Данный проект демонстрирует пример использования ПЛК NLScon-RSB и виртуального ПЛК CODESYS Control Win V3 для обмена данными в сети Modbus RTU в качестве ведущего (Modbus RTU Master) и ведомого (Modbus RTU Slave) устройств. Для разработки проекта использована среда Codesys 3.5.16 patch 4 и установочный пакет компонентов для модулей RealLab! CoDeSys Linux package.

С помощью стандартного функционала Codesys в дерево устройств добавлены компоненты Modbus_COM, Modbus_Master_COM_Port, Modbus_Slave_COM_Port и Modbus_Serial_Device (см. рис. 1).

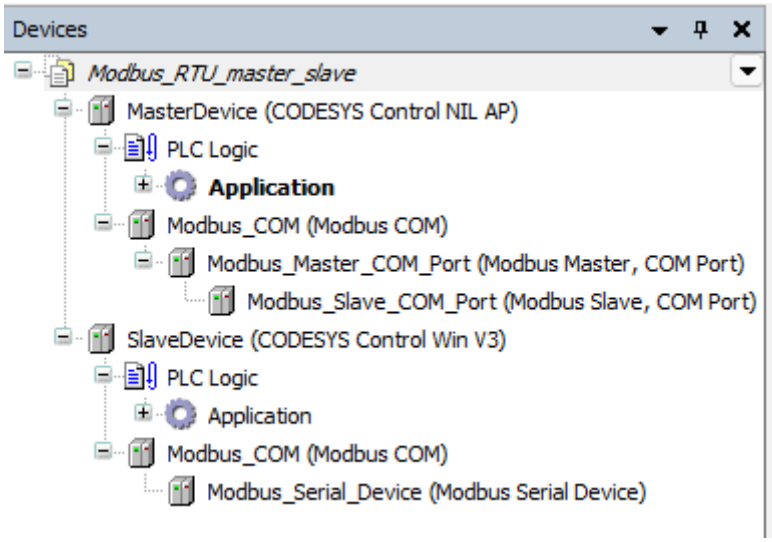

Рисунок 1 — Конфигурация дерева устройств для обмена данными между ПЛК RealLab! и ПЛК CODESYS Control Win V3 по сети Modbus RTU

Для организации обмена данными между ведущим и ведомым устройством требуется настроить добавленные компоненты дерева устройств. С этой целью на вкладке General компонентов Modbus_COM устанавливаются параметры соответствующих COM-портов ПЛК (см. рис. 2). Номер COM-порта виртуального контроллера можно определить, ползуясь Диспетчером устройств ПК (см. рис. 3).

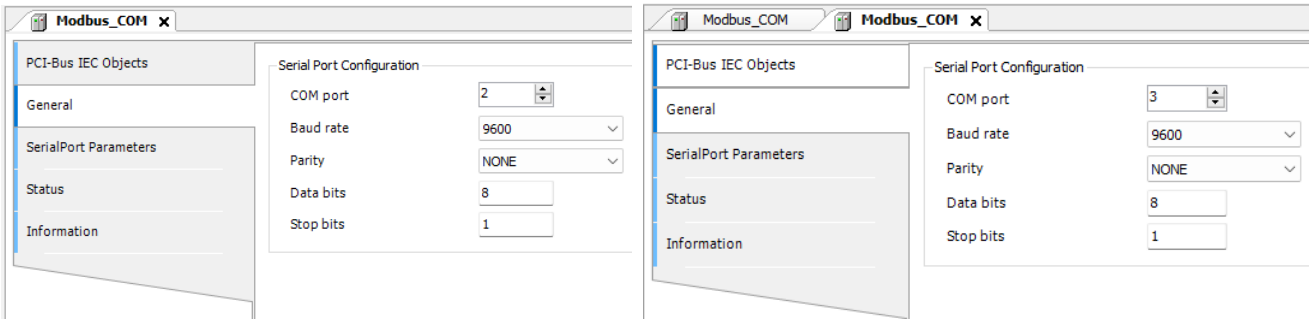

Рисунок 2 — Настройка параметров компонента Modbus_COM ПЛК NLScon-RSB (слева) и виртуального ПЛК (справа)

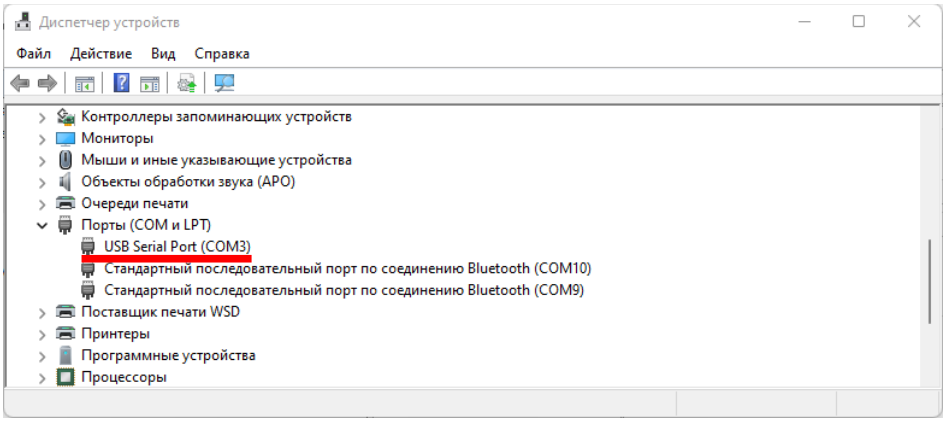

Рисунок 3 — COM-порт виртуального контроллера в Диспетчере устройств ПК

В дереве устройств MasterDevice для компонента Modbus_Master_COM_Port установлен автоперезапуск соединения (см. рис. 4), для Modbus_Slave_COM_Port установлен вручную адрес устройства в сети Modbus RTU (см. рис. 5).

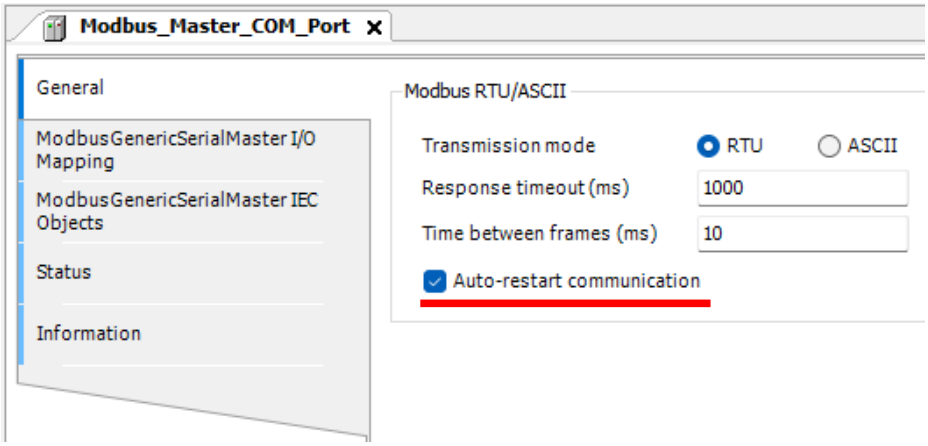

Рисунок 4 — Настройка параметров компонента Modbus_Master_COM_Port в дереве MasterDevice

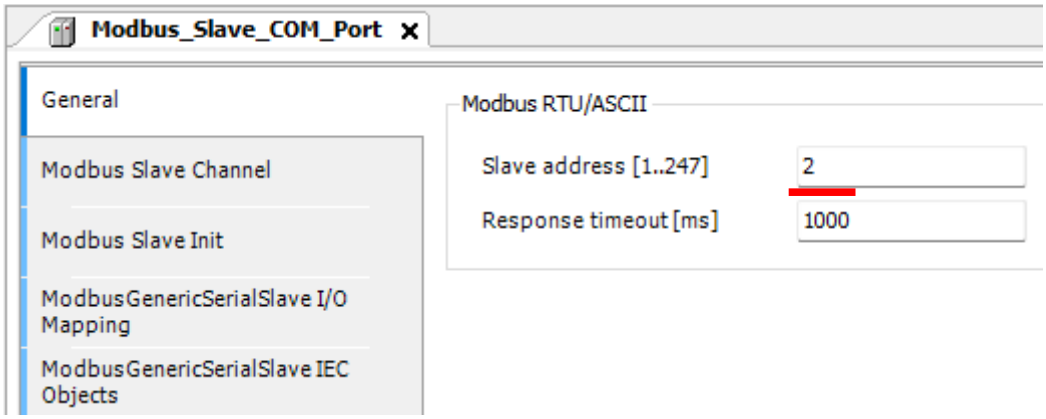

Рисунок 5 — Настройка параметров компонента Modbus_Slave_COM_Port в дереве MasterDevice

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_Slave_COM_Port в дереве MasterDevice добавлены указанные выше каналы (см. рис. 6). На вкладке ModbusGenericSerialSlave I/O Mapping выполнена привязка каналов к соответствующим переменным, используемым в программе устройства MasterDevice (рис. 7). Кроме привязки переменных необходимо установить свойство каналов «Always update variables» – «Enable 2» («Всегда обновлять переменные» - «Вкл. 2 (всегда в задаче цикла шины)»).

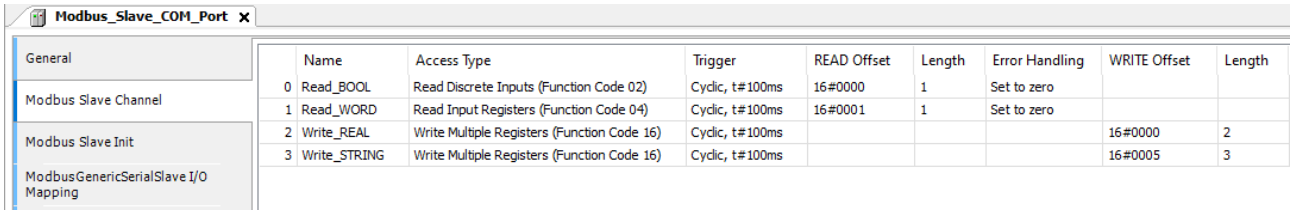

Рисунок 6 — Настройка каналов компонента Modbus_Slave_COM_Port в дереве MasterDevice

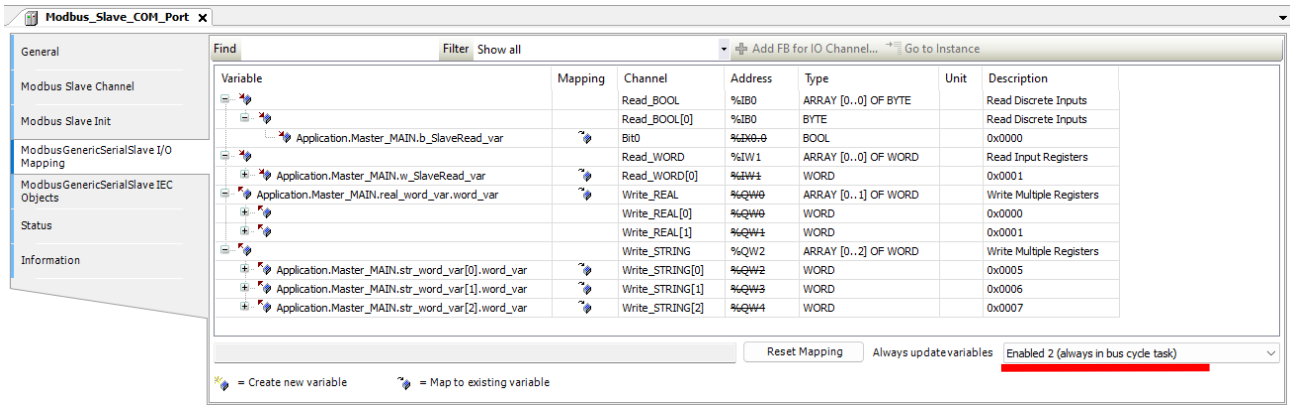

Рисунок 7 — Привязка переменных к каналам компонента Modbus_TCP_Slave в дереве MasterDevice

Для настройки обмена данными в устройстве SerialDevice для компонента Modbus_Serial_Device на вкладке General требуется определить количество регистров, участвующих в обмене (см. рис. 8). Здесь же в поле Unit ID указывается адрес Slave-устройства.

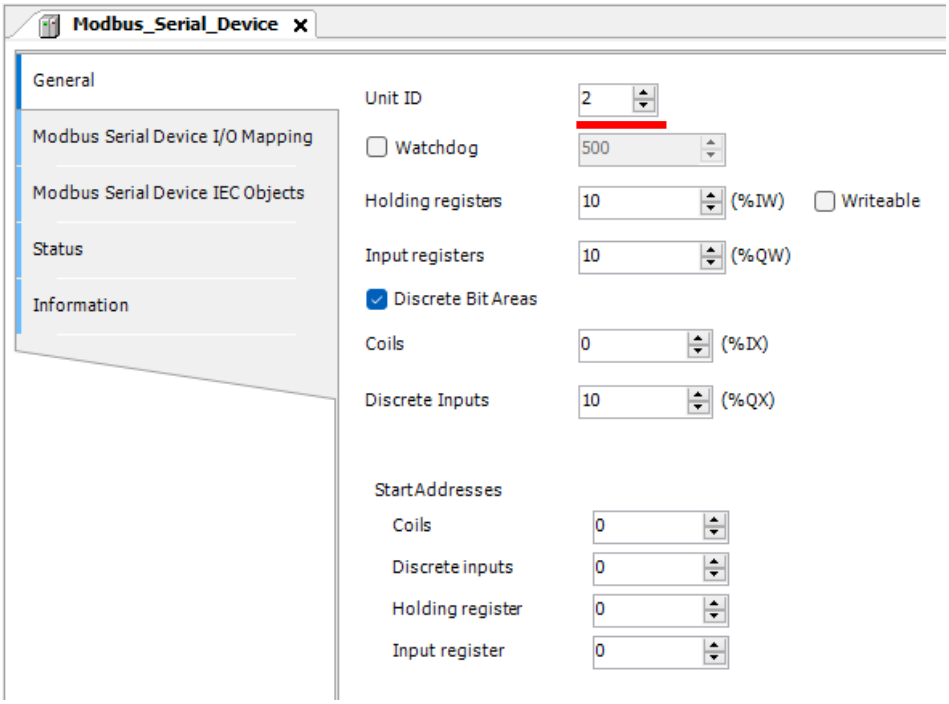

Рисунок 8 — Настройка регистров компонента Modbus_Serial_Device в дереве SlaveDevice

Необходимо отметить, что для компонента Modbus_Serial_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. 8), 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. 9 показана привязка регистров SlaveDevice к переменным программы Slave_MAIN. Кроме привязки переменных необходимо установить свойство каналов «Always update variables» – «Enable 2» («Всегда обновлять переменные» - «Вкл. 2 (всегда в задаче цикла шины)»).

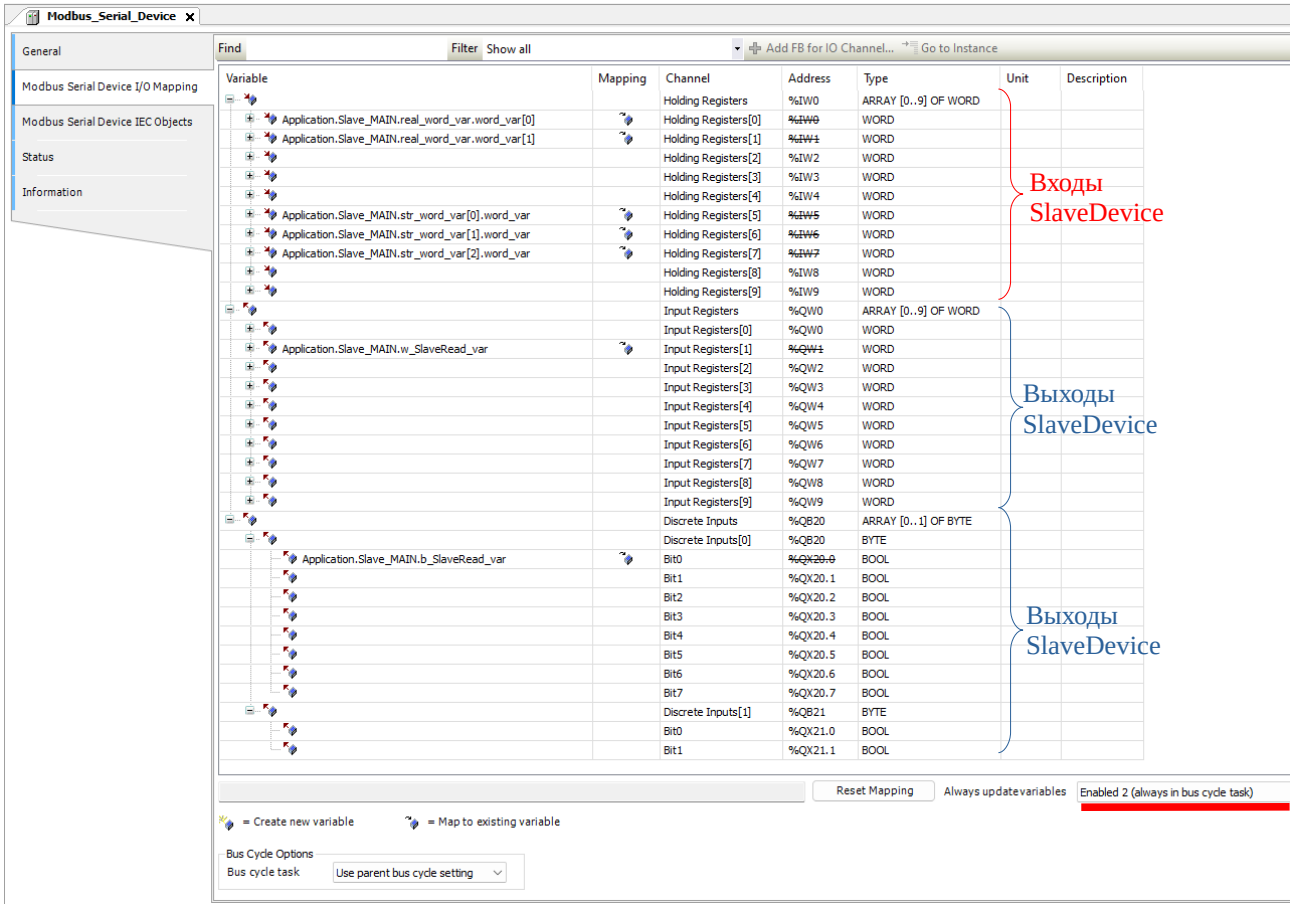

Рисунок 9 — Настройка регистров компонента Modbus_Serial_Device в дереве SlaveDevice

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. 10).

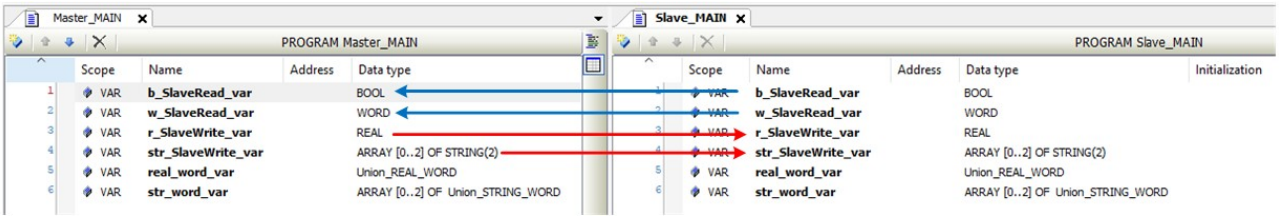

Рисунок 10 — Переменные, используемые в программах MasterDevice и SlaveDevice

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. 11 и 12).

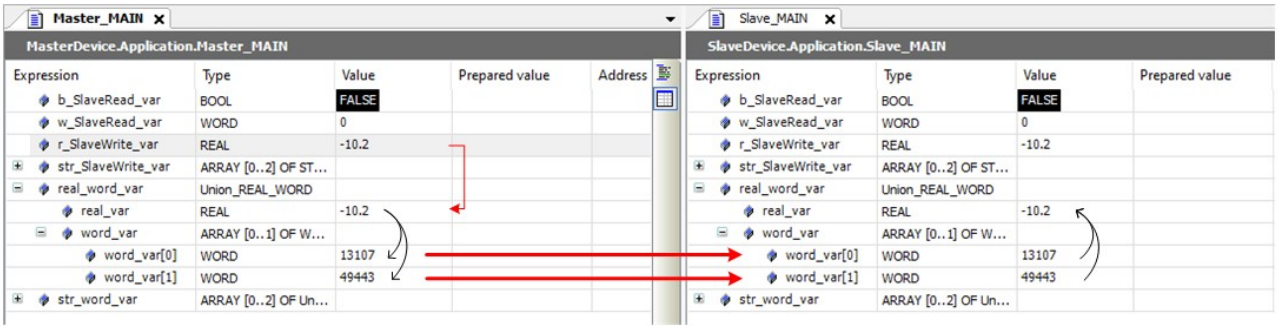

Рисунок 11 — Запись значения r_SlaveWrite_var в SlaveDevice

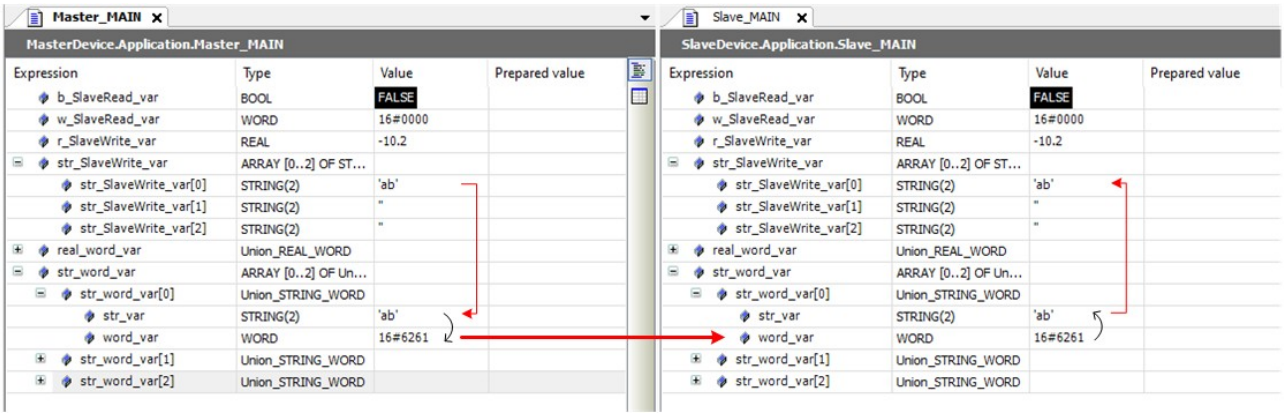

Рисунок 12 — Запись значения str_SlaveWrite_var[0] в SlaveDevice

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. 13).

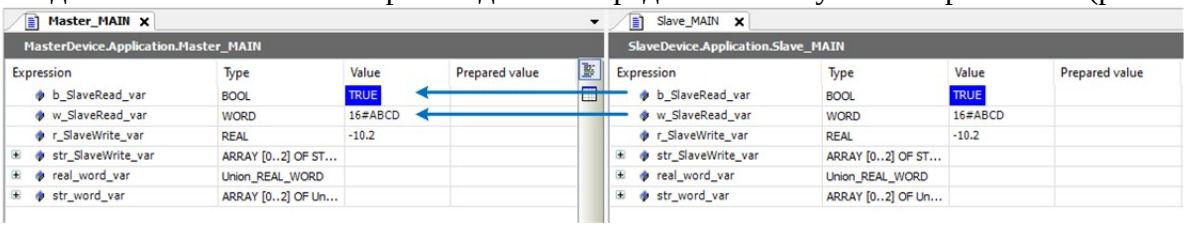

Рисунок 13 — Чтение данных из SlaveDevice