Modbus Universal MasterOPC сервер

Подключение контроллеров ОВЕН серии ПЛК1xx по протоколу Modbus TCP

Руководство пользователя

ОГЛАВЛЕНИЕ

1 Введение

Разработчикам систем управления требуется подключать к SCADA системам различные устройства – модули ввода, регуляторы, программируемые контроллеры. Наиболее распространенным протоколом обмена в промышленности является протокол ***Modbus***. Ранее нами была выпущена документация по подключению контроллеров ***ABB AC500***, ***Siemens S7-1200*** и ***Delta Electronics*** к нашему ***Modbus Universal MasterOPC*** серверу. В данной статье мы рассмотрим подключение контроллеров фирмы ***ОВЕН*** серии ***ПЛК1xx***.

2 Описание контроллеров ОВЕН серии ПЛК1xx

Контроллеры ОВЕН предназначены для построения систем управления котельными установками, объектов водоснабжения, систем вентиляции и кондиционирования, а также различного промышленного оборудования.

Программирование контроллеров осуществляется в среде разработки ***Codesys v2.3***. Контроллеры имеют несколько встроенных сетевых интерфейсов – ***RS-232***, ***RS-485***, ***Ethernet***, а также имеют поддержку работы с модемами. Контроллеры поддерживают несколько протоколов – ***Codesys Gateway*** (для связи со средой разработки), протокол ***ОВЕН***, а также поддерживают протокол ***Modbus*** – версий ***Modbus RTU***, ***ASCII*** и ***TCP***, как в режиме *Master* (ведущий), так и в режиме *Slave* (ведомый).

3 Настройка контроллера и OPC сервера на протокол Modbus RTU

3.1 Настройка контроллера

В качестве примера мы подключим *контроллер* ***ОВЕН ПЛК100***, к ***Modbus Universal MasterOPC*** серверу по протоколу ***Modbus RTU***. Опустим описание создание проекта, выбора целевой платформы контроллера, написание программы и перейдем сразу к настройке ***Modbus*** протокола.

Настройка ***Modbus*** осуществляется в *окне* ***Конфигурация ПЛК***, на *закладке* ***Ресурсы*** (*Рисунок 3-1*).

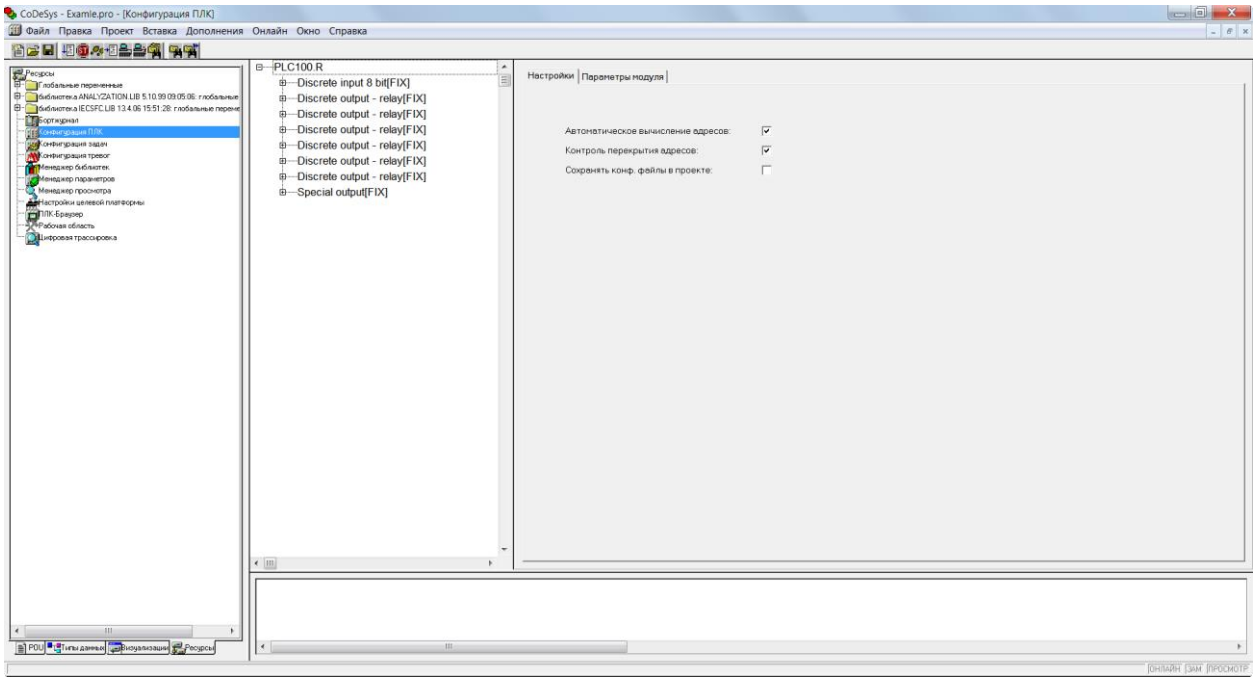

Рисунок 3-1

Сначала в контроллер, через *контекстное меню* нужно добавить *модуль* ***Modbus(Slave)***.

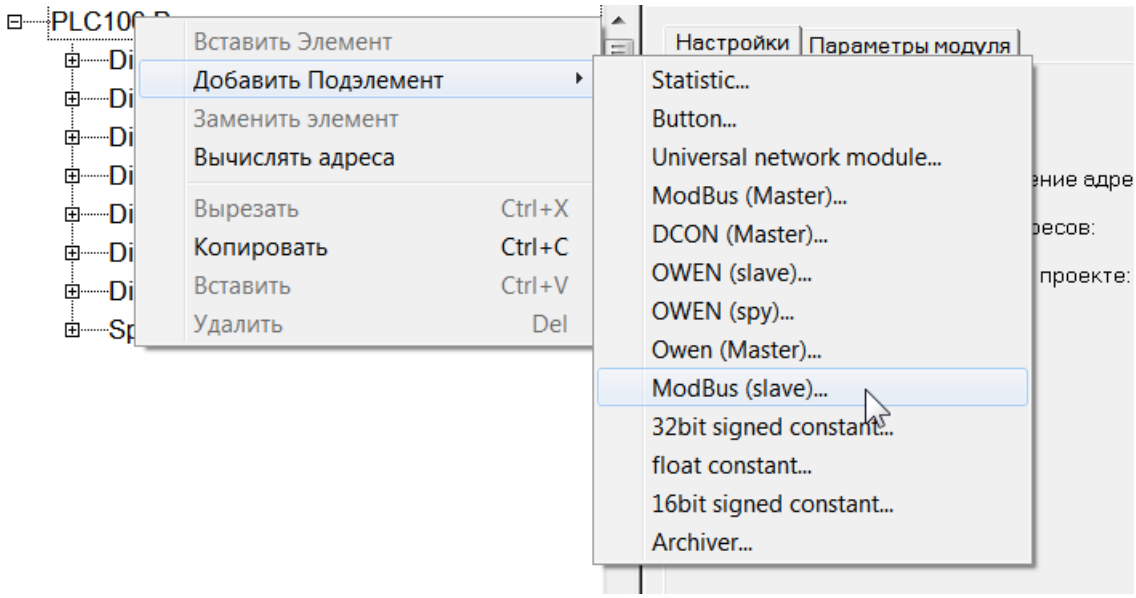

Рисунок 3-2

В дерево контроллера добавился новый *модуль*. В его дочерний элемент – ***Modbus [Fix]*** – нужно добавить *интерфейс* по которому будет происходить обмен с верхним уровнем. В нашем случае это будет порт ***RS-485*** (*Рисунок 3-3*).

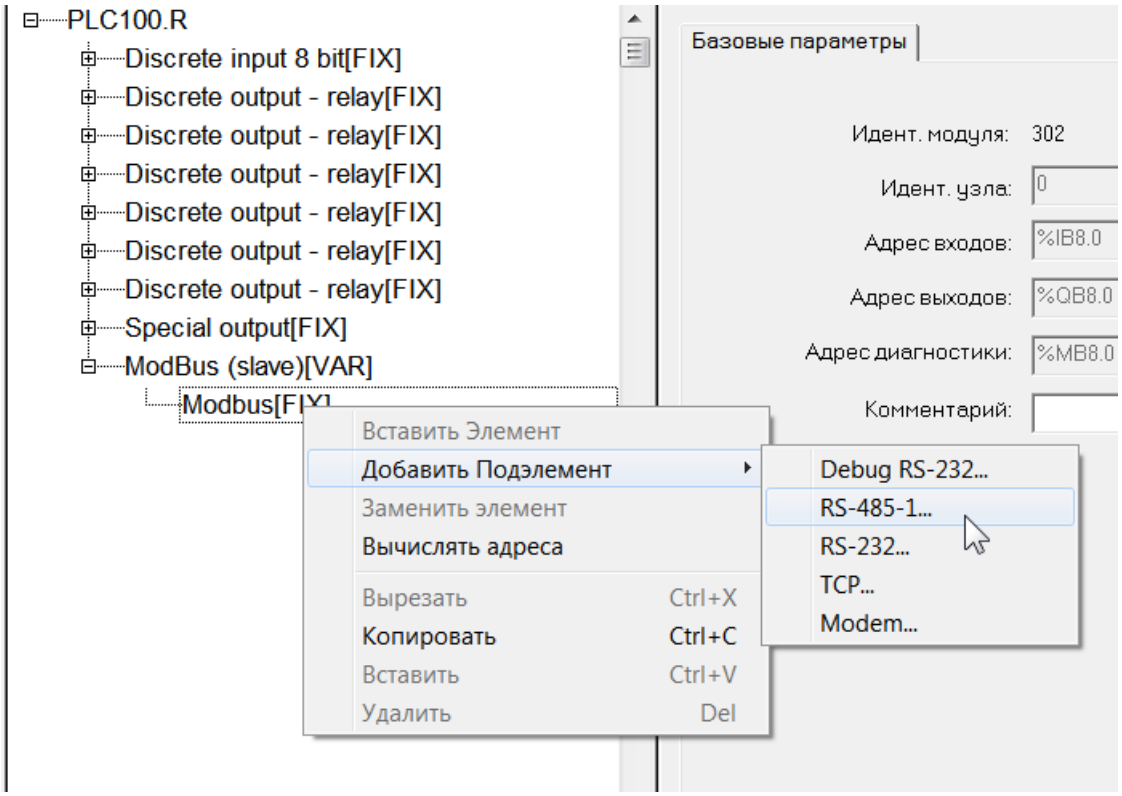

Рисунок 3-3

На *закладке* ***Параметры*** модуля можно задать *параметры обмена* – *скорость*, *четность*, *стоп-биты*, *тип протокола* (***Modbus RTU*** или ***Modbus ASCII***), *задержка ответа*. Протокол обмена установим ***Modbus RTU***, а остальные параметры оставим по умолчанию.

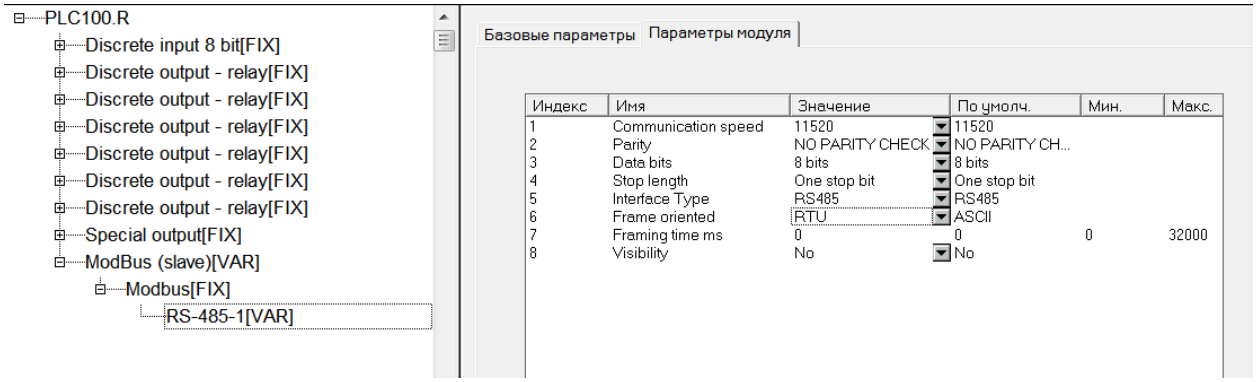

Рисунок 3-4

Теперь укажем *адрес* нашего контроллера на шине ***Modbus***. Данная настройка осуществляется на *закладке* ***Параметры модуля***, элемента ***Modbus (Slave)***. Оставим адрес стандартным – 1 (*Рисунок 3-5*)

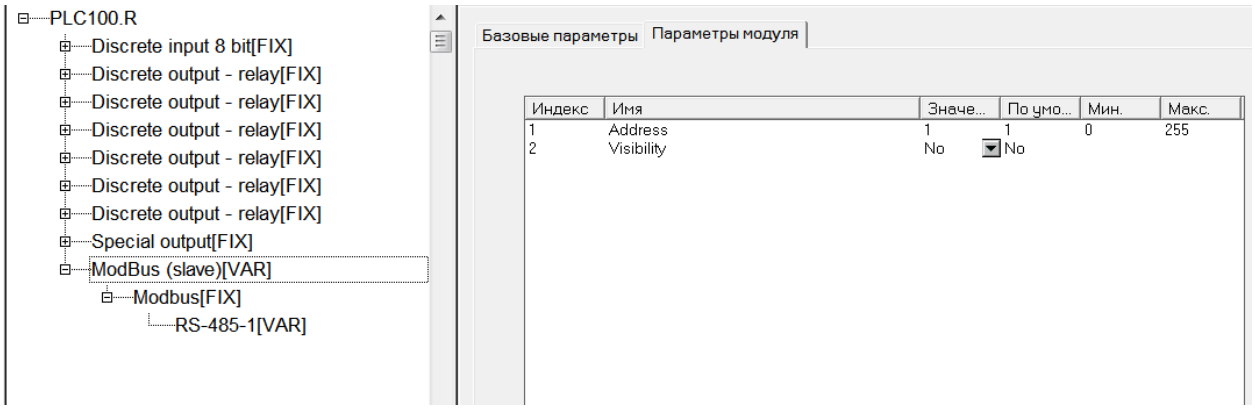

Рисунок 3-5

3.2 Настройка OPC сервера

Настроим OPC сервер на работу с нашим контроллером. Создадим новую конфигурацию OPC сервера и добавим в ***Server*** новый *узел*. *Тип узла* установим – ***COM***, укажем параметры связи такие же, как в настройках контроллера (*Рисунок 3-6*).

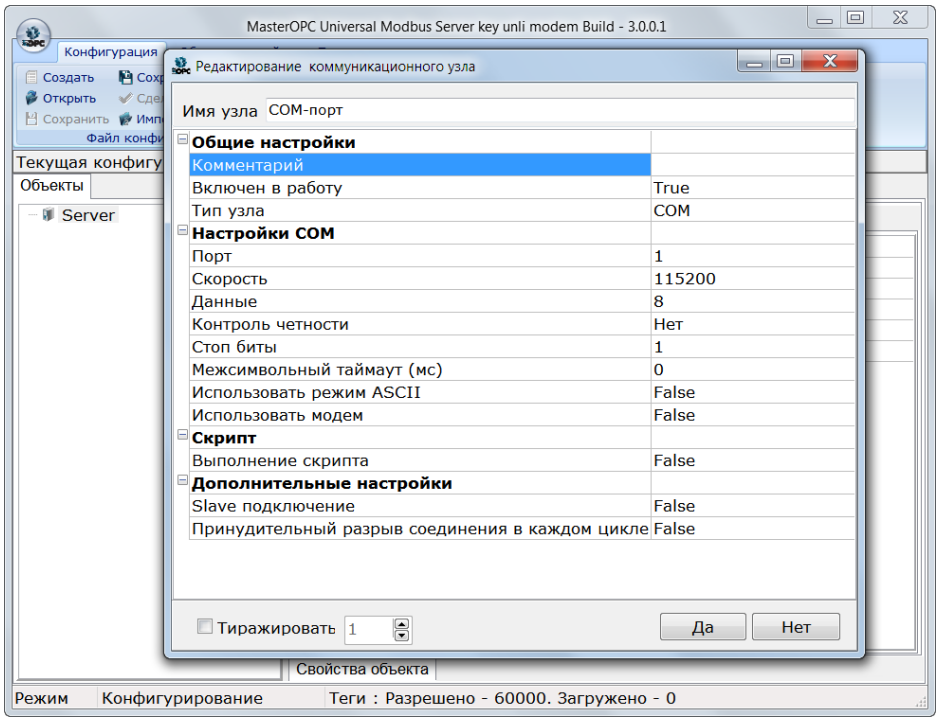

Рисунок 3-6

Добавим в узел *устройство*, через *контекстное меню узла* (*Рисунок 3-7*).

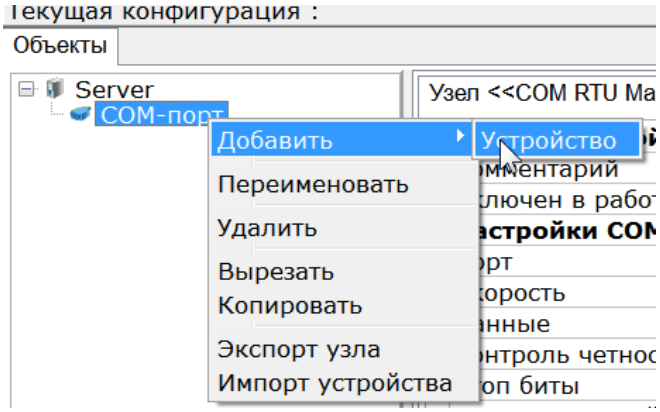

Рисунок 3-7

В окне настройки устройства дадим имя устройству, и зададим адрес – 1 (*Рисунок 3-8*). Кроме того, необходимо установить настройку ***Не использовать команду Write Single Coils*** в ***False***, так как запись битов в контроллер производится функцией 0x05 (если вы планируете работать с отдельными битами памяти).

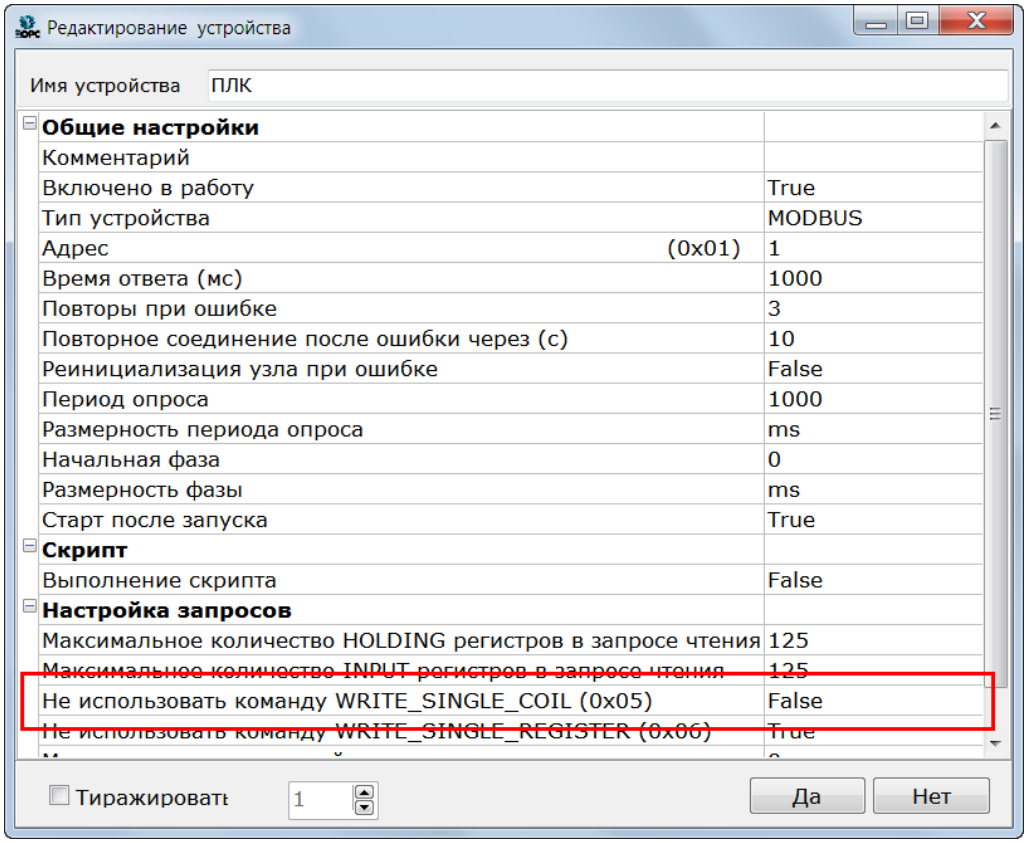

Рисунок 3-8

Устройство будет добавлено в дерево (*Рисунок 3-9*).

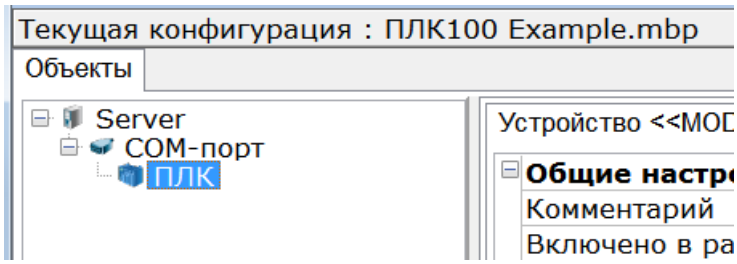

Рисунок 3-9

4 Добавление Modbus переменных

4.1 Адресация переменных в контроллере

Контроллеры ОВЕН могут передавать по ***Modbus*** переменные следующих типов – ***Byte*** (1 байт), ***Word*** (2 байта), ***DWord*** (4 байта), ***Real*** (4 байта). В контроллерах ***ОВЕН*** все *Modbus переменные* находятся в одном сегменте памяти, доступ к которой осуществляется с помощью следующих функций:

0x01 – чтение битов, 0x05 – запись бита, регион ***Coils***.

0x03 – чтение регистров, 0x10 – запись регистров, регион ***Holding Registers***.

Память контроллера можно представить в виде следующей таблицы:

Адрес контроллера	Адрес Modbus бита (регион Coils)								Адрес Modbus регистра (регион Holding Registers)
0x0000	0	1	2	3	4	5	6	7	0x0000
0x0001	8	9	10	11	12	13	14	15	
0x0002	16	17	18	19	20	21	22	23	0x0001
0x0003	24	25	26	27	28	29	30	31	
0x0004	32	33	34	35	36	37	38	39	0x0002
0x0005	40	41	42	43	44	45	46	47	
0x0006	48	49	50	51	52	53	54	55	0x0003
0x0007	56	57	58	59	60	61	62	63	

Таким образом, к байтам памяти контроллера обращаться через регион ***Holding Registers***, или обратится через конкретный бит – используя регион ***Coils***.

Кроме того, при добавлении переменных используется *выравнивание области памяти*.

Выравнивание можно описать следующими правилами:

- 1-байтовая переменная (***Byte***) может располагаться в любом адресе памяти контроллера;
- 2-байтовая переменная (***Word***) может располагаться только в четных адресах памяти контроллера;
- 4-байтовая переменная (***DWord*** и ***Real***) может располагаться только адресах памяти кратных четырем.

Таким образом возможна ситуация, когда отдельные адреса памяти контроллера не будут использоваться.

Подробнее про выравнивание памяти можно прочитать в специальной документации компании ОВЕН. Кроме того, далее, мы разберем несколько примеров добавления различных типов переменных.

4.2 Добавление переменных в контроллер и OPC сервер

Добавление Modbus переменных осуществляется через *контекстное меню модуля* ***Modbus (Slave)*** – *Добавить Подэлемент*.

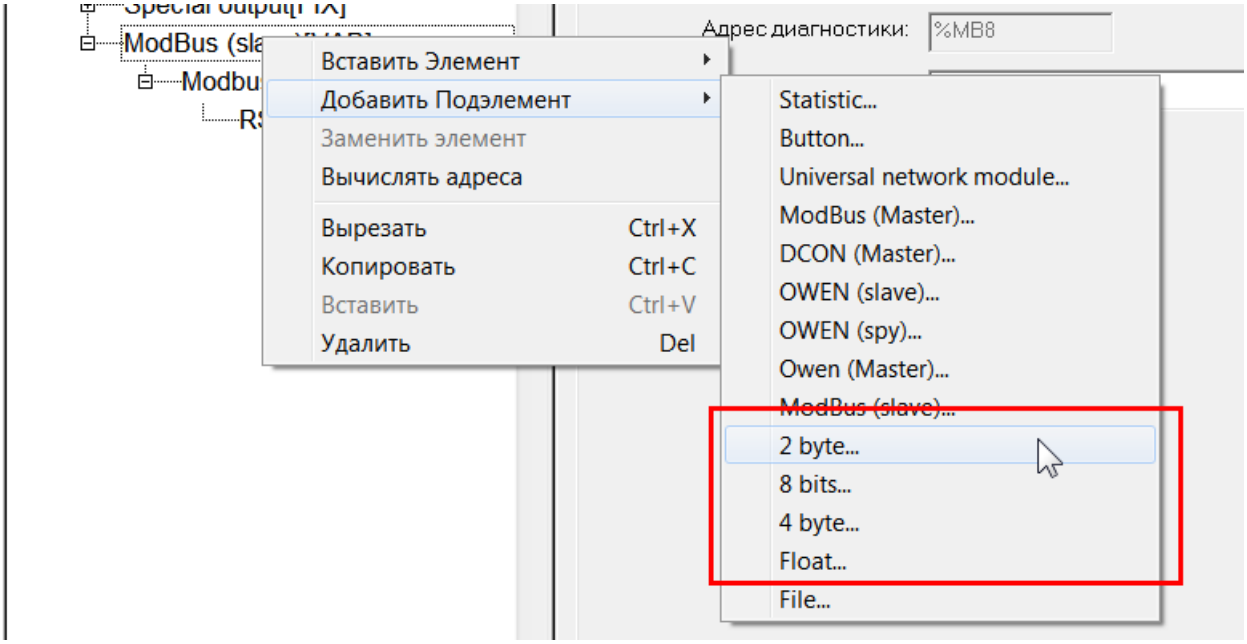

Рисунок 4-1

К *Modbus переменным* относятся элементы – ***2 byte*** (тип ***Word***), ***8 bits*** (тип ***Byte***), ***4 byte*** (тип ***DWord***), ***Float*** (тип ***Real***) (*Рисунок 4-1*).

4.2.1 Добавление переменных типа Byte

Добавим нескольких различных ***Modbus*** переменных. Сначала добавим переменную ***8 bits*** (*Рисунок 4-2*).

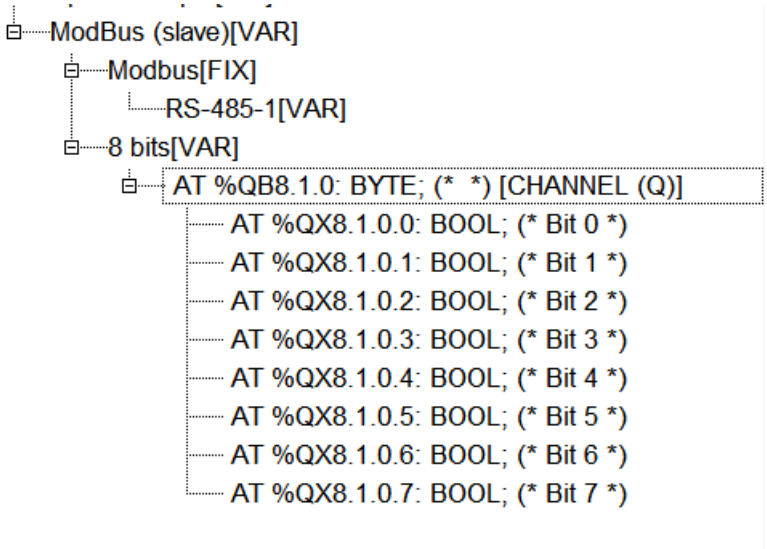

Рисунок 4-2

Данная переменная имеет тип ***Byte***, размер переменной составляет 1 байт, при этом к каждому биту переменной можно обратится через переменную, поэтому, как правило, данную переменную используют для передачи дискретных значений. Присвоим первым двум битам переменные, назовем их ***Discrete1*** и ***Discrete2*** (*Рисунок 4-3*).

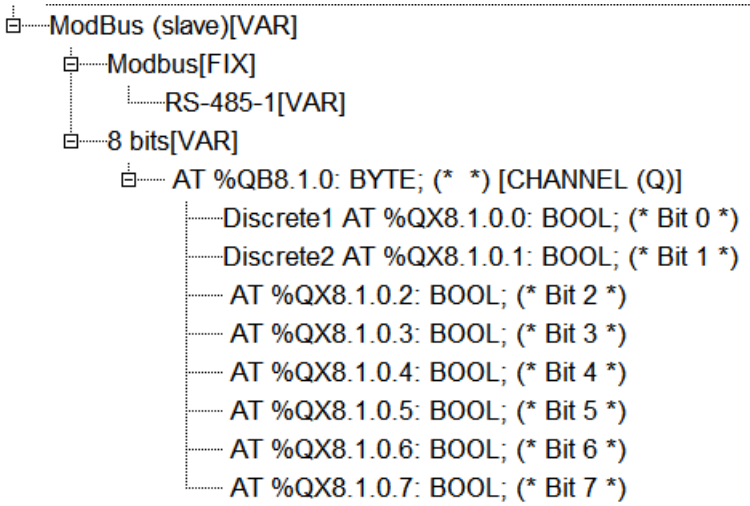

Рисунок 4-3

Теперь добавим такие же переменные в OPC сервер. Для этого через контекстное меню устройства добавим тег. Для удобства зададим ему такое же имя в ***Codesys*** – ***Discrete1***.

Регион тега будет ***Coils***. *Адрес* тега будет равен ***0***. Остальные параметры можно оставить стандартными (*Рисунок 4-4*).

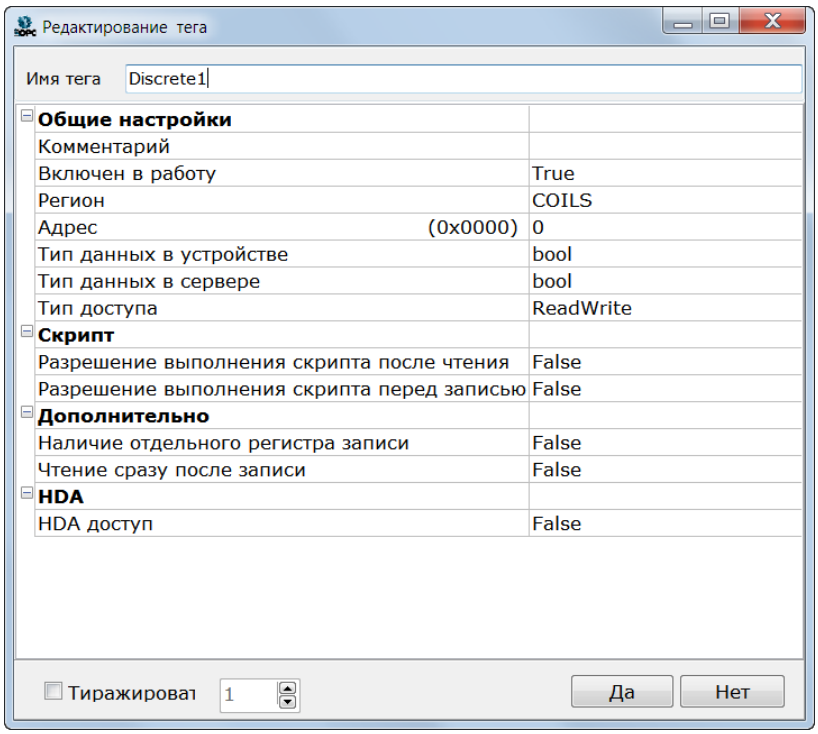

Рисунок 4-4

Аналогично добавим второй тег, дадим ему имя ***Discrete2***, адрес укажем ***1*** (*Рисунок 4-5*).

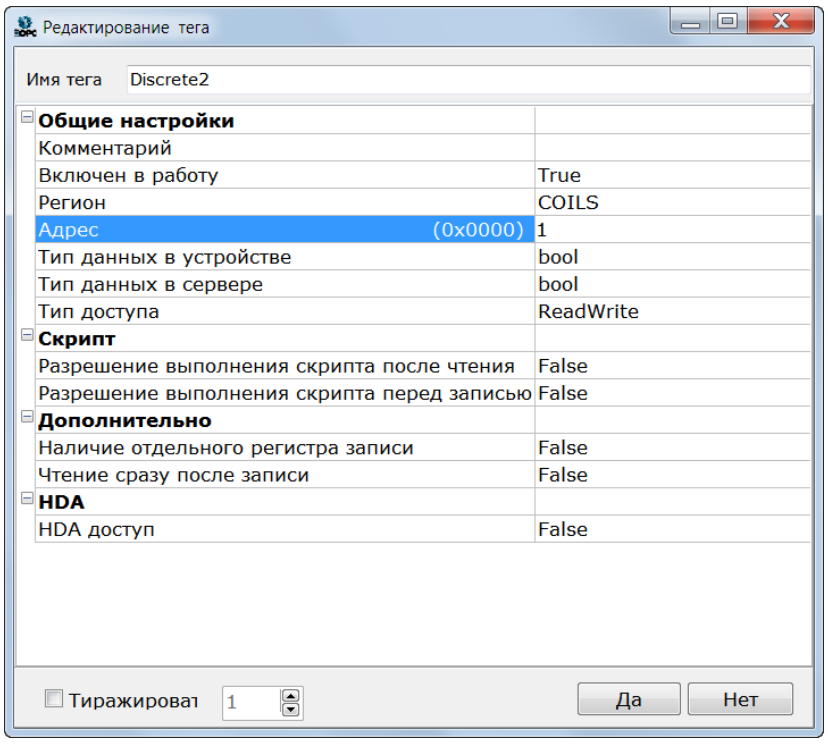

Рисунок 4-5

Проверим получение данных. Подключимся к контроллеру, запишем в него программу, а также запустим OPC сервер в режим исполнения.

Из среды разработки изменим состояние одного из битов переменной – изменение отобразилось в OPC сервере (*Рисунок 4-6*).

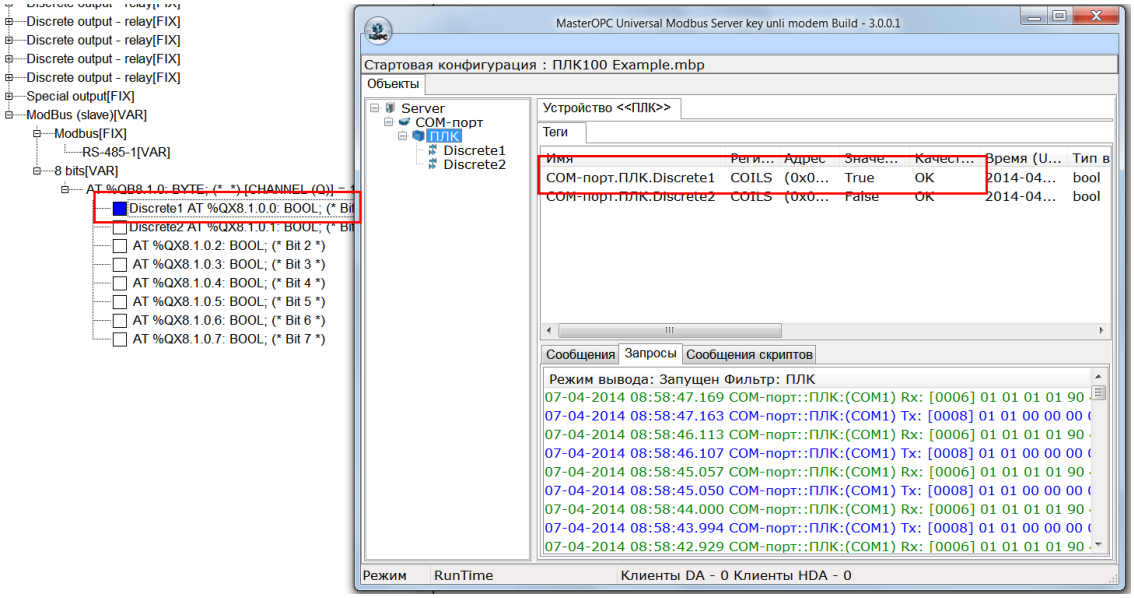

Рисунок 4-6

Отключимся от контроллера, и остановим режим исполнения OPC сервера.

4.2.2 *Добавление переменных типа Word (uint16)*

Теперь добавим в контроллер две целочисленных переменных типа ***Word*** – элемент ***2 Byte*** (*Рисунок 4-7*).

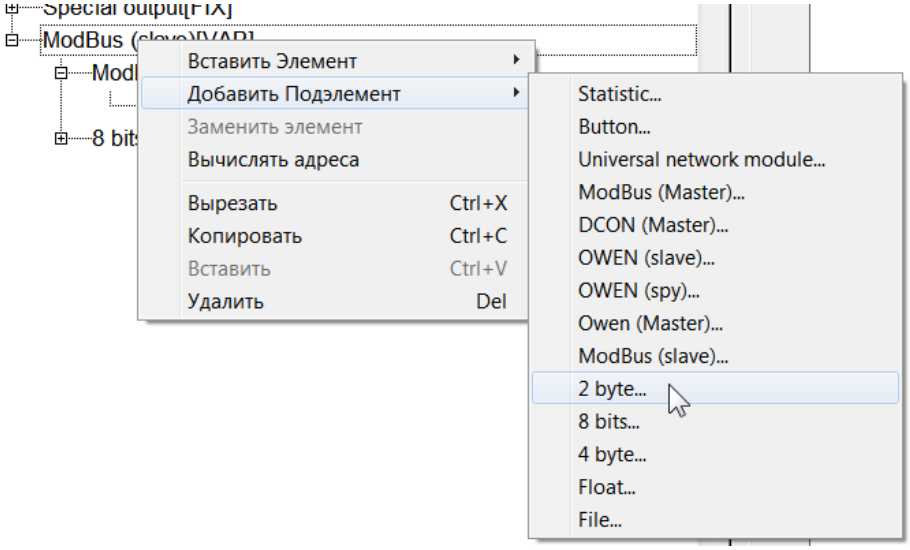

Рисунок 4-7

Дадим имена переменным, назовем из ***VarWord1*** и ***VarWord2*** (*Рисунок 4-8*)

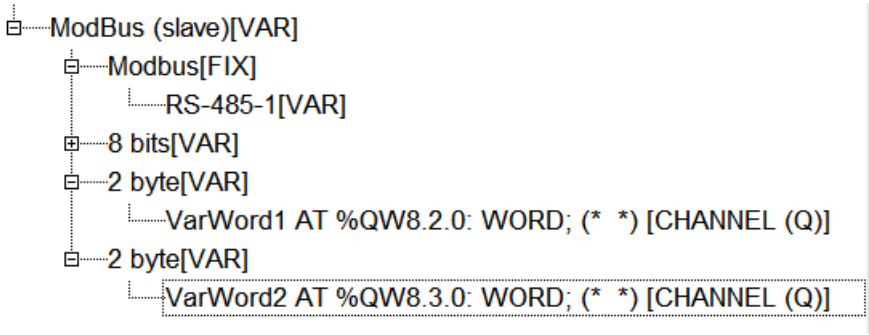

Рисунок 4-8

Добавим теги в OPC сервер. *Регион* тега будет использоваться ***Holding Registers***. Нулевой адрес памяти контроллера уже занят байтовой переменной, а значит и занят весь нулевой Modbus адрес. Поэтому у переменной ***VarWord1*** адрес Modbus регистра будет равен ***1***.

Ситуацию можно проиллюстрировать с помощью таблицы:

<table>
<tr><th>Адрес контроллера</th><th colspan="8">Расположение переменных</th><th>Адрес Modbus регистра (регион Holding Registers)</th></tr>
<tr><td>0x0000</td><td>Discrete1 (бит0)</td><td>Discrete2 (бит1)</td><td></td><td></td><td></td><td></td><td></td><td></td><td rowspan="2">0x0000</td></tr>
<tr><td>0x0001</td><td colspan="8">Незанятое пространство</td></tr>
<tr><td>0x0002</td><td colspan="8" rowspan="2">VarWord1</td><td rowspan="2">0x0001</td></tr>
<tr><td>0x0003</td></tr>
<tr><td>0x0004</td><td colspan="8" rowspan="2">VarWord2</td><td rowspan="2">0x0002</td></tr>
<tr><td>0x0005</td></tr>
</table>

Добавим тег, имя также дадим ***VarWord1***, *регион* – ***Holding Registers***, *адрес* – ***1***, *тип данных в устройстве* – ***uint16*** (соответствует типу ***Word***), *тип данных в сервере* – ***uint32***. Остальные параметры можно оставить по умолчанию (*Рисунок 4-9*).

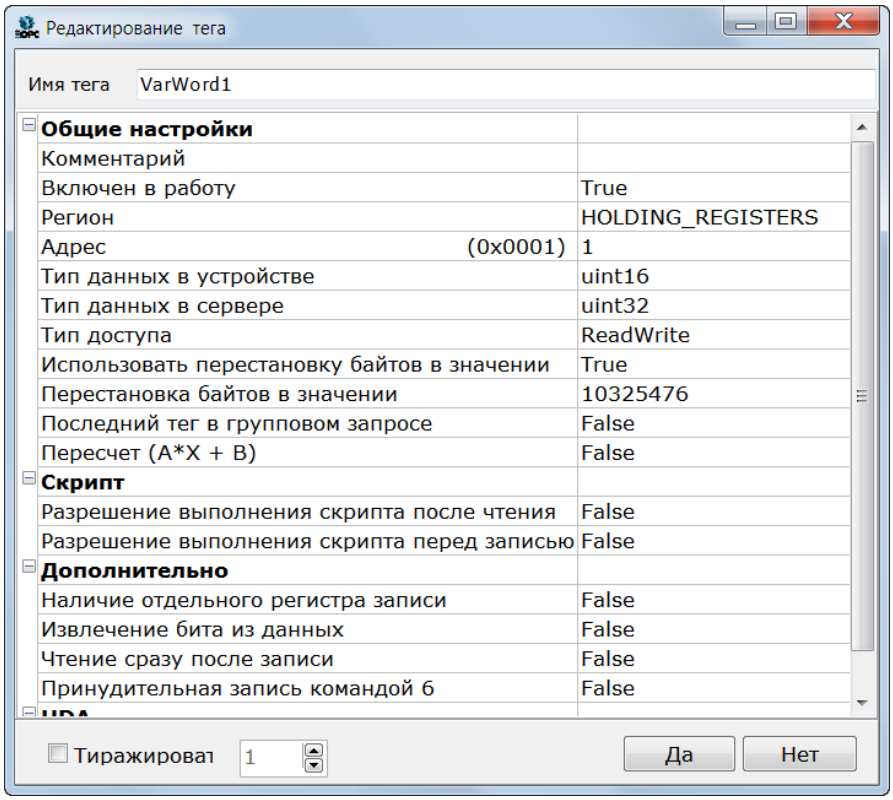

Рисунок 4-9

Аналогично добавим второй тег, его *адрес* Modbus регистра будет равен ***2*** (*Рисунок 4-10*).

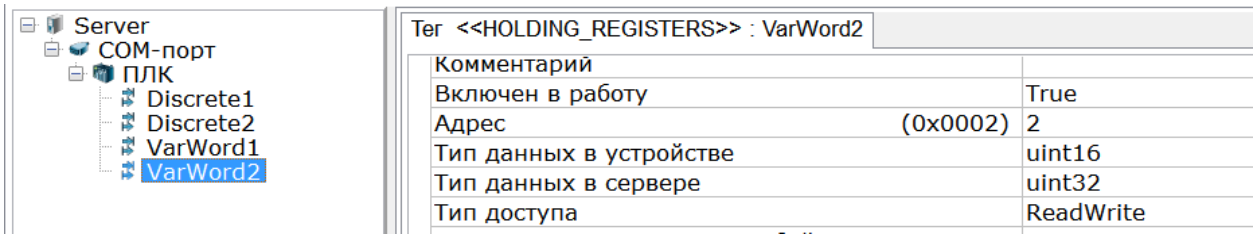

Рисунок 4-10

Проверим получение данных – подключимся к контроллеру и обновим программу, а также запустим OPC сервер в режим исполнения.

Изменим в контроллере одно из значений – значение отобразилось в OPC сервере (*Рисунок 4-11*).

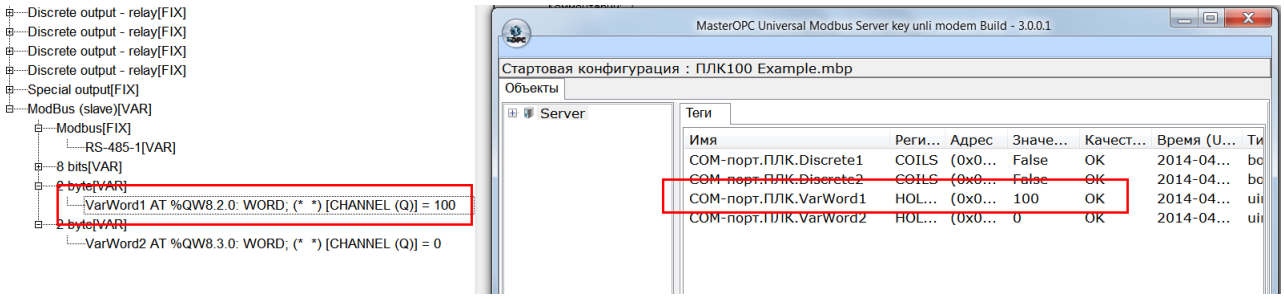

Рисунок 4-11

4.2.3 Добавление переменной типа Real (Float)

Добавим 4-байтовую переменную – переменную типа ***Real***. Добавим *подэлемент* ***Float*** (*Рисунок 4-12*).

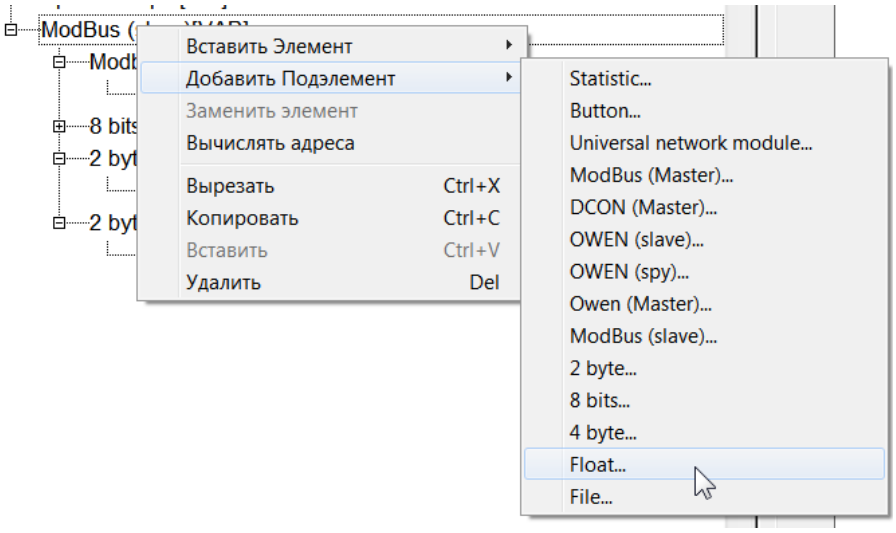

Рисунок 4-12

Дадим ей имя ***VarFloat1*** (*Рисунок 4-13*).

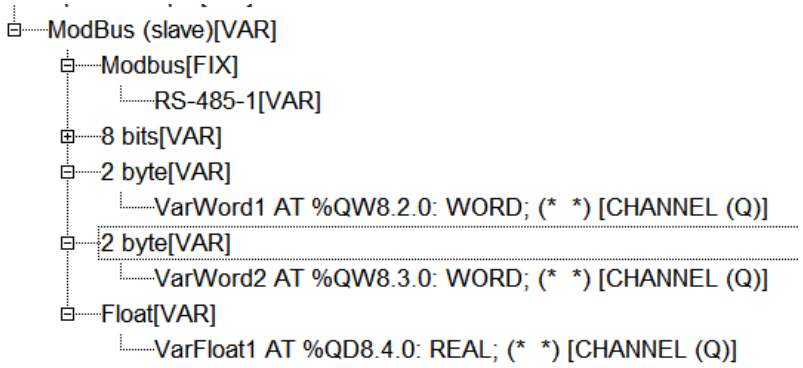

Рисунок 4-13

Определим адрес этой Modbus переменной. Последний использованный адрес памяти - ***0x0005***, занят переменной ***VarWord2***. Следующий за ним адрес – ***0x0006***. Однако согласно

правилам выравнивания, 4 байтовые переменные (которой является переменная ***Float***) могут располагаться только в адресах памяти кратных четырем. Значит переменная будет находится в следующем ближайшем адресе, который будет делиться на 4, то есть - ***0x0008***.

Ситуацию можно проиллюстрировать на таблице:

<table>
<tr><th>Адрес контроллера</th><th colspan="8">Расположение переменных</th><th>Адрес Modbus регистра (регион Holding Registers)</th></tr>
<tr><td>0x0000</td><td>Discrete1 (бит0)</td><td>Discrete2 (бит1)</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0x0000</td></tr>
<tr><td>0x0001</td><td colspan="8">Незанятое пространство</td></tr>
<tr><td>0x0002</td><td colspan="8" rowspan="2">VarWord1</td><td rowspan="2">0x0001</td></tr>
<tr><td>0x0003</td></tr>
<tr><td>0x0004</td><td colspan="8" rowspan="2">VarWord2</td><td rowspan="2">0x0002</td></tr>
<tr><td>0x0005</td></tr>
<tr><td>0x0006</td><td colspan="8" rowspan="2">Незанятое пространство</td><td rowspan="2">0x0003</td></tr>
<tr><td>0x0007</td></tr>
<tr><td>0x0008</td><td colspan="8" rowspan="4">VarFloat1</td><td rowspan="2">0x0004</td></tr>
<tr><td>0x0009</td></tr>
<tr><td>0x000A</td><td rowspan="2">0x0005</td></tr>
<tr><td>0x000B</td></tr>
</table>

Таким образом *адрес* ***0x0003*** останется не использованным, а переменная ***VarFloat*** будет занимать ***Modbus*** адреса ***0x0004*** и ***0x0005***.

Добавим в OPC сервер *тег*. Зададим ему имя – ***VarFloat1***, адрес – ***4***, *тип в устройстве* – ***Float***, *тип в сервере* – ***Float***. Также нужно указать правильное *чередование байт.* Обычно для четырехбайтовых переменных чередование устанавливается в режим ***Старшим словом вперед (32107654)***, однако в контроллерах ОВЕН ПЛК1хх чередование байт у четырехбайтовых переменных такое же, как и двухбайтовых – ***Старшим байтом вперед (10325476).*** (*Рисунок 4-14*)

***Примечание**. Данная особенность свойственна только контроллерам **ОВЕН ПЛК1xx**. Остальные приборы данного производителя (регуляторы, модули ввода-вывода) для четырехбайтных переменных используют чередование байт **«Старшим словом вперед»**.*

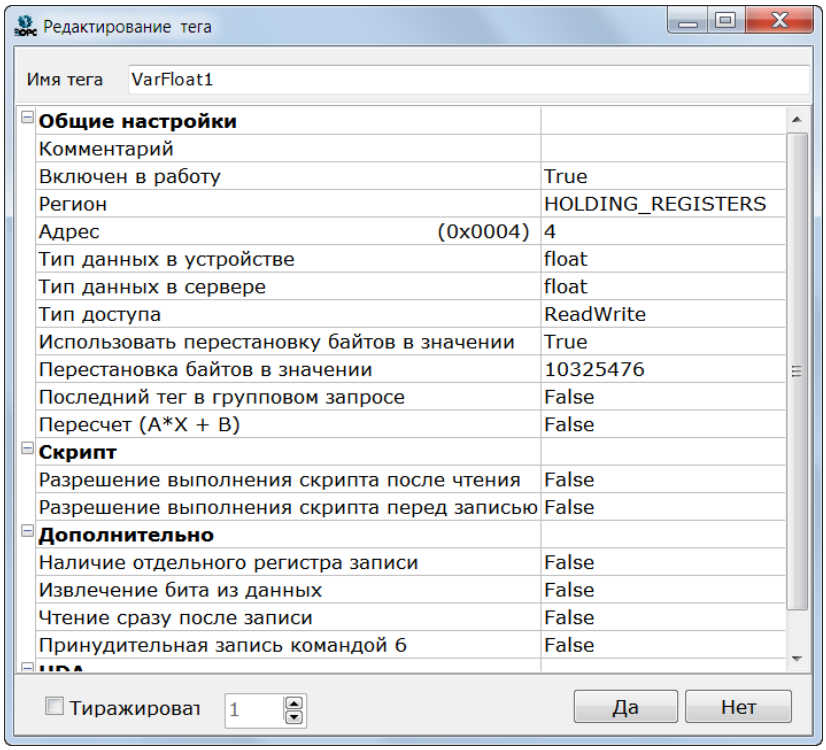

Рисунок 4-14

Тег добавится в устройство (*Рисунок 4-15*)

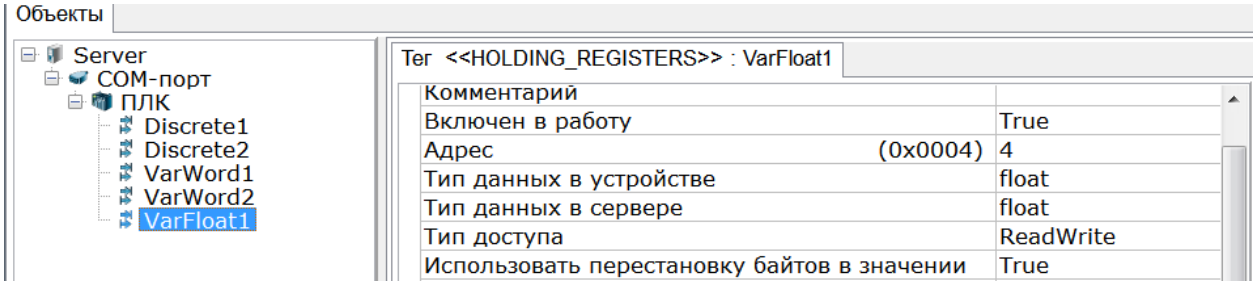

Рисунок 4-15

Аналогично проверим получение данных с контроллера (*Рисунок 4-16*).

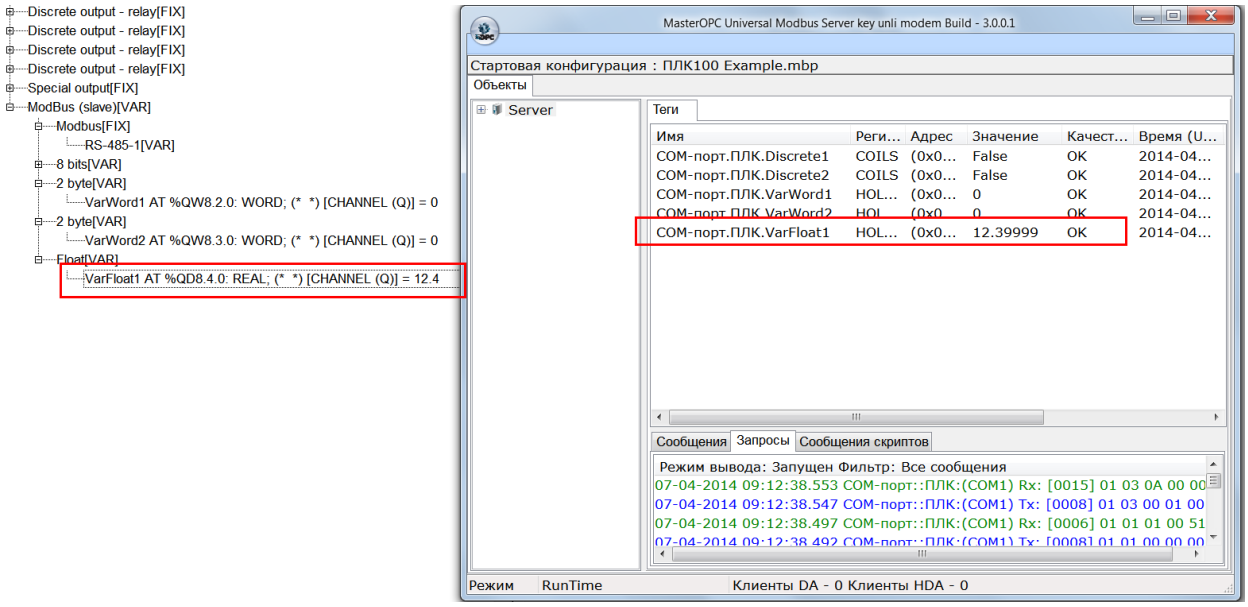

Рисунок 4-16

4.2.4 *Добавление переменной типа DWord*

Переменная типа ***DWord*** добавляется через *подэлемент* ***4 Byte*** (*Рисунок 4-17 и Рисунок 4-18*).

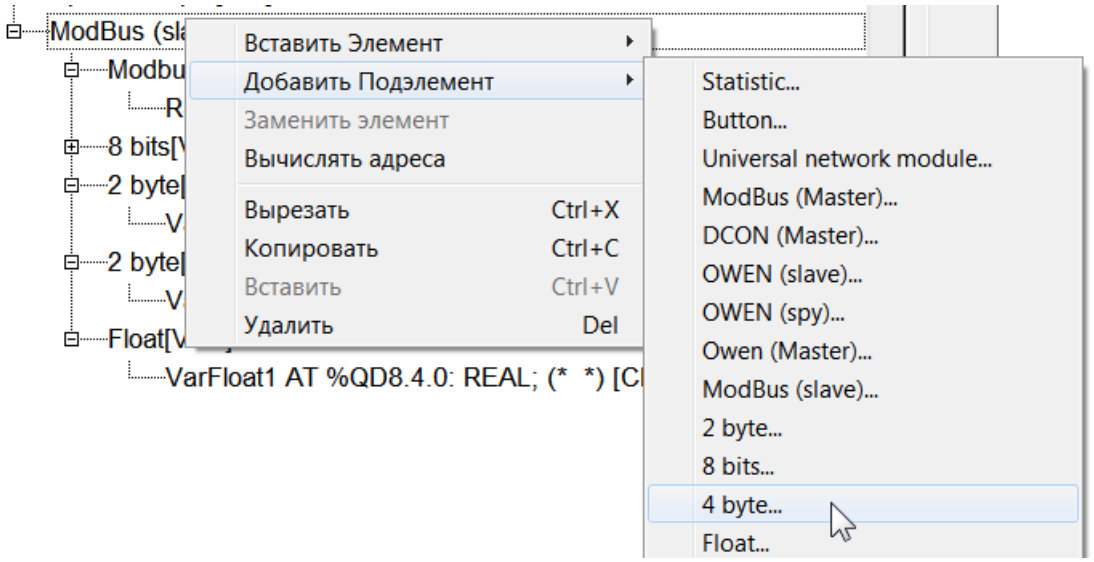

Рисунок 4-17

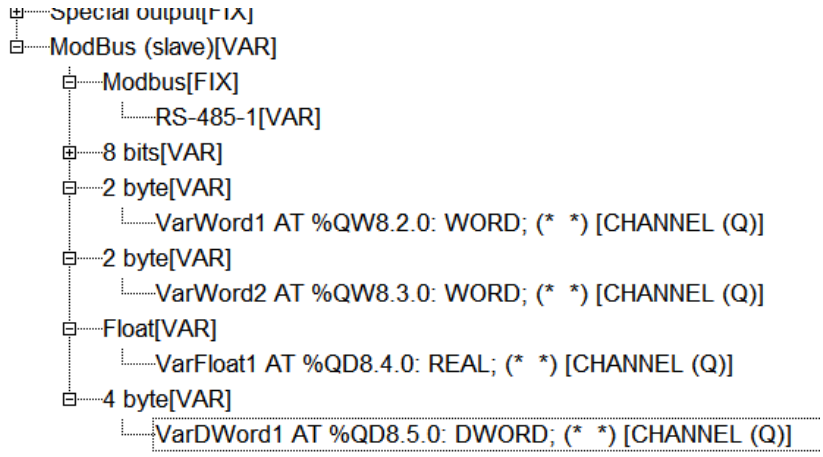

Рисунок 4-18

Данная переменная является 4-байтовой, поэтому к ней применимы те же правила что и для переменной типа ***Float***. В данном случае переменная ***VarDWord1*** будет располагаться в памяти контроллера по адресам 0x000C – 0x000F, которым соответствуют Modbus адреса ***6*** и ***7***.

В OPC сервере настройки тега *Тип данных в устройстве* и *Тип данных в сервере*, нужно задать ***uint32***. *Чередование байт* – также, как и у ***Float***, ***старшим байтом вперед*** (*Рисунок 4-19*).

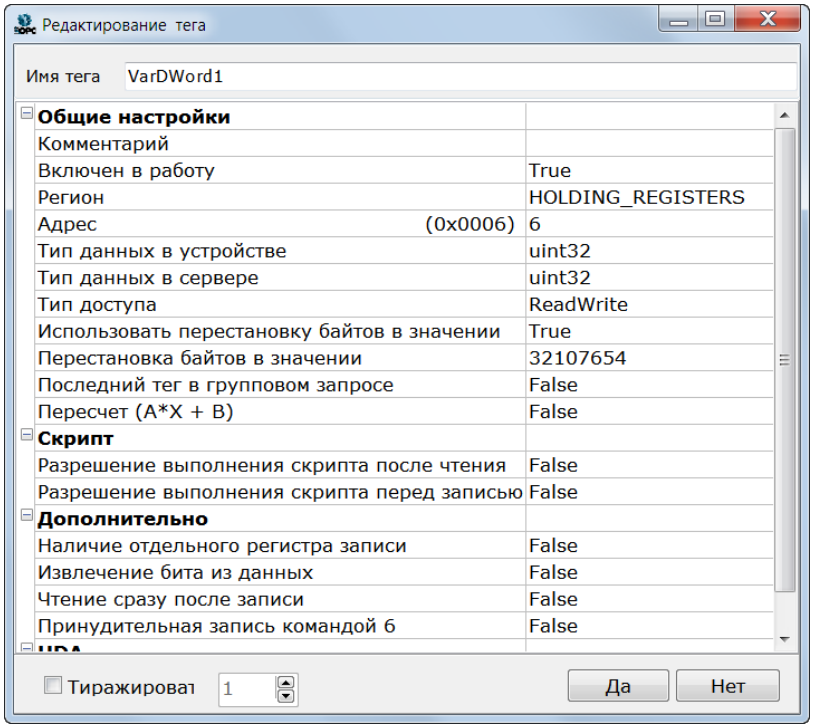

Рисунок 4-19

5 Настройка контроллера и OPC сервера на протокол Modbus TCP

Если опрос контроллера планируется вести по протоколу ***Modbus TCP***, то в *модуль* ***Modbus[FIX]*** нужно добавить *элемент* ***TCP***. (*Рисунок 5-1*)

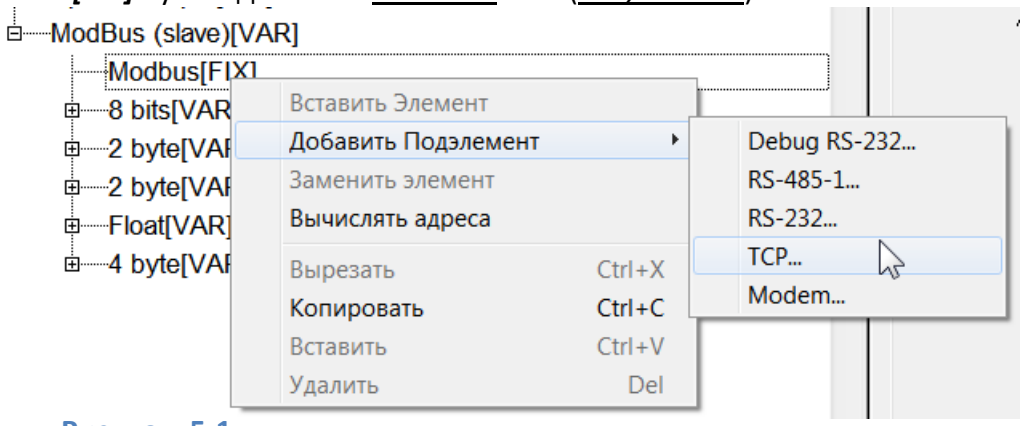

Рисунок 5-1

На *закладке* ***Параметры модуля*** есть лишь одна настройка – *номер порта TCP*, по которому будет происходит обмен. По умолчанию – ***502*** (*Рисунок 5-2*).

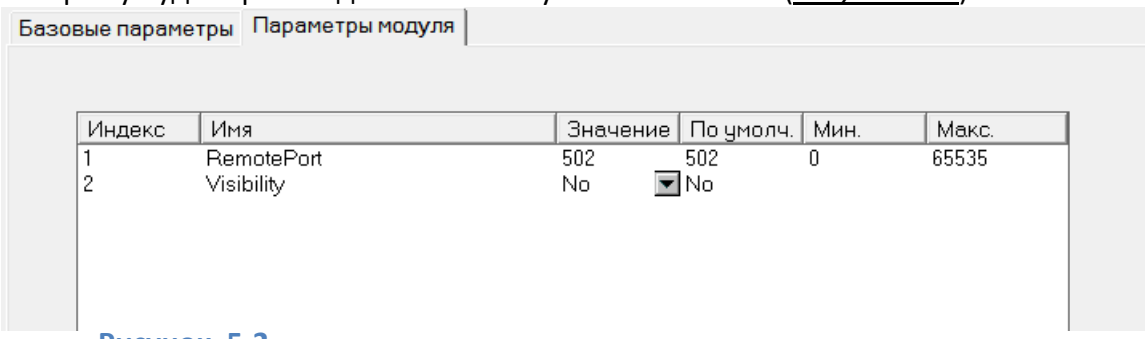

Рисунок 5-2

IP задается самому контроллеру. Это делается при установленном соединении, с помощью окна ***ПЛК-Браузер***. Для задания IP адреса используется команда ***SetIP*** (*Рисунок 5-3*).

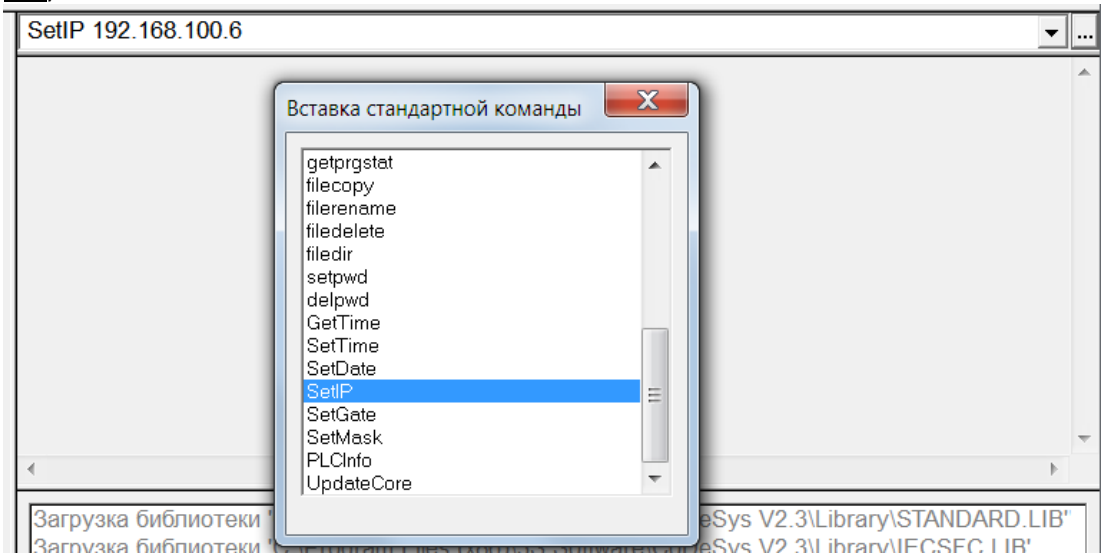

Рисунок 5-3

После выполнения команды контроллер необходимо перезагрузить.

В OPC сервере необходимо добавить узел, *тип узла* задать – ***TCP/IP***. Задать необходимые *параметры связи* – ***IP адрес*** контроллера, и заданный в настройках ***порт*** (*Рисунок 5-4*).

Также можно включить настройку ***Отслеживать Transaction ID*** – если данная настройка включена, то в специальном поле ***Modbus TCP*** запроса, будет меняться поле *идентификатора запроса*, что позволяет избежать коллизий разных запросов, при медленном ответе со стороны устройств.

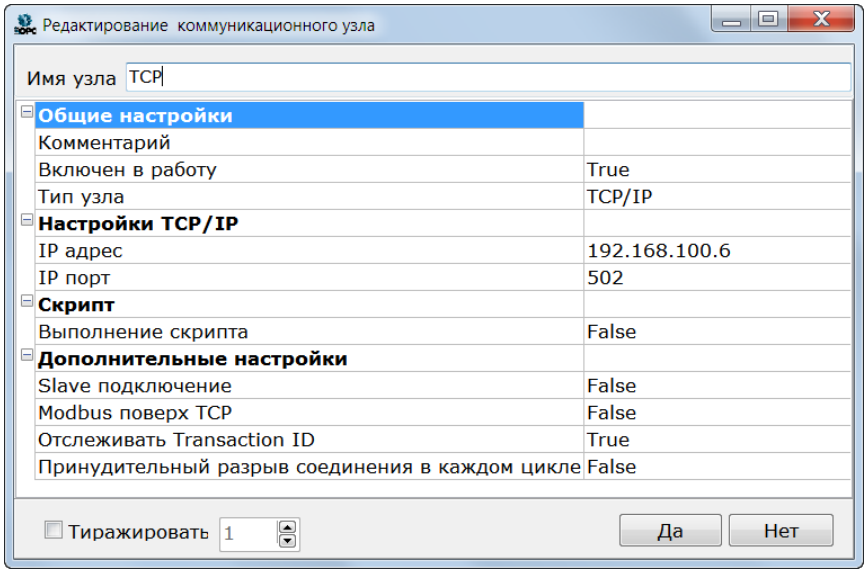

Рисунок 5-4

После добавления *узла*, в него добавляется *устройство*, и указывается ***адрес***, установленный в настройках *модуля* ***Modbus (Slave)*** в контроллере (*Рисунок 5-5*). Как правило данный адрес оставляют равным ***1***.

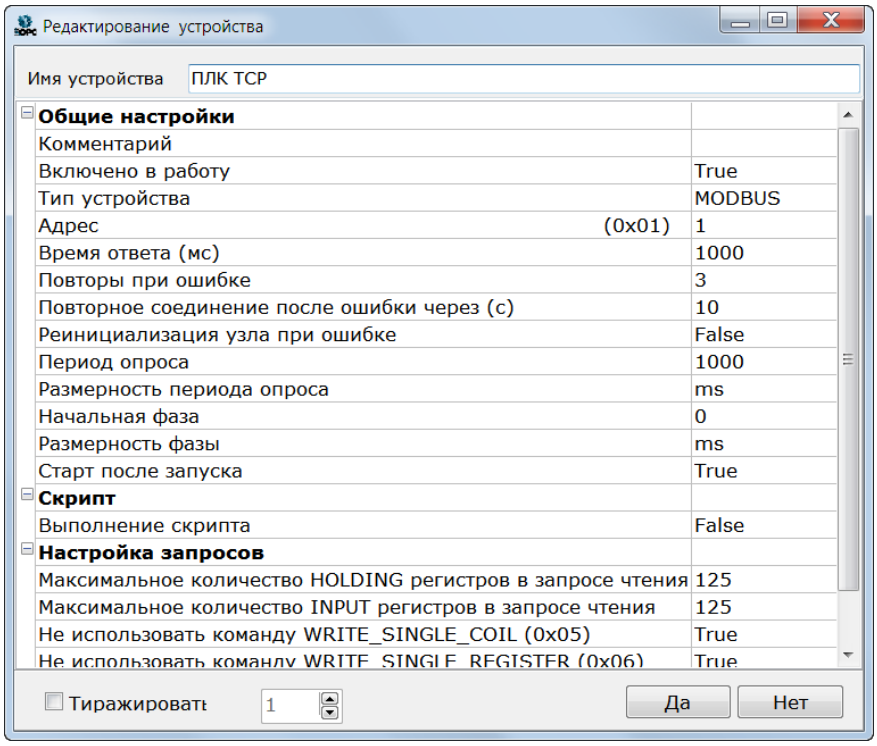

Рисунок 5-5

Добавление тегов и определение их адресов осуществляется аналогично, как и при работе по ***Modbus RTU***.

При необходимости можно добавить в ***Modbus[FIX]*** несколько *интерфейсов* – например ***RS-485*** и ***TCP*** (*Рисунок 5-6*),что позволяет опрашивать контроллер одновременно несколькими *ведущими* (например, *OPC сервером* и *панелью оператора*).

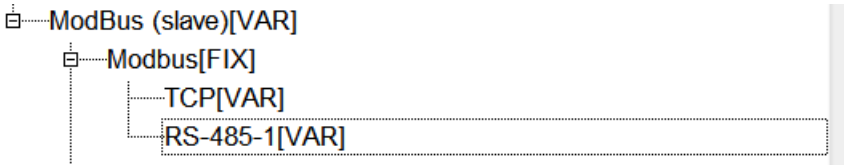

Рисунок 5-6

Также можно добавить два модуля ***TCP*** – для опроса несколькими сетевыми устройствами по локальной сети. Однако в этом случае, необходимо в настройках каждого элемента TCP, задать разные *порты* (например, ***502*** и ***503***) по которым и вести опрос.

6 Рекомендации по организации переменных

Для упрощения настройки обмена по Modbus далее будут приведены несколько рекомендаций.

6.1 Не использовать переменную 8 Bits

Как было сказано ранее в контроллере ***ОВЕН***, все ***Modbus*** переменные размещены в одном сегменте памяти, к которому можно обращаться через *регион* ***Holding Registers*** или ***Coils***.

Переменную ***8 bits***, обычно используют для передачи отдельных бит. Однако гораздо эффективнее использовать для этих целей переменную типа ***Word*** (*подэлемент* ***2 Byte***).

Для записи и чтения отдельных битов в ***Codesys*** можно использовать специальные функциональные блоки – ***Pack*** и ***Unpack*** (библиотека ***«Util.lib»***), а на языке ***ST*** можно обращаться к отдельным битам через точку (например ***VarWord1.0:=true***).

Если на верхнем уровне используется MasterSCADA, то для упаковки и извлечения битов можно использовать *ФБ* ***«Упаковка 32-битного значения»*** и ***«Распаковка 32-битного значения»***.

Отказ от работы с отдельными битами через *регион* ***Coils*** позволит сэкономить как лицензионные теги OPC сервера, так и SCADA системы, а кроме того снизит сетевую нагрузку по обмену данных.

6.2 Задать настройку «Максимально допустимый разрыв адресов в запросе чтения».

Из-за правил выравнивания может получится, что некоторые промежуточные ***Modbus*** адреса окажутся неиспользованными. В нашем примере сначала идут переменные ***VarWord1*** и ***VarWord2*** с адресами ***1*** и ***2***, а затем ***VarFloat1*** с адресом ***4***, и ***VarDWord1*** с адресом ***6***, т.е. адрес ***3*** не используется. В этом случае OPC сервер выполнит два Modbus запроса – сначала опросит адреса ***1*** и ***2***, а затем адреса с ***4*** по ***7***.

При необходимости можно сделать, чтобы подобные регистры были опрошены за один запрос чтения. Для этого необходимо у устройства задать настройку ***Максимально допустимый разрыв в запросе чтения*** (*Рисунок 6-1*). По умолчанию данная настройка равна нулю. Если же задать данный параметр, то все разрывы адресов меньшие заданного значения будут игнорироваться.

То есть, если в данном случае мы установим этот параметр равным ***1***, то OPC сервер запросит регистр с ***1*** по ***7*** одним запросом. Пустой, третий, Modbus регистр также будет опрошен, но его значение сервер просто проигнорирует.

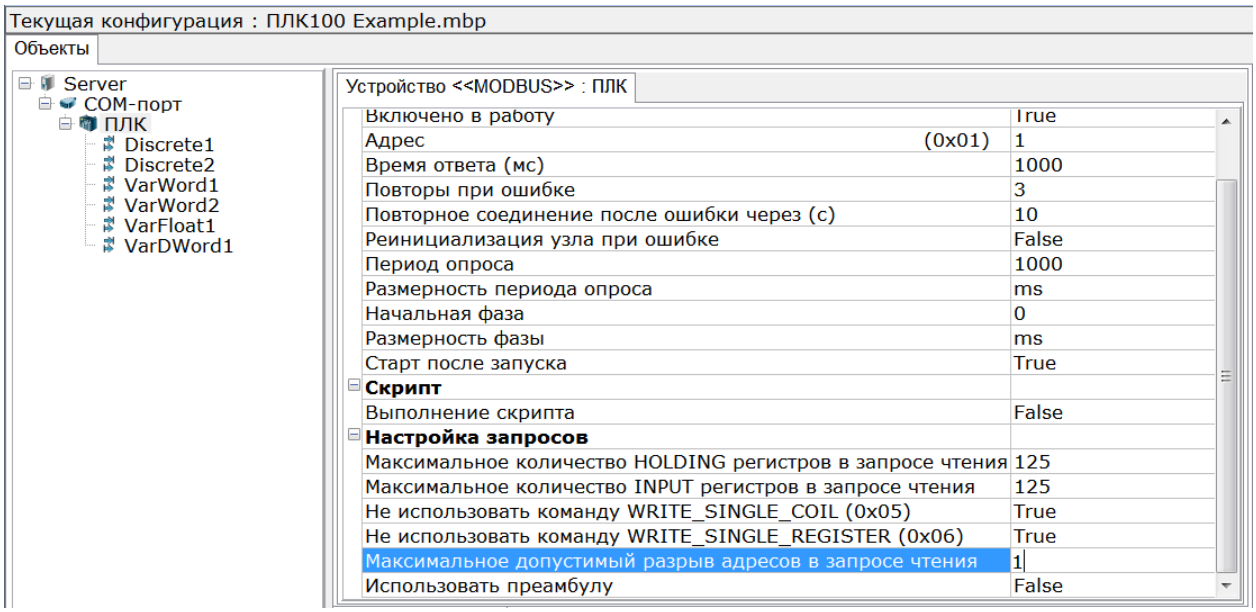

Рисунок 6-1

6.3 Формировать адреса в определенной последовательности

Чтобы уменьшить количество неиспользуемых ячеек памяти, и упростить подсчет Modbus адресов, рекомендуем структурировать переменные по типам. Например – сначала добавляем переменные типа ***Word*** (***2 Byte***), затем добавляем переменные типа ***Real*** (***Float***).

6.4 Вычислять адреса с помощью функции «Групповые операции»

При добавлении OPC переменных можно легко ошибиться при задании ***Modbus*** адресов. В третьей версии ***Modbus Universal MasterOPC*** сервера появилась новая функция – ***Групповые операции***. С помощью данной функции можно быстро вносить изменения в группе тегов – менять адреса, тип данных в устройстве, чередование байт. Данная функция может облегчить и задание адресов для опроса контроллеров ОВЕН.

Рассмотрим следующим пример. В модуль ***Modbus (Slave)*** добавлено ***7*** переменных типа ***Word***, ***4*** переменных типа ***Float*** и ***2*** переменных типа ***DWord*** (*Рисунок 6-2*).

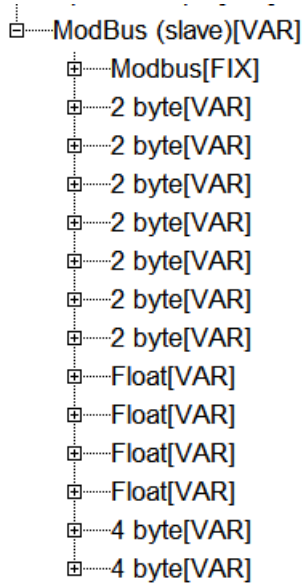

Рисунок 6-2

Создадим конфигурацию OPC сервера для их опроса.

Добавим в *устройство* теги в таком же порядке, как и модуля ***Modbus (Slave)*** и соответствующими типами данных (***2 byte*** – ***uint16***, ***Float*** – ***Float***, ***4 Byte*** – ***uint32***). Адрес у всех тегов можно оставлять равным нулю или оставить вариант предлагаемым сервером – позже мы вычислим через *групповые операции*.

В итоге у нас получилась следующая OPC конфигурация:

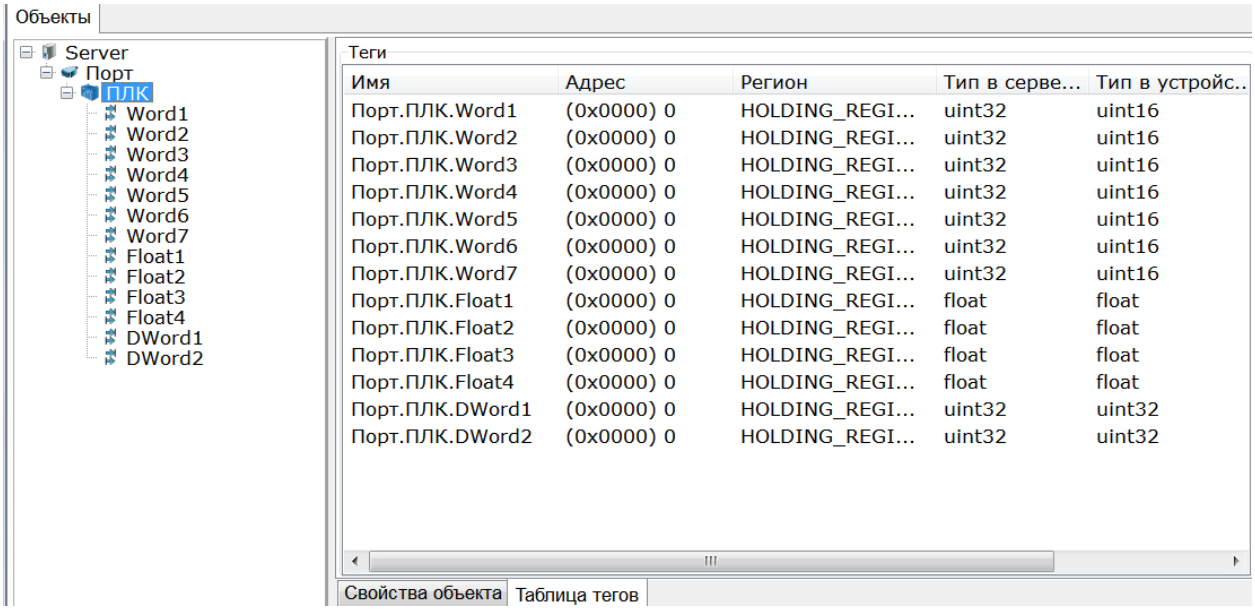

Рисунок 6-3

Через *контекстное меню* устройства вызовем команду ***Групповые операции*** (*Рисунок 6-4*).

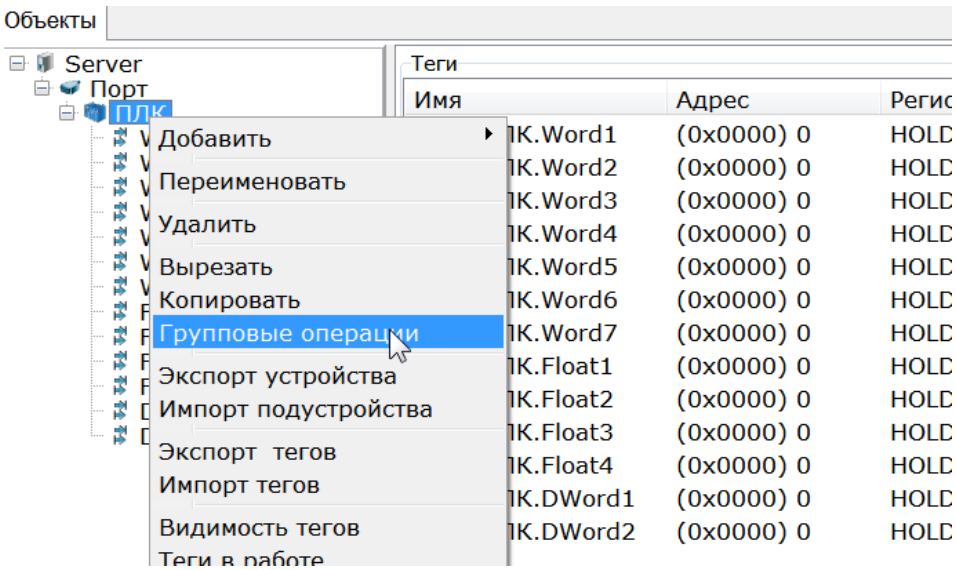

Рисунок 6-4

Выберем *операцию* ***Изменить адрес***, укажем *способ изменения* – ***По типам***, *базовый адрес* укажем – ***0***.

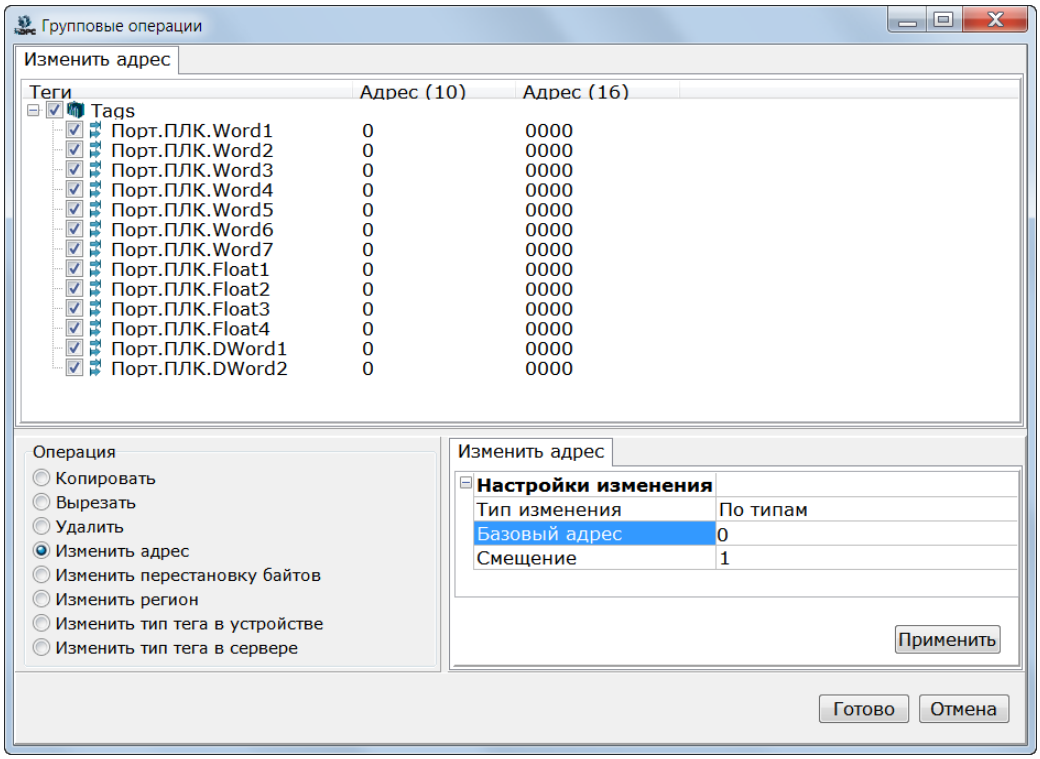

Рисунок 6-5

Нажмем кнопку ***Применить***.

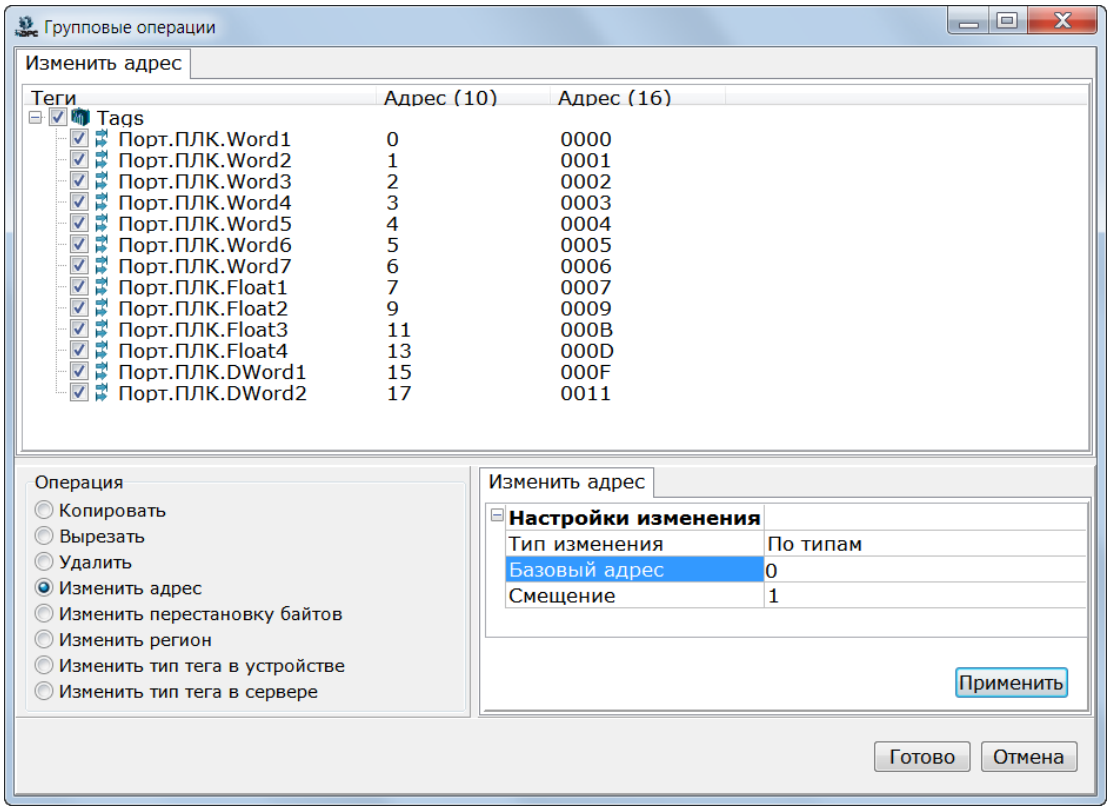

Рисунок 6-6

Теги получили последовательно идущие адреса с шагом в зависимости от типа (*Рисунок 6-6*).

Однако 4 байтовые теги (***Float*** и ***DWord*** переменные) попали на нечетные адреса, в то время как по правилам 4 байтовые переменные должны находится в адресах памяти кратных 4. То есть Modbus адреса 4 байтовых переменных должны начинаться с четного числа. Исправим эту ошибку.

Нажмем кнопку ***Готово***. Затем у устройства, на *закладке* ***Таблица тегов*** выделим все 4-байтовые переменные, вызовем *контекстное меню*, а затем команду ***Групповые операции*** (*Рисунок 6-7*).

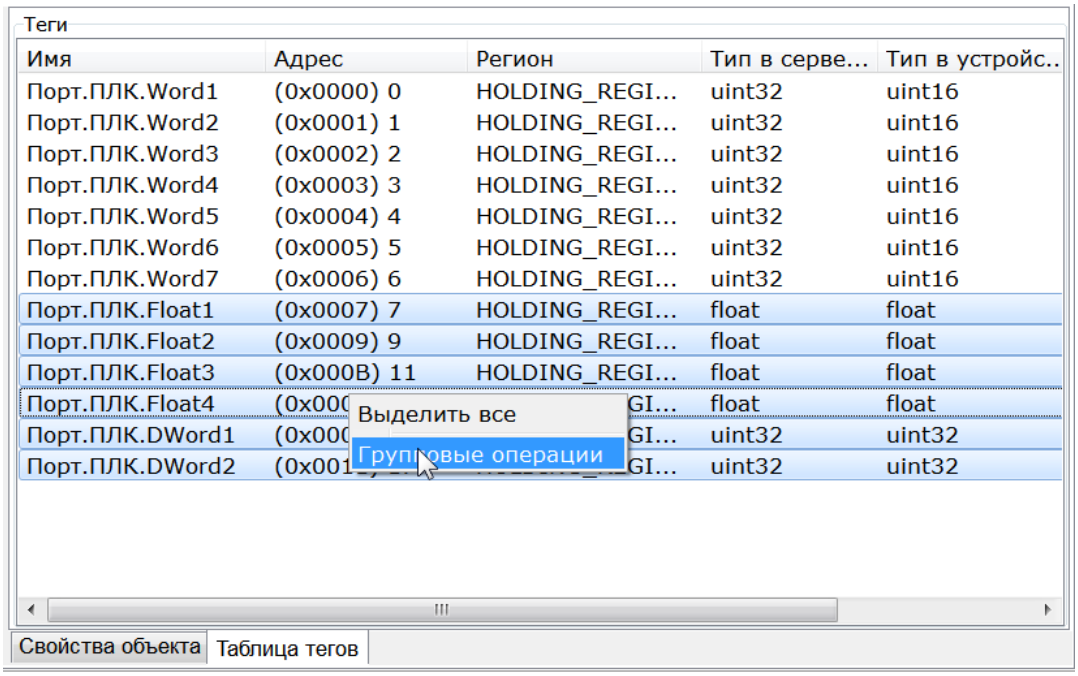

Рисунок 6-7

Переменная ***Float*** начинается с адреса ***7***. Укажем в качестве *базового адреса* следующее четное число, то есть ***8*** (*Рисунок 6-8*).

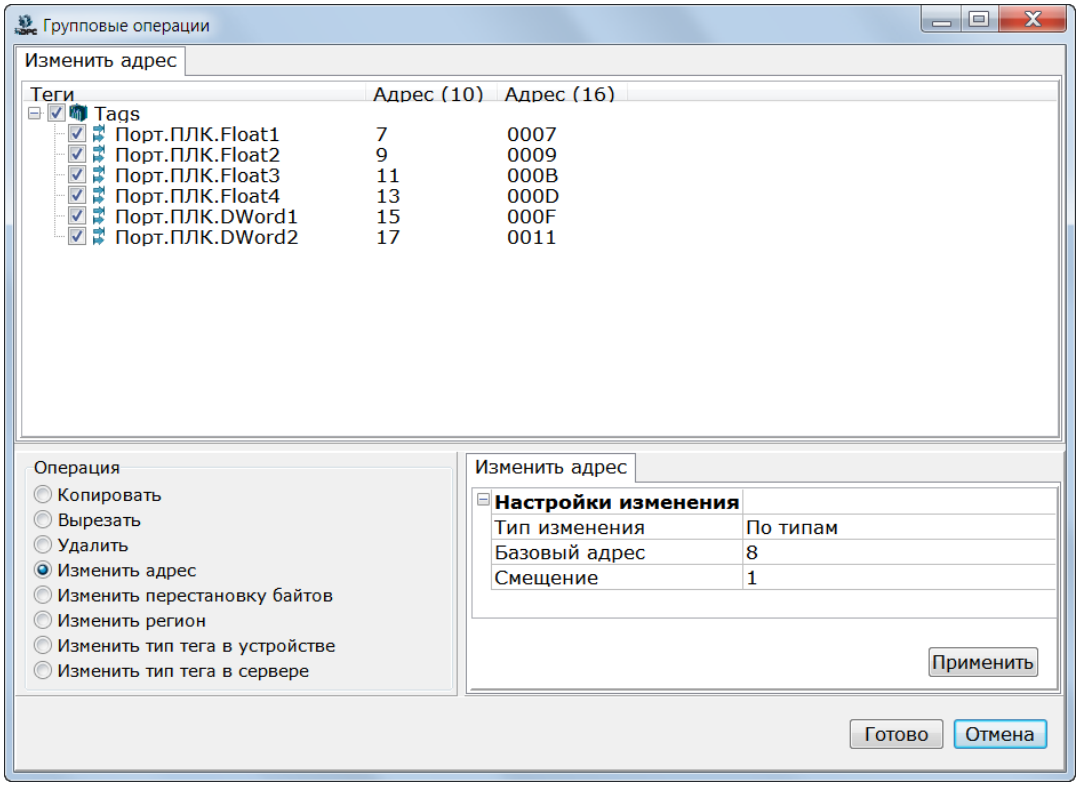

Рисунок 6-8

Нажмем кнопку ***Применить***.

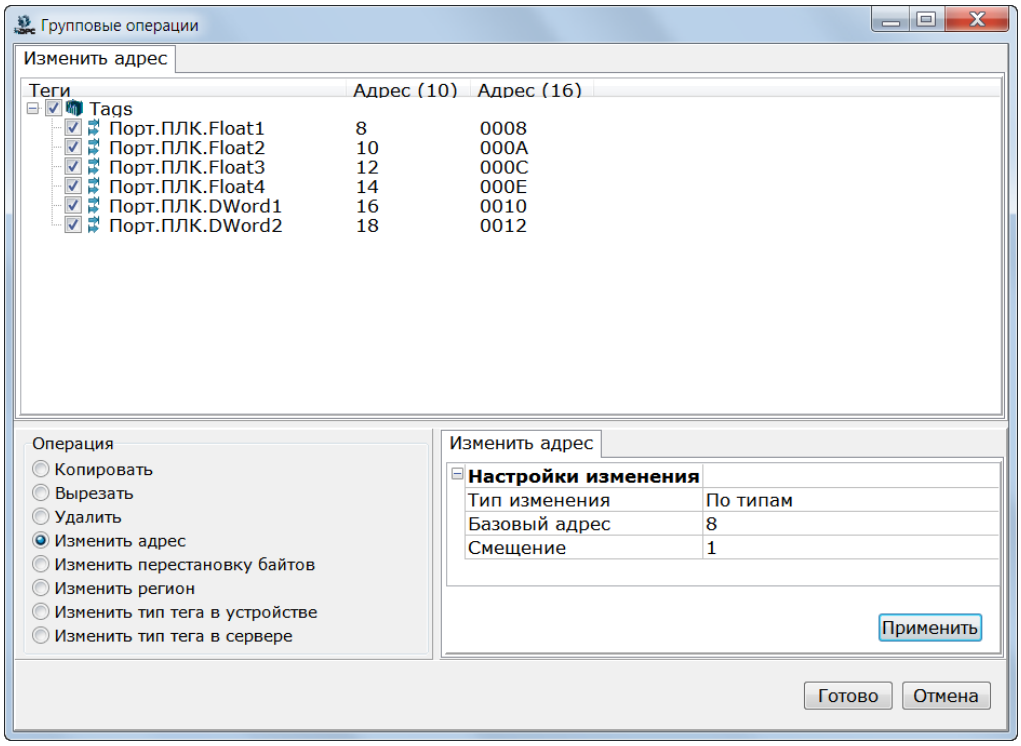

Рисунок 6-9

Теперь все адреса корректные (*Рисунок 6-9*). Нажмем на кнопку ***Готово***.

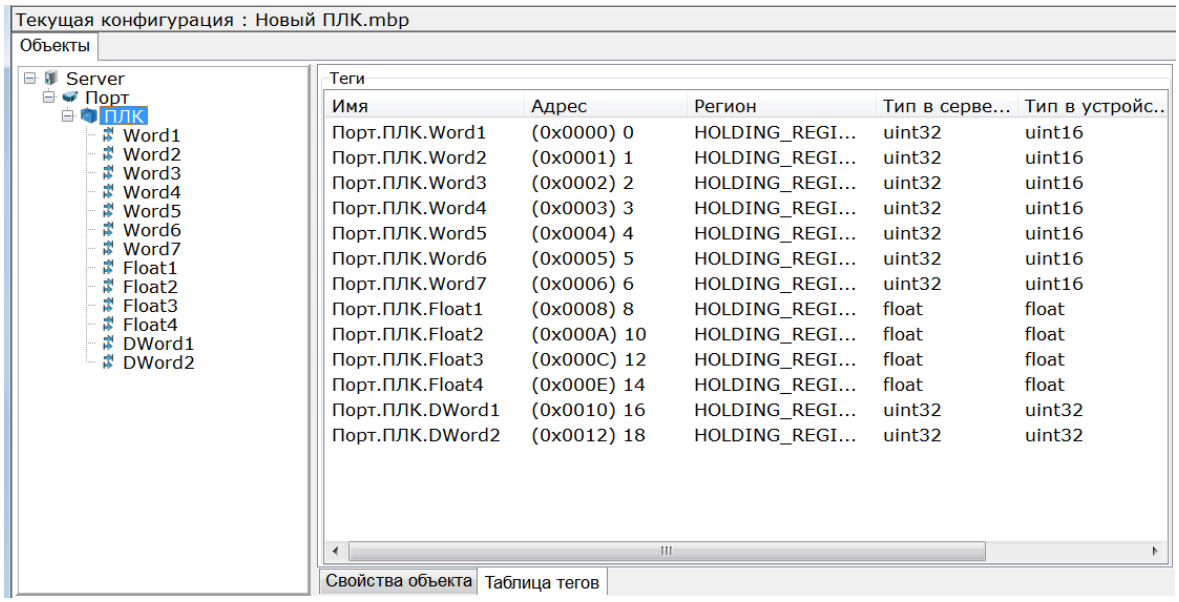

Рисунок 6-10

***Примечание**. Проект Codesys v2.3 для контроллера ОВЕН ПЛК100-Р.М с полным кодом данного примера, а также конфигурация ОРС сервера приложены к документации.*