[4]
	MEM.ReverseBYTEsInWORD	
uStringWord OpcRead.awModbusString[5]	wInput ReverseBYTEsInWORD	uStringWord OpcRead.awModbusString[5]
	MEM.ReverseBYTEsInWORD 14	_
uStringWord OpcRead.awModbusString[6]	winput ReverseBYTEsInWORD	uStringWord OpcRead.awModbusString[6]
	MEM. ReverseBYTEsInWORD	-
uStringWord_OpcRead.awModbusString[7]	winput ReverseBYTEsinWORD	uStringWord OpcRead awModbusString[7]
dowing rold_operioda.amiloabadoumg[/]		Soungfrord_openedatamicabasering[/]
wVar_OpcWrite – uRealWord_OpcWrite.rRealValue	rVar_OpcWrite 18	
	MEM ReverseBYTEsInWORD	_
uStringWord_OpcWrite awModbusString[0]	winnut BeverseBYTEsinWORD	uStringWord OpcWrite awModbus String[0]
Seamgrend_opermisianilesseering[e]		
	MEM ReverseBYTEsInWORD 21	_
uStringWord_OpcWrite awModbusString[1]	winput ReverseBYTEsInWORD	uStringWord_OpcWrite awModbusString[1]
	MEM ReverseBYTEsInWORD 23	
uStringWord OpcWrite.awModbusString[2]	winput ReverseBYTEsInWORD	
		uStringWord OpcWrite.awModbusString[2]
		uStringWord_OpcWrite.awModbusString[2]
	MEM.ReverseBYTEsInWORD 25	uStringWord_OpcWrite.awModbusString[2]
uStringWord OpcWrite.awModbusString[3]	MEM.ReverseBYTEsInWORD	uStringWord_OpcWrite.awModbusString[2]
uStringWord_OpcWrite.awModbusString[3]	MEM.ReverseBYTEsInWORD 23 winput ReverseBYTEsInWORD	uStringWord_OpcWrite.awModbusString[2]
uStringWord_OpcWrite.awModbusString[3]	MEM.ReverseBYTEsInWORD winput ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD	uStringWord_OpcWrite.awModbusString[2] 24
uStringWord_OpcWrite.awModbusString[3]	MEM.ReverseBYTEsInWORD 25 Winput ReverseBYTEsInWORD 27 Winput ReverseBYTEsInWORD 27 winput ReverseBYTEsInWORD 27	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4]	MEM.ReverseBYTEsInWORD 23 Winput ReverseBYTEsInWORD 27 Winput ReverseBYTEsInWORD 27 winput ReverseBYTEsInWORD 27	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4]	MEM.ReverseBYTEsInWORD Winput ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD Winput ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD 23 23	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28
uStringWord_OpcWrite.awModbusString[3]	MEM.ReverseBYTEsInWORD Winput ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD Winput ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD 23 Winput ReverseBYTEsInWORD 23 Winput ReverseBYTEsInWORD 23 Winput ReverseBYTEsInWORD 23 23 23 23 23 23 23 2	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28 uStringWord_OpcWrite.awModbusString[5] 30
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4] uStringWord_OpcWrite.awModbusString[5]	MEM.ReverseBYTEsInWORD WInput ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD WInput ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD 23 WInput ReverseBYTEsInWORD 23 WInput ReverseBYTEsInWORD 23 23 24 23 24 24 25 27 27 27 27 27 27 27	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28 uStringWord_OpcWrite.awModbusString[5] 30
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4] uStringWord_OpcWrite.awModbusString[5]	MEM.ReverseBYTEsInWORD 25 WInput ReverseBYTEsInWORD 27 WInput ReverseBYTEsInWORD 27 WInput ReverseBYTEsInWORD 29 WInput ReverseBYTEsInWORD 29 WInput ReverseBYTEsInWORD 31	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28 uStringWord_OpcWrite.awModbusString[5] 30
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4] uStringWord_OpcWrite.awModbusString[5]	MEM.ReverseBYTEsInWORD 25 WInput ReverseBYTEsInWORD 27 WInput ReverseBYTEsInWORD 27 WInput ReverseBYTEsInWORD 23 WInput ReverseBYTEsInWORD 31 WInput	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28 uStringWord_OpcWrite.awModbusString[5] 30 uStringWord_OpcWrite.awModbusString[5] 32
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4] uStringWord_OpcWrite.awModbusString[5] uStringWord_OpcWrite.awModbusString[6]	MEM.ReverseBYTEsInWORD 23 Winput ReverseBYTEsInWORD 27 Winput ReverseBYTEsInWORD 29 Winput ReverseBYTEsInWORD 29 Winput ReverseBYTEsInWORD 31 Winput ReverseBYTEsInWORD 31 Winput ReverseBYTEsInWORD 31	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28 uStringWord_OpcWrite.awModbusString[5] 30 uStringWord_OpcWrite.awModbusString[5] 32
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4] uStringWord_OpcWrite.awModbusString[5] uStringWord_OpcWrite.awModbusString[6]	MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD 31 Winput MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD 33	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28 uStringWord_OpcWrite.awModbusString[5] 30 uStringWord_OpcWrite.awModbusString[5] 32
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4] uStringWord_OpcWrite.awModbusString[5] uStringWord_OpcWrite.awModbusString[6]	MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD 31 Winput ReverseBYTEsInWORD 32 Winput MEM.ReverseBYTEsInWORD 33 Winput MEM.ReverseBYTEsInWORD 33 Winput MEM.ReverseBYTEsInWORD 33 Winput 33 Winput 34 34 35 35 35 35 35 35	uStringWord_OpcWrite.awModbusString[2]  uStringWord_OpcWrite.awModbusString[3]  uStringWord_OpcWrite.awModbusString[4]  uStringWord_OpcWrite.awModbusString[5]  uStringWord_OpcWrite.awModbusString[6]  uStringWord_OpcWrite.awModbusString[6]  uStringWord_OpcWrite.awModbusString[6]
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4] uStringWord_OpcWrite.awModbusString[5] uStringWord_OpcWrite.awModbusString[6] uStringWord_OpcWrite.awModbusString[7]	MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD 31 Winput ReverseBYTEsInWORD 32 MEM.ReverseBYTESInWORD 33 Winput ReverseBYTEsInWORD 33 Winput 34 35 35 36 36 36 37 31 31 31 32 33 33 34 34 34 35 35 35	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28 uStringWord_OpcWrite.awModbusString[5] 30 uStringWord_OpcWrite.awModbusString[6] 32 uStringWord_OpcWrite.awModbusString[6] 32 34
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4] uStringWord_OpcWrite.awModbusString[5] uStringWord_OpcWrite.awModbusString[6] uStringWord_OpcWrite.awModbusString[7] uStringWord_OpcWrite.sStringValue	MEM.ReverseBYTEsInWORD 25 WInput ReverseBYTEsInWORD 27 WInput ReverseBYTEsInWORD 27 WInput ReverseBYTEsInWORD 29 WInput ReverseBYTEsInWORD 31 WInput ReverseBYTEsInWORD 31 WInput ReverseBYTEsInWORD 33 WInput SVar OpcWrite 35	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28 uStringWord_OpcWrite.awModbusString[5] 30 uStringWord_OpcWrite.awModbusString[6] 32 uStringWord_OpcWrite.awModbusString[6] 32 34
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4] uStringWord_OpcWrite.awModbusString[5] uStringWord_OpcWrite.awModbusString[6] uStringWord_OpcWrite.awModbusString[7] uStringWord_OpcWrite.sStringValue	MEM.ReverseBYTEsInWORD 23 WInput ReverseBYTEsInWORD 27 WInput ReverseBYTEsInWORD 27 WInput ReverseBYTEsInWORD 23 WInput ReverseBYTEsInWORD 31 WInput ReverseBYTEsInWORD 31 WInput ReverseBYTEsInWORD 31 WInput ReverseBYTEsInWORD 31 WInput ReverseBYTEsInWORD 33 WInput SVar_OpcWrite 33	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28 uStringWord_OpcWrite.awModbusString[5] 30 uStringWord_OpcWrite.awModbusString[6] 32 uStringWord_OpcWrite.awModbusString[7] 34
uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4] uStringWord_OpcWrite.awModbusString[5] uStringWord_OpcWrite.awModbusString[6] uStringWord_OpcWrite.awModbusString[7] uStringWord_OpcWrite.sStringValue	MEM.ReverseBYTEsInWORD 23 Winput ReverseBYTEsInWORD 27 Winput ReverseBYTEsInWORD 29 Winput ReverseBYTEsInWORD 31 Winput ReverseBYTEsInWORD 31 Winput ReverseBYTEsInWORD 31 Winput ReverseBYTEsInWORD 31 Winput ReverseBYTEsInWORD 33 Winput SVar_OpcWrite 33	uStringWord_OpcWrite.awModbusString[2] 24 uStringWord_OpcWrite.awModbusString[3] 26 uStringWord_OpcWrite.awModbusString[4] 28 uStringWord_OpcWrite.awModbusString[5] 30 uStringWord_OpcWrite.awModbusString[6] 32 uStringWord_OpcWrite.awModbusString[7] 34

Код программы будет выглядеть следующим образом:

Рисунок 4.10.7 – Код программы PLC_PRG

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

**7.** Добавить компоненты **Modbus COM** и **Modbus Serial Device** в соответствии с <u>п. 4.3</u>. Настроить компоненты в соответствии с <u>таблицей 4.10.1</u>.

Modbus_COM X			
Общее	Конфигурация последоват	ельного порта	
SerialPort Конфигурация	СОМ-порт	1	
Co	Скорость передачи	115200 🔻	
Состояние	Четность	NONE	
Информация	Биты информации	8	
	Стоповые биты	1	



Modbus_Serial_Device 🗙		
Modbus Serial Device	ID:	1
Modbus Serial Device Соотнесение входов/выходов	Сторожевой таймер:	500
Информация	Регистр хранения (%IW):	12
	Входные регистры (%QW): 1	12
	Регистры:	0
	Дискретные входы:	0
	Регистр временного хранен	ния: 0
	Регистр ввода:	0

Рисунок 4.10.9 – Настройки компонента Modbus Serial Device

**8.** Привязать к каналам компонента **Modbus Serial Device** переменные программы в соответствии с <u>таблицей 4.10.2</u>. Установить галочку **Вкл. 2 (Всегда в задаче цикла шины)**.

Найти переменную Фильтр Пока	зать все		•			
ие Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
8- <b>%</b>		Входы	%IW0	ARRAY [011] OF WORD		Регистры временного хранения Modbu
÷.*		Входы[0]	%IW0	WORD		Holding perистр 0 (coils 0-15)
* Application.PLC_PRG.xVar_OpcWrite	~ <b>`</b> #	Bit0	%JX0.0	BOOL		
		Bit1	%IX0.1	BOOL		
		Bit2	%IX0.2	BOOL		
- *		Bit3	%IX0.3	BOOL		
		Bit4	%IX0.4	BOOL		
- *		Bit5	%IX0.5	BOOL		
		Bit6	%IX0.6	BOOL		
- *		Bit7	%IX0.7	BOOL		
		Bit8	%IX1.0	BOOL		
- *		Bit9	%IX1.1	BOOL		
		Bit10	%IX1.2	BOOL		
- *		Bit11	%IX1.3	BOOL		
		Bit12	%IX1.4	BOOL		
- *		Bit13	%IX1.5	BOOL		
		Bit14	%IX1.6	BOOL		
		Bit15	%IX1.7	BOOL		
Application.PLC_PRG.wVar_OpcWrite	<b>*</b>	Входы[1]	96IW1	WORD		Holding peructip 1
Application.PLC_PRG.uRealWord_OpcWrite.awModbusReal[0]	<b>~</b>	Входы[2]	%IW2	WORD		Holding peructp 2
#- * Application.PLC_PRG.uRealWord_OpcWrite.awModbusReal[1]	<b>~</b>	Входы[3]	%IW3	WORD		Holding peructip 3
Application.PLC_PRG.uStringWord_OpcWrite.awModbusString[0]	<b>*</b>	Входы[4]	9GEW-4	WORD		Holding peructp 4
#- * Application.PLC_PRG.uStringWord_OpcWrite.awModbusString[1]	<b>~</b>	Входы[5]	%IW5	WORD		Holding peructip 5
Application.PLC_PRG.uStringWord_OpcWrite.awModbusString[2]	<b>*</b>	Входы[6]	<del>%IW6</del>	WORD		Holding perистр 6
#- * Application.PLC_PRG.uStringWord_OpcWrite.awModbusString[3]	<b>~</b>	Входы[7]	%IW7	WORD		Holding peructip 7
Application.PLC_PRG.uStringWord_OpcWrite.awModbusString[4]	<b>*</b>	Входы[8]	96EW8	WORD		Holding peructp 8
Application.PLC_PRG.uStringWord_OpcWrite.awModbusString[5]	~ <b>*</b>	Входы[9]	96IW9	WORD		Holding peructp 9
Application.PLC_PRG.uStringWord_OpcWrite.awModbusString[6]	<b>*</b>	Входы[10]	96FW-10	WORD		Holding perистр 10
Application.PLC_PRG.uStringWord_OpcWrite.awModbusString[7]	<b>~</b>	Входы[11]	%IW11	WORD		Holding peructp 11
8.7		Выходы	%QW0	ARRAY [011] OF WORD		Входные регистры Modbus
8- <b>5</b> 0		Выходы[0]	%QW0	WORD		Input perистр 0 (discrete inputs 0-15)
Application.PLC_PRG.wVar_OpcRead	<b>*</b>	Выходы[1]	%QW1	WORD		Input pervictp 1

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

Modbus Serial Device	Найти переменную	Фильтр Показать все		-				
Modbus Serial Device Соотнесение	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание	
входов/выходов	* * Application.PLC_PRG.uStringWord_OpcWrite.	awModbusString[5]	Входы[9]	<del>%IW9</del>	WORD		Holding peructp 9	
нформация	Application.PLC_PRG.uStringWord_OpcWrite.	awModbusString[6]	Входы[10]	%IW 10	WORD		Holding peructip 10	
	Application.PLC_PRG.uStringWord_OpcWrite.	awModbusString[7]	Входы[11]	%IW11	WORD		Holding peructp 11	
	😑 - 🍫		Выходы	%QW0	ARRAY [011] OF WORD		Входные регистры Modbus	
	÷-**		Выходы[0]	%QW0	WORD		Input perистр 0 (discrete inputs 0-15)	
	Application.PLC_PRG.xVar_OpcRead	۵.	BitO	%QX0.0	BOOL			
	**		Bit1	%QX0.1	BOOL			
	- *		Bit2	%QX0.2	BOOL			
	**		Bit3	%QX0.3	BOOL			
	- *		Bit4	%QX0.4	BOOL			
			Bit5	%QX0.5	BOOL			
	- *		Bit6	%QX0.6	BOOL			
	- <b>*</b>		Bit7	%QX0.7	BOOL			
	- 10		Bit8	%QX1.0	BOOL			
	······································		Bit9	%QX1.1	BOOL			
	- 10		Bit10	%QX1.2	BOOL			
			Bit11	%QX1.3	BOOL			
	- 14		Bit12	%QX1.4	BOOL			
			Bit13	%QX1.5	BOOL			
	· · · · · · · · · · · · · · · · · · ·		Bit14	%QX1.6	BOOL			
	<b>*</b>		Bit15	%QX1.7	BOOL			
	Application.PLC_PRG.wVar_OpcRead	<b>*</b>	Выходы[1]	%QW1	WORD		Input perистр 1	
	Application.PLC_PRG.uRealWord_OpcRead.ar	wModbusReal[0] 🏾 🖗	Выходы[2]	%QW2	WORD		Input perистр 2	
	Application.PLC_PRG.uRealWord_OpcRead.ar	wModbusReal[1]	Выходы[3]	%QW3	WORD		Input регистр 3	
	Application.PLC_PRG.uStringWord_OpcRead.	awModbusString[0]	Выходы[4]	%QW4	WORD		Input perистр 4	
	Application.PLC_PRG.uStringWord_OpcRead.	awModbusString[1]	Выходы[5]	%QW5	WORD		Input perистр 5	
	Application.PLC_PRG.uStringWord_OpcRead.	awModbusString[2]	Выходы[6]	%QW6	WORD		Input perистр 6	
	Application.PLC_PRG.uStringWord_OpcRead.	awModbusString[3]	Выходы[7]	%QW7	WORD		Input perистр 7	
	Application.PLC_PRG.uStringWord_OpcRead.	awModbusString[4]	Выходы[8]	%QW8	WORD		Input регистр 8	
	Application.PLC_PRG.uStringWord_OpcRead.	awModbusString[5]	Выходы[9]	%QW9	WORD		Input perистр 9	
	Application.PLC_PRG.uStringWord_OpcRead.	awModbusString[6]	Выходы[10]	%QW10	WORD		Input регистр 10	
	Application.PLC_PRG.uStringWord_OpcRead.	awModbusString[7] 👘	Выходы[11]	%QW11	WORD		Input perucrp 11	

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

9. Установить и запустить MasterOPC Universal Modbus Server.

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

	MasterOPC Universal Modbus Server Demo 60000 1 hour Build - 4.2.28						
Конфигурация Общие настрой	ки Помощь						
🗐 Создать 💾 Сохранить как	🗓 Добавить узел 👘 Д	1обавить тег	🖃 Переименовать 🗙 Удалить				
🖗 Открыть 🛛 🛷 Сделать стартовым	🕅 Добавить устройство 🔺 Г	Іереместить вверх	🗐 Копировать 🛛 🖛 Отменить				
陷 Сохранить 📋 Импорт из версии 2.0	🔍 Добавить группу 🛛 📲	😓 Редактирование комм	иуникационного узла				
Файл конфигурации	Сервер		· · ·				
Текущая конфигурация : 653.mbp		Имя узла Node1					
Объекты		🗉 Общие настройк	и				
		Комментарий					
Добавить Ко	оммуникационный узел	Включен в работу	, ,	True			
Групповые операции		Тип узла		COM			
		🗏 Настройки СОМ					
импорт узла		Порт		4			
Узлы в работе		Скорость		115200			
		·Данные		8			
		Контроль четност	и	Не используется			
		Стоп биты		1			
		Межсимвольный т	аймаут (мс)	0			
		Использовать реж	KUM ASCII	False			
		Использовать мод	em	False			
		Скрипт		E-l			
		выполнение скриг	1Ta 	Faise			
		<ul> <li>Дополнительны</li> <li>Сажа во включение</li> </ul>	енастроики	False			
		- Slave подключени Принудитольный р		False			
		Использовать роз		False			
		vicito/ibsobarb pes		T disc			
		🔲 Тиражировать	1 D	а Нет			
Режим Конфигурирование	Теги : Разрешено - 600	00. Загружено - 8					

Рисунок 4.10.12 – Добавление коммуникационного узла

**11**. Нажать **ПКМ** на коммуникационный узел и добавить устройство с настройками по умолчанию (Slave ID = 1 в соответствии с <u>таблицей 4.10.1</u>).

MasterOPC Univ	versal Modbus Server Demo 60000 1 hour Build - 4.2.28	
Конфигурация Общие настройки Помощь		
🗐 Создать 🛛 💾 Сохранить как 🔅 Добавить узел	а 🗏 Добавить тег 📃 Переименовать 🗙 Удалить	
🖗 Открыть 🛛 🛷 Сделать стартовым 🛛 🍓 Добавить устр	ойство 👚 Переместить вверх 🛛 🕼 Копировать 🖛 Отменить	
Сохранить Димпорт из версии 2.0 Добавить групп Файл конфигурации	ПУ Се Редактирование устройства	- 0 <mark>-</mark> )
екущая конфигурация : 653.mbp	Имя устройства Device1	
Объекты	Вобщие настройки	
Pull Server	Комментарий	
Node1	Включено в работу	True
Добавить Устройство	Тип устройства	MODBUS
Переименовать	Адрес (0х01)	1
Дублировать	Время ответа (мс)	1000
Улапить	Повторы при ошибке	3
	Повторное соединение после ошибки через (с)	10
Вырезать	Реинициализация узла при ошибке	False
копировать	Период опроса	1000
трупповые операции	Размерность периода опроса	ms
Экспорт узла	Начальная фаза	0
Импорт устройства	Размерность фазы	ms
Устройства в работе		True 4
	Бадержка запроса после получения ответа (мс)	BHROR
		55555
	Выполнение скрипта	False
	🗉 Настройка запросов	
	Максимальное количество HOLDING регистров в запросе чтения	125
	Максимальное количество INPUT регистров в запросе чтения	125
	Не использовать команду WRITE_SINGLE_COIL (0x05)	True
	Птиражировать 1 🕞 Да	Нет
жим Конфигурирование Теги : Разре	ешено - 60000. Загружено - 8	

Рисунок 4.10.13 – Добавление устройства

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

MasterOPC Universal Modbus Server Demo 60000 1 hour Build - 41.1						
Конфигурация Общие настройки	Помощь					
🗐 Создать 💾 Сохранить как 🗈 До	обавить узел 🛛 💐 Добавить тег	🖃 Переименовать 🗙 Удалить				
🖗 Открыть 🛷 Сделать стартовым 🕅 До	обавить устройство 👚 Переместить в	Редактирование тега				
陷 Сохранить 📋 Импорт из версии 2.0 🍕 Добавить группу 🛛 👆 Переместить в						
Файл конфигурации Сервер		Имя тега xVar_OpcWrite				
Текущая конфигурация : 11.mbp		🗉 Общие настройки				
Объекты		Комментарий				
🖻 🗊 Сервер Устройст	TB0 < <modbus>&gt; : Device1</modbus>	Включен в работу	True			
🖻 🛷 Node1	0. UD CTDOЙИИ	Регион	COILS			
Device1	А Так	Адрес	(0x0000) 0			
дооавить		Тип данных в устроистве	bool			
Переименовать	Подустройство	Тип доступа	ReadWrite			
Дуолировать	вета (мс)	В Скрипт				
Удалить	при ошибке	ри ошибке Разрешение выполнения скрипта после чтен				
Вырезать	е соединение после ошибки через	Разрешение выполнения скрипта перед з	аписью False			
Копировать	злизация узла при ошибке	🗏 Дополнительно				
Групповые операции	npoca	Наличие отдельного регистра записи	False			
Экспорт устройства	ия фаза	Чтение сразу после записи	Faise			
Импорт подустройства	сть фазы		False			
Экспорт тегов	сле запуска		T disc			
Импорт тегов	а запроса после получения ответа					
Видимость тегов	овка байтов в значении					
Теги в работе						
Дополнительные свойства	ие скрипта					
Maken	мальное количество HOLDING регистр		II			
Макси	мальное количество INPUT регистров	🗌 Тиражировать 1 😭	Да Нет			
Не исг	пользовать команду WRITE_SINGLE_CO					
	UNDER OTHER DE	CTCTED (0.00) T	· · · · ·			
Свойства	объекта Таблица тегов					
Режим Конфигурирование	Теги : Разрешено - 60000. Загружено	<ul> <li>о - 0. В текущем элементе - 0</li> </ul>	h.			

Рисунок 4.10.14 – Добавление тега xVar_OpcWrite

И	мя тега	wVar_OpcWrite		
	Общие н	астройки		
	Коммента	рий		
	Включен	в работу	True	
	Регион		HOLDING_REGISTERS	
	Адрес	(0x0001)	1	
	Тип данны	ых в устройстве	uint16	
	Тип данны	ых в сервере	uint32	
l	Тип досту	ла	ReadWrite	
	Использо	зать перестановку байтов устройства	True	e
	Последни	й тег в групповом запросе	False	
	Пересчет	(A*X + B)	False	

Рисунок 4.10.15 – Добавление тега wVar_OpcWrite

Имя тега		rVar_OpcWrite					
•	■ Юбщие настройки						
	Коммента	рий					
ſ	Включен	в работу		True			
	Регион			HOLDING_REGISTERS			
	Адрес	(0x00	2				
	Тип данн	іх в устройстве		float			
	Тип данн	ых в сервере		float			
	Тип досту	упа		ReadWrite			
	Использо	вать перестановку байтов устройств	a	False			
l	Перестан	овка байтов в значении		10325476			
	Последни	й тег в групповом запросе		False			
	Пересчет	(A*X + B)		False			
	-						

Рисунок 4.10.16 – Добавление тега rVar_OpcWrite

И	мя тега	sVar_OpcWrite					
3	Общие настройки						
	Комментарий						
Γ	Включен	в работу		True			
I	Регион			HOLDING_REGISTERS			
I	Адрес	(	0x0004)	4			
I	Тип данны	ых в устройстве		string			
I	Тип данны	ых в сервере		string			
I	Количест	зо байт для строкового типа		16			
I	Тип строн	ки для строкового типа		ascii			
	Тип досту	ла		ReadWrite			
1	Использо	зать перестановку байтов устроі	йства	True			
	Последни	й тег в групповом запросе		False			
	Пересчет	(A*X + B)		False			

## Рисунок 4.10.17 – Добавление тега sVar_OpcWrite

Имя тега		xVar_OpcRead	
Общие настройки		астройки	
	Коммента	рий	
Γ	Включен	в работу	True
I	Регион		DISCRETE_INPUTS
I	Адрес	(	0x0001)0
I	Тип данных в устройстве		bool
I	Тип данных в сервере		bool
L	Тип досту	/na	ReadOnly

Рисунок 4.10.18 – Добавление тега xVar_OpcRead

И	мя тега	wVar_OpcRead				
🗆 Общие настройки						
	Коммента	рий				
	Включен	в работу		True	1	
	Регион			INPUT_REGISTERS		
	Адрес	(	0x0001)	1		
	Тип данны	ых в устройстве		uint16		
	Тип данны	ых в сервере		uint32		
	Тип досту	па		ReadOnly		

Рисунок 4.10.19 – Добавление тега wVar_OpcRead

и	мя тега	rVar_OpcRead					
	⊒ Юбщие настройки						
_	Коммента	рий					
	Включен	в работу		True			
	Регион			INPUT_REGISTERS			
	Адрес	(0x00)	02)	2			
	Тип данн	ых в устройстве		float			
	Тип данн	ых в сервере		float			
	Тип досту	ла		ReadOnly			
	Использо	вать перестановку байтов устройства	э	False			
L	Перестан	овка байтов в значении		10325476			
	Последни	й тег в групповом запросе		False			
	Пересчет	(A*X + B)		False			

Рисунок 4.10.20 – Добавление тега rVar_OpcRead

Имя тега		sVar_OpcRead			
-	∃ Юбщие настройки				
_	Коммента	рий			
	Включен	в работу		True	
	Регион			INPUT_REGISTERS	
	Адрес		(0x0004)	4	
	Тип данн	ых в устройстве		string	
	Тип данных в сервере			string	
	Количество байт для строкового типа			16	
	Тип строки для строкового типа			ascii	
L	Тип дост	упа		ReadOnly	

Рисунок 4.10.21 – Добавление тега sVar_OpcRead

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

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

Device Application.PLC_PRG				
Выражение	Тип	Значение		
🚸 xVar_OpcRead	BOOL	TRUE		
🚸 wVar_OpcRead	WORD	11		
rVar_OpcRead	REAL	22.33		
🚸 sVar_OpcRead	STRING(16)	'тест'		
WRealWord_OpcRead	Real_Word			
Word_OpcRead	String_Word			
xVar_OpcWrite	BOOL	TRUE		
WVar_OpcWrite	WORD	44		
🖗 rVar_OpcWrite	REAL	55.66		
🕸 sVar_OpcWrite	STRING(16)	'test'		
uRealWord_OpcWrite	Real_Word			
🗉 < uStringWord_OpcWrite	String_Word			

	MasterOPC Universa	l Modbus Server Demo 6	0000 1 hour Build - 4.2.2	8		
Стартовая конфигурация : 653.mbp						
Объекты						
Server	Устройство < <device< th=""><th>21&gt;&gt;</th><th></th><th></th><th></th><th></th></device<>	21>>				
Type: Content of the system of the syst	Имя Node1.Device1.xVa Node1.Device1.wV Node1.Device1.rVa Node1.Device1.sVa Node1.Device1.xVa Node1.Device1.rVa Node1.Device1.rVa Node1.Device1.sVa	Регион         Адрес           DISC         (0x00           INPU         (0x00           INPU         (0x00           Oxnet         (0x00           Oxnet         (0x00           Oxnet         (0x00           HOL         (0x00           HOL         (0x00           HOL         (0x00	Значение True 11 22.330000 тест True 44 55.660000 test	Качество GOOD GOOD GOOD GOOD GOOD GOOD GOOD	Время (UTC) 2019-07-0 2019-07-0 2019-07-0 2019-07-0 2019-07-0 2019-07-0 2019-07-0 2019-07-0	Тип в сер bool uint32 float string bool uint32 float string

Рисунок 4.10.22 – Чтение и запись данных через ОРС-сервер

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

В качестве примера будет рассмотрена настройка обмена с модулями Mx210 (MB210-101 и MK210-301) с использованием **шаблонов**. В примере используются шаблоны версии **3.5.11.4**.

**Реализуемый алгоритм**: если значение первого аналогового входа модуля **MB210-101** превышает **30** и при этом первый дискретный вход модуля **MK210-301** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **MK210-301** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).

Структурная схема примера приведена на рисунке ниже:



Рисунок 4.11.1 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.x. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания: <u>Example_CodesysModbusTcpMasterMx210_3511v1.projectarchive</u> Видеоверсия примера доступна по <u>ссылке</u>. Сетевые параметры устройств приведены в таблице ниже:

Параметр	СПК1хх [М01]	MK210-301	MB210-101	
Режим работы	master	slave	slave	
IP-адрес	10.2.11.170	10.2.11.180	10.2.11.181	
Маска подсети	255.255.0.0			
IP-адрес шлюза		10.2.1.1		
Порт		502		
Unit ID	-	1	1	

Таблица 4.11.1 – Сетевые параметры устройств

Переменные примера описаны в таблице ниже:

Таблица 4.11.2 – Список переменных примера

Модуль	Имя переменной	Тип	Описание
	awModbusPeal	ARRAY [01]	Значение температуры в виде двух <b>WORD</b> ,
	awiwioubusivear	OF WORD	считываемое с модуля
MB210-101			Значение температуры в виде числа с
	rRealValue	REAL	плавающей точкой для использования в
			программе
			Значение дискретных входов в виде битовой
			маски. При обращении к отдельным входам
	WDI	WORD	указывается их номер, начиная с 0:
	WDI	WORLD	<b>wDI.0</b> – состояние первого входа (TRUE/FALSE)
			wDI.1 – состояние второго входа
MK210-301			
	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
			Значение дискретных выходов в виде битовой
			маски из предыдущего цикла программы.
_	wPrevDO	WORD	Используется для отправки команды записи
	WITCODO	WORD	только в случае изменения значений выходов
			(иначе будет производиться циклическая запись
			последнего значения)
			Триггерная переменная, управляющая функцией
-	xTrigger	BOOL	записи дискретного выхода (запись происходит
			по переднему фронту переменной)

Для настройки обмена следует:

1. Настроить модули Mx210 с помощью программы OBEH Конфигуратор в соответствии с <u>таблицей 4.11.1</u> (см. руководство Mx210. Примеры настройки обмена). Подключить модули к контроллеру.

2. Создать новый проект CODESYS с программой PLC_PRG на языке CFC:

Стандартны	й проект		x
	Вы собираете следующие о - Одно прогр - Программа - Циклическа:	сь создать новый стандартный проект. При этом будут созданы бъекты: аммируемое устройство, как задано ниже PLC_PRG на языке, выбранном ниже я задача, вызывающая PLC PRG	
	- Ссылка на н Устройство:	овейшую установленную версию библиотеки Standard library. SPK1xx[M01] (Production association OWEN)	•
	PLC_PRG на:	Непрерывные функциональные схемы (CFC)	•
		ОК От	мена

Рисунок 4.11.2 – Создание проекта CODESYS

3. Добавить в проект объединение с именем Real_Word:



Рисунок 4.11.3 – Добавление в проект объединения

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

Рисунок 4.11.4 – Объявление переменных объединения

**5.** В программе PLC_PRG объявить экземпляр объединения Real_Word с названием _2WORD_TO_REAL, переменные wDI, wDO и wPrevDO типа WORD и переменную xTrigger типа BOOL. Описание переменных приведено в таблице 4.11.2.

<u> </u>	D PLC_PRG X				
	1	PROGRAM PLC_PRG			
	2	VAR			
	3	_2WORD_TO_REAL:	Real_Word;	// значение 1-го входа МВ210-101	
	4	wDI:	WORD;	// битовая маска входов MK210-301	
	5	wDO:	WORD;	// битовая маска выходов МК210-301	
	6	wPrevDO:	WORD;	// битовая маска предыдущей записи выходов МК210-301	
	7	xTrigger:	BOOL;	// триггер записи выходов	
	8	END_VAR			

Рисунок 4.11.5 – Объявление переменных программы

Код программы будет выглядеть следующим образом:



Рисунок 4.11.6 – Код программы PLC_PRG

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

6. Добавить в проект компонент Ethernet.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 4.11.7 – Добавление компонента Ethernet

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

Device X				
Установки соединения	Сканировать сеть Gatewa	ау 👻 Устройство	•	
Приложения				
Резервное копирование и восстановление				
Файлы				•
Журнал		Gateway-1	Gateway 👻	[0000.0BAA] (актив.) — —
Установки ПЛК		IP-Address:		Имя устройства: kie
Оболочка ПЛК		Port:		Адрес устройства:
Пользователи и группы		1217		0000.0BAA
Размещение задачи				1628 0072
Состояние				Тип таргета: 4096
Информация				Производитель таргета: Owen
				Версия таргета: 3.5.11.50

Рисунок 4.11.8 – Подключение к контроллеру

👔 Ethernet 🗙	
Конфигурация ethernet	Интерфейс: евно
Состояние	IP-agpec 10 . 2 . 11 . 170
Ethernet Device Соотнесение	Маска подсети 255 . 255 . 0 . 0
Информация	Gateway по умолчанию 10 . 2 . 1 . 1 Adjust Operating System Settings
	Сегевые адаптеры
	Интерфейсы:
	Имя Описание IP-адрес
	usb0 10.0.6.10
	IP-адрес 10 . 2 . 11 . 170
	Маска подсети 255.255.0.0
	Gateway по умолчанию 10 . 2 . 1 . 1
	MAC-agpec 6A:77:00:DD:BB:AA
	ОК Отмена

Рисунок 4.11.9 – Выбор используемого интерфейса



#### ПРИМЕЧАНИЕ

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

7. В компонент Ethernet добавить компонент Modbus TCP Master.



### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.

йства		- + ×		🖪 Добави	ть устройство				×
Example_TemplatesMx210_3511v1									
Device (SPK1xx[M01])				Имя: М	odbus_TCP_Master				
= Eli Pic Logic				Лействи					
Application				0.006	- Berasura Serasura veri		© 06новить	vernoverso	
П менеджер биолио	отек			C Acco	Sand yerpowerbo	O noticine teleporeneo	Occupant	yerpowerbo	
E Kowhermane sar	nau			Введите	строку для полнотекстового поиска	п: Производитель: <all vendors=""></all>			-
B-SS MainTask	404			Mug		Beoutee surrent	Rencue	Opurchung	
B) PLC PRG				- Gil -		производитель	берски	Oniconne	
😑 🍪 OwenCloudTas	sk				dan.				
- @ OwenStora	age.C	LOUD_PRG			Ethernet/IP				
Ethernet (Ethernet)	V	Remarker		8-1	Modbus				
OwenRTC (OwenRTC)	0	bopesarb			HIM Macrep Modbus TCP				
CovenCloud (OwenCloud)	-	а Копировать			- 📶 Modbus TCP Master	35 - Smart Software Solutions GmbH	3.4.0.0	A device that works as a Modbus Master on Ethernet.	
NB Buzzer (Buzzer)	16	Вставить			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.4.1.0	A device that works as a Modbus Master on Ethernet.	
UU Drives (Drives)	>	Удалить			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.4.2.0	A device that works as a Modbus Master on Ethernet.	
Contract (Network)		Обзор			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.4.3.0	A device that works as a Modbus Master on Ethernet.	-
Debug (Debug)		Pedaktonen:			Modbus TCP Master	3S - Smart Software Solutions GmbH	3.5.0.0	A device that works as a Modbus Master on Ethernet.	-
A Info (Info)					Modbus TCP Master	3S - Smart Software Solutions GmbH	3.5.1.0	A device that works as a Modbus Master on Ethernet.	
Watchdog (Watchdog)	1E	своиства			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.2.0	A device that works as a Modbus Master on Ethernet.	
	ê	Добавление объекта			Modbus TCP Master	35 - Smart Software Solutions GmbH	3540	A device that works as a Moduus Master on Ethernet	
	6	Добавить папку			- fil Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.5.0	A device that works as a Modbus Master on Ethernet.	
		Добавить устройство	<b>→</b>		Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.6.0	A device that works as a Modbus Master on Ethernet.	
		Вставить устройство			- 📶 Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.8.10	A device that works as a Modbus Master on Ethernet.	
		Откл. устройство			- 🚮 Modbus TCP Master	3S - Smart Software Solutions GmbH	3.5.9.0	A device that works as a Modbus Master on Ethernet.	
		Обновить устройство			- Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.10.0	A device that works as a Modbus Master on Ethernet.	
	18	Panartimopath officert			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.11.0	A device that works as a Modbus Master on Ethernet.	
	14				Modbus TCP Master	3S - Smart Software Solutions GmbH	3.5.11.29	A device that works as a Modbus Master on Ethernet.	
		Редактировать объект в			Modbus TCP Master	35 - Smart Software Solutions GmbH	3.5.11.30	A device that works as a Modbus Master on Ethernet.	
		Изменить I/O-соотнесение			Houbus TCP Haster	35 - Smart Software Solutions GmbH	3.3.12.0	A device that works as a modous master on Ethernet.	*
		Импорт соотнесений из CSV							
		Экспортировать соотнесения в CSV							
		Эмуляция		🗸 Груп	ировать по категориям 👿 Отобра	жать все версии (для экспертов) 🛛 🕅 По	жазать устар	евшие версии	
			-						
					we Module TCP Maeter				
					роизводитель: 3S - Smart Software	Solutions GmbH			
					руппы: мастер модрия ГСР ерсия: 3.5.11.30				
					юмер моделя: -	diver Master on Ethernat			3
					Announce A dence dide norks as a Mo	active reason of continee		*	-
				Etherne	гь выоранное устройство как по Н	следнего потомка			
				<b>0</b> (M					
				<b>3</b> (M	ижно евороте другой таргетчузен, п				
								Добавить устройство	Закоыть
								Account Actbourge	Lenberg

Рисунок 4.11.10 – Добавление компонента Modbus TCP Master

В настройках компонента вкладке Общее следует установить галочку Автоподключение.

Modbus_TCP_Master 🗙	
Общее	Modbus-TCP
ModbusTCPMaster Соотнесение входов/выходов	Таймаут ответа (мс) 1000 🚔 Таймаут сокета (мс) 10 🚔
ModbusTCPMaster Конфигурация	и автоподключение
Состояние	
Информация	

Рисунок 4.11.11 – Настройки компонента Modbus TCP Master

8. В компонент Modbus TCP Master следует добавить компоненты Modbus TCP Slave с именами MV210_101 и MK210_301.



#### ПРИМЕЧАНИЕ

Версия компонента должна соответствовать версии таргет-файла. Для отображения всех доступных версий компонента следует установить галочку **Отображать все версии**. См. рекомендации в <u>приложении A</u>.



Рисунок 4.11.12 – Добавление slave-устройств в проект CODESYS

В настройках компонентов следует указать IP-адреса согласно <u>таблице 4.11.1</u> (**МК210-301** – **10.2.11.180**, **МВ210-101** – **10.2.11.181**). На вкладке **Modbus TCPSIave Конфигурация** следует установить для параметра **Unit ID** значение **1**.

Общее	-Modbus-TCP		
Канал Modbus Slave	IP-адрес слейва:	10 . 2 . 11 . 181	MODBUS
Modbus Slave Init	Unit-ID [1247]	1	
ModbusTCPSlave Конфигурация	Таймаут ответа (мс) Порт	1000 502	
ModbusTCPSlave Соотнесение входов/выходов			

Рисунок 4.11.13 – Настройки компонента Modbus TCP Slave, вкладка Общее

MV210_101 X								
Общее	Параметр	Тип	Значение	Значение по умолчанию	Единица	Описание		
	💬 🌵 NewChannelConfig	BOOL	true	true		Use the new Channel-Config format		
Канал Modbus Slave	🖤 < Unit-ID	USINT	1	16#FF		Unit-ID of the Device		
Modbus Slave Init	ResponseTimeout	DWORD	1000	1000		Maximum time for a Slave to respond in ms		
Houbus Slave Inc	IPAddress	ARRAY[03] OF BYTE	[10, 2, 11, 180]	[192, 168, 0, 1]		Configure IP Address of TCP SLave.		
ModbusTCPSlave Конфигурация	🖤 < Port	UINT	502	502		Port where the slave is listening		
	ConfigVersion	UDINT	16#03050B00	16#03050B00				
ModbusTCPSlave Соотнесение входов/выходов								

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

9. В настройках компонента MV210_101 на вкладке Канал Modbus Slave следует добавить канал, в котором с помощью функции Read Holding Registers будет считываться значение 4000 и 4001 регистров модуля. В данных регистрах содержится значение входа 1 в представлении с плавающей точкой. Таблица регистров модуля и поддерживаемые функции приведены в руководстве по эксплуатации.

MV210_101 X	
Общее	Имя Тип доступа Триггер Сдвиг READ Длина Обработка ошибок. Сдвиг WRITE Длина Комментарий
Канал Modbus Slave	Канал Modbus
Modbus Slave Init	Канал
ModbusTCPSlave Конфигурация	Vitra Channel 0
ModbusTCPSlave Соотнесение входов/выходов	Тип доступа Read Holding Registers (Код функции 3)  Триггер Цикл.  Время цикла (нс) 100
Состояние	Комментарий
Информация	Регистр READ Савиг 4000 Длина 2 Сроботла анибах Салариан ала аниан
	Регистр WRITE Сдвиг Охоооо Длина 1
	ОК Оттена
	Добавить канал Удалить Изменить

Рисунок 4.11.15 – Добавление канала в конфигурацию slave-устройства MV210_101

На вкладке ModbusGenericSerialSlave Соотнесение входов/выходов следует привязать к каналу элементы объединения _2WORD_TO_REAL.

Для параметра Всегда обновлять переменные следует установить значение Включено 2.

щее	Найти переменную Фильтр	Показать все		-			
wan Modbus Slave	Переменная	Соотнесение	Канал	Адрес	Тип	Едини	Описание
nan nodbus slave	8-16		Channel 0	%IW0	ARRAY [01] OF WORD		Read Holding Register
dbus Slave Init	Application.PLC_PRG2WORD_TO_REAL.awModbusReal[0]	٦	Channel 0[0]	%IW0	WORD		0x0FA0
	Application.PLC_PRG2WORD_TO_REAL.awModbusReal[1]	۵	Channel 0[1]	%IW1	WORD		0x0FA1
bub rer blare rempin ypagisi		-					
dbusTCPSlave Соотнесение							
дов/выходов							
тояние							
тояние							
ормация							
ормация							
тояние рормация							
ормация							
ормация							
ормация							
рмация							
ояние							
ормация							
ормация							
орнация							

Рисунок 4.11.16 – Привязка переменных к каналу

10. В настройках компонента MK210_301 на вкладке Канал Modbus Slave следует добавить канал, в котором с помощью функции Read Holding Registers будет считываться значение регистра 51. В данном регистре содержится битовая маска состояний дискретных входов. Также следует добавить канал, в котором с помощью функции Write Multiple Registers будет записываться значение в регистр 470. В данном регистре содержатся значения выходов модуля в виде битовой маски. У параметра Триггер следует установить значение Передний фронт, чтобы иметь возможность управлять записью в модуль с помощью логической переменной.

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

MK210_301 X	
Общее	Имя Тип доступа Триггер Сдвиг READ Длина Обработка ошибок Сдвиг WRITE Длина Комментарий
Канал Modbus Slave	
Modbus Slave Init	Kanan Modbus
ModbusTCPSlave Конфигурация	Kanan Viria Channel 0
ModbusTCPSlave Соотнесение входов/выходов	Тип доступа Read Holding Registers (Код функции 3) 💌
Состояние	Тритгер Цикл. 🔻 Время цикла (мс) 100
Информация	Комментарий
	Сдвиг 51
	Длина 1
	Обработка ошибок Сохранить посл. значен 🔻
	Регистр WRITE Сдвиг 0х0000 ч
	Длина 1
	ОК Отмена
L	
	доовынь канал Удалить Изменить

Общее	Имя	Тип доступа	Триггер	Сдвиг READ	Длина	Обработка ошибок	Cдвиг WRITE	Длина	Комментари
Канал Modbus Slave	0 Channel 0	Read Holding Registers (Код функции 03)	Цикл., t#100ms	16#0033	1	Сохранить посл. значение			
Modbus Slave Init	ſ	Канал Modbus			×				
1odbusTCPSlave Конфигурация		Канал							
1odbusTCPSlave Соотнесение іходов/выходов		Имя Channel 1							
Состояние		Тип доступа Write Multiple Registers	(Код функции 16)	(MC) 100					
Інформация		Комментарий		(11)					
		Регистр READ			_ I <b>!</b>				
		Сдвиг				$\mathbf{A}$			
		Длина 1				$\backslash$			
		Обработка ошибок Сохранить посл. :	значен 🔻						
		Регистр WRITE							
		Сдвиг 470		-					
		40 mina 1			_				
			0	К Отмен	на				
	L L		_	_					

Рисунок 4.11.17 – Добавление каналов в конфигурацию slave-устройства MK210_301

MK210_301 X								
Общее	Найти переменную		Фильтр Показать все					
Канал Modbus Slave	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание	
		1	Channel 0	%IW2	ARRAY [00] OF WORD		Read Holding Registers	
Modbus Slave Init	Application.PLC_PRG.wDI	٩	Channel 0[0]	%IW2	WORD		0x0033	
	Application.PLC_PRG.xTrigger	٩	Channel 1	%QX0.0	BIT		Триггерная переменная	
ModbusTCPSlave Конфигурация	i = - <b>*</b> ≱		Channel 1	%QW1	ARRAY [00] OF WORD		Write Multiple Registers	
ModbusTCPSlave Coothecenne	■ K Application.PLC_PRG.wDO	~⊘	Channel 1[0]	<del>%QW1</del>	WORD		0x01D6	
входов/выходов	-							
Состояние								
cocrossine								
Информация								
·								
				Cot	оос соотнесения Всегда о	оновлять пер	еменные: Вкл. 2 (всегда в з	адаче цикла шины) 🔹 🔻

Рисунок 4.11.18 – Привязка переменных к каналам

12. Загрузить проект в контроллер и запустить его.

В переменной _2WORD_TO_REAL.rRealValue будет отображаться текущее значение первого аналогового входа модуля MV210_101. В нулевом бите переменной wDI (wDI.0) будет отображаться текущее значение первого дискретного входа модуля MK210_301.

Если значение _2WORD_TO_REAL.rRealValue превысит 30 и при этом значение wDI.0 будет равно TRUE, то в нулевой бит переменной wDO (wDO.0) будет однократно (по триггеру) записано значение TRUE, что приведет к замыканию первого дискретного выхода модуля MK210_301. Если одно из условий перестанет выполняться, то выход будет разомкнут.



Рисунок 4.11.19 – Выполнение программы в режиме Online

# 4.12 Пример: СПК1xx [M01] (Modbus TCP Slave) + MasterOPC Universal Modbus Server

В качестве примера будет рассмотрена настройка обмена с OPC-сервером <u>Insat MasterOPC Universal</u> <u>Modbus Server</u>, который будет использоваться в режиме **Modbus TCP Master**.

Структурная схема примера приведена на рисунке ниже:



Рисунок 4.12.1 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.x. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания: <u>Example_CodesysModbusTcpSlave_3511v1.zip</u>

Сетевые параметры устройств приведены в таблице ниже:

Таблица 4.12.1 – Сетевые параметры устройств

Параметр	СПК1хх [М01]	ПК с ОРС-сервером		
IP-адрес	10.2.11.170	любой адрес из сети, к которой подключен контроллер		
Порт		502		
Режим работы	slave	master		

Переменные примера описаны в таблице ниже:

### Таблица 4.12.2 – Список переменных примера

Имя	Тип	Адрес регистра/бита							
Переменные, считываемые ОРС-сервером									
xVar_OpcRead	BOOL	Coils	0/0						
wVar_OpcRead	WORD		1						
rVar_OpcRead	REAL	Holding регистры	2–3						
sVar_OpcRead	STRING(16)	4–11							
	Переменные, запи	сываемые ОРС-сервером							
xVar_OpcWrite	BOOL	Discrete inputs	0						
wVar_OpcWrite	WORD		1						
rVar_OpcWrite	REAL	Input регистры	2–3						
sVar_OpcWrite STRING(16) 4–11									

Для настройки обмена следует:

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

Стандартн	ый проект	×							
1	Вы собираетесь создать новый стандартный проект. При этом будут созданы следующие объекты: - Одно программируемое устройство, как задано ниже								
	- Программа PLC_PRG на языке, выбранном ниже - Циклическая задача, вызывающая PLC_PRG - Ссылка на новейшую установленную версию библиотеки Standard library.								
	Устройство: SPK1xx[M01] (Production association OWEN)								
	ОК Отм	іена							

Рисунок 4.12.2 – Создание проекта CODESYS

3. Добавить в проект объединения с именами Real_Word и String_Word:

Устройства		<b>→</b> ∓ X					
SPK_Union		-					
Device (CODESYS Control V	Vin V3	3)				Добавить DUT	
П Менеджер би	*	Вырезать				🔶 Создать новый объект типа данных	
PLC_PRG (PRG		Копировать					
😑 🧱 Конфигурация	12	Вставить				Имя:	
🖹 🍪 MainTask	×	Удалить				Real_Word	
···렌] PLC_PI	C.	Свойства				Тип:	
	4:2	Deferments	- •			 💿 Структура	
		доравить объе	CT P	1	DUT	 Расширение для:	
		дооавить папку		T	Persistent-переменные	Переинсление	
	Ľ	чедактировать	ооъект	<b>a</b>	POU		
		Редактировать	объект в	<b>#</b> 1	РОU для неявных проверок	Баровый тип	
	ОŞ	Логин		ø	Redundancy Configuration		
				•	Symbol configuration	Объединение	
				1	Trend recording manager		
				0	Unit conversion		
				•	Визуализация		
				M	Внешний файл		
				~0	Интерфейс		
				<b>**</b>	Конфигурация тревог		
				-	Менеджер визуализации	Добавить	Отмена
				<b>A</b>	Менеджер просмотра		
				0	Приложение		
					Пул изображений		
					Сервер данных		
					Список глобальных переменных		
					Список сетевых переменных (Отправитель)		
				6	Список сетевых переменных (Получатель)		
					Список текстов		
				<u>0</u> 9	Трассировка		
				-			

Рисунок 4.12.3 – Добавление в проект объединения Real_Word

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

4. В объединениях объявить следующие переменные:



Рисунок 4.12.4 – Объявление переменных объединения Real_Word



Рисунок 4.12.5 – Объявление переменных объединения String_Word

- 5. В менеджере библиотек добавить библиотеку CAA Memory.
- 6. В программе PLC_PRG объявить переменные в соответствии с таблицей 4.13.



Рисунок 4.12.6 – Объявление переменных программы

r/mc_OpeRed       uRenived OpeRed Shing/3ue         uShing/Vord_OpeRed       uShing/Vord_OpeRed Shing/3ue         uShing/Vord_OpeRed avModusShing[0]       winput         ReverseBYTEshNORD       uShing/Vord_OpeRed avModusShing[0]         uShing/Vord_OpeRed avModusShing[1]       winput         ReverseBYTEshNORD       uShing/Vord_OpeRed avModusShing[1]         uShing/Vord_OpeRed avModusShing[2]       winput         ReverseBYTEshNORD       uShing/Vord_OpeRed avModusShing[2]         uShing/Vord_OpeRead avModusShing[3]       winput         ReverseBYTEshNORD       uShing/Vord_OpeRead avModusShing[3]         uShing/Vord_OpeRead avModusShing[3]       winput         ReverseBYTEshNORD       uShing/Vord_OpeRead avModusShing[3]         uShing/Vord_OpeRead avModusShing[3]       winput         ReverseBYTEshNORD       uShing/Vord_OpeRead avModusShing[3]         uShing/Vord_OpeRead avModusShing[5]       winput         ReverseBYTEshNORD       uShing/Vord_OpeRead avModusShing[5]         uShing/Vord_OpeRead avModusShing[5]       winput         ReverseBYTEshNORD       uShing/Vord_OpeRead avModusShing[5]         uShing/Vord_OpeRead avModusShing[6]       winput         uShing/Vord_OpeRead avModusShing[7]       winput         uShing/Vord_OpeRead avModusShing[7]       winput         u	wVar_OpcRead		
	rVar_OpcRead uRealWord_	OpcRead.rRealValue	
MEM. Reversel/FTEsh/WORD         Ustring/Vord_OpcRead.awModbusString[0]           uString/Vord_OpcRead.awModbusString[1]         witput         Reversel/FTEsh/WORD         uString/Vord_OpcRead.awModbusString[1]           uString/Vord_OpcRead.awModbusString[2]         witput         Reversel/FTEsh/WORD         uString/Vord_OpcRead.awModbusString[2]           uString/Vord_OpcRead.awModbusString[2]         witput         Reversel/FTEsh/WORD         uString/Vord_OpcRead.awModbusString[2]           uString/Vord_OpcRead.awModbusString[2]         witput         Reversel/FTEsh/WORD         uString/Vord_OpcRead.awModbusString[2]           uString/Vord_OpcRead.awModbusString[2]         witput         Reversel/FTEsh/WORD         uString/Vord_OpcRead.awModbusString[3]           uString/Vord_OpcRead.awModbusString[3]         witput         Reversel/FTEsh/WORD         uString/Vord_OpcRead.awModbusString[3]           uString/Vord_OpcRead.awModbusString[5]         witput         Reversel/FTEsh/WORD         uString/Vord_OpcRead.awModbusString[5]           uString/Vord_OpcRead.awModbusString[6]         witput         Reversel/FTEsh/WORD         uString/Vord_OpcRead.awModbusString[6]           uString/Vord_OpcRead.awModbusString[7]         witput         Reversel/FTEsh/WORD         uString/Vord_OpcRead.awModbusString[7]           uString/Vord_OpcRead.awModbusString[7]         witput         Reversel/FTEsh/WORD         uString/Vord_OpcRead.awModbusString[7] <tr< th=""><th>svar_openead</th><th>_Operread.souringvalue</th><th></th></tr<>	svar_openead	_Operread.souringvalue	
UString/Vord_OpcRead avModbusString[0]         URE// ReverseBYTEsIn/VORD         UString/Vord_OpcRead avModbusString[0]           UString/Vord_OpcRead avModbusString[1]         MEM_ReverseBYTEsIn/VORD         UString/Vord_OpcRead avModbusString[1]           UString/Vord_OpcRead avModbusString[2]         winput         ReverseBYTEsIn/VORD         UString/Vord_OpcRead avModbusString[2]           UString/Vord_OpcRead avModbusString[2]         winput         ReverseBYTEsIn/VORD         UString/Vord_OpcRead avModbusString[2]           UString/Vord_OpcRead avModbusString[3]         WInput         ReverseBYTEsIn/VORD         UString/Vord_OpcRead avModbusString[3]           UString/Vord_OpcRead avModbusString[3]         WInput         ReverseBYTEsIn/VORD         UString/Vord_OpcRead avModbusString[3]           UString/Vord_OpcRead avModbusString[5]         WInput         ReverseBYTEsIn/VORD         UString/Vord_OpcRead avModbusString[5]           UString/Vord_OpcRead avModbusString[5]         WInput         ReverseBYTEsIn/VORD         UString/Vord_OpcRead avModbusString[6]           UString/Vord_OpcRead avModbusString[7]         WInput         ReverseBYTEsIn/VORD         UString/Vord_OpcRead avModbusString[7]           UString/Vord_OpcRead avModbusString[7]         WInput         ReverseBYTEsIn/VORD         UString/Vord_OpcVinte avModbusString[7]           UString/Vord_OpcVinte avModbusString[7]         WInput         ReverseBYTEsIn/VORD         UString/Vord_OpcVinte		2	
Interest	uStringWord_OpcRead_awModbusString[0]	MEM.ReverseBYTEsInWORD	uStringWord_OpcRead awModbusString[0]
MEM. ReverseBYTEshNORD         uStringtVord_OpcRead awModbusString[1]         uStringtVord_OpcRead awModbusString[2]           uStringtVord_OpcRead awModbusString[2]         wfmput         ReverseBYTEshNORD         uStringtVord_OpcRead awModbusString[2]           uStringtVord_OpcRead awModbusString[2]         wfmput         ReverseBYTEshNORD         uStringtVord_OpcRead awModbusString[2]           uStringtVord_OpcRead awModbusString[3]         wfmput         ReverseBYTEshNORD         uStringtVord_OpcRead awModbusString[3]           uStringtVord_OpcRead awModbusString[4]         wfmput         ReverseBYTEshNORD         uStringtVord_OpcRead awModbusString[4]           uStringtVord_OpcRead awModbusString[5]         wfmput         ReverseBYTEshNORD         uStringtVord_OpcRead awModbusString[5]           uStringtVord_OpcRead awModbusString[5]         wfmput         ReverseBYTEshNORD         uStringtVord_OpcRead awModbusString[5]           uStringtVord_OpcRead awModbusString[5]         wfmput         ReverseBYTEshNORD         uStringtVord_OpcRead awModbusString[7]           uStringtVord_OpcRead awModbusString[7]         wfmput         ReverseBYTEshNORD         uStringtVord_OpcRead awModbusString[7]           uStringtVord_OpcVinite         uStringtVord_OpcRead awModbusString[7]         wfmput         ReverseBYTEshNORD         uStringtVord_OpcVinite awModbusString[7]           uStringtVord_OpcVinite awModbusString[7]         wfmput         ReverseBYTEshNORD<			(-)
Ustring/Vord_OpcRead awModbusString[1] [	0. · · · · · 0. · · · · · · · · · · · ·	MEM.ReverseBYTEsInWORD	
MEM. Reverseb?TEshWORD         Ustring/Word_OpcRead awModbusString[2]           ustring/Word_OpcRead awModbusString[3]         winput         Reverseb?TEshWORD         ustring/Word_OpcRead awModbusString[3]           ustring/Word_OpcRead awModbusString[3]         Winput         Reverseb?TEshWORD         ustring/Word_OpcRead awModbusString[3]           ustring/Word_OpcRead awModbusString[3]         Winput         Reverseb?TEshWORD         ustring/Word_OpcRead awModbusString[3]           ustring/Word_OpcRead awModbusString[5]         Winput         Reverseb?TEshWORD         ustring/Word_OpcRead awModbusString[5]           ustring/Word_OpcRead awModbusString[5]         Winput         Reverseb?TEshWORD         ustring/Word_OpcRead awModbusString[6]           ustring/Word_OpcRead awModbusString[6]         Winput         Reverseb?TEshWORD         ustring/Word_OpcRead awModbusString[7]           ustring/Word_OpcRead awModbusString[7]         Winput         Reverseb?TEshWORD         ustring/Word_OpcRead awModbusString[7]           ustring/Word_OpcRead awModbusString[7]         Winput         Reverseb?TEshWORD         ustring/Word_OpcRead awModbusString[7]           ustring/Word_OpcWrite         winput         Reverseb?TEshWORD         ustring/Word_OpcRead awModbusString[7]           ustring/Word_OpcWrite         wModbusString[7]         Winput         Reverseb?TEshWORD         ustring/Word_Opc/Write awModbusString[7]           ustrin	uStringWord_OpcRead.awModbusString[1]	winput ReverseBYTESInWORD	uStringWord_OpcRead.awModbusString[1]
UStringVord_OpcRead awModbusString[2] whput ReverseBYTEsInVORD UStringVord_OpcRead awModbusString[2] WEM. ReverseBYTEsInVORD UStringVord_OpcRead awModbusString[3] WEM. ReverseBYTEsInVORD UStringVord_OpcRead awModbusString[4] WEM. ReverseBYTEsInVORD UStringVord_OpcRead awModbusString[5] WEM. ReverseBYTEsInVORD UStringVord_OpcRead awModbusString[6] WEM. ReverseBYTEsInVORD UStringVord_OpcRead awModbusString[7] Wenput ReverseBYTEsInVORD UStringVord_OpcRead awModbusString[7] Wenput ReverseBYTEsInVORD UStringVord_OpcRead awModbusString[7] Wenput ReverseBYTEsInVORD UStringVord_OpcRead awModbusString[7] Winput ReverseBYTEsInVORD UStringVord_OpcRead awModbusString[7] Winput ReverseBYTEsInVORD UStringVord_OpcWrite awModbusString[8] Winput ReverseBYTEsInVORD UStringVord_OpcWrite awModbusString[9] Winput ReverseBYTEsInVORD UStrin		MEM.ReverseBYTEsInWORD	(
MEM.ReverseBYTEsInWORD     UstringWord_OpcRead awModbusString[3]     Winput ReverseBYTEsInWORD     UstringWord_OpcRead awModbusString[4]     WEM.ReverseBYTEsInWORD     UstringWord_OpcRead awModbusString[5]     Winput ReverseBYTEsInWORD     UstringWord_OpcRead awModbusString[7]     Winput ReverseBYTEsInWORD     UstringWord_OpcRead awModbusString[7]     Winput ReverseBYTEsInWORD     UstringWord_OpcRead awModbusString[7]     Winput ReverseBYTEsInWORD     UstringWord_OpcRead awModbusString[7]     Winput ReverseBYTEsInWORD     UstringWord_OpcWrite awModbusString[8]     Winput ReverseBY	uStringWord_OpcRead.awModbusString[2]	wInput ReverseBYTEsInWORD	uStringWord_OpcRead.awModbusString[2]
UString/Word_OpcRead awModbusString[3] winput ReverseBYTEsIn/WORD UString/Word_OpcRead awModbusString[4] UString/Word_OpcRead awModbusString[4] Winput ReverseBYTEsIn/WORD UString/Word_OpcRead awModbusString[5] UString/Word_OpcRead awModbusString[5] Winput ReverseBYTEsIn/WORD UString/Word_OpcRead awModbusString[5] Winput ReverseBYTEsIn/WORD UString/Word_OpcRead awModbusString[5] Winput ReverseBYTEsIn/WORD UString/Word_OpcRead awModbusString[5] Winput ReverseBYTEsIn/WORD UString/Word_OpcRead awModbusString[6] Winput ReverseBYTEsIn/WORD UString/Word_OpcRead awModbusString[7] Winput ReverseBYTEsIn/WORD UString/Word_OpcWrite awModbusString[8] Winput ReverseBYTEsIn/WORD UString/Word_OpcWrite awModbusString[9] Winput ReverseBYTEsIn/WORD UString/W		MEM ReverseBYTEsInWORD	
WEM. ReverseBYTEsInWORD         UString/Word_OpcRead awModbusString[4]           uString/Word_OpcRead awModbusString[5]         Winput         ReverseBYTEsInWORD         UString/Word_OpcRead awModbusString[6]           uString/Word_OpcRead awModbusString[5]         Winput         ReverseBYTEsInWORD         UString/Word_OpcRead awModbusString[6]           uString/Word_OpcRead awModbusString[6]         Winput         ReverseBYTEsInWORD         UString/Word_OpcRead awModbusString[6]           uString/Word_OpcRead awModbusString[7]         Winput         ReverseBYTEsInWORD         UString/Word_OpcRead awModbusString[7]           uString/Word_OpcRead awModbusString[7]         winput         ReverseBYTEsInWORD         UString/Word_OpcRead awModbusString[7]           uString/Word_OpcRead awModbusString[7]         winput         ReverseBYTEsInWORD         UString/Word_OpcRead awModbusString[7]           x/dsr_Opc/Write         winput         ReverseBYTEsInWORD         UString/Word_OpcRead awModbusString[7]           uString/Word_Opc/Write awModbusString[7]         winput         ReverseBYTEsInWORD         UString/Word_Opc/Write awModbusString[7]           uString/Word_Opc/Write awModbusString[7]         winput         ReverseBYTEsIn/WORD         UString/Word_Opc/Write awModbusString[7]           uString/Word_Opc/Write awModbusString[7]         winput         ReverseBYTEsIn/WORD         UString/Word_Opc/Write awModbusString[7]           uStrin	uStringWord_OpcRead.awModbusString[3]	winput ReverseBYTEsInWORD	uStringWord_OpcRead.awModbusString[3]
MEM. ReverseBYTEsInWORD       uString/Word_OpcRead awModbusString[4]         uString/Word_OpcRead awModbusString[5]       winput       ReverseBYTEsInWORD       uString/Word_OpcRead awModbusString[5]         uString/Word_OpcRead awModbusString[5]       winput       ReverseBYTEsInWORD       uString/Word_OpcRead awModbusString[5]         uString/Word_OpcRead awModbusString[6]       winput       ReverseBYTEsInWORD       uString/Word_OpcRead awModbusString[5]         uString/Word_OpcRead awModbusString[6]       winput       ReverseBYTEsInWORD       uString/Word_OpcRead awModbusString[7]         vWring/Word_OpcRead awModbusString[7]       winput       ReverseBYTEsInWORD       uString/Word_OpcRead awModbusString[7]         xVar_Opc/Write wwar_Opc/Write uReal/Word_Opc/Write awModbusString[7]       winput       ReverseBYTEsInWORD       uString/Word_OpcWrite awModbusString[7]         uString/Word_Opc/Write awModbusString[7]       winput       ReverseBYTEsInWORD       uString/Word_Opc/Write awModbusString[7]         uString/Word_Opc/Write awMo			
ustring!Word_OpcRead awModbusString[5]       MEM.ReverseBYTEsInWORD       ustring!Word_OpcRead awModbusString[5]         ustring!Word_OpcRead awModbusString[6]       winput       ReverseBYTEsInWORD       ustring!Word_OpcRead awModbusString[6]         ustring!Word_OpcRead awModbusString[6]       winput       ReverseBYTEsInWORD       ustring!Word_OpcRead awModbusString[6]         ustring!Word_OpcRead awModbusString[7]       winput       ReverseBYTEsInWORD       ustring!Word_OpcRead awModbusString[7]         ustring!Word_OpcRead awModbusString[7]       winput       ReverseBYTEsInWORD       ustring!Word_OpcRead awModbusString[7]         ustring!Word_OpcRead awModbusString[7]       winput       ReverseBYTEsInWORD       ustring!Word_OpcRead awModbusString[7]         ustring!Word_OpcWrite       www.awModbusString[7]       winput       ReverseBYTEsInWORD       ustring!Word_Opc/Write awModbusString[7]         ustring!Word_OpcWrite       reverseBYTEsInWORD       ustring!Word_Opc/Write awModbusString[7]       ustring!Word_Opc/Write awModbusString[7]         ustring!Word_Opc/Write awModbusString[8]       winput       ReverseBYTEsInWORD       ustring!Word_Opc/Write awModbusString[8]         ustring!Word_Opc/Write awModbusString[8]       winput       ReverseBYTEsInWORD       ustring!Word_Opc/Write awModbusString[7]         ustring!Word_Opc/Write awModbusString[8]       winput       ReverseBYTEsInWORD       ustring!Word_Opc/Write awModbusString[8] </td <td>uStringWord OpeRead awModbus String[/]</td> <td>MEM.ReverseBYTEsInWORD</td> <td>uString//ord_OpeRead aw/Modbus String[/]</td>	uStringWord OpeRead awModbus String[/]	MEM.ReverseBYTEsInWORD	uString//ord_OpeRead aw/Modbus String[/]
MEM.ReverseBYTEsINVORD     UStringWord_OpcRead awModbusString[5]     Whput ReverseBYTEsINVORD     uStringWord_OpcRead awModbusString[6]     MEM.ReverseBYTEsINVORD     uStringWord_OpcRead awModbusString[6]     Whput ReverseBYTEsINVORD     uStringWord_OpcRead awModbusString[7]     Whput ReverseBYTEsINVORD     uStringWord_OpcRead awModbusString[7]     whput ReverseBYTEsINVORD     uStringWord_OpcRead awModbusString[7]     whput ReverseBYTEsINVORD     uStringWord_OpcRead awModbusString[7]     whput ReverseBYTEsINVORD     uStringWord_OpcWrite awModbusString[8]     whput ReverseBYTEsINVORD     uStringWord_OpcWrite awModbusString[8]     whput ReverseBYTEsINVORD     uStringWord_OpcWrite awModbusString[8]     whput ReverseBYTEsINVORD     uStringWord_OpcWrite awModbusString[6]     winput ReverseBYTEsINVORD     uStringWord_OpcWrite awModbusString[6]     winput ReverseBYTEsINVORD     uStri	dounigword_openead.awmodbusounig[4]	winput NeversebTTEsinivonb	usungword_openead.awmodbussung[+]
uString/Word_OpcRead awModbusString[5] winput ReverseBYTEsihWORD uString/Word_OpcRead awModbusString[6] winput ReverseBYTEsihWORD uString/Word_OpcRead awModbusString[7] winput ReverseBYTEsihWORD uString/Word_OpcRead awModbusString[7] winput ReverseBYTEsihWORD uString/Word_OpcRead awModbusString[7] winput ReverseBYTEsihWORD uString/Word_OpcWrite awModbusString[8] winput ReverseBYTEsihWORD uString/Word_OpcWrite awModbusString[9] winput ReverseBYTEsihWORD uStri		MEM.ReverseBYTEsInWORD	
MEM.ReverseBYTEsInVORD         Mem.ReverseBYTEsInVORD           uStringWord_OpcRead awModbusString[5]         winput         ReverseBYTEsInVORD         uStringWord_OpcRead awModbusString[7]           uStringWord_OpcRead awModbusString[7]         winput         ReverseBYTEsInWORD         uStringWord_OpcRead awModbusString[7]           xVar_OpcWrite	uStringWord_OpcRead.awModbusString[5]	wInput ReverseBYTEsInWORD	uStringWord_OpcRead.awModbusString[5]
uStringWord_OpcRead.awModbusString[6] winput ReverseBYTEsInWORD uStringWord_OpcRead.awModbusString[7] winput ReverseBYTEsInWORD uStringWord_OpcRead.awModbusString[7] xVar_OpcWrite		MEM. ReverseBYTEsInWORD	
MEM.ReverseBYTEsInWORD     uStringWord_OpcRead.awModbusString[7]     winput ReverseBYTEsInWORD     uStringWord_OpcWrite_rRealValue     rVar_OpcWrite     uRealVord_OpcWrite_rRealValue     rVar_OpcWrite     uRealVord_OpcWrite_rRealValue     rVar_OpcWrite     uRealVord_OpcWrite_rRealValue     rVar_OpcWrite     uReverseBYTEsInWORD     uStringWord_OpcWrite_awModbusString[0]     winput ReverseBYTEsInWORD     uStringWord_OpcWrite_awModbusString[2]     winput ReverseBYTEsInWORD     uStringWord_OpcWrite_awModbusString[2]     winput ReverseBYTEsInWORD     uStringWord_OpcWrite_awModbusString[2]     winput ReverseBYTEsInWORD     uStringWord_OpcWrite_awModbusString[3]     winput ReverseBYTEsInWORD     uStringWord_OpcWrite_awModbusString[4]     winput ReverseBYTEsInWORD     uStringWord_OpcWrite_awModbusString[5]     winput ReverseBYTEsInWORD     uStringWord_OpcWrite_awModbusString[6]     winput ReverseBYTEsInWORD     uStringWord_OpcWrite_awModbus	uStringWord_OpcRead.awModbusString[6]	wInput ReverseBYTEsInWORD	uStringWord_OpcRead.awModbusString[6]
uStringWord_OpcRead.awModbusString[7]       whput       ReverseBYTEsInWORD       uStringWord_OpcRead.awModbusString[7]         xVar_OpcWrite       whput       ReverseBYTEsInWORD       uStringWord_OpcRead.awModbusString[7]         uStringWord_OpcWrite rRealValue       rVar_OpcWrite       18         uStringWord_OpcWrite rRealValue       rVar_OpcWrite       18         uStringWord_OpcWrite awModbusString[0]       whput       ReverseBYTEsInWORD       19         uStringWord_OpcWrite awModbusString[1]       whput       ReverseBYTEsInWORD       21         uStringWord_OpcWrite awModbusString[1]       whput       ReverseBYTEsInWORD       23         uStringWord_OpcWrite awModbusString[2]       whput       ReverseBYTEsInWORD       23         uStringWord_OpcWrite awModbusString[2]       whput       ReverseBYTEsInWORD       23         uStringWord_OpcWrite awModbusString[3]       whput       ReverseBYTEsInWORD       27         uStringWord_OpcWrite awModbusString[4]       whput       ReverseBYTEsInWORD       27         uStringWord_OpcWrite awModbusString[5]       whput       ReverseBYTEsInWORD       28         uStringWord_OpcWrite awModbusString[6]       whput       ReverseBYTEsInWORD       29         uStringWord_OpcWrite awModbusString[6]       whput       ReverseBYTEsInWORD       28		MEM D	
xVar_Opc/Vrite	uStringWord OpcRead.awModbusString[7]	winput ReverseBYTEsInWORD	uStringWord OpcRead.awModbusString[7]
xVar_OpcWrite wVar_OpcWrite wRealWord_OpcWrite.rRealValue wRealWord_OpcWrite.rRealValue wRealWord_OpcWrite.rRealValue wRealWord_OpcWrite.rRealValue wRealWord_OpcWrite.awModbusString[0] wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTEsInWORD wRealPreseRPTESINWORD wRealPreseRPTESINWORD wRealPreseRPTESINWORD wRealPreseRPTESINWORD wRealPreseRPTESINWORD wRealPreseRPTESINWORD wRealPreseRPTESINWORD wRealPreseRPTESINWORD wRealPreseRPTE			
MEM.ReverseBYTEsInWORD         13           uStringWord_OpcWrite.awModbusString[0]         winput         ReverseBYTEsInWORD         uStringWord_OpcWrite.awModbusString[0]           uStringWord_OpcWrite.awModbusString[1]         MEM.ReverseBYTEsInWORD         21           uStringWord_OpcWrite.awModbusString[1]         winput         ReverseBYTEsInWORD         uStringWord_OpcWrite.awModbusString[1]           uStringWord_OpcWrite.awModbusString[2]         MEM.ReverseBYTEsInWORD         uStringWord_OpcWrite.awModbusString[2]           uStringWord_OpcWrite.awModbusString[2]         Winput         ReverseBYTEsInWORD         uStringWord_OpcWrite.awModbusString[3]           uStringWord_OpcWrite.awModbusString[3]         Winput         ReverseBYTEsInWORD         uStringWord_OpcWrite.awModbusString[3]           uStringWord_OpcWrite.awModbusString[3]         Winput         ReverseBYTEsInWORD         uStringWord_OpcWrite.awModbusString[3]           uStringWord_OpcWrite.awModbusString[4]         Winput         ReverseBYTEsInWORD         uStringWord_OpcWrite.awModbusString[5]           uStringWord_OpcWrite.awModbusString[5]         Winput         ReverseBYTEsInWORD         uStringWord_OpcWrite.awModbusString[5]           uStringWord_OpcWrite.awModbusString[6]         Winput         ReverseBYTEsInWORD         uStringWord_OpcWrite.awModbusString[6]           uStringWord_OpcWrite.awModbusString[6]         Winput         ReverseBYTEsInWORD	xVar_OpcWrite	18	
uStringWord_OpcWrite.awModbusString[0] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[0] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[1] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[2] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[2] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[2] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[3] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[3] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[3] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[3] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[4] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[5] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[5] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[6] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[7] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusSt	xVar_OpcWrite - wVar_OpcWrite - uRealWord_OpcWrite.rRealValue -	rVar_OpcWrite	
MEM.ReverseBYTEsInWORD         Interview           ustringWord_OpcWrite.awModbusString[1]         whput         ReverseBYTEsInWORD         ustringWord_OpcWrite.awModbusString[1]           ustringWord_OpcWrite.awModbusString[2]         whput         ReverseBYTEsInWORD         23           ustringWord_OpcWrite.awModbusString[2]         whput         ReverseBYTEsInWORD         23           ustringWord_OpcWrite.awModbusString[3]         whput         ReverseBYTEsInWORD         23           ustringWord_OpcWrite.awModbusString[3]         whput         ReverseBYTEsInWORD         23           ustringWord_OpcWrite.awModbusString[4]         whput         ReverseBYTEsInWORD         ustringWord_OpcWrite awModbusString[4]           ustringWord_OpcWrite.awModbusString[5]         whput         ReverseBYTEsInWORD         ustringWord_OpcWrite awModbusString[5]           ustringWord_OpcWrite.awModbusString[5]         whput         ReverseBYTEsInWORD         ustringWord_OpcWrite awModbusString[5]           ustringWord_OpcWrite.awModbusString[6]         whput         ReverseBYTEsInWORD         ustringWord_OpcWrite awModbusString[6]           ustringWord_OpcWrite.awModbusString[6]         whput         ReverseBYTEsInWORD         ustringWord_OpcWrite awModbusString[7]           ustringWord_OpcWrite.awModbusString[7]         whput         ReverseBYTEsInWORD         ustringWord_OpcWrite awModbusString[7]	xVar_OpcWrite	rVar_OpcWrite	
uStringWord_OpcWrite.awModbusString[1]       whput       ReverseBTTEsInWORD       uStringWord_OpcWrite.awModbusString[2]         uStringWord_OpcWrite.awModbusString[2]       whput       ReverseBTTEsInWORD       33         uStringWord_OpcWrite.awModbusString[3]       whput       ReverseBTTEsInWORD       33         uStringWord_OpcWrite.awModbusString[3]       whput       ReverseBTTEsInWORD       33         uStringWord_OpcWrite.awModbusString[4]       whput       ReverseBTTEsInWORD       33         uStringWord_OpcWrite.awModbusString[5]       whput       ReverseBTTEsInWORD       33         uStringWord_OpcWrite.awModbusString[6]       whput       ReverseBTTEsInWORD       33         uStringWord_OpcWrite.awModbusString[7]       whput       ReverseBTTEsInWORD       33         uStringWord_OpcWrite.awModbusString[7]       whput       ReverseBTTEsInWORD       33         uStringWord_OpcWrite.awModbusString[7]       whput       ReverseBTTEsInWORD       33         uStringWord_OpcWrite.awModbusString[7]       whput       ReverseBTTEsInWORD       33         uStringWord_OpcWrite.astringValue       sVar OpcWrite       33	xVar_OpcWrite	rVar_OpcWrite	uStringWord_OpcWrite.awModbusString[0]
Image: String Word_OpcWrite.awModbusString[2]       Image: String Word_OpcWrite.awModbusString[2]         Image: String Word_OpcWrite.awModbusString[2]       Image: String Word_OpcWrite.awModbusString[2]         Image: String Word_OpcWrite.awModbusString[3]       Image: String Word_OpcWrite.awModbusString[3]         Image: String Word_OpcWrite.awModbusString[3]       Image: String Word_OpcWrite.awModbusString[3]         Image: String Word_OpcWrite.awModbusString[4]       Image: String Word_OpcWrite.awModbusString[4]         Image: String Word_OpcWrite.awModbusString[5]       Image: String Word_OpcWrite.awModbusString[5]         Image: String Word_OpcWrite.awModbusString[6]       Image: String Word_OpcWrite.awModbusString[6]         Image: String Word_OpcWrite.awModbusString[6]       Image: String Word_OpcWrite.awModbusString[6]         Image: String Word_OpcWrite.awModbusString[7]       Image: String Word_OpcWrite.awModbusString[7]         Image: String Word_OpcWrite.astring Vord_OpcWrite.astring V	xVar_OpcWrite	MEM.ReverseBYTEsInWORD  MEM.ReverseBYTEsInWORD  Winput ReverseBYTEsInWORD  21  21	uStringWord_OpcWrite.awModbusString[0]
MEM.ReverseBYTEsInWORD     uStringWord_OpcWrite.awModbusString[2]     whnput ReverseBYTEsInWORD     uStringWord_OpcWrite.awModbusString[3]     whnput ReverseBYTEsInWORD     uStringWord_OpcWrite.awModbusString[4]     whnput ReverseBYTEsInWORD     uStringWord_OpcWrite.awModbusString[5]     whnput ReverseBYTEsInWORD     uStringWord_OpcWrite.awModbusString[5]     whnput ReverseBYTEsInWORD     uStringWord_OpcWrite.awModbusString[6]     whnput ReverseBYTEsInWORD     uStringWord_OpcWrite.awModbusString[7]	xVar_OpcWrite	MEM.ReverseBYTEsInWORD  MEM.ReverseBYTEsInWORD  MEM.ReverseBYTEsInWORD  21  winput ReverseBYTEsInWORD  21  winput ReverseBYTEsInWORD	uString\Vord_Opc\Vrite.awModbusString[0] {
ustring/vord_Opc/vrite.aw/ModbusString[2]       winput       ReversebTTEsinWORD       ustring/vord_Opc/vrite.aw/ModbusString[2]         ustring/vord_Opc/vrite.aw/ModbusString[3]       winput       ReversebTTEsinWORD       ustring/vord_Opc/vrite.aw/ModbusString[3]         ustring/vord_Opc/vrite.aw/ModbusString[3]       winput       ReversebTTEsinWORD       ustring/vord_Opc/vrite.aw/ModbusString[3]         ustring/vord_Opc/vrite.aw/ModbusString[4]       winput       ReversebTTEsinWORD       ustring/vord_Opc/vrite.aw/ModbusString[4]         ustring/vord_Opc/vrite.aw/ModbusString[5]       winput       ReversebTTEsinWORD       ustring/vord_Opc/vrite.aw/ModbusString[5]         ustring/vord_Opc/vrite.aw/ModbusString[6]       winput       ReversebTTEsinWORD       ustring/vord_Opc/vrite.aw/ModbusString[6]         ustring/vord_Opc/vrite.aw/ModbusString[6]       winput       ReversebTTEsinWORD       ustring/vord_Opc/vrite.aw/ModbusString[6]         ustring/vord_Opc/vrite.aw/ModbusString[6]       winput       ReversebTTEsinWORD       ustring/vord_Opc/vrite.aw/ModbusString[7]         ustring/vord_Opc/vrite.aw/ModbusString[7]       winput       ReversebTEsinWORD       ustring/vord_Opc/vrite.aw/ModbusString[7]         ustring/vord_Opc/vrite.aw/ModbusString[7]       winput       ReversebTEsinWORD       ustring/vord_Opc/vrite.aw/ModbusString[7]         ustring/vord_Opc/vrite.aw/ModbusString[7]       winput       ReversebTEsinWORD       ustring/vord_Op	xVar_DpcWrite - wVar_DpcWrite - uRealWord_OpcWrite.rRealValue uStringWord_OpcWrite.awModbusString[0]	MEM.ReverseBYTEsInWORD	uStringWord_OpcWrite.awModbusString[0]
MEM. ReverseBYTEsInWORD         [25]           uString/Vord_Opc/Vrite awModbusString[3]         wInput         ReverseBYTEsInWORD         uString/Vord_Opc/Vrite awModbusString[3]           uString/Vord_Opc/Vrite awModbusString[4]         wInput         ReverseBYTEsInWORD         [27]           uString/Vord_Opc/Vrite awModbusString[4]         wInput         ReverseBYTEsInWORD         [27]           uString/Vord_Opc/Vrite awModbusString[4]         wInput         ReverseBYTEsInWORD         [28]           uString/Vord_Opc/Vrite awModbusString[5]         wInput         ReverseBYTEsInWORD         [28]           uString/Vord_Opc/Vrite awModbusString[6]         wInput         ReverseBYTEsInWORD         [29]           uString/Vord_Opc/Vrite awModbusString[6]         wInput         ReverseBYTEsInWORD         [31]           uString/Vord_Opc/Vrite awModbusString[6]         wInput         ReverseBYTEsInWORD         [33]           uString/Vord_Opc/Vrite awModbusString[7]         wInput         [33]         [33]	xVar_UpcV/rite	MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD 21 MEM.ReverseBYTESINWORD 22 MEM.ReverseBYTESINWORD 23 MEM.ReverseBYTESINWORD 23 MEM.ReverseBYTESINWORD 23 24 24 25 25 25 25 25 25	uStringWord_OpcWrite.awModbusString[0] ( uStringWord_OpcWrite.awModbusString[1] (
uStringWord_OpcWrite awModbusString[3] wInput ReverseBYTEsInWORD uStringWord_OpcWrite awModbusString[3] WEM.ReverseBYTEsInWORD uStringWord_OpcWrite awModbusString[4] wInput ReverseBYTEsInWORD uStringWord_OpcWrite awModbusString[5] WINput ReverseBYTEsInWORD uStringWord_OpcWrite awModbusString[5] wInput ReverseBYTEsInWORD uStringWord_OpcWrite awModbusString[6] WINput ReverseBYTEsInWORD uStringWord_OpcWrite awModbusString[7] Winput ReverseBYTESInWORD UStringWord_OpcWrite awModbusStr	xVar_UppcV/rite	MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD 21 winput ReverseBYTEsInWORD 23 winput ReverseBYTEsInWORD 23 24 23 24 24 25 25 25 26 26 27 27 27 28 29 29 20 20 20 20 20 20	uStringWord_OpcWrite.awModbusString[0] { uStringWord_OpcWrite.awModbusString[1] { uStringWord_OpcWrite.awModbusString[2] {
	xVar_Opc\Write - wVar_Opc\Write - uReal\Word_Opc\Write.rRealValue uString\Word_Opc\Vrite.awModbusString[0] uString\Word_Opc\Vrite.awModbusString[1] uString\Word_Opc\Vrite.awModbusString[2]		uStringWord_OpcWrite.awModbusString[0]     uStringWord_OpcWrite.awModbusString[1]     uStringWord_OpcWrite.awModbusString[2]
uStringWord_OpcWrite.awModbusString[4] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[4] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[5] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[6] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[6] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[6] wInput ReverseBYTEsInWORD uStringWord_OpcWrite.awModbusString[7] uStringWord_OpcWrite.awModbusString[7] uStringWord_OpcWrite.awModbusString[7] uStringWord_OpcWrite.awModbusString[7] uStringWord_OpcWrite.awModbusString[7] uStringWord_OpcWrite.astringValue uStringWord_OpcWrite.astringValue uStringWord_OpcWrite.astringValue uStringWord_OpcWrite.astringValue uStringWord_OpcWrite.astringValue uStringVord_OpcWrite.astringValue uStringWord_OpcWrite.astringValue uStringVord_OpcWrite.astringValue uStringVord	xVar_Opc\Write - wVar_Opc\Write - uReal\Word_Opc\Write.rReal\Value uString\Word_Opc\Vrite.awModbusString[0] uString\Word_Opc\Vrite.awModbusString[1] uString\Word_Opc\Vrite.awModbusString[2] uString\Word_Opc\Vrite.awModbusString[3]	rVar_OpcWrite     19       MEM.ReverseBYTEsInWORD     19       wInput     ReverseBYTEsInWORD       MEM.ReverseBYTEsInWORD     21       wInput     ReverseBYTEsInWORD       MEM.ReverseBYTEsInWORD     23       wInput     ReverseBYTEsInWORD       MEM.ReverseBYTEsInWORD     23       wInput     ReverseBYTEsInWORD       MEM.ReverseBYTEsInWORD     23       wInput     ReverseBYTEsInWORD       wInput     ReverseBYTEsInWORD	uStringWord_OpcWrite_awModbusString[0] {     uStringWord_OpcWrite_awModbusString[1] {     uStringWord_OpcWrite_awModbusString[2] {     uStringWord_OpcWrite_awModbusString[3] { }
	xVar_OpclWrite - wVar_OpclWrite - uRealWord_OpcWrite.rRealValue		uStringWord_OpcWrite_awModbusString[0]     uStringWord_OpcWrite_awModbusString[1]     uStringWord_OpcWrite_awModbusString[2]     uStringWord_OpcWrite_awModbusString[3]
MEM.ReverseBTIEsInWORD     uStringWord_OpcWrite.awModbusString[5]     wInput ReverseBTIEsInWORD     uStringWord_OpcWrite.awModbusString[6]     MEM.ReverseBTIEsInWORD     uStringWord_OpcWrite.awModbusString[6]     MEM.ReverseBTIEsInWORD     uStringWord_OpcWrite.awModbusString[7]     wInput ReverseBTIEsInWORD     uStringWord_OpcWrite.awModbusString[7]	xVar_Opc\Write wVar_Opc\Write uRealWord_Opc\Write.rRealValue uStringWord_Opc\Write.awModbusString[0] uStringWord_Opc\Write.awModbusString[1] uStringWord_Opc\Write.awModbusString[2] uStringWord_Opc\Write.awModbusString[3] uStringWord_Opc\Write.awModbusString[3] uStringWord_Opc\Write.awModbusString[4]	rVar_OpcWrite         MEM.ReverseBYTEsInWORD         winput       ReverseBYTEsInWORD         MEM.ReverseBYTEsInWORD       21         winput       ReverseBYTEsInWORD	uStringWord_OpcWrite.awModbusString[0]     uStringWord_OpcWrite.awModbusString[1]     uStringWord_OpcWrite.awModbusString[2]     uStringWord_OpcWrite.awModbusString[3]     uStringWord_OpcWrite.awModbusString[3]
	xVar_DpcWrite - wVar_DpcWrite - uRealWord_OpcWrite.rRealValue uStringWord_OpcWrite.awModbusString[0] uStringWord_OpcWrite.awModbusString[1] uStringWord_OpcWrite.awModbusString[2] uStringWord_OpcWrite.awModbusString[3] uStringWord_OpcWrite.awModbusString[4]		UStringWord_OpcWrite.awModbusString[0] UStringWord_OpcWrite.awModbusString[1] UStringWord_OpcWrite.awModbusString[2] UStringWord_OpcWrite.awModbusString[3] UStringWord_OpcWrite.awModbusString[3] UStringWord_OpcWrite.awModbusString[4]
MEM.ReverseBYTEsInWORD     UstringWord_OpcWrite.awModbusString[6]     MEM.ReverseBYTEsInWORD     uStringWord_OpcWrite.awModbusString[6]     MEM.ReverseBYTEsInWORD     uStringWord_OpcWrite.awModbusString[7]     wInput ReverseBYTEsInWORD     uStringWord_OpcWrite.awModbusString[7]     uStringWord_OpcWrite.awModbusString[7]	xVar_DpcWrite		UStringWord_OpcWrite.awModbusString[0] UStringWord_OpcWrite.awModbusString[1] UStringWord_OpcWrite.awModbusString[2] UStringWord_OpcWrite.awModbusString[3] UStringWord_OpcWrite.awModbusString[4] UStringWord_OpcWrite.awModbusString[4] UStringWord_OpcWrite.awModbusString[4]
ustringWord_OpcWrite.awModbusString[0]	xVar_OpcV/rite	MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTEsInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESInWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.ReverseBYTESINWORD MEM.Re	UStringWord_OpcWrite.awModbusString[0]
UString/Vord_Opc/Vrite.awModbusString[7]	xVar_Opc/Write - wVar_Opc/Write - uReal/Word_Opc/Write.rReal/Value uString/Word_Opc/Write.awModbusString[0] uString/Word_Opc/Write.awModbusString[1] uString/Word_Opc/Write.awModbusString[2] uString/Word_Opc/Write.awModbusString[3] uString/Word_Opc/Write.awModbusString[4] UString/Word_Opc/Write.awModbusString[5]	War_OpcWrite         MEM.ReverseBYTEsInWORD         winput         ReverseBYTEsInWORD         MEM.ReverseBYTESINWORD         MEM.ReverseBYTESINWORD	uStringWord_OpcWrite.awModbusString[0] ( uStringWord_OpcWrite.awModbusString[1] ( uStringWord_OpcWrite.awModbusString[2] ( uStringWord_OpcWrite.awModbusString[3] ( uStringWord_OpcWrite.awModbusString[4] ( uStringWord_OpcWrite.awModbusString[5] (
uStringWord_OpcWrite awModbusString[7]	xVar_OpcWrite - wVar_OpcWrite - uRealWord_OpcWrite.rRealValue uStringWord_OpcWrite.awModbusString[0] - uStringWord_OpcWrite.awModbusString[1] - uStringWord_OpcWrite.awModbusString[2] - uStringWord_OpcWrite.awModbusString[3] - uStringWord_OpcWrite.awModbusString[4] - uStringWord_OpcWrite.awModbusString[5] - uStringWord_OpcWrite.awModbusString[5] -	rVar_Opc/Write         MEM.ReverseBYTEsIn/vORD         winput         ReverseBYTEsIn/vORD	UStringWord_OpcWrite.awModbusString[0] UStringWord_OpcWrite.awModbusString[1] UStringWord_OpcWrite.awModbusString[2] UStringWord_OpcWrite.awModbusString[3] UStringWord_OpcWrite.awModbusString[4] UStringWord_OpcWrite.awModbusString[5] UStringWord_OpcWrite.awModbusString[5] UStringWord_OpcWrite.awModbusString[5]
uStringWord OpcWrite.sStringValue	xVar_OpclWrite - wVar_OpclWrite - uRealWord_OpclWrite.rRealValue uStringWord_OpclWrite.awModbusString[0] uStringWord_OpclWrite.awModbusString[1] uStringWord_OpclWrite.awModbusString[2] uStringWord_OpclWrite.awModbusString[4] uStringWord_OpclWrite.awModbusString[5] uStringWord_OpclWrite.awModbusString[5] uStringWord_OpclWrite.awModbusString[6]	rVar_Opc/Write     19       wInput     ReverseBYTEsInWORD       MEM.ReverseBYTEsInWORD     23       WInput     ReverseBYTEsInWORD       MEM.ReverseBYTEsInWORD     31       WInput     ReverseBYTEsInWORD       MEM.ReverseBYTEsInWORD     31       WInput     ReverseBYTEsInWORD       MEM.ReverseBYTESInWORD     31       WInput     ReverseBYTESInWORD	UStringWord_OpcWrite.awModbusString[0] UStringWord_OpcWrite.awModbusString[1] UStringWord_OpcWrite.awModbusString[2] UStringWord_OpcWrite.awModbusString[3] UStringWord_OpcWrite.awModbusString[4] UStringWord_OpcWrite.awModbusString[5] UStringWord_OpcWrite.awModbusString[5] UStringWord_OpcWrite.awModbusString[5] UStringWord_OpcWrite.awModbusString[5] UStringWord_OpcWrite.awModbusString[5] UStringWord_OpcWrite.awModbusString[6] UStringWord_OpcWrite.awModbusString[6]
	xVar_OpclWrite - wVar_OpclWrite - uRealWord_OpclWrite.rRealValue uStringWord_OpclWrite.awModbusString[0] - uStringWord_OpclWrite.awModbusString[1] - uStringWord_OpclWrite.awModbusString[2] - uStringWord_OpclWrite.awModbusString[3] - uStringWord_OpclWrite.awModbusString[4] uStringWord_OpclWrite.awModbusString[5] uStringWord_OpclWrite.awModbusString[6] uStringWord_OpclWrite.awModbusString[6] uStringWord_OpclWrite.awModbusString[7]	rVar_OpcWrite         MEM.ReverseBYTEsInWORD         winput         ReverseBYTEsInWORD         Winput         Rev	UStringWord_OpcWrite.awModbusString[0] { UStringWord_OpcWrite.awModbusString[1] { UStringWord_OpcWrite.awModbusString[2] { UStringWord_OpcWrite.awModbusString[3] { UStringWord_OpcWrite.awModbusString[4] { UStringWord_OpcWrite.awModbusString[5] { UStringWord_OpcWrite.awModbusString[6] { UStringWord_OpcWrite.awModbusString[6] { UStringWord_OpcWrite.awModbusString[6] { UStringWord_OpcWrite.awModbusString[6] { UStringWord_OpcWrite.awModbusString[6] { UStringWord_OpcWrite.awModbusString[6] { UStringWord_OpcWrite.awModbusString[7] { USTRINGWOrd_OpcWrite.awModbusS

Код программы будет выглядеть следующим образом:

Рисунок 4.12.7 – Код программы PLC_PRG

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

**7.** Добавить компоненты **Ethernet** и **Modbus TCP Slave Device** в соответствии с <u>п. 4.4</u>. Настроить компонент **Modbus TCP Slave Device** в соответствии с рисунком ниже:

ница конфигурации	Заданные параметры						
bus TCP Slave Device	🔲 Сторожевой таймер:	500	* *	(мс)			
есение входов/выходов	Slave-nopt:	502	*				
мация	Unit ID:						
	Регистры временного хранения (%IW):	12	*				
	Входные регистры (% OW):	12	*				
				1			
	-Модель данных Начальные адреса:			1			
	Модель данных Начальные адреса: Регистры:		)				
	Модель данных Начальные адреса: Регистры: Дискретные входы:	C	)	×			
	Модель данных Начальные адреса: Регистры: Дискретные входы: Регистр временного хра	с анения: С	)				

Рисунок 4.12.8 – Настройки компонента Modbus TCP Slave Device

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

конфигурации	Найти переменную Фильт	•					
'CP Slave Device	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
ние входов/выходов			Входы	%IW0	ARRAY [011] OF WORD		Регистры временного хранения Modb
ция	😑 - 🍫		Входы[0]	%IW0	WORD		
	Application.PLC_PRG.xVar_OpcWrite	کې	Bit0	%IX0.0	BOOL		
			Bit1	%IXO.1	BOOL		
	🔍 🛛 🗠 🐎		Bit2	%IX0.2	BOOL		
			Bit3	%IX0.3	BOOL		
	*>		Bit4	%IX0.4	BOOL		
	<b>*</b> >		Bit5	%IX0.5	BOOL		
	🐐		Bit6	%IX0.6	BOOL		
	*>		Bit7	%IX0.7	BOOL		
	<b>*</b> >		Bit8	%IX1.0	BOOL		
			Bit9	%IX1.1	BOOL		
			Bit10	%IX1.2	BOOL		
			Bit11	%IX1.3	BOOL		
	**		Bit12	%IX1.4	BOOL		
	**		Bit13	%IX1.5	BOOL		
	*>		Bit14	%IX1.6	BOOL		
			Bit15	%IX1.7	BOOL		
	Application.PLC_PRG.wVar_OpcWrite	<b>*</b>	Входы[1]	%IW1	WORD		
	Application.PLC_PRG.uRealWord_OpcWrite.awModbusReal	[0] 🌍	Входы[2]	%IW2	WORD		
	Application.PLC_PRG.uRealWord_OpcWrite.awModbusReal	[1] 🌍	Входы[3]	%IW3	WORD		
	Application.PLC_PRG.uStringWord_OpcWrite.awModbusStri	ng[0] 👘	Входы[4]	96IW4	WORD		
	Application.PLC_PRG.uStringWord_OpcWrite.awModbusStri	ng[1] 🗳	Входы[5]	%IW5	WORD		
	Application.PLC_PRG.uStringWord_OpcWrite.awModbusStri	ng[2] 🏾 🌍	Входы[6]	%IW6	WORD		
	Application.PLC_PRG.uStringWord_OpcWrite.awModbusStri	ng[3] 🖓	Входы[7]	%IW7	WORD		
	Application.PLC_PRG.uStringWord_OpcWrite.awModbusStri	ng[4] 🛛 🖗	Входы[8]	<del>%IW8</del>	WORD		
	Application.PLC_PRG.uStringWord_OpcWrite.awModbusStri	ng[5] 🔷 🖗	Входы[9]	<del>%IW9</del>	WORD		
	Application.PLC_PRG.uStringWord_OpcWrite.awModbusStri	ng[6] 🗳	Входы[10]	%IW 10	WORD		
	Application.PLC_PRG.uStringWord_OpcWrite.awModbusStri	ng[7] 🗳	Входы[11]	%IW11	WORD		
	B- <b>%</b>		Выходы	%QW0	ARRAY [011] OF WORD		Входные регистры Modbus
	1 - <b>*</b>		Выходы[0]	%QW0	WORD		
	Application.PLC_PRG.wVar_OpcRead	<b>*</b>	Выходы[1]	%QW1	WORD		

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

Страница конфигурации	Наити переменную	Фильтр Показа	ть все	•					
Modbus TCP Slave Device	Переменная		Соотнесение	Канал	Адрес	Тип	Единица	Описание	*
Соотнесение входов/выходов	Application.PLC_PRG.uStringWord	_OpcWrite.awModbusString[5]	<b>*</b>	Входы[9]	%IW9	WORD			
Информация	Application.PLC_PRG.uStringWord	OpcWrite.awModbusString[6]	2	Входы[10]	%IW10	WORD			
	Application.PLC_PRG.uStringWord	OpcWrite.awModbusString[7]	2	Входы[11]	%IW11	WORD			
	ii			Выходы	%QW0	ARRAY [011] OF WORD		Входные регистры Modbus	
				Выходы[0]	%QW0	WORD			
	Application.PLC_PRG.xVar_Op	cRead	~ <b>*</b>	Bit0	%QX0.0	BOOL			
	***			Bit1	%QX0.1	BOOL			
	<b>*</b> ø			Bit2	%QX0.2	BOOL			
	<b>*</b> •			Bit3	%QX0.3	BOOL			
				Bit4	%QX0.4	BOOL			
				Bit5	%QX0.5	BOOL			
	- <b>*</b> ø			Bit6	%QX0.6	BOOL			
	<b>*</b> @			Bit7	%QX0.7	BOOL			
	***			Bit8	%QX1.0	BOOL			
	<b>*</b> @			Bit9	%QX1.1	BOOL			
				Bit10	%QX1.2	BOOL			
				Bit11	%QX1.3	BOOL			
	<b>*</b> •			Bit12	%QX1.4	BOOL			
	<b>*</b> ø			Bit13	%QX1.5	BOOL			
	<b>*</b> ø			Bit14	%QX1.6	BOOL			=
	<b>*</b> ø			Bit15	%QX1.7	BOOL			
	Application.PLC_PRG.wVar_OpcRe	ad	<b>*</b>	Выходы[1]	%QW1	WORD			
	Application.PLC_PRG.uRealWord_(	OpcRead.awModbusReal[0]	<b>*</b>	Выходы[2]	<del>%Q₩2</del>	WORD			
	Application.PLC_PRG.uRealWord_(	OpcRead.awModbusReal[1]	<b>*</b>	Выходы[3]	%QW3	WORD			
	Application.PLC_PRG.uStringWord	_OpcRead.awModbusString[0]	<b>*</b>	Выходы[4]	%QW4	WORD			
	Application.PLC_PRG.uStringWord	_OpcRead.awModbusString[1]	۵.	Выходы[5]	%QW5	WORD			
	Application.PLC_PRG.uStringWord	_OpcRead.awModbusString[2]	۵.	Выходы[6]	<del>%QW6</del>	WORD			
	Application.PLC_PRG.uStringWord	_OpcRead.awModbusString[3]	<b>*</b>	Выходы[7]	%QW7	WORD			
	Application.PLC_PRG.uStringWord	_OpcRead.awModbusString[4]	~ <b>`</b> }	Выходы[8]	%QW8	WORD			
	Application.PLC_PRG.uStringWord	_OpcRead.awModbusString[5]	°.	Выходы[9]	%QW9	WORD			
	Application.PLC_PRG.uStringWord	_OpcRead.awModbusString[6]	°.	Выходы[10]	%QW10	WORD			
	KA ANTANYA PLO PRO PRO PROVINCIA	0.0.1.1.1.1.0.1.1.171	· · ·	a seat		141000			

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

9. Установить и запустить MasterOPC Universal Modbus Server.

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

	-	MasterOPC Universal M	odbus Server Demo 60000 1 hour Build -	4.2.28	
Конф	оигурация Общие настрой	іки Помощь			
🗍 Создать	💾 Сохранить как	🗓 Добавить узел 👘 Добавить тег	🖃 Переименовать 🗙 Удалить	b	
🖗 Открыть	🖋 Сделать стартовым	🧠 Добавить устройство  🕆 Переместить вверх	🗍 Копировать 👘 Отмени	ть	
📔 Сохраните	ь 📋 Импорт из версии 2.0	🍕 Добавить группу 🛛 🗍 Переместить вниз	📋 Вставить 🛛 🔎 Вернуть	ь	
Φai	йл конфигурации	Сервер	Правка	Редактирование коммуникационного узла	
Текущая конс	фигурация :		°	·	
Объекты				Имя узла Node1	
Server	Лобавить	Соммуникационный узел		⊟ Юбщие настройки	
	En una anno 1		-	Комментарий	
	групповые операции			Включен в работу	True
	Импорт узла			Пип узла	ТСР/ІР
	Узлы в работе			ПР адрес	10 2 11 170
				ІР порт	502
				Время ожидания соединения (с)	10
				Повторы при ошибке	3
				Межсимвольный таймаут (мс)	1000
			e	ВСкрипт	
				Выполнение скрипта	False
				<ul> <li>Дополнительные настройки</li> </ul>	Calae
					False
				Принудительный разрыв соединения в каждом цикле	False
				Отслеживать Transaction ID	True
				Подключение в режиме ТСР сервера	False
				Использовать резервные каналы	False
			-		
Режим	Конфигурирование	Теги : Разрешено - 60000. Загружено - 0		П Тиражировать 1	Да Нет

Рисунок 4.12.11 – Добавление коммуникационного узла
11. Нажать ПКМ на коммуникационный узел и добавить устройство с настройками по умолчанию.

<b>P</b>		MasterOPC Universal Mod	lbus Server Demo 60000 1 hour B	luild - 4.2.28	
Конфиг	урация Общие настр	йки Помощь			
🗐 Создать	💾 Сохранить как	Добавить узел	🖏 Добавить тег	🖃 Переименовать 🇙 Удалить	
🖗 Открыть	🖋 Сделать стартовым	🕅 Добавить устройство	👚 Переместить вверх	🖲 Копировать 🖛 Отменить	
П Сохранить	П Импорт из версии 2.0	Добавить группу			_ 0 _
Файл	конфигурации	Cer	Редактирование устройств	la (	
екушая конфи	15VD2UM9 : 653 mbp		Имя устройства Devic	ce1	
бъекты	a ypaquin : ooonnop				
			Сощие настроики		
Server			Вилюцена в воботи		True
Добавить Устройство Типусториства Мос					MODBUS
				1	
Переименовать Адрес (0x01) 1				1000	
	дуолировать		Повторы при ощибке		
	Удалить		Повторы при ошиоке		10
	Вырезать		Реинициализация узла	а при ошибке	False
	Копировать		Период опроса		1000
	Групповые операции		Размерность периода	опроса	ms
	Экспорт узла		Начальная фаза		0
	Импорт устройства		Размерность фазы		ms
	Verneŭerne o nefere		Старт после запуска		True
	устроиства в работе		Задержка запроса по	сле получения ответа (мс)	4
			Перестановка байтов	в значении	Вызов
			Скрипт		
			Выполнение скрипта		False
			Настройка запросов	B	
			Максимальное количе	ество HOLDING регистров в запросе чтения	125
			Максимальное количе	ество INPUT регистров в запросе чтения	125
			Не использовать кома	анду WRITE_SINGLE_COIL (0х05)	True
			П Тиражировать	Да	Нет
жим Кон	нфигурирование	Теги : Разрешено -	60000. Загружено - 8		

Рисунок 4.12.12 – Добавление устройства

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

	MasterOPC Universal Modbus Se	rver Demo 60000 1 hour Build - 4.1.1		x	
Конфигурация Общие настройки	Помощь				
🗐 Создать 💾 Сохранить как 🕅	Добавить узел 📫 Добавить тег	🖃 Переименовать 🗙 Удалить			
🖗 Открыть 🛛 🛷 Сделать стартовым 🛛 🦏	Добавить устройство 🔺 Переместить в				
Кохранить П Импорт из версии 2.0 4	Добавить группу 🚽 Переместить в	Гедактирование тега			
Файл конфигурации	Сервер	Имя тега xVar_OpcWrite			
Текущая конфигурация : 11.mbp		В Общие настройки			
Объекты Комментарий					
Сепвер     Ключен в работу     Тие					
🗎 🛷 Node1		Регион	COILS		
🗆 🍘 Device1	щие настройки	Адрес	(0x0000) 0	<u>^</u>	
Добавить	• Ter	Тип данных в устройстве	bool		
Переименовать	Группу	Тип данных в сервере	bool		
Дублировать	Подустройство	Тип доступа	ReadWrite		
Manager	вета (мс)	Скрипт			
удалить	при ошибке	Разрешение выполнения скрипта после ч	чтения False		
Вырезать	е соединение после ошиоки через	ре соединение после ошиоки через Разрешение выполнения скрипта перед записью			
Копировать	лизация узла при ошиоке	В Дополнительно			
Групповые операции		Наличие отдельного регистра записи	False		
Экспорт устройства	и фара	Чтение сразу после записи	False	-==	
Импорт подустройства		HDA			
Экспорт тегов	сле запуска	ных доступ	False		
Импорт тегов	а запроса после получения ответа				
BURNMOCTH THEOR	овка байтов в значении				
Теги в работе					
Лополнительные свойс	тва ние скрипта				
	строика запросов				
Мак	ссимальное количество HOLDING регистр				
Ман	ссимальное количество INPUT регистров	ПТиражировать 1	Да Нет		
Hei	использовать команду WRITE_SINGLE_CO			-	
		CTCTED (0.02) T=			
Свойст	тва объекта Таблица тегов				
Режим Конфигурирование	Теги : Разрешено - 60000. Загружено	- О. В текущем элементе - О			

Рисунок 4.12.13 – Добавление тега xVar_OpcWrite

И	мя тега	wVar_OpcWrite				
=	Общие на	астройки				
	Коммента	рий				
	Включен	в работу	True	1		
	Регион		HOLDING_REGISTERS			
	Адрес	(0x0001)	1			
	Тип данны	ых в устройстве	uint16			
	Тип данны	ых в сервере	uint32			
	Тип досту	ла	ReadWrite			
	Использов	зать перестановку байтов устройства	True	e		
	Последни	й тег в групповом запросе	False			
	Пересчет	(A*X + B)	False			
	1					

# Рисунок 4.12.14 – Добавление тега wVar_OpcWrite

и	мя тега	rVar_OpcWrite		
	Общие н	астройки		
	Коммента	рий		
Γ	Включен	в работу		True
	Регион			HOLDING_REGISTERS
	Адрес	(0)	(0002)	2
	Тип данн	ых в устройстве		float
	Тип данн	ых в сервере		float
	Тип доступа			ReadWrite
	Использо	вать перестановку байтов устройс	тва	False
L	Перестан	овка байтов в значении		10325476
	Последни	й тег в групповом запросе		False
	Пересчет	(A*X + B)		False

# Рисунок 4.12.15 – Добавление тега rVar_OpcWrite

и	Імя тега	sVar_OpcWrite		
٦	Общие н	астройки		
	Коммента	рий		
ſ	Включен	в работу		True
I	Регион			HOLDING_REGISTERS
I	Адрес	(	(0x0004)	4
I	Тип данн	ых в устройстве		string
I	Тип данн	ых в сервере		string
I	Количест	во байт для строкового типа		16
I	Тип строн	ки для строкового типа		ascii
l	Тип досту	ла		ReadWrite
1	Использо	вать перестановку байтов устро	йства	True
	Последни	й тег в групповом запросе		False
	Пересчет	(A*X + B)		False

# Рисунок 4.12.16 – Добавление тега sVar_OpcWrite

И	Імя тега	xVar_OpcRead	
•	Общие н	астройки	
	Коммента	рий	
Γ	Включен	в работу	True
I	Регион		DISCRETE_INPUTS
I	Адрес	(	0x0001)0
I	Тип данн	ых в устройстве	bool
I	Тип данн	ых в сервере	bool
L	Тип досту	/na	ReadOnly

Рисунок 4.12.17 – Добавление тега xVar_OpcRead

И	мя тега	wVar_OpcRead			
	Общие на	астройки			
	Коммента	рий			
	Включен	в работу		True	
	Регион			INPUT_REGISTERS	
	Адрес	(	0x0001)	1	
	Тип данны	ых в устройстве		uint16	
	Тип данны	ых в сервере		uint32	
	Тип досту	па		ReadOnly	

Рисунок 4.12.18 – Добавление тега wVar_OpcRead

И	мя тега	rVar_OpcRead					
	∃ Юбщие настройки						
_	Коммента	рий					
	Включен	в работу	True				
	Регион		INPUT_REGISTERS				
	Адрес	(0x000	2) 2				
	Тип данн	ых в устройстве	float				
	Тип данн	ых в сервере	float				
	Тип досту	ла	ReadOnly				
	Использо	вать перестановку байтов устройства	False				
L	Перестан	овка байтов в значении	10325476				
	Последни	й тег в групповом запросе	False				
	Пересчет	(A*X + B)	False				

Рисунок 4.12.19 – Добавление тега rVar_OpcRead

И	мя тега	sVar_OpcRead		
-	Общие н	астройки		
_	Коммента	рий		
	Включен	в работу		True
	Регион			INPUT_REGISTERS
	Адрес		(0x0004)	4
	Тип данн	ых в устройстве		string
	Тип данн	ых в сервере		string
	Количест	во байт для строкового типа		16
	Тип строн	ки для строкового типа		ascii
L	Тип досту	упа		ReadOnly

Рисунок 4.12.20 – Добавление тега sVar_OpcRead

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

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

Device.Application.PLC_PRG				
Выражение	Тип	Значение		
🛷 xVar_OpcRead	BOOL	TRUE		
😻 wVar_OpcRead	WORD	11		
🚸 rVar_OpcRead	REAL	22.33		
🚸 sVar_OpcRead	STRING(16)	'тест'		
WealWord_OpcRead	Real_Word			
Word_OpcRead	String_Word			
xVar_OpcWrite	BOOL	TRUE		
WVar_OpcWrite	WORD	44		
vVar_OpcWrite	REAL	55.66		
🚸 sVar_OpcWrite	STRING(16)	'test'		
WealWord_OpcWrite	Real_Word			
🗉 < uStringWord_OpcWrite	String_Word			

	MasterOPC Universa	l Modbus Server Demo 6	0000 1 hour Build - 4.2.2	8		
Стартовая конфигурация : 653.mbp						
Объекты						
Server	Устройство < <device< th=""><th>21&gt;&gt;</th><th></th><th></th><th></th><th></th></device<>	21>>				
Type: Content of the system of the syst	Имя Node1.Device1.xVa Node1.Device1.wV Node1.Device1.rVa Node1.Device1.sVa Node1.Device1.xVa Node1.Device1.rVa Node1.Device1.rVa Node1.Device1.sVa	Регион         Адрес           DISC         (0x00           INPU         (0x00           INPU         (0x00           Oxnet         (0x00           Oxnet         (0x00           Oxnet         (0x00           HOL         (0x00           HOL         (0x00           HOL         (0x00	Значение True 11 22.330000 тест True 44 55.660000 test	Качество GOOD GOOD GOOD GOOD GOOD GOOD GOOD	Время (UTC) 2019-07-0 2019-07-0 2019-07-0 2019-07-0 2019-07-0 2019-07-0 2019-07-0 2019-07-0	Тип в сер bool uint32 float string bool uint32 float string

Рисунок 4.12.21 – Считывание и запись данных через ОРС-сервер

# 5 Библиотека OwenCommunication

# 5.1 Основная информация

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

- ФБ для настройки интерфейсов;
- <u>ФБ обмена по Modbus;</u>
- ФБ для реализации нестандартных протоколов;
- функции и ФБ преобразования данных.



### ПРИМЕЧАНИЕ

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



# ПРИМЕЧАНИЕ

Библиотека соответствует PLCopen Behavior Model.

# 5.2 Установка библиотеки

Библиотека **OwenCommunication** доступна на сайте компании <u>OBEH</u> в разделе <u>CODESYS</u> <u>V3/Библиотеки и компоненты</u>. Для установки библиотеки в **CODESYS** в меню **Инструменты** следует выбрать пункт **Репозиторий библиотек**, после чего нажать **Установить** и указать путь к файлу библиотеки:



Рисунок 5.2.1 – Установка библиотеки

# 5.3 Добавление библиотеки в проект CODESYS

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



#### Рисунок 5.3.1 – Добавление библиотеки OwenCommunication

После добавления библиотека появится в списке Менеджера библиотек:

🎢 Менеджер библиотек 🗙						
🎦 Добавить библиотеку 🔀 Удалить библиотеку 🕼 Свойства 💿 Детали 🔄 Плейсхолдеры 🎁 Репозиторий библиотек						
Имя Дополнительное имя Действующая версия						
🗑 📲 SLicense = 3SLicense, 3.5.11.50 (3S - Smart Software Solutions GmbH)	_3S_LICENSE	3.5.11.50				
🕼 🕬 BreakpointLogging = Breakpoint Logging Functions, 3.5.5.0 (3S - Smart Software Solutions GmbH)	BPLog	3.5.5.0				
CAA Memory = CAA Memory, 3.5.11.0 (CAA Technical Workgroup)	MEM	3.5.11.0				
□ •• • IoDrvEthernet = IoDrvEthernet, 3.5.11.0 (3S - Smart Software Solutions GmbH)	IoDrvEthernet	3.5.11.0				
i IoStandard = IoStandard, 3.5.10.0 (System)	IoStandard	3.5.10.0				
OwenCommunication = OwenCommunication, 3.5.11.1 (Production association OWEN)	OCL	3.5.11.1				
🖶 🚥 OwenStorage, 3.5.11.30 (Production association OWEN)	OwenStorage	3.5.11.30				

#### Рисунок 5.3.2 – Список библиотек проекта

# i

# ПРИМЕЧАНИЕ

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

# 5.4 Структуры и перечисления

# 5.4.1 Перечисление ERROR

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

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

Таблица 5.4.1 – Описание элементов перечисления ERR(
------------------------------------------------------

## 5.4.2 Перечисление COM_PARITY

Перечисление **COM_PARITY** описывает режим контроля четности COM-порта.

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

Таблица 5.4.2 – Описание элементов перечисления COM_PARITY

#### 5.4.3 Перечисление COM_STOPBIT

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

	Таблица 5.4.3 -	Описание элеме	нтов перечислен	ия СОМ	STOPBIT
--	-----------------	----------------	-----------------	--------	---------

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

#### 5.4.4 Перечисление MB_FC

Перечисление **MB_FC** описывает используемую функцию Modbus.

#### Таблица 5.4.4 – Описание элементов перечисления MB_FC

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

# 5.4.5 Структура MB_REQ_INFO

Структура **MB_REQ_INFO** описывает запрос Modbus, полученный slave-устройством. Структура используется в ФБ <u>MB SerialSlave</u> и <u>MB TcpSlave</u> для предоставления пользователю информации о запросах, поступающих от master-устройства, а также для запрета обработки определенных запросов.

Таблица 5.4.5 – Описание элементов структуры MB REQ INF
---------------------------------------------------------

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

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

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

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

# 5.5 ФБ настройки интерфейсов

# 5.5.1 ΦБ COM_Control

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



# ПРИМЕЧАНИЕ

Не допускается открытие уже используемого СОМ-порта (например, добавленного в проект с помощью <u>стандартных средств конфигурирования</u>).



### Рисунок 5.5.1 – Внешний вид ФБ СОМ_Control на языке СFC

### Таблица 5.5.1 – Описание входов и выходов ФБ СОМ_Control

Название	Тип	Описание		
Входы				
xEnable	BOOL	По переднему фронту происходит открытие СОМ-порта, по заднему – закрытие		
udiComPort	UDINT	Номер СОМ-порта		
udiBaudrate	UDINT	Скорость обмена в бодах. Стандартные возможные значения: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200		
udiByteSize	UDINT(78)	Число бит данных (7 или 8)		
eParity	COM_PARITY	Режим контроля четности		
eStopBit	COM_STOPBIT	Число стоп-бит		
Выходы				
xDone	BOOL	Принимает TRUE на один цикл ПЛК при успешном открытии порта		
xError	BOOL	Принимает значение TRUE в случае возникновения ошибки		
eError	ERROR	Статус работы ФБ (или код ошибки)		
xActive	BOOL	Пока порт открыт, данный выход имеет значение TRUE		
hCom	CAA.HANDLE	Дескриптор СОМ-порта		

# 5.5.2 ΦБ TCP_Client



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

Рисунок 5.5.2 – Внешний вид ФБ TCP_Client на языке CFC

# Таблица 5.5.2 – Описание входов и выходов ФБ TCP_Client

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

# 5.6 ФБ протокола Modbus

# 5.6.1 ΦБ MB_SerialRequest

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

Ответ от slave-устройства ожидается в течение времени **tTimeout**. В случае отсутствия ответа ФБ повторяет запрос. Число переповторов определяется входом **usiRetry** (значение **0** соответствует отсутствию переповторов). Если ни на один из запросов не был получен ответ, то выход **xError** принимает значение **TRUE**, а выход **eError = TIME_OUT**. В случае получения корректного ответа выход **xDone** принимает значение **TRUE**, а выход **eError = NO_ERROR**. В случае получения ответа с кодом ошибки Modbus **xError** принимает значение **TRUE**, а выход **eError = NO_ERROR**. В случае получения ответа с кодом ошибки Modbus **xError** принимает значение **TRUE**, а выход **eError = NO_ERROR**. В случае получения ответа с кодом ошибки Modbus **xError** принимает значение **TRUE**). Для отправки следующего запроса следует создать передний фронт на входе **xExecute**.



# ПРИМЕЧАНИЕ

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

fbModbusSerialMasterRequest					
	OCL.MB_SerialRequest				
_	xExecute xDone	_			
_	tTimeout xBusy	_			
_	usiRetry xError	_			
_	hCom eError	_			
_	xIsAsciiMode				
_	usiSlaveld				
_	eFuncCode				
_	uiDataAddr				
_	uiDataCount				
_	pData				
_	szSize				

Рисунок 5.6.1 – Внешний вид ФБ MB_SerialRequest на языке CFC

Таблица 5.6.1 – Описание входов и выходов ФБ MB_SerialRequest	

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

### 5.6.2 **ΦБ MB_SerialSlave**

Функциональный блок **MB_SerialSlave** используется для работы в режиме **Modbus Serial Slave**. Пока вход **xEnable** имеет значение **TRUE**, блок находится в работе. На вход **hCom** следует подать дескриптор используемого COM-порта, полученный с помощью ФБ <u>COM_Control</u>. Вход **usiSlaveld** определяет адрес slave'a. Под регистры slave'a выделяется область памяти по указателю **pData** размером **szSize** байт.

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

Блок поддерживает все стандартные функции Modbus, приведенные в таблице 2.2.

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

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

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

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

	fbModbusSerialSlave		
	OCL.MB_SerialSlave		
_	xEnable xActive	_	
_	hCom xError	_	
_	usiSlaveld eError	_	
_	pData xNewRequest	_	
_	szSize stRequestInfo	_	
_	pastForbiddenRequests		
_	szForbiddenReguests		
	Параметры		

Рисунок 5.6.2 – Внешний вид ФБ MB_SerialSlave на языке CFC

# Таблица 5.6.2 – Описание входов и выходов ФБ MB_SerialSlave

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

# 5.6.3 ΦБ MB_TcpRequest

Функциональный блок MB_TcpRequest используется для работы в режиме Modbus TCP Master. По переднему фронту на входе xExecute происходит отправка запроса, определяемого параметрами usiUnitld, eFuncCode, uiDataAddr и uiDataCount по протоколу Modbus TCP (если xIsRtuOverTcp = FALSE) или Modbus RTU over TCP (если xIsRtuOverTcp = TRUE) через TCP-соединение, определяемое дескриптором hConnection, полученным от ФБ <u>TCP_Client</u>. Считываемые или записываемые данные размещаются в буфере, расположенном по указателю pData размером szSize байт.

Ответ от slave-устройства ожидается в течение времени **tTimeout**. В случае отсутствия ответа ФБ повторяет запрос. Число переповторов определяется входом **usiRetry** (значение **0** соответствует отсутствию переповторов). Если ни на один из запросов не был получен ответ, то выход **xError** принимает значение **TRUE**, а выход **eError = TIME_OUT**. В случае получения корректного ответа выход **xDone** принимает значение **TRUE**, а выход **eError = NO_ERROR**. В случае получения ответа с кодом ошибки Modbus **xError** принимает значение **TRUE**, а выход **eError = NO_ERROR**. В случае получения ответа с кодом ошибки Modbus **xError** принимает значение **TRUE**, а выход **eError** содержит код ошибки (при этом выход **xDone** не принимает значение **TRUE**). Для отправки следующего запроса следует создать передний фронт на входе **xExecute**.

	fbModbusTcpMasterRequest		
	OCL.MB_TcpRequest		
_	xExecute xDone	_	
_	tTimeout xBusy	_	
_	usiRetry xError	_	
_	hConnection eError	_	
_	xlsRtuOverTcpMode		
_	usiUnitId		
_	eFuncCode		
_	uiDataAddr		
_	uiDataCount		
_	pData		
_	szSize		

Рисунок 5.6.3 – Внешний вид ФБ MB_TcpRequest на языке CFC

Таблица 5.6.3 – Описание входов и выходов ФБ MB_TcpRequest
------------------------------------------------------------

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

#### 5.6.4 ΦБ MB_TcpSlave

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

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

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

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

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

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

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

	fbModbusTcpSlave				
	OCL.MB_TcpSlave				
_	xEnable xActive	_			
-	slpAddr xError	_			
-	uiPort eError	_			
_	usiUnitId xNewRequest	_			
-	pData stRequestInfo	_			
_	szSize byFileAccessMask	_			
-	pastForbiddenRequests usiCurrentCountClients	_			
_	szForbiddenRequests				
_	pasFilePath				
	Параметры				

Рисунок 5.6.4 – Внешний вид ФБ MB_TcpSlave на языке CFC

Таблица 5.6.4 – Описание входов и выходов ФБ MB_TcpSlave

Название Тип		Описание		
Входы				
vEnable	BOOL	По переднему фронту происходит включение slave'a, по		
	BOOL	заднему – отключение		
sInAddr	STRING	IP-адрес используемого сетевого интерфейса в формате		
		<u>IPv4</u> ('xxx.xxx.xxx').		
uiPort	UINT	Используемый порт		
usiUnitId	USINT	Адрес slave-устройства		
pData	CAA.PVOID	Указатель на буфер данных slave'a		
szSize	CAA.SIZE	Размер буфера в байтах		
pastForbiddenRequest	CAA.PVOID	Указатель на массив структур запрещенных запросов		
szForbiddenRequests	CAA.SIZE	Размер массива структур запрещенных запросов в байтах		
n a c File Dath	POINTER TO	Указатель на массив путей к файлам архива (для		
pasrilePath	OF STRING	функции <b>20</b> )		
		Выходы		
xActive	BOOL	<b>TRUE</b> – ФБ находится в работе		
	DOOL	Принимает значение TRUE в случае возникновения		
XError	BOOL	ошибки		
eError	ERROR	Статус работы ФБ (или код ошибки)		
xNowPoquest	DOOL	Принимает TRUE на один цикл контроллера при		
xnewRequest	BOOL	получении запроса от master-устройства		
stRequestInfo	MB REO INFO	Информация о полученном запросе (актуальна, пока		
		xNewRequest = TRUE)		
byFileAccessMask	BYTE	Битовая маска открытых файлов (для функции <b>20</b> )		
		Число клиентов, подключенных к slave'y. Максимальное		
		число клиентов определяется глобальным параметром		
usiCurrentCountClients	USINT	библиотеки g_c_usiMaxCountClients (вкладка Settings		
		в дереве библиотеки в Менеджере библиотек).		
		Максимальное число клиентов – 16		
		Параметры		
c_xReverseByteOrder	BOOL	IRUE – изменить порядок оаит в оуфере данных slave a		
		на противоположный		
		пла функции 20 на противоположный. Пла работы с		
c_xReverseByteOrderFiles	BOOL	MasterOPC Universal Modbus Server спелиет		
		установить значение <b>TRUE</b>		
		Начальный адрес slave'а. При получении запроса, в		
	UINT	котором присутствует регистр с адресом <		
c_uiStartAddr		с uiStartAddr, slave отправит master-устройству ошибку		
		ILLEGAL_DATA_ADDRESS		
		Размер записи для функции 20. По спецификации		
c usi∆mountRvtes		Modbus этот параметр должен иметь значение 2. Для		
	03111	работы с MasterOPC Universal Modbus Server следует		
		установить значение 10		

# 5.7 ФБ нестандартных протоколов

# 5.7.1 ΦБ UNM_SerialRequest

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

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

В случае получения корректного ответа выход **xDone** принимает значение **TRUE**, выход **eError** = **NO_ERROR**, а на выходе **uiResponseSize** отображается размер ответа в байтах. Полученные данные помещается в буфер, расположенный по указателю **pResponse** и имеющий размер **szResponse** байт.

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

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

#### ПРИМЕЧАНИЕ

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



Рисунок 5.7.1 – Внешний вид ФБ UNM_SerialRequest на языке CFC

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

# Таблица 5.7.1 – Описание входов и выходов ФБ UNM_SerialRequest

# 5.7.2 ΦБ UNM_TcpRequest

Функциональный блок UNM_TcpRequest используется для реализации нестандартного протокола поверх протокола TCP. По переднему фронту на входе xExecute происходит отправка содержимого буфера запроса, расположенного по указателю pRequest, размером szRequest байт через соединение, определяемое дескриптором hConnection, полученным от ФБ <u>TCP Client</u>. Ответ от slave-устройства ожидается в течение времени tTimeout. При получении ответа происходит его проверка на основании значений входов szExpectedSize и wStopChar:

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

В случае получения корректного ответа выход **xDone** принимает значение **TRUE**, выход **eError** = **NO_ERROR**, а на выходе **uiResponseSize** отображается размер ответа в байтах. Полученные данные помещается в буфер, расположенный по указателю **pResponse** и имеющий размер **szResponse** байт.

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

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

#### ПРИМЕЧАНИЕ

i

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



Рисунок 5.7.2 – Внешний вид ФБ UNM_TcpRequest на языке CFC

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

Таблица 5.7.2 – Описание входов и выходов ФБ UNM_TcpRequest

# 5.7.3 ΦБ UNM_UdpRequest

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

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

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

В случае получения корректного ответа выход **xDone** принимает значение **TRUE**, выход **eError** = **NO_ERROR**, а на выходе **uiResponseSize** отображается размер ответа в байтах. Полученные данные помещается в буфер, расположенный по указателю **pResponse** и имеющий размер **szResponse** байт.

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

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

#### ПРИМЕЧАНИЕ

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

fbUnmUdpRequest				
	OCL.UNM_UdpRe	equest 🗳		
_	xExecute	xDone	-	
_	tTimeout	xBusy	-	
_	usiRetry	xError	$\vdash$	
_	uiLocalPort	eError	$\vdash$	
_	sServerIpAddr u	uiResponseSize	-	
_	uiServerPort			
_	pRequest			
_	szRequest			
_	pResponse			
_	szResponse			
_	szExpectedSize			
-	wStopChar			

Рисунок 5.7.3 – Внешний вид ФБ UNM_UdpRequest на языке CFC

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

# Таблица 5.7.3 – Описание входов и выходов ФБ UNM_UdpRequest

# 5.8 Функции и ФБ преобразования данных

# 5.8.1 Φ6 DWORD_TO_WORD2

Функциональный блок **DWORD_TO_WORD2** используется для преобразования переменной типа **DWORD** в две переменные типа **WORD**.

	fbDwordToWord2	2	
	OCL.DWORD_TO_V	VORD2	
_	dwInput	wOutput1	_
	xSwapBytes	wOutput2	_

### Рисунок 5.8.1 – Внешний вид ФБ DWORD_TO_WORD2 на языке CFC

### Таблица 5.8.1 – Описание входов и выходов ФБ DWORD_TO_WORD2

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

				fbDwordTo	Word2	1
		10	#41000004	OCL.DWORD	TO_WORD2	1640041
16#A1B2C3D	)4	10	#AIDZU3D4	dwInput	wOutput1	10#DZAT
TRUE				xSwapBytes	wOutput2	16#D4C3

# Рисунок 5.8.2 – Пример использования ФБ DWORD_TO_WORD2 на языке CFC

# 5.8.2 Φ**5** REAL_TO_WORD2

Функциональный блок **REAL_TO_WORD2** используется для преобразования переменной типа **REAL** в две переменные типа **WORD**.

	fbRealToWo	ord2	
	OCL.REAL_TO	_WORD2	
_	rInput	wOutput1	_
_	xSwapBytes	wOutput2	_

### Рисунок 5.8.3 – Внешний вид ФБ REAL_TO_WORD2 на языке CFC

Таблица 5.8.2 – Описание входов и выходов ФБ REA	L TO	WORD2

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

		fbRealTo	Word2
	11.0	OCL.REAL_	TO_WORD2
11.22	11.2	rInput	wOutput1
FALSE		xSwapBytes	wOutput2

# Рисунок 5.8.4 – Пример использования ФБ REAL_TO_WORD2 на языке CFC (см. <u>онлайн-</u> конвертер для проверки)

# 5.8.3 Функция WORD2_TO_DWORD

Функция **WORD2_TO_DWORD** используется для преобразования двух переменных типа **WORD** в переменную типа **DWORD**.



Рисунок 5.8.5 – Внешний вид функции WORD2_TO_DWORD на языке CFC

|--|

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

		OCL.WORD2_TO_DWORD		
16#A1B2	 wInput1	WORD2_TO_DWORD	 dwVar	16#BZA1D4C3
16#C3D4	 wInput2			
TRUE	xSwapB	ytes		

Рисунок 5.8.6 – Пример использования функции WORD2_TO_DWORD на языке CFC

# 5.8.4 Функция WORD2_TO_REAL

Функция **WORD2_TO_REAL** используется для преобразования двух переменных типа **WORD** в переменную типа **REAL**.

		0	)
	OCL.WORD2	_TO_REAL	1
_	wInput1	WORD2_TO_REAL	⊢
	wInput2		
_	xSwapBytes		

# Рисунок 5.8.7 – Внешний вид функции WORD2_TO_REAL на языке CFC

#### Таблица 5.8.4 – Описание входов и выходов функции WORD2_TO_REAL

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

		OCL.WORD2_TO_REAL	
16#4133	wInput1	WORD2_TO_REAL	rVar
16#851F	 wInput2		11.22
FALSE	xSwapBy	tes	

Рисунок 5.8.8 – Пример использования функции WORD2_TO_REAL на языке CFC (см. <u>онлайн-</u> конвертер для проверки)

# 5.8.5 Функция SWAP_DATA

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

	OCL.SWAP_DATA	
_	pSource SWAP_DATA	
_	pDestination	
_	szSize	
_	xSwapBytes	
_	xSwapWord	
_	xReverseByteOrder	

Рисунок 5.8.9 – Внешний вид функции SWAP_DATA на языке CFC

Название Тип Описание						
	Входы					
pSource	CAA.PVOID	Указатель на буфер исходных данных				
	CAA.PVOID	Указатель на буфер, в который будут скопированы данные.				
pDestination		Может совпадать с <b>pSource</b> – тогда после изменения данные				
		будут помещены в тот же буфер				
679i70	CAA.SIZE	Размер копируемых данных в байтах (должен не превышать				
52.512E		размеры буферов)				
vSwapBytes	BOOL	TRUE – выполнить перестановку байт				
xSwapDytes		(A1 B2 C3 D4> B2 A1 D4 C3)				
xSwanWord	BOOL	<b>TRUE</b> – выполнить перестановку регистров				
xSwapvvoru		(A1 B2 C3 D4> C3 D4 A1 B2)				
	BOOL	TRUE – изменить порядок байт на противоположный				
vReverseRvteOrder		(A1 B2 C3 D4> D4 C3 B2 A1).				
xitevelsebyteoldel		Если данный вход имеет значение <b>TRUE</b> , то входы				
		xSwapBytes и xSwapWord не обрабатываются				
Выходы						
SWAP_DATA	BOOL	TRUE – операция выполнена				

### Таблица 5.8.5 – Описание входов и выходов функции SWAP_DATA

# 5.9 Примеры

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

В качестве примера будет рассмотрена настройка обмена с модулями <u>Mx110</u> (MB110-8A, MB110-16Д, MУ110-16Р) с использованием библиотеки **OwenCommunication**. В примере используется библиотека версии **3.5.11.1**.

**Реализуемый алгоритм**: если значение первого аналогового входа модуля **MB110-8A** превышает **30** и при этом первый дискретный вход модуля **MB110-16Д** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **MУ110-16P** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).

Структурная схема примера приведена на рисунке ниже:



### Рисунок 5.9.1 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.x. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания:

Example OwenCommunicationModbusRtuMaster 3511v1.projectarchive

Сетевые параметры модулей приведены в таблице ниже:

Таблица 5.9.1 – Сетевые	параметры модулей Мх110
-------------------------	-------------------------

Параметр	MB110-8A	МВ110-16Д	МУ110-16Р
СОМ-порт контроллера, к которому	COM1	COM2	
подключен модуль	CONT		
ID COM-порта	1		2
Адрес модуля	1	1	17
Скорость обмена		115200	
Количество бит данных		8	
Контроль четности	Отсутствует		
Количество стоп-бит	1		

Для настройки обмена следует:

**1**. Настроить модули **Mx110** с помощью программы **Конфигуратор Mx110** в соответствии с таблицей 5.9.1. Подключить модули к СОМ-портам контроллера в соответствии с <u>рисунком 5.9.1</u>.

- 2. Установить в CODESYS библиотеку OwenCommunication (см. п. 5.2).
- 3. Создать новый проект CODESYS с программой на языке ST или CFC:

Стандартный проект						
	Вы собираетесь создать новый стандартный проект. При этом будут созданы следующие объекты: - Одно программируемое устройство, как задано ниже - Программа PLC_PRG на языке, выбранном ниже - Циклическая задача, вызывающая PLC_PRG - Ссылка на новейшую установленную версию библиотеки Standard library.					
	Устройство: PLC_PRG на:	SPK1xx[M01] (Production association OWEN) Непрерывные функциональные схемы (CFC) ОК От	• • 1eHa			

Рисунок 5.9.2 – Создание проекта CODESYS



## ПРИМЕЧАНИЕ

Проект примера содержит программы на языках СFC и ST. По умолчанию используется программа на CFC (**PLC_PRG_CFC**). Для работы с программой на ST следует в конфигурации задач для задачи **MainTask** удалить вызов **PLC_PRG_CFC** и добавить вызов **PLC_PRG_ST**.

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

# i

#### ПРИМЕЧАНИЕ

Библиотека Util используется только в программе на языке CFC.

5. Объявить в программе следующие переменные:

D PLC_PRG_CFC X					
1	PROGRAM PLC_PRG_CFC				
2	VAR				
3	fbComControl1:	OCL.COM_Control;	// ФБ управления портом COM1		
4	fbComControl2:	OCL.COM_Control;	// ФБ управления портом COM2		
5	fbMV110_8A_AI1:	OCL.MB_SerialRequest;	// ФБ опроса модуля MB110-8A		
6	fbMV110_16D_DI:	OCL.MB_SerialRequest;	// ФБ опроса модуля МВ110-16Д		
7	fbMU110_16R_DO:	OCL.MB_SerialRequest;	// ФБ опроса модуля МУ110-16Р		
8					
9	rAI1:	REAL;	// значение 1-го входа модуля МВ110-8А		
10	wDiMask:	WORD;	// битовая маска входов модуля МВ110-16Д		
11	wDoMask:	WORD;	// битовая маска выходов модуля MV110-16P		
12	xDi0:	BOOL;	// значение 1-го входа модуля МВ110-16Д		
13	xDo0:	BOOL;	// значение 1-го выхода модуля MV110-16P		
14					
15	awAI1:	ARRAY [01] OF WORD;	// регистры, считанные с модуля MB110-8A		
16					
17	fbUnpackWord:	UTIL.WORD_AS_BIT;	// ФБ распаковки битовой маски		
18	fbPackWord:	UTIL.BIT_AS_WORD;	// ФБ упаковки битовой маски		
19					
20	iStateCom1:	INT;	// шаг опроса по порту СОМ1		
21	iStateCom2:	INT;	// шаг опроса по порту СОМ2		
22	END VAR				

Рисунок 5.9.3 – Объявление переменных в программе

#### ПРИМЕЧАНИЕ

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

6. Нажать **ПКМ** на программу, выбрать команду **Добавление объекта – Действие** и добавить действия с названиями **СОМ1** и **СОМ2** (язык реализации действий совпадает с языком программы). В рамках примера действия используются для повышения читабельности кода.



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

Рисунок 5.9.4 – Код действия СОМ1

При первом вызове действия происходит открытие COM-порта с заданными настройками с помощью экземпляра ФБ <u>COM Control</u> (блок 0). После успешного открытия порта на выходе **xDone** генерируется единичный импульс, что приводит к вызову экземпляра ФБ <u>MB SerialRequest</u> (блок 2), который производит опрос первого аналогового входа модуля **MB110-8A** с адресом 1 (usiSlaveld). Требуемый код функции (eFunc), начальный адрес регистра (uiDataAddr) и их количество (uiDataCount) приведены в РЭ на модуль. Полученные данные помещаются в переменную awAl1 типа ARRAY [0..1] OF WORD. С помощью функции <u>WORD2 TO REAL</u> (блоки 4–5) этот массив преобразуется в переменную типа rAl1 типа REAL.

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

Если порт закрыт или находится в состоянии ошибки (выход **xActive** у экземпляра ФБ <u>COM CONTROL</u> имеет значение **FALSE**), то опрос прекращается.



#### ПРИМЕЧАНИЕ

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



Рисунок 5.9.5 – Код действия СОМ2

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

После завершения работы ФБ <u>MB SerialRequest</u> один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**. В любой из этих ситуаций начинается работа следующего блока <u>MB SerialRequest</u> (блоки 5–6), который производит запись переменной **wDoMask** в качестве битовой маски дискретных выходов модуля **МУ110-16Р** с адресом **17**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Экземпляр блока **BIT_AS_WORD** (**блоки 0–1**) позволяет упаковать отдельные переменные типа **BOOL** в битовую маску типа **WORD**.

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

Если порт закрыт или находится в состоянии ошибки (выход **xActive** у ФБ <u>COM CONTROL</u> имеет значение **FALSE**), то опрос прекращается.



#### ПРИМЕЧАНИЕ

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

В программе **PLC_PRG_CFC** производится вызов действий **COM1** и **COM2**, а также выполнение алгоритма, описанного в условии примера.



Рисунок 5.9.6 – Код программы PLC_PRG_CFC

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

```
CASE iStateCom1 OF
 0: // открытие СОМ-порта СОМ1
fbComControl1
 (
 xEnable
 := TRUE,
 usiComPort := 1,
 udiBaudrate := 115200.
 10
 udiByteSize := 8,
 11
 eParity := OCL.COM_PARITY.NONE,
 := OCL.COM_STOPBIT.ONE
 12
 eStopBit
 13
);
 14
 15
 IF fbComControl1.xDone THEN
 16
 iStateCom1 := 1:
 17
 END IF
 18
 19
 20
 1: // опрос MB110-8A
 21
 22
 fbMV110_8A_AI1
 23
 xExecute
 (
 := fbComControl1.xActive,
 24
 25
 := T#200MS,
 usiRetry
 26
 := 3,
 27
 hCom
 := fbComControl1.hCom,
 := FALSE ,
 28
 xIsAsciiMode
 23
 usiSlaveId
 := 1,
 30
 := OCL.MB_FC.READ_HOLDING_REGISTERS,
 eFuncCode
 31
 uiDataAddr
 := 4.
 uiDataCount
 32
 := 2,
 pData
 33
 := ADR(awAI1),
 := SIZEOF(awAI1)
 34
 szSize
 35
);
 36
 37
 IF fbMV110_8A_AI1.xDone OR fbMV110_8A_AI1.xError THEN
 38
 // после выполнения блока его надо сбросить
 39
 fbMV110_8A_AI1(xExecute := FALSE);
 40
 41
 rAI1 := OCL.WORD2 TO REAL(awAI1[0], awAI1[1], FALSE);
 42
 43
 iStateCom1 := 2;
 44
 END IF
 45
 46
 47
 2: // здесь можно добавить опрос следующего устройства
 48
 // после опроса последнего опроса возрашаемся к опросу первого
 49
 iStateCom1 := 1;
 50
 51
 END CASE
```

Рисунок 5.9.7 - Код действия СОМ1

При первом вызове действия в **шаге 0** происходит открытие СОМ-порта с заданными настройками с помощью экземпляра ФБ <u>COM Control</u>. После успешного открытия порта на выходе **xDone** генерируется единичный импульс, что приводит к переходу на **шаг 1**.

На шаге 1 выполняется вызов экземпляра ФБ <u>MB SerialRequest</u>, который производит опрос первого аналогового входа модуля **MB110-8A** с адресом 1 (usiSlaveld). Требуемый код функции (eFunc), начальный адрес регистра (uiDataAddr) и их количество (uiDataCount) приведены в РЭ на модуль. Полученные данные помещаются в переменную awAl1 типа ARRAY [0..1] OF WORD. С помощью функции <u>WORD2 TO REAL</u> этот массив преобразуется в переменную типа rAl1 типа REAL.
После завершения работы экземпляра ФБ <u>MB SerialRequest</u> один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**. В любой из этих ситуаций происходит вызов экземпляра ФБ <u>MB SerialRequest</u> со значением **FALSE** на входе **xExecute** (сброс блока) и переход на **шаг 2**.

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

Если порт закрыт или находится в состоянии ошибки (выход **xActive** у экземпляра ФБ <u>COM Control</u> имеет значение **FALSE**), то опрос прекращается.

```
CASE iStateCom2 OF
 1
-
 2
 3
 0: // открытие СОМ-порта СОМ2
-
 4
 5
 fbComControl2
-
 6
 (
 xEnable
 := TRUE,
 usiComPort := 2,
 8
 9
 udiBaudrate := 115200,
 10
 udiByteSize := 8,
 := OCL.COM_PARITY.NONE,
:= OCL.COM_STOPBIT.ONE
 11
 eParity
 12
 eStopBit
 13
);
 14
 15
 IF fbComControl2.xDone THEN
 16
 iStateCom2 := 1;
 17
 END_IF
 18
 19
 1: // опрос модуля МВ110-16Д
 20
 21
 fbMV110_16D_DI
 22
=
 (
 23
 := fbComControl2.xActive,
 xExecute
 24
 tTimeout
 := T#200MS,
 25
 usiRetry
 := 3,
 26
 hCom
 := fbComControl2.hCom,
 27
 xIsAsciiMode
 := FALSE ,
 28
 := 1,
 usiSlaveId
 29
 eFuncCode
 := OCL.MB_FC.READ_HOLDING_REGISTERS,
 30
 uiDataAddr
 := 51,
 31
 uiDataCount := 1,
 pData
 := ADR(wDiMask),
 32
 33
 szSize
 := SIZEOF(wDiMask)
 34
);
 35
 36
 IF fbMV110_16D_DI.xDone OR fbMV110_16D_DI.xError THEN
37
 // после выполнения блока его надо сбросить
 38
 fbMV110_16D_DI(xExecute := FALSE);
 39
 40
 xDi0 := wDiMask.0;
 41
 42
 iStateCom2 := 2;
 43
 END IF
 44
 45
 2: // опрос модуля МУ110-16Р
 46
 47
 wDoMask.0 := xDo0;
 48
 49
 50
 fbMU110_16R_DO
 51
 (
 := fbComControl2.xActive,
 52
 xExecute
 53
 tTimeout
 := T#200MS,
 54
 usiRetry
 := 17,
 55
 := fbComControl2.hCom,
 hCom
 xIsAsciiMode
 := FALSE ,
 56
 57
 usiSlaveId
 := 60,
 58
 eFuncCode
 := OCL.MB_FC.WRITE_MULTIPLE_REGISTERS,
 59
 uiDataAddr
 := 50,
 60
 uiDataCount
 := 1,
 61
 := ADR(wDoMask),
 pData
 := SIZEOF (wDoMask)
 62
 szSize
 63
);
 64
 65
 IF fbMU110_16R_DO.xDone OR fbMU110_16R_DO.xError THEN
66
 // после выполнения блока его надо сбросить
 67
 fbMU110_16R_DO(xExecute := FALSE);
 68
 69
 // возвращаемся к опросу первого модуля
 70
 iStateCom2 := 1;
 71
 END IF
 72
```

Рисунок 5.9.8 – Код действия СОМ2

При первом вызове действия в **шаге 0** происходит открытие СОМ-порта с заданными настройками с помощью экземпляра ФБ <u>COM Control</u>. После успешного открытия порта на выходе **xDone** генерируется единичный импульс, что приводит к переходу на **шаг 1**.

На шаге 1 выполняется вызов экземпляра ФБ <u>MB_SerialRequest</u>, который производит опрос битовой маски дискретных входов модуля **MB110-16Д** с адресом 1 (usiSlaveld). Требуемый код функции (eFunc), начальный адрес регистра (uiDataAddr) и их количество (uiDataCount) приведены в РЭ на модуль. Полученные данные помещаются в переменную wDiMask типа WORD, из которой происходит копирование данных в нужные переменные типа BOOL с помощью побитового доступа (переменная.номер_бита).

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

На **шаге 2** выполняется вызов экземпляра ФБ <u>MB_SerialRequest</u>, который производит запись переменной **wDoMask** в качестве битовой маски дискретных выходов модуля **MУ110-16P** с адресом **17**. Требуемый код функции (**eFunc**), начальный адрес регистра (**uiDataAddr**) и их количество (**uiDataCount**) приведены в РЭ на модуль. Битовая маска может формироваться из отдельных переменных типа **BOOL** с помощью побитового доступа.

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

Если порт закрыт или находится в состоянии ошибки (выход **xActive** у экземпляра ФБ <u>COM Control</u> имеет значение **FALSE**), то опрос прекращается.

В программе **PLC_PRG_ST** производится вызов действий **COM1** и **COM2**, а также выполнение алгоритма, описанного в условии примера.



Рисунок 5.9.9 – Код программы PLC_PRG_ST

9. Загрузить проект в контроллер и запустить его.

В переменной rAl1 будет отображаться текущее значение первого аналогового входа модуля MB110-8A. В переменной xDi0 будет отображаться текущее значение первого дискретного входа модуля MB110-16Д.

Если значение **rAl1**превысит **30** и при этом значение **xDi0** будет равно **TRUE**, то в переменную **xDo0** будет записано значение **TRUE**, что приведет к замыканию первого дискретного выхода модуля **MY110-16P**. Если одно из условий перестанет выполняться, то выход будет разомкнут.



Рисунок 5.9.10 – Выполнение программы в режиме Online

# 5.9.2 CIIK1xx [M01] (Modbus RTU Slave) + MasterOPC Universal Modbus Server

В качестве примера будет рассмотрена настройка обмена с OPC-сервером <u>Insat MasterOPC Universal</u> <u>Modbus Server</u>, который будет использоваться в режиме **Modbus RTU Master**, с помощью библиотеки **OwenCommunication**. В примере используется библиотека версии **3.5.11.1**.

Структурная схема примера приведена на рисунке ниже:



Рисунок 5.9.11 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.x. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания: Example OwenCommunicationModbusRtuSlave 3511v1.zip

#### Сетевые параметры модулей приведены в таблице ниже:

Таблица 5.9.2 – Сетевые параметры устройств

Параметр	СПК1хх [М01] ПК с ОРС-сервером		
СОМ-порт контроллера, к которому	COM1	COM4	
подключен модуль	COMIT	00114	
ID COM-порта	1	4	
Режим работы	slave	master	
Slave ID	1	-	
Скорость обмена		115200	
Количество бит данных	8		
Контроль четности	Отсутствует		
Количество стоп-бит		1	

Переменные примера описаны в таблице ниже:

#### Таблица 5.9.3 – Список переменных примера

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



#### ПРИМЕЧАНИЕ

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

Для настройки обмена следует:

- 1. Подключить контроллер к ПК (например, с помощью конвертера <u>OBEH AC4</u>).
- 2. Создать новый проект CODESYS с программой на языке ST или CFC:

Стандартнь	ій проект	and the second	×			
	Вы собираетесь создать новый стандартный проект. При этом будут созданы следующие объекты:					
	<ul> <li>Одно программируемое устройство, как задано ниже</li> <li>Программа PLC_PRG на языке, выбранном ниже</li> <li>Циклическая задача, вызывающая PLC_PRG</li> <li>Ссылка на новейшую установленную версию библиотеки Standard library.</li> </ul>					
	Устройство: SPK1xx[M01] (Production association OWEN)					
	PLC_PRG на: Непрерывные функциональные схемы (CFC)					
		ОК От	1ена			

Рисунок 5.9.12 – Создание проекта CODESYS

# ПРИМЕЧАНИЕ

Проект примера содержит программы на языках СFC и ST. По умолчанию используется программа на CFC (**PLC_PRG_CFC**). Для работы с программой на ST следует в конфигурации задач для задачи **MainTask** удалить вызов **PLC_PRG_CFC** и добавить вызов **PLC_PRG_ST**.

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



i

### ПРИМЕЧАНИЕ

Библиотека Util используется только в программе на языке CFC.

4. Объявить в программе следующие переменные:

1	PROGRAM PLC_PRG_CFC		
2	VAR		
3	fbComControl1:	OCL.COM_Control;	// ФБ управления портом COM1
4	fbUnpackWord:	UTIL.WORD AS BIT;	// ФБ распаковки битовой маски
6	fbPackWord:	UTIL.BIT AS WORD;	// ФБ упаковки битовой маски
7	fbRealToWord2:	OCL.REAL TO WORD2;	// ФБ преобразования REAL в две переменные типа WORD
8			,,
9			
10	fbModbusSerialSlave:	OCL.MB SerialSlave:	// ФБ для реализиации Modbus Slave
11			
12	awSlaveData:	ARRAY [015] OF WORD:	// буфер данных Modbus Slave
13			//
14	(* значения, полученные	от OPC *)	
15	xVar_Opc:	BOOL;	
16	wVar_Opc:	WORD;	
17	rVar_Opc:	REAL;	
18	sVar Opc:	STRING(15);	
19	_		
20	(* значения для передачи	ив <i>ОРС *)</i>	
21	xVar_Plc:	BOOL;	
22	wVar_Plc:	WORD;	
23	rVar_Plc:	REAL;	
24	sVar_Plc:	STRING(15);	
25			
26	xWrite:	BOOL;	// команда записи данных из программы в регистры Modbus Slave
27	fbWriteEdge:	R_TRIG;	// тригтер для однократной записи
28	END VAR		

Рисунок 5.9.13 – Объявление переменных программы



#### ПРИМЕЧАНИЕ

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





Рисунок 5.9.14 – Код программы на языке СFC

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

В блоках 2–7 происходит копирование данных из регистров slave'а в переменные программы. Для выделения переменных типа **BOOL** из переменной типа **WORD** используется экземпляр ФБ **WORD_AS_BIT** из библиотеки Util. Для преобразования двух переменных типа **WORD** в переменную типа **REAL** используется функция <u>WORD2_TO_REAL</u>. Для преобразования набора переменных типа **WORD** в переменную типа **STRING** используется функция <u>SWAP_DATA</u>. Для обеспечения порядка байт, принятого в OPC-сервере, в процессе копирования происходит перестановка байт в регистрах (xSwapBytes=TRUE).



#### ПРИМЕЧАНИЕ

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

По переднему фронту переменной **xWrite** (блок 9) происходит запись переменных программы в регистры slave'a. Для записи переменной типа **BOOL** используется экземпляр ФБ **BIT_AS_WORD** из библиотеки **Util**. Для записи переменной типа **REAL** используется экземпляр ФБ <u>REAL TO WORD2</u>. Для записи переменной типа **STRING** используется функция <u>SWAP_DATA</u>.

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

```
// чтобы запустить пример на ST на контроллере требуется:
 // 1. Удалить из задачи MainTask вызов программа PLC_PRG_CFC
 // 2. Привязать к задаче MainTask вызов программы PLC_PRG_ST
 fbComControl1
 := TRUE,
 xEnable
 usiComPort := 1,
 udiBaudrate := 115200,
12
 udiByteSize := 8,
 eParity
13
 := OCL.COM_PARITY.NONE,
 eStopBit := OCL.COM_STOPBIT.ONE
14
15
);
16
17
 fbModbusSerialSlave
18
 (
19
 xEnable
 :=fbComControl1.xActive,
20
 := fbComControl1.hCom,
 hCom
21
 usiSlaveId := 1,
 pData := ADR(awSlaveData),
szSize := SIZEOF(awSlaveData)
22
23
24
):
25
26
 // данные, полученные от ОРС
27
 xVar Opc := awSlaveData[0].0;
28
 wVar_Opc := awSlaveData[1];
29
 rVar_Opc := OCL.WORD2_TO_REAL(awSlaveData[2], awSlaveData[3], FALSE);
30
 OCL.SWAP_DATA(ADR(awSlaveData[4]), ADR(sVar_Opc), SIZEOF(sVar_Opc), TRUE, FALSE, FALSE);
31
32
33
 // по команде записываем переменные из программы в регистры Modbus Slave
34
 fbWriteEdge(CLK := xWrite);
35
36
 IF fbWriteEdge.0 THEN
37
38
 awSlaveData[0].0 := xVar_Plc;
39
 awSlaveData[1] := wVar Plc;
40
 fbRealToWord2(rInput := rVar_Plc, wOutput1 => awSlaveData[2], wOutput2 => awSlaveData[3]);
41
 OCL.SWAP DATA (ADR (sVar Plc), ADR (awSlaveData[4]), SIZEOF (sVar Plc), TRUE, FALSE, FALSE);
42
43
 END IF
```

#### Рисунок 5.9.15 – Код программы на языке ST

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

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



#### ПРИМЕЧАНИЕ

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

По переднему фронту переменной **xWrite** происходит запись переменных программы в регистры slave'a. Для записи переменной типа **REAL** используется экземпляр ФБ <u>REAL_TO_WORD2</u>. Для записи переменной типа **STRING** используется функция <u>SWAP_DATA</u>.

#### 7. Установить и запустить MasterOPC Universal Modbus Server.

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

	MasterOPC Universal Modbus S	ierver Demo 60000 1 hour Build - 4.2.28	_ <b>_</b> ×
Конфигурация Общие настройк	ки Помощь		
🗐 Создать 💾 Сохранить как 🤅	🗓 Добавить узел 👘 🖉	обавить тег 📄 Переименовать 🗙 Удалить	
🖗 Открыть 🛛 🛷 Сделать стартовым 🛛 🕅	🕅 Добавить устройство 🔺 Г	ереместить вверх 🔅 Копировать 🖛 Отменить	
📔 Сохранить 📋 Импорт из версии 2.0	🍕 Добавить группу 🛛 📲	Редактирование коммуникационного узла	
Файл конфигурации	Сервер		
Текущая конфигурация : 653.mbp		Имя узла Node1	
Объекты		🗆 Юбщие настройки	
		Комментарий	
Добавить Ког	ммуникационный узел	Включен в работу	True
Групповые операции		Тип узла	COM
Импорт узда		Настройки СОМ	
in the state		Порт	4
узлы в работе	<b>A</b>	Принко	0
		Контроль четности	Не используется
		Стоп биты	1
		Межсимвольный таймаут (мс)	0
		Использовать режим ASCII	False
		Использовать модем	False
		🗏 Скрипт	
		Выполнение скрипта	False
		Дополнительные настройки	
		Slave подключение	False
		Принудительный разрыв соединения в каждом цикле	False
		использовать резервные каналы	
		Пиражировать 1	нет
Режим Конфигурирование	Теги : Разрешено - 600	00. Загружено - 8	

Рисунок 5.9.16 – Добавление коммуникационного узла

**9**. Нажать **ПКМ** на коммуникационный узел и добавить устройство с настройками по умолчанию (Slave ID = **1** в соответствии с <u>таблицей 5.9.2</u>).

	MasterOPC Universal Mod	bus Server Demo 60000 1 hour Bu	uild - 4.2.28	- 0 X	
Конфигурация Общие настро	іки Помощь				
🗐 Создать 💾 Сохранить как	🗓 Добавить узел	Добавить узел 🚽 Добавить тег 🦳 Переименовать 🗙 Удалить			
🖗 Открыть 🛛 🛷 Сделать стартовым	🕅 Добавить устройство	👚 Переместить вверх	🗐 Копировать 🔊 Отменить		
П Сохранить П Импорт из версии 2.0	🗐 Лобавить группу				
Файа конфистрации		Редактирование устройства			
		Имя устройства Пеуіс	e1		
Объекты					
		• Общие настройки		<b>^</b>	
Server		Комментарии		True	
Добавить	Устройство	Тип устройства		MODBUS	
Переименовать		Алрес	(0x01)	1	
Лублировать		Время ответа (мс)	(0.01)	1000	
A) on inposents		Повторы при ошибке		3	
Удалить		Повторное соединение после ошибки через (с)		10	
Вырезать		Реинициализация узла	а при ошибке	False	
Копировать		Период опроса		1000	
Групповые операции		Размерность периода	ms		
Экспорт узла		Начальная фаза		0	
Импорт устройства		Размерность фазы		ms	
Устройства в работе		Старт после запуска		True	
	-	Задержка запроса пос	сле получения ответа (мс)	4 Dunon	
		Перестановка байтов	взначении	BDI30B	
		Выполнение скрипта		False	
		Настройка запросов			
		Максимальное количе	ство HOLDING регистров в запросе чтения	125	
		Максимальное количе	ство INPUT регистров в запросе чтения	125	
		Не использовать кома	нду WRITE_SINGLE_COIL (0x05)	True _	
				Нет	
Режим Конфигурирование	Теги : Разрешено -	60000. Загружено - 8			

Рисунок 5.7.17 – Добавление устройства

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

Создать 🗖	Сохранить как	дооавить узел ча дооавить тег	Переименовать 🗶 удалить		
открыть 🧳	сделать стартовым	дооавить устроиство тпереместить в	Редактирование тега		
🛚 Сохранить 🔲	Импорт из версии 2.0	Добавить группу 🐳 Переместить в			
Файл к	онфигурации	Сервер	Имя тега xVar		
екущая конфигу	ирация : 11.mbp		🗉 Общие настройки		
бъекты			Комментарий	_	
🖻 🗊 Сервер	Устро	йство < <modbus>&gt; : Device1</modbus>	Включен в работу		True
🖻 🛷 Node1			Регион		COILS
- 🗐 Device	e1 00i	цие настроики	Адрес	(0x0000)	0
	Добавить	, Ter	Тип данных в устроистве	-	bool
	Переименовать	Попустройство	Тип данных в сервере		ReadWrite
	Дублировать	вета (мс)	Скрипт	-	Redutine
	Удалить	при ошибке	Разрешение выполнения скрипта посл	е чтения	False
	Вырезать	е соединение после ошибки через	Разрешение выполнения скрипта перед записью		False
	Копировать	лизация узла при ошибке	🗉 Дополнительно		
	Групповые операции	npoca	Наличие отдельного регистра записи		False
	Экспорт устройства	сть периода опроса	Чтение сразу после записи		False
	Импорт подустройства	я фаза			
	Экспорт тегов	сле запуска	НДА доступ		False
	Импорт тегов	а запроса после получения ответа			
	BURUMOCTH TOFOR	овка байтов в значении			
	Теги в работе				
	Дополнительные свойст	гва ние скрипта			
	на	строяка запросов			
	Mak	ссимальное количество HOLDING регистр	Tuppygapara 1		
	Mak	ссимальное количество INPUT регистров	Пиражировать		Да пет
	He	UCHORDSOBATE KOMAHAY WRITE_SINGLE_CO			
	Casilo	Таблица тегов			

Рисунок 5.9.18 – Добавление тега xVar

Te	er < <u><holding_registers>&gt; : wVa</holding_registers></u> r		
	Общие настройки		
	Комментарий		
	Включен в работу	True	
	Адрес (0х0001)	1	1
	Тип данных в устройстве	uint16	
	Тип данных в сервере	uint32	
	Тип доступа	ReadWrite	
	Использовать перестановку байтов устройства	True	
	Последний тег в групповом запросе	False	
	Пересчет (А*Х + В)	False	

Рисунок 5.9.19 – Добавление тега wVar

Те	er < <u><holding_registers>&gt; : rVar</holding_registers></u>		
=	Общие настройки		
	Комментарий		
	Включен в работу	True	
	Адрес (0x0002)	2	
	Тип данных в устройстве	float	
	Тип данных в сервере	float	
	Тип доступа	ReadWrite	
	Использовать перестановку байтов устройства	True	
	Последний тег в групповом запросе	False	
	Пересчет (А*Х + В)	False	

Рисунок 5.9.20 – Добавление тега rVar

Те	er < <holding_registers>&gt; : sVar</holding_registers>		
	Общие настройки		
	Комментарий		
	Включен в работу	True	
	Адрес (0х0004)	4	
	Тип данных в устройстве	string	
	Тип данных в сервере	string	
	Количество байт для строкового типа	16	
	Тип строки для строкового типа	ascii	
	Тип доступа	ReadWrite	
	Использовать перестановку байтов устройства	True	
	Последний тег в групповом запросе	False	
	Пересчет (А*Х + В)	False	

Рисунок 5.9.21 – Добавление тега sVar

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

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

Device.Application.PLC_PRG_ST		
Выражение	Тип	Значение
🗈 < fbComControl1	OCL.COM_Control	
ø fbUnpackWord     ø	UTIL.WORD_AS_BIT	
🗉 🔌 fbPackWord	UTIL.BIT_AS_WORD	
🗉 🔌 fbRealToWord2	OCL.REAL_TO_WO	
🗉 🚸 fbModbusSerialSlave	OCL.MB_SerialSlave	
🗉 🚸 awSlaveData	ARRAY [015] OF	
vVar_Opc	BOOL	TRUE
WVar_Opc	WORD	11
var_Opc	REAL	22.33
ø sVar_Opc	STRING(15)	'привет'
Var_Plc	BOOL	TRUE
wVar_Plc	WORD	11
var_Pic	REAL	22.33
🖗 sVar_Plc	STRING(15)	'привет'
xWrite	BOOL	TRUE
🗉 < fbWriteEdge	R_TRIG	

Объекты							
⊡¶ Server	Устройство < <device< td=""><td>1&gt;&gt;</td><td></td><td></td><td></td><td></td><td></td></device<>	1>>					
🖻 🕪 Node1 🖻 🖏 Device1	Теги						
🛱 xVar	Имя	Регион	Адрес	Значение	Качество	Время (UTC)	Тип в сер
wVar	Node1.Device1.xVar	COILS	(0x00	True	GOOD	2019-08-1	bool
rvar	Node1.Device1.wVar	HOL	(0x00	11	GOOD	2019-08-1	uint32
i≱ Svai	Node1.Device1.rVar	HOL	(0x00	22.330000	GOOD	2019-08-1	float
	Node1.Device1.sVar	HOL	(0x00	привет	GOOD	2019-08-1	string

Рисунок 5.9.22 – Чтение и запись данных через ОРС-сервер

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

В качестве примера будет рассмотрена настройка обмена с модулями Mx210 (MB210-101 и MK210-301) с использованием библиотеки **OwenCommunication**. В примере используется библиотека версии **3.5.11.1**.

**Реализуемый алгоритм**: если значение первого аналогового входа модуля **MB210-101** превышает **30** и при этом первый дискретный вход модуля **MK210-301** имеет значение **TRUE** (замкнут), то первому дискретному выходу модуля **MK210-301** присваивается значение **TRUE** (замкнут). Во всех остальных случаях дискретному выходу присваивается значение **FALSE** (разомкнут).

Структурная схема примера приведена на рисунке ниже:



Рисунок 5.9.23 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.x. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания:

Example_OwenCommunicationModbusTcpMaster_3511v1.projectarchive

Сетевые параметры устройств приведены в таблице ниже:

Таблица 5.9.4 – Сетевые параметры устройств	
---------------------------------------------	--

Параметр	СПК1хх [М01]	MK210-301	MB210-101
Режим работы	master	slave	slave
IP-адрес	10.2.11.170	10.2.11.180	10.2.11.181
Маска подсети		255.255.0.0	
IP-адрес шлюза		10.2.1.1	
Порт		502	
Unit ID	-	1	1

Для настройки обмена следует:

1. Настроить модули **Mx210** с помощью программы **OBEH Конфигуратор** в соответствии с таблицей 5.9.4 (см. руководство **Mx210. Примеры настройки обмена**). Подключить модули к контроллеру.

- 2. Установить в CODESYS библиотеку OwenCommunication (см. п. 5.2).
- 3. Создать новый проект CODESYS с программой на языке ST или CFC:

Стандартны	й проект	×
	Вы собираетесь создать новый стандартный проект. При этом будут созданы следующие объекты: - Одно программируемое устройство, как задано ниже - Программа PLC_PRG на языке, выбранном ниже	
	- Циклическая задача, вызывающая PLC_PRG - Ссылка на новейшую установленную версию библиотеки Standard library.	
	Устройство: SPK1xx[M01] (Production association OWEN)	<b>_</b>
	PLC_PRG на: Непрерывные функциональные схемы (CFC)	-
	ОК От	іена

Рисунок 5.9.24 – Создание проекта CODESYS



#### ПРИМЕЧАНИЕ

Проект примера содержит программы на языках СFC и ST. По умолчанию используется программа на CFC (**PLC_PRG_CFC**). Для работы с программой на ST следует в конфигурации задач для задачи **MainTask** удалить вызов **PLC_PRG_CFC** и добавить вызов **PLC_PRG_ST**.

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



#### ПРИМЕЧАНИЕ

Библиотека Util используется только в программе на языке CFC.

5. Объявить в программе следующие переменные:

/ 🔊	PLC	_PRG_CFC X		
	1	PROGRAM PLC_PRG_CFC		
	2	VAR		
	3	fbTcpClientMV210:	OCL.TCP_Client;	// ФБ ТСР-подключения к модулю MB210-101
	4	fbTcpClientMK210:	OCL.TCP_Client;	// ФБ TCP-подключения к модулю MK210-301
	5	fbMV210_101_AI1:	OCL.MB_TcpRequest;	// ФБ опроса модуля MB210-101
	6	fbMK210_301_DI:	OCL.MB_TcpRequest;	// ФБ опроса входов модуля МК210-301
	7	fbMK210_301_DO:	OCL.MB_TcpRequest;	// ФБ опроса выходов модуля МК210-301
	8			
	9	rAI1:	REAL;	// значение 1-го входа модуля MB210-101
1	10	wDiMask:	WORD;	// битовая маска входов модуля МК210-301
1	11	wDoMask:	WORD;	// битовая маска выходов модуля МК210-301
1	12	xDi0:	BOOL;	// значение 1-го входа модуля МК210-301
1	13	xDo0:	BOOL;	// значение 1-го выхода модуля МК210-301
1	14			
1	15	awAI1:	ARRAY [01] OF WORD;	// регистры, считанные с модуля MB210-101
1	16			
1	17	fbUnpackWord:	UTIL.WORD_AS_BIT;	// ФБ распаковки битовой маски
1	18	fbPackWord:	UTIL.BIT_AS_WORD;	// ФБ упаковки битовой маски
1	19			
2	20	iStateMV210:	INT;	// шаг опроса модуля MB210-101
2	21	iStateMK210:	INT;	// шаг опроса модуля МК210-301
2	22			
2	23	fbConnectionTimeout	:MV210: TON;	// таймер таймаута соединения с модулем MB210-101
2	2.4	fbConnectionTimeout	:MK210: TON;	// таймер таймаута соединения с модулем МК210-301
2	25	END VAR		

Рисунок 5.9.25 – Объявление переменных в программе

#### ПРИМЕЧАНИЕ

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

6. Нажать **ПКМ** на программу, выбрать команду **Добавление объекта – Действие** и добавить действия с названиями **MV210_101** и **MK210_301** (язык реализации действий соответствует языку программы). В рамках примера действия используются для повышения читабельности кода.

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



Рисунок 5.9.26 - Код действия MV210_101

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

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

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

Если соединение разорвано или экземпляр ФБ <u>TCP Client</u> находится в состоянии ошибки (выход **xActive** имеет значение **FALSE**), то опрос прекращается.

# i

#### ПРИМЕЧАНИЕ

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



Рисунок 5.9.27 – Код действия МК210_301

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

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

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

Если соединение разорвано или экземпляр ФБ <u>TCP Client</u> находится в состоянии ошибки (выход **xActive** имеет значение **FALSE**), то опрос прекращается.



1

#### ПРИМЕЧАНИЕ

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

## ПРИМЕЧАНИЕ

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

В программе PLC_PRG_CFC производится вызов действий MV210_101 и MK210_301, а также выполнение алгоритма, описанного в условии примера.

MV210_101 Вызов действия MV210_101
МК210_301 1 Вызов действия МК210_301
xDi0
GT 2 AND 4 xDo0 4

Рисунок 5.9.28 – Код программы PLC_PRG_CFC

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

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

Рисунок 5.9.29 - Код действия MV210_101

При первом вызове действия в **шаге 0** происходит установка соединения с модулем по заданному IPадресу (**slpAddr**) и порту (**uiPort**) с помощью экземпляра ФБ <u>TCP_Client</u> (**блок 0**). После установки соединения выход **xActive** принимает значение **TRUE**, что приводит к переходу на **шаг 1**. Если соединение не удается установить в течение заданного времени (это определяются с помощью таймера **fbConnectionTimeoutMV210**) или при установке соединения происходит ошибка, то выполняется новая попытка соединения.

На шаге 1 выполняется вызов экземпляра ФБ <u>MB TcpRequest</u>, который производит опрос первого аналогового входа модуля MB210-101 с адресом 1 (usiSlaveld). Требуемый код функции (eFunc), начальный адрес регистра (uiDataAddr) и их количество (uiDataCount) приведены в РЭ на модуль. Полученные данные помещаются в переменную awAl1 типа ARRAY [0..1] OF WORD. С помощью функции <u>WORD2 TO REAL</u> этот массив преобразуется в переменную типа rAl1 типа REAL.

После завершения работы экземпляра ФБ <u>MB_TcpRequest</u> один из его выходов принимает значение **TRUE**: если опрос произведен успешно, то значение **TRUE** принимает выход **xDone**, если ответ не получен или получен ответ с кодом ошибки Modbus – то значение **TRUE** принимает выход **xError**.

В любой из этих ситуаций происходит вызов экземпляра ФБ <u>MB_TcpRequest</u> со значением **FALSE** на входе **xExecute** (сброс блока) и переход на **шаг 2**.

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

```
CASE iStateMK210 OF
 2
 з
 0: // подключение к модулю
 4
 fbConnectionTimeoutMK210(IN := TRUE, PT := T#10S);
 5
 6
 7
 fbTcpClientMK210
 8
 (
 9
 xEnable
 := TRUE,
 tTimeout := T#5S,
10
 sIpAddr := '10.2.11.180',
uiPort := 502
11
12
13
);
14
15
 IF fbTcpClientMV210.xActive THEN
16
 iStateMK210 := 1;
17
 END IF
18
 IF fbTcpClientMK210.xError OR fbConnectionTimeoutMK210.Q THEN
19
20
 fbConnectionTimeoutMK210(IN := FALSE);
21
 fbTcpClientMK210(xEnable := FALSE);
 END IF
22
23
24
 1: // опрос дискретных входов модуля МК210-301
25
26
 fbMK210_301_DI
27
 (
28
 xExecute
 := TRUE,
 := T#50MS,
:= 3,
29
 tTimeout
30
 usiRetry
31
 hConnection
 := fbTcpClientMK210.hConnection,
32
 xIsRtuOverTcpMode := FALSE ,
 usiUnitId
33
 := 1,
:= OCL.MB_FC.READ_HOLDING_REGISTERS,
34
 eFuncCode
 := 51,
:= 1,
35
 uiDataAddr
36
 uiDataCount
37
 pData
 := ADR(wDiMask),
38
 szSize
 := SIZEOF(wDiMask)
39
);
40
41
 IF fbMK210_301_DI.xDone OR fbMK210_301_DI.xError THEN
42
 // после выполнения блока его надо сбросить
 fbMK210_301_DI(xExecute := FALSE);
43
44
45
 xDi0 := wDiMask.0:
46
47
 iStateMK210 := 2;
48
 END IF
49
50
 2: // запись дискретных выходов модуля МК210-301
51
52
 wDoMask.0 := xDo0;
53
54
 fbMK210_301_D0
55
 (
56
 := TRUE,
 xExecute
 := T#50MS,
:= 3,
57
 tTimeout
58
 usiRetry
 hConnection := fbTcpClientMK210.hConnection,
xIsRtuOverTcpMode := FALSE ,
59
60
61
 usiUnitId := 1,
62
 eFuncCode
 := OCL.MB_FC.WRITE_MULTIPLE_REGISTERS,
63
 uiDataAddr
 := 470,
64
 uiDataCount
 := 1,
65
 pData
 := ADR(wDoMask),
66
 := SIZEOF(wDoMask)
 szSize
67
);
68
 IF fbMK210_301_D0.xDone OR fbMK210_301_D0.xError THEN
69
70
 // после выполнения блока его надо сбросить
71
 fbMK210_301_D0(xExecute := FALSE);
72
73
 // перед следующим сеансом опроса проверяем соединение с модулем
74
 iStateMK210 := 0;
75
 END IF
76
77
 END_CASE
```

Рисунок 5.9.30 – Код действия МК210_301

При первом вызове действия в **шаге 0** происходит установка соединения с модулем по заданному IPадресу (**slpAddr**) и порту (**uiPort**) с помощью экземпляра ФБ <u>TCP Client</u> (**блок 0**). После установки соединения выход **xActive** принимает значение **TRUE**, что приводит к переходу на **шаг 1**. Если соединение не удается установить в течение заданного времени (это определяются с помощью таймера **fbConnectionTimeoutMK210**) или при установке соединения происходит ошибка, то выполняется новая попытка соединения.

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

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

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

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

Если соединение разорвано или экземпляр ФБ <u>TCP_Client</u> находится в состоянии ошибки (выход **xActive** имеет значение **FALSE**), то опрос прекращается.

В программе PLC_PRG_ST производится вызов действий MV210_101 и MK210_301, а также выполнение алгоритма, описанного в условии примера.

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

Рисунок 5.9.31 – Код программы PLC_PRG_ST

9. Загрузить проект в контроллер и запустить его.

В переменной **rAl1** будет отображаться текущее значение первого аналогового входа модуля **MB210-101**. В переменной **xDi0** будет отображаться текущее значение первого дискретного входа модуля **MK210-301**.

Если значение **rAl1** превысит **30** и при этом значение **xDi0** будет равно **TRUE**, то в переменную **xDo0** будет записано значение **TRUE**, что приведет к замыканию первого дискретного выхода модуля **MK210-301**. Если одно из условий перестанет выполняться, то выход будет разомкнут.



Рисунок 5.9.32 – Выполнение программы в режиме Online

# 5.9.4 CΠK1xx [M01] (Modbus TCP Slave) + MasterOPC Universal Modbus Server

В качестве примера будет рассмотрена настройка обмена с OPC-сервером <u>Insat MasterOPC Universal</u> <u>Modbus Server</u>, который будет использоваться в режиме **Modbus TCP Master**, с помощью библиотеки **OwenCommunication**. В примере используется библиотека версии **3.5.11.1**.

Структурная схема примера приведена на рисунке ниже:



Рисунок 5.9.33 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.x. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания: Example OwenCommunicationModbusTcpSlave 3511v1.zip

Сетевые параметры модулей приведены в таблице ниже:

#### Таблица 5.9.5 – Сетевые параметры устройств

Параметр	СПК1хх [М01] ПК с ОРС-сервером		
ІР-адрес	10.2.11.170	любой адрес из сети, к которой подключен контроллер	
Порт		502	
Адрес (Unit ID)	1	-	
Режим работы	slave	master	

Переменные примера описаны в таблице ниже:

#### Таблица 5.9.6 – Список переменных примера

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



#### ПРИМЕЧАНИЕ

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

Для настройки обмена следует:

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

Стандартн	ый проект
	Вы собираетесь создать новый стандартный проект. При этом будут созданы следующие объекты: - Одно программируемое устройство, как задано ниже - Программа PLC_PRG на языке, выбранном ниже - Циклическая задача, вызывающая PLC_PRG - Ссылка на новейшую установленную версию библиотеки Standard library.
	Устройство: SPK1xx[M01] (Production association OWEN)   PLC_PRG на: Непрерывные функциональные схемы (CFC)   OK Отмена

Рисунок 5.9.12 – Создание проекта CODESYS

#### ПРИМЕЧАНИЕ

Проект примера содержит программы на языках СFC и ST. По умолчанию используется программа на CFC (**PLC_PRG_CFC**). Для работы с программой на ST следует в конфигурации задач для задачи **MainTask** удалить вызов **PLC_PRG_CFC** и добавить вызов **PLC_PRG_ST**.

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



i

#### ПРИМЕЧАНИЕ

Библиотека Util используется только в программе на языке CFC.

4. Объявить в программе следующие переменные:

1	PROGRAM PLC_PRG_ST		
2	VAR		
3			
4	fbUnpackWord:	UTIL.WORD_AS_BIT;	// ФБ распаковки битовой маски
5	fbPackWord:	UTIL.BIT_AS_WORD;	// ФБ упаковки битовой маски
6	fbRealToWord2:	OCL.REAL_TO_WORD2;	// ФБ преобразования REAL в две переменные типа WORD
7			
8			
9	fbModbusTcpSlave:	OCL.MB_TcpSlave;	// ФБ для реализиации Modbus Slave
10			
11	awSlaveData:	ARRAY [015] OF WORD;	// буфер данных Modbus Slave
12			
13	(* значения, полученные	OT OPC *)	
14	xVar_Opc:	BOOL;	
15	wVar_Opc:	WORD;	
16	rVar_Opc:	REAL;	
17	sVar_Opc:	STRING(15);	
18			
19	(* значения для передачи	и в <i>ОРС *)</i>	
20	xVar_Plc:	BOOL;	
21	wVar_Plc:	WORD;	
22	rVar_Plc:	REAL;	
23	sVar_Plc:	STRING(15);	
24			
25	xWrite:	BOOL;	// команда записи данных из программы в регистры Modbus Slave
26	fbWriteEdge:	R_TRIG;	// триггер для однократной записи
27	END_VAR		
2.0			

Рисунок 5.9.34 – Объявление переменных программы



#### ПРИМЕЧАНИЕ

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



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

Рисунок 5.9.35 – Код программы на языке CFC

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

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

В блоках 1–6 происходит копирование данных из регистров slave'а в переменные программы. Для выделения переменных типа **BOOL** из переменной типа **WORD** используется экземпляр ФБ **WORD_AS_BIT** из библиотеки **Util**. Для преобразования двух переменных типа **WORD** в переменную типа **REAL** используется функция <u>WORD2_TO_REAL</u>. Для преобразования набора переменных типа **WORD** в переменную типа **STRING** используется функция <u>SWAP_DATA</u>. Для обеспечения порядка байт, принятого в OPC-сервере, в процессе копирования происходит перестановка байт в регистрах (**xSwapBytes=TRUE**).

# і Примечание

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

По переднему фронту переменной **xWrite** (блок 8) происходит запись переменных программы в регистры slave'a. Для записи переменной типа **BOOL** используется экземпляр ФБ **BIT_AS_WORD** из библиотеки **Util**. Для записи переменной типа **REAL** используется экземпляр ФБ <u>REAL TO WORD2</u>. Для записи переменной типа **STRING** используется функция <u>SWAP_DATA</u>.

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

```
// чтобы запустить пример на ST на контроллере требуется:
 3
 // 1. Удалить из задачи MainTask вызов программы PLC PRG CFC
 4
 // 2. Привязать к задаче MainTask вызов программы PLC PRG ST
 5
 fbModbusTcpSlave
 8
 (
 := TRUE,
 9
 xEnable
 sIpAddr := '0.0.0.0',
10
11
 uiPort
 := 502,
 usiUnitId := 1,
12
 pData := ADR(aworave____
erSize := SIZEOF(awSlaveData)
13
14
15
);
16
17
 // данные, полученные от OPC
18
 xVar_Opc := awSlaveData[0].0;
19
 wVar_Opc := awSlaveData[1];
20
 rVar Opc := OCL.WORD2 TO REAL(awSlaveData[2], awSlaveData[3], FALSE);
21
 OCL.SWAP_DATA(ADR(awSlaveData[4]), ADR(sVar_Opc), SIZEOF(sVar_Opc), TRUE, FALSE, FALSE);
22
23
24
 // по команде записываем переменные из программы в регистры Modbus Slave
25
 fbWriteEdge(CLK := xWrite);
26
27
 IF fbWriteEdge.Q THEN
28
29
 awSlaveData[0].0 := xVar_Plc;
30
 awSlaveData[1] := wVar_Plc;
31
 fbRealToWord2(rInput := rVar_Plc, wOutput1 => awSlaveData[2], wOutput2 => awSlaveData[3]);
32
 OCL.SWAP_DATA(ADR(sVar_Plc), ADR(awSlaveData[4]), SIZEOF(sVar_Plc), TRUE, FALSE, FALSE);
33
34
 END IF
```

Рисунок 5.9.36 – Код программы на языке ST

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

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



#### ПРИМЕЧАНИЕ

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

По переднему фронту переменной **xWrite** происходит запись переменных программы в регистры slave'a. Для записи переменной типа **REAL** используется экземпляр ФБ <u>REAL TO WORD2</u>. Для записи переменной типа **STRING** используется функция <u>SWAP_DATA</u>.

7. Установить и запустить MasterOPC Universal Modbus Server.

8. Нажать ПКМ на узел Server и добавить коммуникационный узел типа TCP/IP. В узле следует указать сетевые настройки в соответствии с <u>таблицей 5.9.5</u>. Для работы OPC-сервера в режиме Modbus TCP Master параметры Modbus поверх TCP и Slave подключение должны иметь значение FALSE.

	MasterOPC Universal Mo	dbus Server Demo 60000 1 H	nour Build - 4.2.42			- C X	
Конфигурация Общие настро	йки Помощь						
🗐 Создать 💾 Сохранить как	🗓 Добавить узел 👘 Д	обавить тег	🖃 Переименовать	🗙 Удалить			
🖗 Открыть 🛛 🛷 Сделать стартовым	🕅 Добавить устройство 🔺 Г	ереместить вверх	🦲 Копировать	🖍 Отменить			
📔 Сохранить 📋 Импорт из версии 2.0	🍕 Добавить группу 🛛 🗍 Г	Іереместить вниз	🗍 Вставить	<i>🕫</i> Вернуть			
Файл конфигурации	Сервер		Правк	а			
Текущая конфигурация : Example_Modbu	ısRtuSlave.mbp						
Объекты		Редактирование ком				_ 0 <u>_ X</u>	
		у сдактирование коли					
Добавить Ко	оммуникационный узел	Имя узла Node1					
Групповые операции		Общие настройк	и				
Импорт узла		Комментарий					
Узлы в работе	7	Включен в работу	1		True		
Tonia o pocoro		Тип узла			TCP/IP		
		Настройки ТСР/	(P		10 2 11 170	-	
		IP адрес IP порт			502	-	
		Время ожидания о	соединения (с)		10	-	
		Повторы при оши	бке		3		
		Межсимвольный таймаут (мс)			1000		
		Скрипт					
		Выполнение скриг	та		False	0.2.11.170 02 000 alse	
		Дополнительны	е настройки		Cales		
		Modbus popeny T(	rp		False		
		Принудительный р	Эазрыв соединения в	з каждом цикле	False		
		Отслеживать Trar	isaction ID		True		
		Подключение в ре	ежиме ТСР сервера		False		
		Использовать рез	ервные каналы		False		
		🔄 Тиражировать	1		Да	Нет	
	U					Свойства объ	
Режим Конфигурирование	Теги : Разрешено - 600	00. Загружено - О				.di	

Рисунок 5.9.37 – Добавление коммуникационного узла

конфи	гурация	Общие настрой	іки Помощь			
Создать	💾 Coxpa	нить как	🗓 Добавить узел	🖷 Добавить тег	🖃 Переименовать 🗙 Удалить	
Открыть	🖉 Сдела	ть стартовым	П. Добавить устройство	🕆 Переместить вверх	П Копировать Отменить	
Соуранить	ПИмпол	т из версии 2.0	Побавить группу			
осохранито Файг	- Konquer		Cer	Редактирование устройств	Ba	
(Carl)	тконфиту	, CED mba	00			
сущая конфі	игурация	: 653.mbp		имя устроиства Devic	cei	
Бекты				Общие настройки		
Server				Комментарий		
Sode Node	1		Устройство	Включено в работу		True
	дооа	БИТО	Устроиство	Тип устройства		MODBUS
	Пере	именовать		Адрес	(0x01	) 1
	Дубл	ировать	3	время ответа (мс)		1000
	Удал	ить		Повторы при ошиоке		10
	Выре:	зать		Ремнициализация узл		False
	Копи	оовать		Период опроса		1000
	Групп	повые операции		Размерность периода	опроса	ms
	Эксп			Начальная фаза		0
	Импо	рт устройства		Размерность фазы		ms
				Старт после запуска		True
	устро	риства в работе		Задержка запроса по	сле получения ответа (мс)	4
				Перестановка байтов	в значении	Вызов .
				😑 Скрипт		
				Выполнение скрипта		False
				Настройка запросо	8	
				Максимальное количе	ество HOLDING регистров в запросе чтения	125
				Максимальное количе	ество INPUT регистров в запросе чтения	125
				Не использовать кома	анду WRITE_SINGLE_COIL (0х05)	True

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

Рисунок 5.9.38 – Добавление устройства

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

Создать 🖪	Сохранить как	Доб	авить узел 🖷 Добавить тег		Переименовать Удалить			
	Cooperty creetophik	EL Dof		-				
P OTKPOTO -	сделать стартовым	ap doc	авить устроиство Птереместить	1	Редактирование тега			
Сохранить	Импорт из версии 2.	о 🭕 Доб	авить группу 🔮 Переместить	B				
Файл ко	онфигурации		Сервер		Имя тега XVar			
Гекущая конфигу	рация : 11.mbp			6	Общие настройки			
Объекты					Комментарий			
🕀 🕅 Сервер	1	Устройств	MODBUS>> : Device1		Включен в работу	( )	True	
🖻 🛷 Node1		responent			Регион		COILS	
🗌 🗐 Device	e <b>1</b>	🗉 Общие	настройки	-	Адрес	(0x0000)	0	
	Добавить	•	Ter		Тип данных в устройстве		bool	
	Переименовать		Группу		Тип данных в сервере		bool	
	Дублировать		Подустройство		Тип доступа		ReadWrite	
			вета (мс)	-	В Скрипт			
	удалить	при ошибке			Разрешение выполнения скрипта после чтения		False	
	Вырезать		е соединение после ошибки через илизация узла при ошибке импроса		Разрешение выполнения скрипта пе	False		
	Копировать				Дополнительно			
	Групповые опера	ции			Наличие отдельного регистра запис	ельного регистра записи		
	Экспорт устройс	тва	сть периода опроса		Чтение сразу после записи		False	
	Импорт подустро	йства	я фаза	- 6	HDA			
	2		сть фазы		HDA доступ		False	
	Экспорт тегов		сле запуска					
	импорт тегов		в запроса после получения ответа					
	Видимость тегов		овка одитов в значении					
	Теги в работе							
	Дополнительные	свойства	ие скрипта					
		Maxan	настрояка запросов					
		Максим	альное количество носолю регист	P	Птиражировать 1		Ла Нет	
		Houcon	and the construction in the second of the se					
		He MCHC	WOULD ROMANDY WRITE_SINGLE_C		CTED (0.00) T			
	1							

Рисунок 5.9.39 – Добавление тега xVar

Т	Ter < <holding_registers>&gt; : wVar</holding_registers>							
E	Общие настройки							
	Комментарий							
	Включен в работу	True						
	Адрес (0х0001)	1						
	Тип данных в устройстве	uint16						
	Тип данных в сервере	uint32						
	Тип доступа	ReadWrite						
	Использовать перестановку байтов устройства	True						
	Последний тег в групповом запросе	False						
	Пересчет (А*Х + В)	False						

Рисунок 5.9.40 – Добавление тега wVar

Те	Ter < <u><holding_registers>&gt; : rVar</holding_registers></u>						
=	Общие настройки						
	Комментарий						
	Включен в работу	True					
	Адрес (0x0002)	2					
	Тип данных в устройстве	float					
	Тип данных в сервере	float					
	Тип доступа	ReadWrite					
	Использовать перестановку байтов устройства	True					
	Последний тег в групповом запросе	False					
	Пересчет (А*Х + В)	False					

Рисунок 5.9.41 – Добавление тега rVar

Те	er < <holding_registers>&gt; : sVar</holding_registers>		
	Общие настройки		
	Комментарий		
	Включен в работу	True	
	Адрес (0х0004)	4	
	Тип данных в устройстве	string	
	Тип данных в сервере	string	
	Количество байт для строкового типа	16	
	Тип строки для строкового типа	ascii	
	Тип доступа	ReadWrite	
	Использовать перестановку байтов устройства	True	
	Последний тег в групповом запросе	False	
	Пересчет (А*Х + В)	False	

Рисунок 5.9.42 – Добавление тега sVar

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

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

Device.Application.PLC_PRG_ST					
Выражение	Тип	Значение			
🗈 < fbUnpackWord	UTIL.WORD_AS_BIT				
Image: Book of the second s	UTIL.BIT_AS_WORD				
Image: Book of the second s	OCL.REAL_TO_WO				
Image: Book of the second s	OCL.MB_TcpSlave				
🗄 < awSlaveData	ARRAY [015] OF				
<pre>     xVar_Opc </pre>	BOOL	TRUE			
wVar_Opc	WORD	11			
rVar_Opc	REAL	22.33			
🕸 sVar_Opc	STRING(15)	'привет'			
xVar_Plc	BOOL	TRUE			
wVar_Plc	WORD	11			
var_Plc	REAL	22.33			
🕸 sVar_Pic	STRING(15)	'привет'			
xWrite	BOOL	TRUE			
🗄 < fbWriteEdge	R_TRIG				

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

Рисунок 5.9.43 – Считывание и запись данных через ОРС-сервер

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

Функциональный блок <u>MB TcpSlave</u> поддерживает 20 функцию Modbus (**Read File Record**), что позволяет считывать файлы контроллера. Устройство, которое является Modbus TCP Master'ом, должно также поддерживать эту функцию.

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

Данный ОРС-сервер включает скрипт, который позволяет считывать только файлы определенного формата (формата архива, поддержанного в контроллерах ОВЕН ПЛК1xx, режим ASCII). Описание этого формата доступно по <u>ссылке</u>. Пользователь может отредактировать скрипт для поддержки другого формата. Скрипт имеет название **OwenPicHDA.lua** и расположен в директории **C:\ProgramData\InSAT\MasterOPC Universal Modbus Server\MODULES**. Описание принципов работы со скриптами OPC-сервера доступно по <u>ссылке</u>.

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

Структурная схема примера приведена на рисунке ниже:



Рисунок 5.9.44 – Структурная схема примера

Пример создан в среде CODESYS V3.5 SP11 Patch 5 и подразумевает запуск на CПК1xx [M01] с таргет-файлом 3.5.11.x. В случае необходимости запуска проекта на другом устройстве следует изменить таргет-файл в проекте (ПКМ на узел Device – Обновить устройство).

Пример доступен для скачивания: Example OwenCommunicationModbusTcpSlaveFiles 3511v1.zip

Сетевые параметры модулей приведены в таблице ниже:

#### Таблица 5.9.7 – Сетевые параметры устройств

Параметр	СПК1хх [М01] ПК с ОРС-сервером		
IP-адрес	10.2.11.170	любой адрес из сети, к которой подключен контроллер	
Порт	502		
Адрес (Unit ID)	- 1		
Режим работы	slave master		

Для настройки обмена следует:

1. Подключить контроллер и ПК к общей локальной сети.

2. Убедиться, что в контроллере по пути /mnt/ufs/home/root находится файл Arc1.log. В рамках примера этот файл может быть загружен в контроллер вручную через утилиту <u>WinSCP</u> (см. руководство CODESYS V3.5. FAQ, п. 13.5).

3. Создать новый проект CODESYS с программой на языке ST или CFC:



Рисунок 5.9.45 – Создание проекта CODESYS

#### ПРИМЕЧАНИЕ

Проект примера содержит программы на языках СFC и ST. По умолчанию используется программа на CFC (**PLC_PRG_CFC**). Для работы с программой на ST следует в конфигурации задач для задачи **MainTask** удалить вызов **PLC_PRG_CFC** и добавить вызов **PLC_PRG_ST**.

4. Объявить в программе следующие переменные:

1	PROGRAM PLC_PRG_CFC		
2	VAR		
3			
4	fbModbusTcpSlave:	OCL.MB_TcpSlave;	// ФБ для peaлизиации Modbus Slave
5	awSlaveData:	ARRAY [015] OF WORD;	// буфер данных Modbus Slave
6			
7	// путь к файлам архивов		
8	asFilePath:	ARRAY [18] OF STRING	:= ['/mnt/ufs/home/root/Arc1.log', 7('')];
9	END VAR		

Рисунок 5.9.46 – Объявление переменных программы



5. Код программы будет выглядеть следующим образом:

Рисунок 5.9.47 – Код программы на языке CFC

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

Рисунок 5.9.48 – Код программы на языке ST

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

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

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

Для работы с <u>Insat MasterOPC Universal Modbus Server</u> параметр **c_xReverseByteOrderFiles** (порядок байт при передаче файлов) должен иметь значение **TRUE**, а параметр **c_usiAmountBytes** (размер записи в файле) – значение **10**.

6. Установить и запустить MasterOPC Universal Modbus Server.

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

	MasterOPC Universal Mo	odbus Server Demo 60000 1 h	nour Build - 4.2.42		
Конфигурация Общие настро	і́ки Помощь				
🗐 Создать 🛛 💾 Сохранить как	🐧 Добавить узел 🛛 👒 ,	Добавить тег	🖃 Переименовать 🗙 Удалить		
🖗 Открыть 🛛 🖋 Сделать стартовым	🕅 Добавить устройство 🔺 🛙	Переместить вверх	🗍 Копировать 🛛 🖛 Отменить		
📔 Сохранить 📋 Импорт из версии 2.0	🍕 Добавить группу 💦 🦊 I	Переместить вниз	📋 Вставить 🖉 Вернуть		
Файл конфигурации	Сервер		Правка		
	sRtuSlave.mbp				
Объекты	(				
		Редактирование комм	куникационного узла		
Добавить Ко	ммуникационный узел	Имя узла Node1			
Групповые операции	<u> </u>	Е общио изстройи			
MADODT V202		Комментарий	n		
in the second	*	Включен в работу		True	
узлы в работе		Тип узла		TCP/IP	
		Настройки ТСР/1	(P		
		IP адрес		10.2.11.170	
		IP порт	502		
		Время ожидания с	соединения (с)	10	
		Повторы при ошиб	3		
		Межсимвольный т	аймаут (мс)	1000	
		Скрипт			
		Выполнение скриг	False		
		Дополнительные			
		Slave подключени	False		
		Modbus noBepx TCP		False	
		Принудительный разрыв соединения в каждом цикле		e False	
		отслеживать Iran	isaction ID	True	
		Подключение в ре	ежиме ТСР сервера	False	
		использовать рез	ервные каналы	Faise	
		П пиражировать	1	да нет	
				Свойства	
Режим Конфигурирование	Теги : Разрешено - 600	00. Загружено - 0			

Рисунок 5.9.49 – Добавление коммуникационного узла

MasterOPC Universal Modbus Server Demo 60000 1 hour Build - 4.2.28 Конфигурация Общие настройки Помощь Создать Н Сохранить как...
 О Добавить узел Добавить тег
 Переименовать Худалить
 Открыть Сделать стартовым
 Добавить устройство Переместить вверх
 Копировать Отменить
 Копировать Отменить
 Стартовать 🖹 Сохранить 🗋 Импорт из версии 2.0 🍕 Добавить группу 🛛 🎯 Редактирование устройства - • × Файл конфигурации Ce Имя устройства Device1 Текущая конфигурация : 653.mbp Объекты Общие настройки Server Комментарий Включено в работу True MODBUS Тип устройства іство Адрес Время ответа (мс) Переименовать (0x01) 1 1000 Дублировать время ответа (мс) Повторы при ошибке Повторное соединение после ошибки через (с) 3 Удалить 10 Вырезать Реинициализация узла при ошибке False Копировать Период опроса 1000 Групповые операции Размерность периода опроса ms Начальная фаза 0 Экспорт узла Импорт устройства Размерность фазь ms Старт после запуска True Устройства в работе Задержка запроса после получения ответа (мс) Перестановка байтов в значении Вызов .... Скрипт Выполнение скрипта False Настройка запросов Максимальное количество HOLDING регистров в запросе чтения 125 Максимальное количество INPUT регистров в запросе чтения Не использовать команду WRITE_SINGLE_COIL (0x05) 125 True П Тиражировать 1 Да Нет Теги : Разрешено - 60000. Загружено - 8 Конфигурирование Режим

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

Рисунок 5.9.50 – Добавление устройства

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

	MasterOPC Universal Modbus Se	rver Demo 60000 1 hour Build	1 - 4.2.42		
Конфигурация Общие настройки Помощь					
🗐 Создать 💾 Сохранить как 🗈 Добавить узел	🔩 Добавить тег	🖃 Переименовать 💙	🗙 Удалить		
🖗 Открыть 🛭 🖋 Сделать стартовым 🛛 🧠 Добавить устройс	тво 🔺 Переместить вверх	🔲 Копировать 🛛 🖛	• Отменить		
🗎 Сохранить 📋 Импорт из версии 2.0 🍕 Добавить группу	🖶 Переместить вниз	🗍 Вставить 🛛 🧖	≌ Вернуть		
Файл конфигурации	Сервер	Правила			
Текущая конфигурация : Example_ModbusTcpSlaveFiles.mbp	00	↓	ARY > OWEN	- 4 Nouck: OWEN	
	Vacua			8=	
□ I Node1	5100		▲ I/I	-	
Добавить •	🗙 и	збранное Заспузки		Owen Hirtony HDA cod	10/
Переименовать	<u> </u>	Недавние места		_ Owen Thistory TIDA.ssu	154
Дублировать		Рабочий стол			
Удалить	<b>5</b>	иблиотеки	-		
Вырезать Копировать		Subversion	-		Нет данных
Групповые операции		Видео			для :дварительн
Экспорт устройства		Документы Изображения			просмотра.
Импорт подустройства		Музыка			
Экспорт тегов Импорт тегов					
Видимость тегов		Локальный диск (С:)			
Теги в работе		Локальный диск (D:)		III	•
Дополнительные свойства		Имя файла:	Owen History HDA se	sd 🗸 Файлы импорта	устройств (ф 💌
			onen istory ribris.	Открыть	Отмена
Режим Конфигурирование Теги : Разреше	но - 60000. Загружено - О. В	текущем элементе - 0	)		i.

Рисунок 5.9.51 – Импорт подустройства

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

MasterOPC Universal Modbus Server Demo 60000 1 hour Build - 4.2.42							
Конфигурация Общие настройки Помощь							
🗐 Создать 💾 Сохранить как 🛛 🗎	Добавить узел 👒 Добавить тег 📄 Переименовать 🗙 Удалить						
🖗 Открыть 🛛 🛷 Сделать стартовым 🛛 🧌	Добавить устройство 🔺 Переместить вверх 🛛 🤁 Копировать 🛛 🖛 Отменить						
📔 Сохранить 📋 Импорт из версии 2.0 🍕	Добавить группу 🐇 Переместить вниз 🔲 Вставить 🛹 Вернуть						
Файл конфигурации Сервер Правка							
Текущая конфигурация : Example_ModbusTe	cpSlaveFiles.mbp						
Объекты							
🖃 🗊 Server	Подустройство <<Архив >>						
□ # Node1	🗉 Общие настройки						
🖹 🎲 Архив	Комментарий Архив1						
➡ Var1	Включено в работу True						
► Var2	Использовать цикл опроса устройства False						
	Период опроса 10000						
	Размерность периода опроса ms						
	Начальная фаза 0						
	Размерность фазы ms						
	Старт после запуска Тrue						
	Скрипт						
	Выполнение скрипта True						
	Редактирование кода скрипта Вызов редактора скрипта						
	Дополнительные свойства						
	Номер файла 1						
	Максимальное количество записеи 20000						
	количество считываемых записеи в запросе 24						
	использовать докачивание фаила						
<u></u>	Свойства объекта Таблица тегов						
Режим Конфигурирование	Теги : Разрешено - 60000. Загружено - 2. В текущем элементе - 2	.4					

Рисунок 5.9.52 – Настройки подустройства

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

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

	And and and and and and a	MasterOPC Universal Modbus Server Demo 60000	) 1 hour Build - 4.2.42				
Конфигурация Общие настройки Помощь							
Конфигурация Общие наст     Содат     Соданть Стартовыя     Соданть Стартовыя     Соденить Стартовыя     Соденить Стартовыя     Соденить Стартовыя     Соденить Стартовыя     Соданть Инмпорт из версия 2     Файл конфигурация     Собъекты     Фолосі      Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі     Фолосі      Фолосі     Фолосі     Фолосі     Фол	ройки Помощь 1 Лобавить узел 1 Лобавить тег 2 Добавить устройство Переместить виерх 3 Добавить сруппу Переместить вика Сервер Тег Группу рации ройства е свойства	МазегОРС Universal Modbus Sever Demo 6000	Редактирование тега Иня тега Var0 Общие настройки Конентарий Вилочен в работу Регион Тип данных в сервере Тип доступа Воличество записей в архиве (100 - 4080) Аагисантическая запись Запись по изменению значения тега	True SERVER_ONLY uint2 ReadOnly HA False True 1000 False False			
			🗐 Тиражировать 1 🕞	Да Нет			
ежим Конфигурирование	Теги : Разрешено - 60000. Загружено - 2. І	В текущем элементе - 2					

Рисунок 5.9.53 – Добавление тега в подустройство



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

Рисунок 5.9.54 – Перемещение тегов в дереве ОРС-сервера

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

🖃 🗊 Server	Подустройство <<Архив>> Архив1			
⊖ I vode1 i i i Device1 i i i Aрхив Var0 i Var1 i Var2	Теги			
	Имя	Регион	Адрес	Значение
	Node1.Device1.Apx	SER		54433
	Node1.Device1.Apx	SER		4294847327
	Node1.Device1.Apx	SER		-0.531566

Рисунок 5.9.55 – Считывание и запись данных через ОРС-сервер
Для просмотра истории тега следует выделить его в дереве проекта и перейти на вкладку **HDA** (в OPC формат отображения **DEC**, порядок – от новых записей к старым. В файле архива формат отображения **HEX**, порядок – от старых записей к новым):

Стартовая конфигурация : Example_	Modbus	TcpSlaveFiles.m	bp					
Объекты								
E-1 Server	Теги	7						
🖮 🛷 Node1								
🖻 👹 Device1	Имя		Регион	Адрес	Значение	Качество	Время (UTC)	Тип в сер
🖹 🍿 Архив	Node1	Device1.Apx	SER		54433	GOOD	2000-01-0	uint32
→ Var0 → Var1								
Var2								
			1					
	Сообц	цения Запросы	Сообще	ния скри	птов HDA : Var0			
	Nº.	Значение	Качеств	о Время	а записи			
	481	52824	GOOD	2000-	01-01 03:14:21.000			
	480	47824	GOOD	2000-	01-01 03:14:16.000			
	479	42824	GOOD	2000-	01-01 03:14:11.000			
	478	37824	GOOD	2000-	01-01 03:14:06.000			
	477	32824	GOOD	2000-	01-01 03:14:01.000			
	476	27824	GOOD	2000-	01-01 03:13:56.000			
	475	22824	GOOD	2000-	01-01 03:13:51.000			
	474	17824	GOOD	2000-	01-01 03:13:46.000			
	473	12824	GOOD	2000-	01-01 03:13:41.000			
	472	7824	GOOD	2000-	01-01 03:13:36.000			
	471	2824	GOOD	2000-	01-01 03:13:31.000			
	470	63360	GOOD	2000-	01-01 03:13:26.000			
	469	58360	GOOD	2000-	01-01 03:13:21.000			
	468	53360	GOOD	2000-	01-01 03:13:16.000			
	467	48360	GOOD	2000-	01-01 03:13:11.000			
	466	43360	GOOD	2000-	01-01 03:13:06.000			
	465	38360	GOOD	2000-	01-01 03:13:01.000			
	464	33360	GOOD	2000-	01-01 03:12:56.000			

Рисунок 5.9.56 – Считывание архивных данных через ОРС-сервер

			-	
2000.01.01	03:12:56	#000=8250	≠001=ffe07db0	#002=-0.377153
2000.01.01	03:13:01	#000=95d8	≠001=ffe06a28	#002=-0.796073
2000.01.01	03:13:06	#000=a960	#001=ffe056a0	#002=-0.993284
2000.01.01	03:13:11	#000=bce8	#001=ffe04318	#002=-0.913863
2000.01.01	03:13:16	#000=d070	#001=ffe02f90	#002=-0.579929
2000.01.01	03:13:21	#000=e3f8	#001=ffe01c08	#002=-0.0844832
2000.01.01	03:13:26	#000=f780	#001=ffe00880	#002=0.434491
2000.01.01	03:13:31	#000=0b08	≠001=ffdff4f8	#002=0.832459
2000.01.01	03:13:36	#000=1e90	#001=ffdfe170	#002=0.998585
2000.01.01	03:13:41	#000=3218	≠001=ffdfcde8	#002=0.886602
2000.01.01	03:13:46	#000=45a0	≠001=ffdfba60	#002=0.527698
2000.01.01	03:13:51	#000=5928	≠001=ffdfa6d8	#002=0.0218292
2000.01.01	03:13:56	#000=6cb0	#001=ffdf9350	#002=-0.490119
2000.01.01	03:14:01	#000=8038	≠001=ffdf7fc8	#002=-0.865568
2000.01.01	03:14:06	#000=93c0	#001=ffdf6c40	#002=-0.999954
2000.01.01	03:14:11	#000=a748	#001=ffdf58b8	#002=-0.855851
2000.01.01	03:14:16	#000=bad0	#001=ffdf4530	#002=-0.47339
2000.01.01	03:14:21	#000=ce58	#001=ffdf31a8	#002=0.0409107
	2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01 2000.01.01	2000.01.01         03:12:56           2000.01.01         03:13:01           2000.01.01         03:13:01           2000.01.01         03:13:11           2000.01.01         03:13:11           2000.01.01         03:13:12           2000.01.01         03:13:21           2000.01.01         03:13:21           2000.01.01         03:13:26           2000.01.01         03:13:31           2000.01.01         03:13:31           2000.01.01         03:13:41           2000.01.01         03:13:51           2000.01.01         03:13:51           2000.01.01         03:13:51           2000.01.01         03:14:61           2000.01.01         03:14:11           2000.01.01         03:14:11           2000.01.01         03:14:11           2000.01.01         03:14:11           2000.01.01         03:14:11	2000.01.01         03:12:56         #000=8250           2000.01.01         03:13:01         #000=95d8           2000.01.01         03:13:01         #000=a960           2000.01.01         03:13:11         #000=c826           2000.01.01         03:13:11         #000=c878           2000.01.01         03:13:11         #000=c878           2000.01.01         03:13:21         #000=c780           2000.01.01         03:13:31         #000=1690           2000.01.01         03:13:31         #000=1290           2000.01.01         03:13:41         #000=3218           2000.01.01         03:13:51         #000=5928           2000.01.01         03:13:56         #000=6cb0           2000.01.01         03:14:01         #000=8038           2000.01.01         03:14:10         #000=8038           2000.01.01         03:14:11         #000=8038           2000.01.01         03:14:11         #000=8038           2000.01.01         03:14:11         #000=8038           2000.01.01         03:14:11         #000=8038           2000.01.01         03:14:11         #000=8038           2000.01.01         03:14:11         #000=8038           2000.01.01	2000.01.01         03:12:56         #000=8250         #001=ffe07db0           2000.01.01         03:13:01         #000=95d8         #001=ffe06a28           2000.01.01         03:13:01         #000=a960         #001=ffe06a28           2000.01.01         03:13:01         #000=a960         #001=ffe04318           2000.01.01         03:13:11         #000=bc8         #001=ffe04318           2000.01.01         03:13:16         #000=d070         #001=ffe02f90           2000.01.01         03:13:21         #000=a780         #001=ffe01c08           2000.01.01         03:13:21         #000=f780         #001=ffe01880           2000.01.01         03:13:31         #000=b08         #001=ffdf4f48           2000.01.01         03:13:31         #000=a1e90         #001=ffdff4f8           2000.01.01         03:13:41         #000=3218         #001=ffdfcc48           2000.01.01         03:13:51         #000=5928         #001=ffdfa6d8           2000.01.01         03:13:51         #000=6cb0         #001=ffdf9350           2000.01.01         03:14:01         #000=a748         #001=ffdf58b8           2000.01.01         03:14:11         #000=a6d8         #001=ffdf4530           2000.01.01         03:14:11         #000=

Рисунок 5.9.57 – Содержимое файла архива

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

## 6 FAQ

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

В случае возникновения проблем при настройке обмена рекомендуется:

1. Проверить (прозвонить) линию связи. Проверить распиновку кабеля (в случае использования кабелей с разъемом DB9). Проверить, что контакты **A** и **B** (или **RXD** и **TXD**) не перепутаны местами.

2. Проверить используемый номер СОМ-порта в CODESYS.

**3**. Проверить соответствие сетевых настроек контроллера и подключаемых приборов (скорость обмена, количество стоп-бит, адреса slave-устройств и т. д.).

**4**. Проверить <u>настройки опроса регистров</u>: используемые коды функций, адреса регистров, типы данных и т. д.

**5**. Проверить, нет ли разрывов в карте регистров slave-устройства (в случае использования групповых запросов).

6. Проверить, что на шине находится только одно master-устройство (для Modbus RTU/ASCII).

7. Проверить, что в сети нет slave-устройств с одинаковыми адресами.

**8**. В случае опроса модулей **Mx110** – с помощью программы <u>Конфигуратор Mx110</u> проверить, что модулям заданы корректные сетевые настройки.

9. В случае использования <u>стандартных средств конфигурирования</u> – проверить, что на вкладке привязки переменных для параметра Всегда обновлять переменные установлено значение Вкл. 2 (Всегда в задаче цикла шины).

Общее	Каналы						
	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описани
Канал Modbus Slave	🖃 🍫		Channel 0	%IW0	ARRAY [00] O		Read Hold
Modbus Slave Init	🖻 👋 Application.PLC_PRG.wVar	<b>°</b> ∳	Channel 0[0]	%IW0	WORD		0000
ModbusGenericSerialSlave							
Соотнесение входов/выходов							
Соотнесение входов/выходов Состояние							
Соотнесение входов/выходов Состояние							
Соотнесение входов/выходов Состояние Информация							
Соотнесение входов/выходов Состояние Информация							
Соотнесение входов/выходов Состояние Информация							
Соотнесение входов/выходов Состояние Информация							
Соотнесение входов/выходов Состояние Информация							
Соотнесение входов/выходов Состояние Информация	•		m				

Рисунок 6.1 – Настройка параметра Всегда обновлять переменные

Если успешное выполнение всех вышеперечисленных пунктов не приведет к настройке обмена, то следует обратиться в <u>техническую поддержку компании Овен</u>. При обращении следует предоставить указанную ниже информацию и материалы:

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

- маркировку используемых кабелей, информацию об изоляции и заземлении, наличии согласующих резисторов (терминаторов) и повторителей, а также условиях, в которых находятся приборы (например, о присутствии в шкафу автоматики силового оборудования);
- архивы проектов для программируемого оборудования Овен (СПК, ПЛК, панели оператора и т. д.), скриншоты сетевых настроек конфигурируемых приборов (модули Мх110, ТРМ и т. д.) и приборов других производителей (а также карты регистров этих устройств).

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

См. п. 4.8 (перечисления, указатели) и п. 5.8 (функции библиотеки OwenCommunication).

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

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

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

## 6.4 Вопросы по стандартным средствам конфигурирования

## 6.4.1 Какие версии компонентов рекомендуются к использованию?

Таблица рекомендуемых версий компонентов Modbus приведена в <u>приложении A</u>.

Узнать используемую версию компонента можно на вкладке Информация:

Modbus_Serial_Device 🗙		
Modbus Serial Device	Общее	Имя: Modbus Serial Device
Modbus Serial Device Соотнесение входов/выходов		Производитель: 3S - Smart Software Solutions GmbH Группы: Устройство Modbus Serial Типо 94
Информация		ПП: 94 ID: 0000 0001 Версия: <u>3.5.11.0</u> Номер модели: -
		Описание: A device that works as a Modbus Serial standalone slave. Версия конфигурации: 3.5.5.0

#### Рисунок 6.2 – Версия компонента Modbus Slave Serial Device

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

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

Канал	
Имя	Channel 0
Тип доступа	Read Holding Registers (Код функции 3) 🔹
Триггер	Передний фронт 🔻 Время цикла (мс) 100
Комментарий	
Регистр READ	
Сдвиг	0 🗸
Длина	1
Обработка оши	бок Сохранить посл. значен 🔻
Регистр WRITE	
Сдвиг	<b></b>
Длина	1

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

Рисунок 6.3 – Настройка опроса канала по триггеру

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

Modbus_Slave_COM_Port X							
Общее	Найти переменную		Фильтр Показ	ать все	-		
Канал Modbus Slave	Переменная	Соотнесение	Канал	Адрес	Тип	Единица	Описание
Randon House as State	Application.PLC_PRG.xTrigger	€7	Channel 0	<del>%QX0.0</del>	BIT		Триггерная переменная
Modbus Slave Init	🖹 - 🍫		Channel 0	%IW0	ARRAY [00] OF WORD		Read Holding Registers
	. ▲ Application.PLC_PRG.wVar	°¢	Channel 0[0]	<del>%IW0</del>	WORD		0x0000
ModbusGenericSerialSlave Соотнесение входов/выходов Состояние							
Информация							

Рисунок 6.4 – Привязка триггерной переменной

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

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

Modbus_Serial_Device 🗙		
Modbus Serial Device	ID:	1
Modbus Serial Device Соотнесение входов/выходов	📄 Сторожевой таймер:	500
Информация	Регистр хранения (%IW):	10
	Входные регистры (%QW):	10

Рисунок 6.5 – Настройки компонента Modbus Serial Device

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

Нет, стандартные средства конфигурирования **CODESYS** в явном виде запрещают запись значений в **holding** регистры из программы контроллера. Если это необходимо, то следует использовать ФБ <u>MB_SerialSlave</u> или <u>MB_TcpSlave</u> из библиотеки <u>OwenCommunication</u>.

## 6.4.5 Как произвести диагностику обмена в программе?

См. п. 4.6. В случае использования шаблонов модулей Мх110 и 210 см. п. 3.4.

## 6.4.6 Как расшифровываются пиктограммы статуса обмена?

Расшифровка пиктограмм статуса обмена приведена в таблице 6.1:

## Таблица 6.1 – Расшифровка пиктограмм статуса обмена

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

## 6.5 Вопросы по библиотеке OwenCommunication

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

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



Рисунок 6.6 – Добавление действия в проект CODESYS

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

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

## Приложение А. Рекомендуемые версии компонентов Modbus

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

Система исполнения	3.5.14.3	3.5.11.5	3.5.4.3
Среда программирования	V3.5 SP14 Patch 3	V3.5 SP11 Patch 5 HF 4	V3.5 SP5 Patch 5
Modbus COM	3.5.11.20	3.5.11.20	3.4.0.0
Modbus RTU Master	3.5.14.0	3.5.11.20	3.5.5.0
Modbus RTU Slave	3.5.10.30	3.5.10.30	3.5.4.0
Шаблоны Мх110	3.5.11.x	3.5.11.x	3.5.4.13
Modbus RTU Serial Device	3.5.14.0	3.5.11.0	3.5.5.0
Ethernet-адаптер	3.5.14.0	3.5.11.0	3.4.2.0
Modbus TCP Master	3.5.14.0	3.5.11.30	3.5.5.0
Modbus TCP Slave	3.5.12.0	3.5.11.30	3.5.4.0
Шаблоны Мх210	3.5.11.x	3.5.11.x	-
Modbus TCP Device	3.5.14.0	3.5.11.0	3.5.2.0

## Таблица А.1 – Рекомендуемые версии компонентов Modbus

## Приложение Б. Листинги примеров

## Б1 Листинг примера из п. 5.9.1

## Б.1.1 Код программы PLC_PRG_ST

## PROGRAM PLC_PRG_ST

## VAR

fbComControl1: OCL.COM_Control; //	^и ФБ управления портом COM1
fbComControl2: OCL.COM_Control; //	ФБ управления портом СОМ2
fbMV110_8A_Al1: OCL.MB_SerialReque	est; // ФБ опроса модуля MB110-8A
fbMV110_16D_DI: OCL.MB_SerialReque	est; // ФБ опроса модуля МВ110-16Д
fbMU110_16R_DO: OCL.MB_SerialReque	est; // ФБ опроса модуля МУ110-16Р

rAI1:	REAL;	// значение 1-го входа модуля MB110-8A
wDiMask:	WORD;	// битовая маска входов модуля MB110-16Д
wDoMask:	WORD;	// битовая маска выходов модуля МУ110-16Р
xDi0:	BOOL;	// значение 1-го входа модуля MB110-16Д
xDo0:	BOOL;	// значение 1-го выхода модуля МУ110-16Р
awAl1:	ARRAY [01] OF WORD;	// регистры, считанные с модуля MB110-8A
iStateCom1:	INT;	// шаг опроса по порту СОМ1
iStateCom2:	INT;	// шаг опроса по порту СОМ2
END_VAR		

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

COM1(); COM2();

xDo0 := xDi0 AND (rAl1 > 30.0);

## Б.1.2 Код действия СОМ1

#### CASE iStateCom1 OF

0:

```
1: // опрос МВ110-8А
```

```
fbMV110_8A_AI1
```

#### (

xExecute	:= fbComControl1.xActive,
tTimeout	:= T#200MS,
usiRetry	:= 3,
hCom	:= fbComControl1.hCom,
xIsAsciiMode	:= FALSE ,
usiSlaveId	:= 1,
eFuncCode	:= OCL.MB_FC.READ_HOLDING_REGISTERS,
uiDataAddr	:= 4,
uiDataCount	:= 2,
pData	:= ADR(awAI1),
szSize	:= SIZEOF(awAl1)

```
IF fbMV110_8A_AI1.xDone OR fbMV110_8A_AI1.xError THEN

// после выполнения блока его надо сбросить

fbMV110_8A_AI1(xExecute := FALSE);

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

iStateCom1 := 2;
```

## END_IF

);

## 2: // здесь можно добавить опрос следующего устройства

```
// после опроса последнего устройства возвращаемся к опросу первого
iStateCom1 := 1;
```

## Б.1.3 Код действия СОМ2

#### CASE iStateCom2 OF

0: // открытие СОМ-порта СОМ2

```
IF fbComControl2.xDone THEN
iStateCom2 := 1;
END_IF
```

## 1: // опрос модуля МВ110-16Д

fbMV110_16D_DI

#### (

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

IF fbMV110_16D_DI.xDone OR fbMV110_16D_DI.xError THEN // после выполнения блока его надо сбросить fbMV110_16D_DI(xExecute := FALSE);

xDi0 := wDiMask.0;

iStateCom2 := 2;

END_IF

);

## 2: // опрос модуля МУ110-16Р

wDoMask.0 := xDo0;

```
fbMU110_16R_DO
```

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

);

(

## IF fbMU110_16R_DO.xDone OR fbMU110_16R_DO.xError THEN

// после выполнения блока его надо сбросить fbMU110_16R_DO(xExecute := FALSE);

```
// возвращаемся к опросу первого модуля
iStateCom2 := 1;
```

## END_IF

END_CASE

## Б2 Листинг примера из п. 5.9.2

```
PROGRAM PLC_PRG_ST
VAR
 fbComControl1: OCL.COM_Control;
 // ФБ управления портом COM1
 fbRealToWord2: OCL.REAL_TO_WORD2;
 // ФБ преобразования REAL в две...
 // ...переменные типа WORD
 fbModbusSerialSlave:
 OCL.MB_SerialSlave;
 // ФБ для реализиации Modbus Slave
 awSlaveData:
 ARRAY [0..15] OF WORD; // буфер данных Modbus Slave
 (* значения, полученные от ОРС *)
 xVar_Opc:
 BOOL;
 wVar Opc:
 WORD;
 rVar_Opc:
 REAL;
 sVar_Opc:
 STRING(15);
 (* значения для передачи в ОРС *)
 xVar Plc:
 BOOL;
 wVar_Plc:
 WORD;
 rVar_Plc:
 REAL;
 sVar_Plc:
 STRING(15);
 BOOL; // команда записи данных из программы в регистры Modbus Slave
 xWrite:
 R TRIG; // триггер для однократной записи
 fbWriteEdge:
END_VAR
```

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

```
fbComControl1
 xEnable := TRUE,
 usiComPort := 1,
 udiBaudrate := 115200,
 udiByteSize := 8,
 eParity := OCL.COM PARITY.NONE,
 eStopBit := OCL.COM_STOPBIT.ONE
```

);

(

fbModbusSerialSlave

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

);

(

#### // данные, полученные от ОРС

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

## // по команде записываем переменные из программы в регистры Modbus Slave fbWriteEdge(CLK := xWrite);

#### IF fbWriteEdge.Q THEN

```
awSlaveData[0].0 := xVar_Plc;
awSlaveData[1] := wVar_Plc;
fbRealToWord2(rInput := rVar_Plc, wOutput1 => awSlaveData[2], wOutput2 => awSlaveData[3]);
OCL.SWAP_DATA(ADR(sVar_Plc), ADR(awSlaveData[4]), SIZEOF(sVar_Plc), TRUE, FALSE, FALSE);
```

END_IF

## БЗ Листинг примера из п. 5.9.3

## Б.3.1 Код программы PLC_PRG_ST

```
PROGRAM PLC_PRG_ST
```

#### VAR

fbTcpClientMV210:	OCL.TCP_Client; // ФБ ТСР-подключения к модулю MB210-101		
fbTcpClientMK210:	OCL.TCP_Client; // ФБ ТСР-подключения к модулю МК210-301		
fbMV210_101_AI1:	OCL.MB TcpRequest; // ФБ опроса модуля MB210-101		
fbMK210_301_DI:	OCL.MB_TcpRequest;	// ФБ опроса входов модуля МК210-301	
fbMK210_301_DO:	OCL.MB_TcpRequest;	// ФБ опроса выходов модуля МК210-301	
rAl1:	REAL;	// значение 1-го входа модуля MB210-101	
wDiMask:	WORD;	// битовая маска входов модуля MK210-301	
wDoMask:	WORD;	// битовая маска выходов модуля МК210-301	
xDi0:	BOOL;	// значение 1-го входа модуля МК210-301	
xDo0:	BOOL;	// значение 1-го выхода модуля МК210-301	
awAl1:	ARRAY [01] OF WORD;	// регистры, считанные с модуля MB210-101	
iStateMV210:	INT;	// шаг опроса модуля МВ210-101	
iStateMK210:	INT;	// шаг опроса модуля МК210-301	
B			

## END_VAR

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

MV210_101(); MK210_301();

xDo0 := xDi0 AND (rAl1 > 30.0);

## Б.3.2 Код действия MV210_101

#### CASE iStateMV210 OF

```
0:
 // подключение к модулю
 fbTcpClientMV210
 (
 xEnable := TRUE,
 tTimeout
 := T#5S,
 sIpAddr
 := '10.2.11.181',
 uiPort
 := 502
);
 IF fbTcpClientMV210.xActive THEN
 iStateMV210 := 1;
 END_IF
```

## 1: // опрос МВ210-101

fbMV210_101_AI1

#### (

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

);

IF fbMV210_101_Al1.xDone OR fbMV210_101_Al1.xError THEN // после выполнения блока его надо сбросить fbMV210_101_AI1(xExecute := FALSE);

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

```
iStateMV210 := 2;
```

## END_IF

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

## Б.3.3 Код действия МК210_301

## CASE iStateMK210 OF

0: // подключение к модулю

```
END_IF
```

1: // опрос дискретных входов модуля МК210-301

fbMK210_301_DI

#### (

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

);

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

```
xDi0 := wDiMask.0;
```

iStateMK210 := 2;

END_IF

2: // запись дискретных выходов модуля МК210-301

```
wDoMask.0 := xDo0;
fbMK210_301_DO
(
 xExecute
 := fbTcpClientMK210.xActive,
 tTimeout
 := T#50MS,
 usiRetry
 := 3,
 := fbTcpClientMK210.hConnection,
 hConnection
 xlsRtuOverTcpMode
 := FALSE ,
 usiUnitId
 := 1,
 eFuncCode
 := OCL.MB_FC.WRITE_MULTIPLE_REGISTERS,
 uiDataAddr
 := 470,
 uiDataCount
 := 1,
 pData
 := ADR(wDoMask),
 szSize
 := SIZEOF(wDoMask)
);
IF fbMK210_301_DO.xDone OR fbMK210_301_DO.xError THEN
 // после выполнения блока его надо сбросить
 fbMK210_301_DO(xExecute := FALSE);
 // возвращаемся к первому запросу
 iStateMK210 := 1;
END_IF
```

END_CASE

## Б4 Листинг примера из п. 5.9.4

PROGRAM PLC_PRG_ST

VAR

fbRealToWord2: OCL.REAL_TO_WORD2;

// ФБ преобразования REAL в две... // ...переменные типа WORD

fbModbusTcpSlave:	OCL.MB_TcpSlave;	// ФБ для реализиации Modbus Slave
awSlaveData:	ARRAY [015] OF WORD;	// буфер данных Modbus Slave

## (* значения, полученные от ОРС *)

xVar_Opc:	BOOL;
wVar_Opc:	WORD;
rVar_Opc:	REAL;
sVar_Opc:	STRING(15);
(* значения для передачи в ОРС *)	
xVar_Plc:	BOOL;
wVar_Plc:	WORD;
rVar Plc:	REAL;

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

STRING(15);

END_VAR

sVar_Plc:

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

#### fbModbusTcpSlave

(

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

);

## // данные, полученные от ОРС

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

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

fbWriteEdge(CLK := xWrite);

## IF fbWriteEdge.Q THEN

awSlaveData[0].0 := xVar_Plc; awSlaveData[1] := wVar_Plc; fbRealToWord2(rInput := rVar_Plc, wOutput1 => awSlaveData[2], wOutput2 => awSlaveData[3]); OCL.SWAP_DATA( ADR(sVar_Plc), ADR(awSlaveData[4]), SIZEOF(sVar_Plc), TRUE, FALSE, FALSE );

END_IF

## Б5 Листинг примера из п. 5.9.5

## PROGRAM PLC_PRG_ST

## VAR

fbModbusTcpSlave:	OCL.MB_TcpSlave;	// ФБ для реализации Modbus Slave
awSlaveData:	ARRAY [015] OF WORD; //	буфер данных Modbus Slave

#### // пути к файлам архивов

asFilePath: ARRAY [1..8] OF STRING := ['/mnt/ufs/home/root/Arc1.log', 7('')]; END_VAR

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

#### fbModbusTcpSlave

(

xEnable	:= TRUE,
sIpAddr	:= '0.0.0.0',
uiPort	:= 502,
usiUnitId	:= 1,
pData	:= ADR(awSlaveData),
szSize	:= SIZEOF(awSlaveData),
pasFilePath	:= ADR(asFilePath),
c_xReverseByteOrderFiles	:= TRUE,
c_usiAmountBytes	:= 10

);