Данный проект демонстрирует пример использования ПЛК NLScon-RSB и ПЛК NLcon-CE для обмена данными в сети Modbus TCP в качестве ведущего (Modbus TCP Master) и ведомого (Modbus TCP Slave) устройств. Для разработки проекта использована среда Codesys 3.5.16 patch 4 и установочные пакеты компонентов для модулей RealLab! CoDeSys Linux package и CoDeSys Windows CE package.

С помощью стандартного функционала Codesys в дерево устройств добавлены компоненты Ethernet, Modbus_TCP_Master, Modbus_TCP_Slave (см. рис. 1).

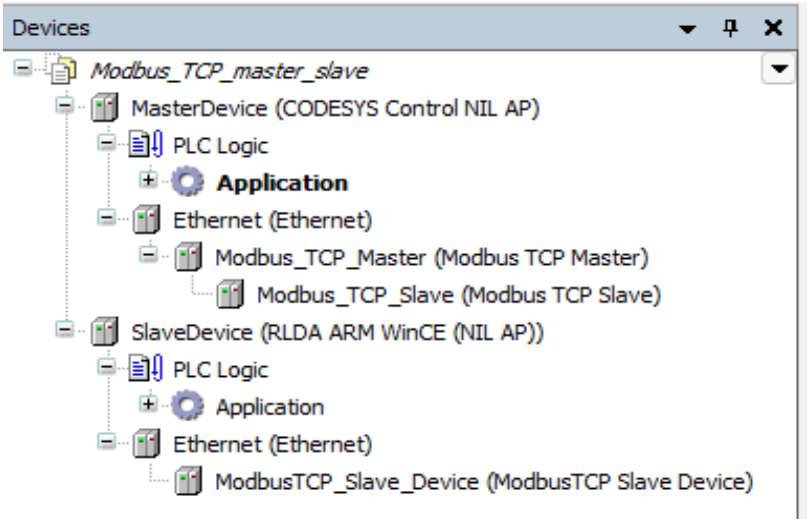

Рисунок 1 — Конфигурация дерева устройств для обмена данными между ПЛК RealLab! по сети Modbus TCP

Для организации обмена данными между ведущим и ведомым устройством требуется настроить добавленные компоненты дерева устройств. С этой целью на вкладке General компонентов Ethernet устанавливаются параметры соответствующего Ethernet-порта ПЛК (см. рис. 2).

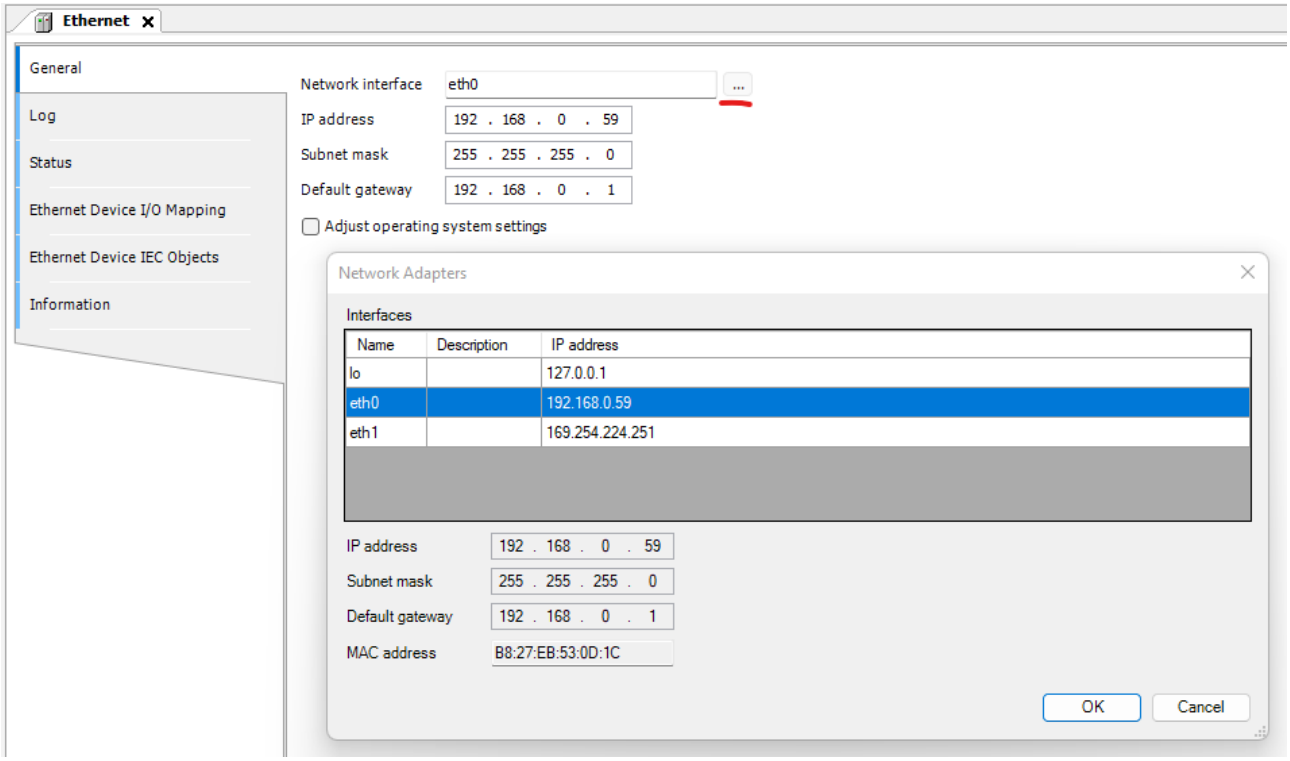

Рисунок 2 — Настройка параметров компонента Ethernet

В дереве устройств MasterDevice для компонента Modbus_TCP_Master установлен автоперезапуск соединения (см. рис. 3), для Modbus_TCP_Slave установлен вручную IP адрес устройства, с которым нужно обмениваться данными по протоколу Modbus TCP (см. рис. 4).

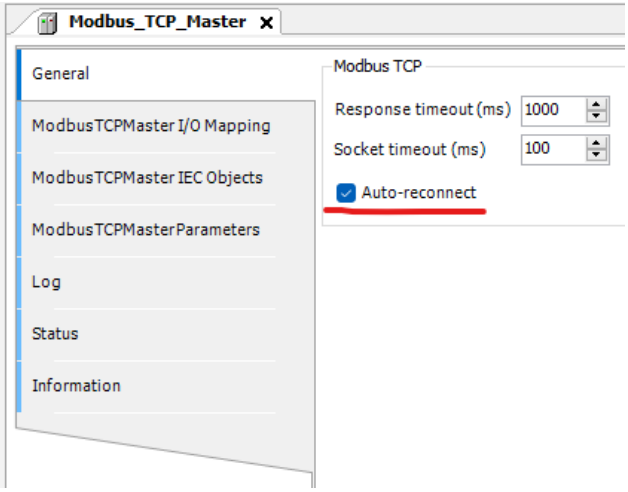

Рисунок 3 — Настройка параметров компонента Modbus_TCP_Master в дереве MasterDevice

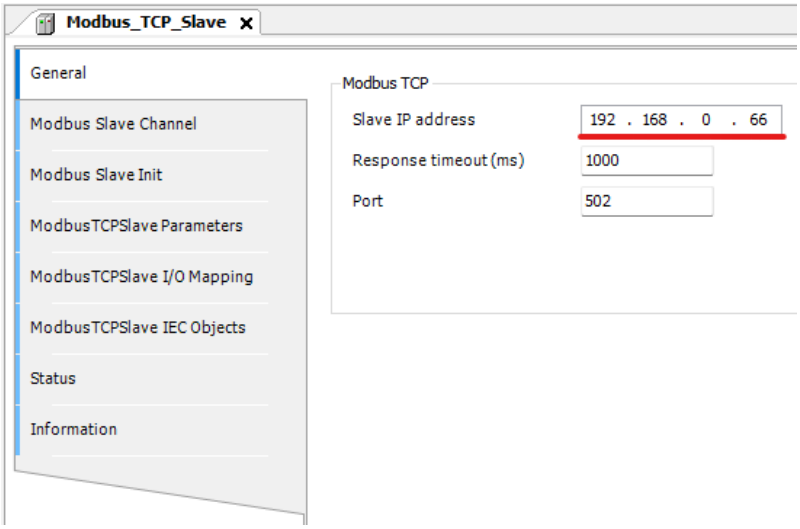

Рисунок 4 — Настройка параметров компонента Modbus_TCP_Slave в дереве MasterDevice

В данном примере ведущее устройство (MasterDevice) обменивается данными с ведомым (SlaveDevice) по 4 каналам:

- чтение одного дискретного входа с адресом 0x0000;
- чтение одного входного регистра с адресом 0x0001;
- запись двух регистров хранения с адресами 0x0000 и 0x0001;
- запись трех регистров хранения с адресами 0x0005, 0x0006 и 0x0007.

На вкладке Modbus Slave Channel устройства Modbus_TCP_Slave в дереве MasterDevice добавлены указанные выше каналы (см. рис. 5). На вкладке ModbusTCPSlave I/O Mapping выполнена привязка каналов к соответствующим переменным, используемым в программе устройства MasterDevice (рис. 6). Кроме привязки переменных необходимо установить свойство каналов «Always update variables» – «Enable 2» («Всегда обновлять переменные» - «Вкл. 2 (всегда в задаче цикла шины)»).

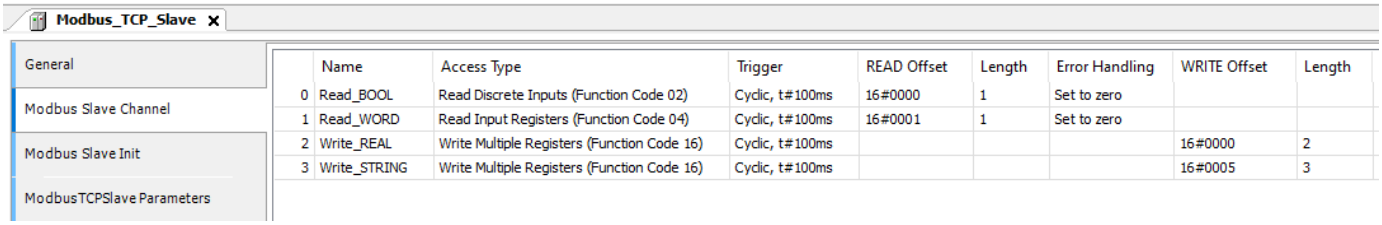

Рисунок 5 — Настройка каналов компонента Modbus_TCP_Slave в дереве MasterDevice

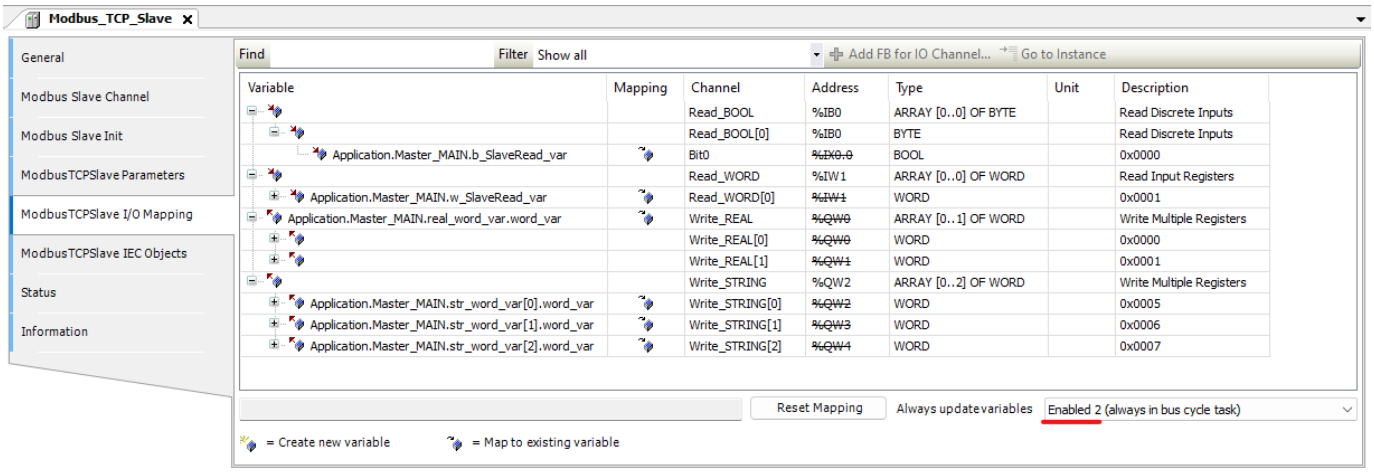

Рисунок 6 — Привязка переменных к каналам компонента Modbus_TCP_Slave в дереве MasterDevice

Для настройки обмена данными в устройстве SerialDevice для компонента ModbusTCP_Slave_Device на вкладке General требуется определить количество регистров, участвующих в обмене (см. рис. 7).

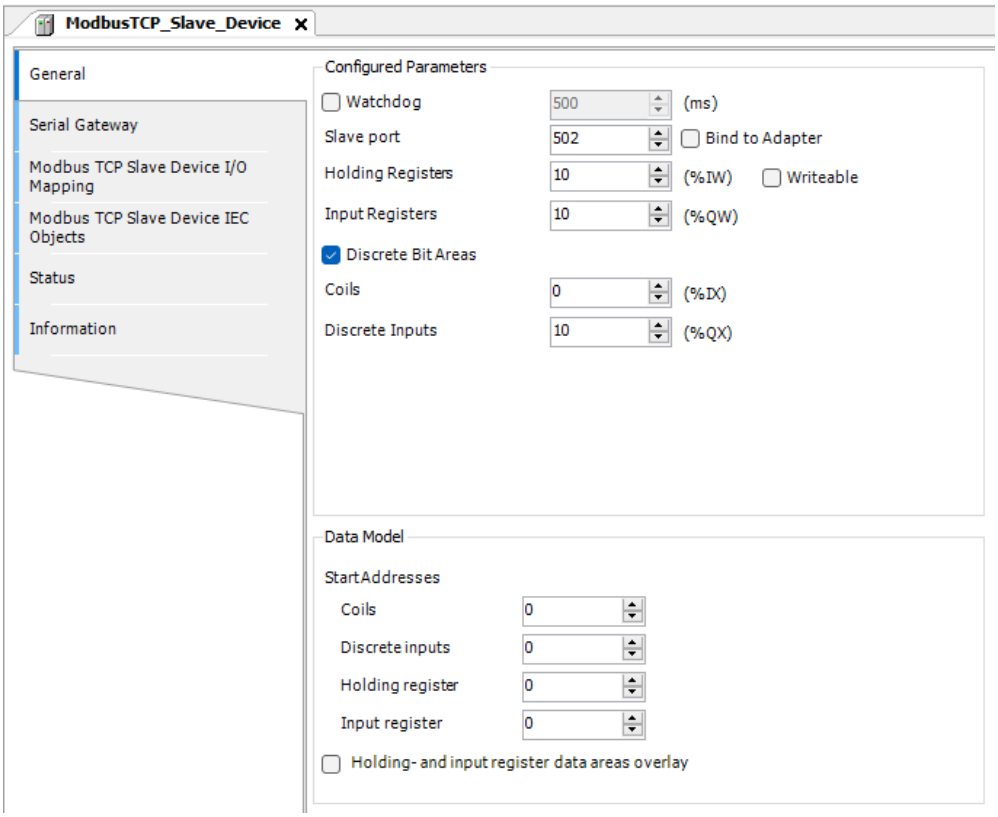

Рисунок 7 — Настройка регистров компонента ModbusTCP_Slave_Device в дереве SlaveDevice

Необходимо отметить, что для компонента ModbusTCP_Slave_Device в дереве SlaveDevice имеет место следующее распределение регистров:

Holding Registers (Регистры временного хранения) – регистры для записи данных из Master в Slave;

Input Registers (Входные регистры) — регистры для чтения данных из Slave;

Coils (Регистры) — регситры для записи данных из Master в Slave;

Discrete Inputs (Дискретные входы) — регистры для чтения данных из Slave.

Если установить галочку «Writable» напротив Holding Registers (см. рис. 7), то будет возможно изменять значения этих регистров из программы Codesys, загруженной в SlaveDevice. Это возможно делать, меняя в коде программы SlaveDevice переменные, привязанные к Holding registers. Однако из визуализации SlaveDevice значения переменных, привязанных к каналам Holding регистров, нельзя изменить напрямую. Это связано с тем, что обработка компонентов Modbus и визуализации выполняется в разных задачах проекта.

На рис. 8 показана привязка регистров SlaveDevice к переменным программы Slave_MAIN.

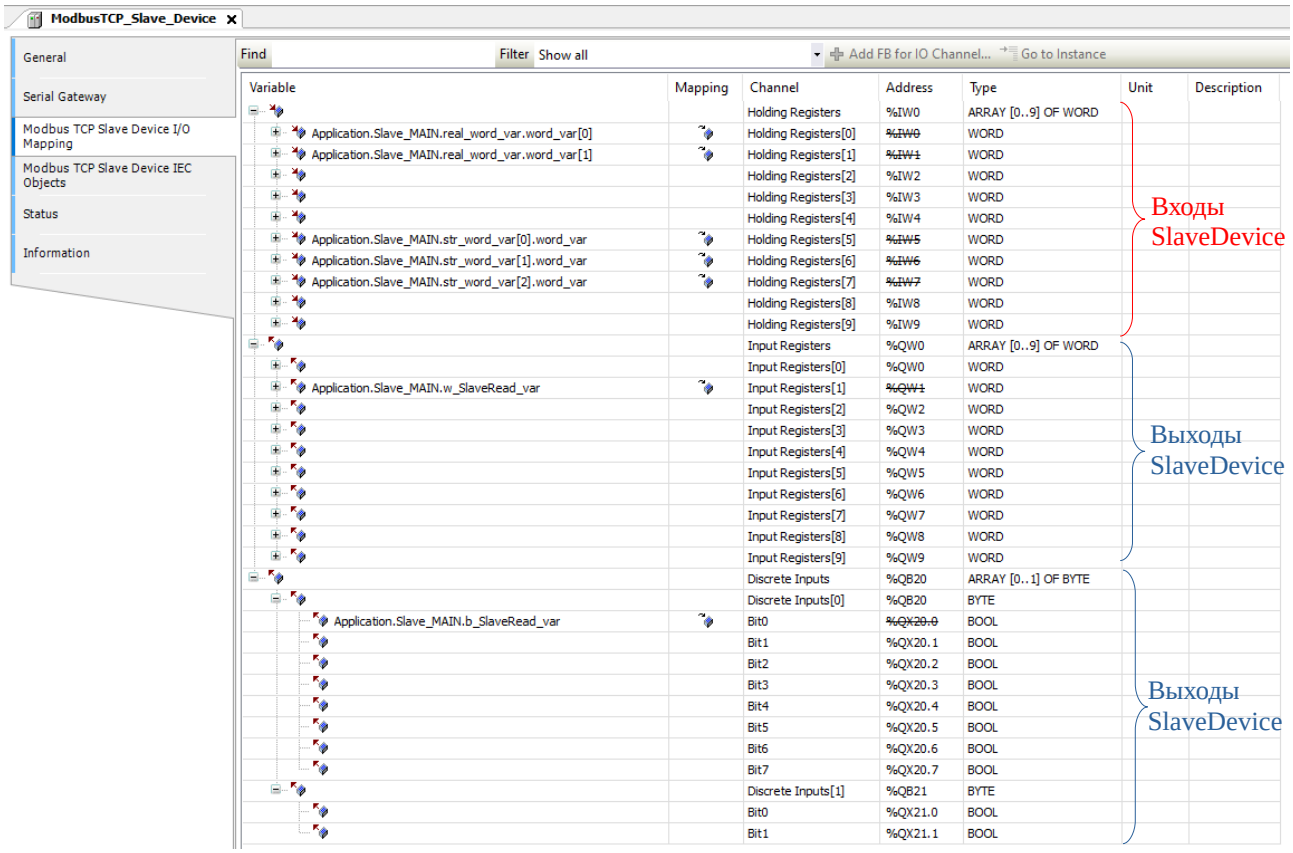

Рисунок 8 — Настройка регистров компонента ModbusTCP_Slave_Device в дереве SlaveDevice

Для проверки работы проекта требуется загрузить в MasterDevice и SlaveDevice соответствующие приложения. Далее в режиме отладки можно формировать значения переменных, учитывая направление передачи данных (см. рис. 9).

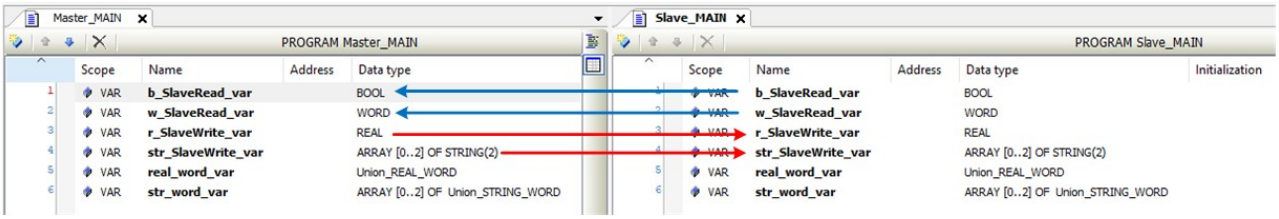

Рисунок 9 — Переменные, используемые в программах MasterDevice и SlaveDevice

Для корректной записи данных типов REAL и STRING из MasterDevice в SlaveDevice в приложениях обоих ПЛК используются объединения Union_REAL_WORD и Union_STRING_WORD. В программе Master_MAIN можно указать значение r_SlaveWrite_var и элементов массива str_SlaveWrite_var, а затем проверить их передачу в SlaveDevice в соответствующие переменные (см. рис. 10 и 11).

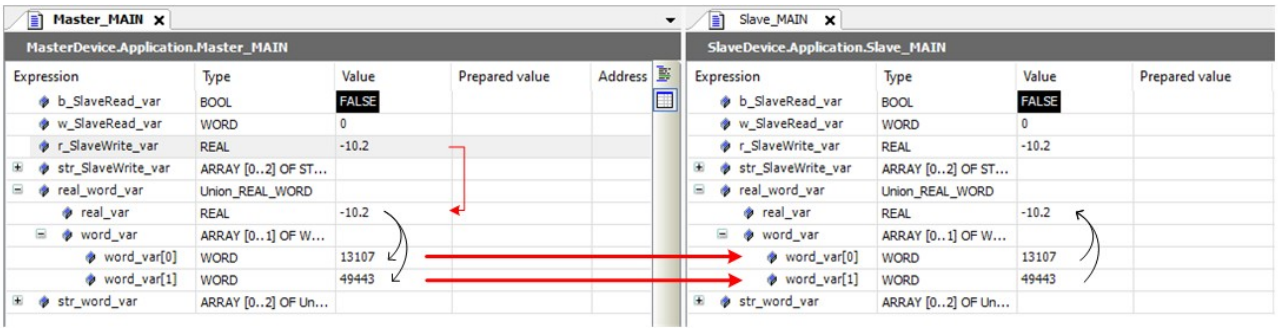

Рисунок 10 — Запись значения r_SlaveWrite_var в SlaveDevice

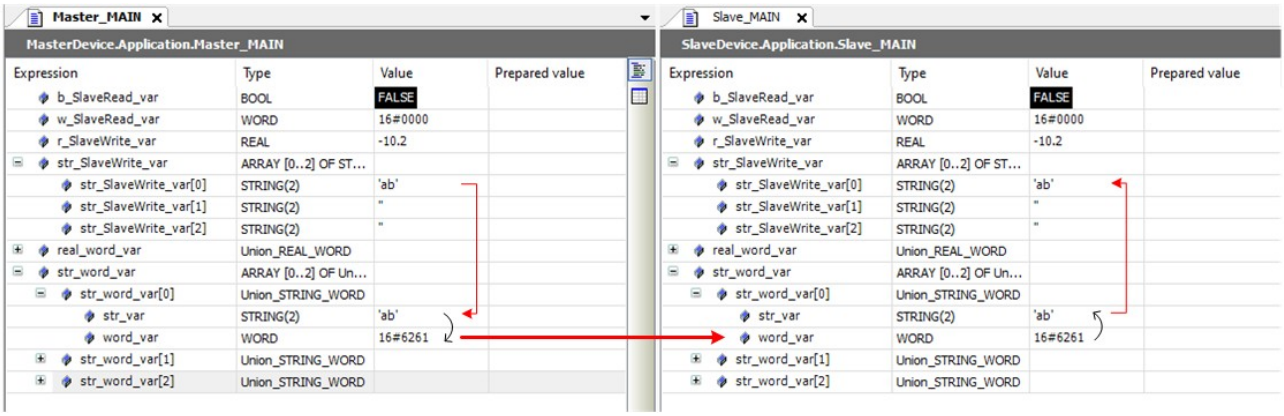

Рисунок 11 — Запись значения str_SlaveWrite_var[0] в SlaveDevice

В приложении для SlaveDevice также используется функция WORD2_AS_REAL(wVar), формирующая в качестве выходного значения переменную r_SlaveWrite_var типа REAL из входного массива типа WORD:

```
r_SlaveWrite_var   -10.2   := WORD2_AS_REAL(real_word_var.word_var);
```

Чтение данных из SlaveDevice происходит непосредственно в нужные переменные (рис. 12).

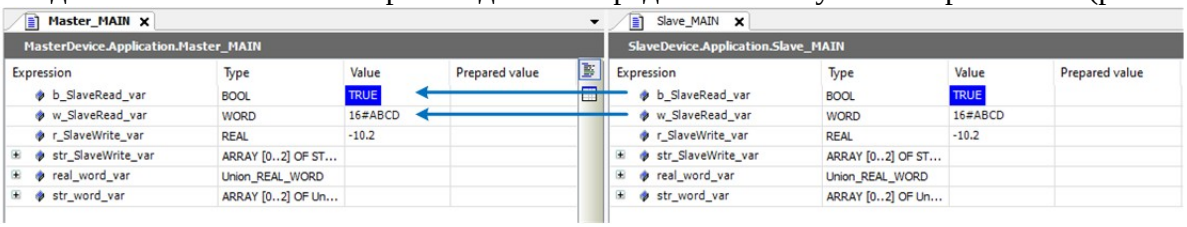

Рисунок 12 — Чтение данных из SlaveDevice