Данный проект демонстрирует пример использования библиотеки ModbusFB (CODESYS Forge - MODBUS / Home / Home) для настройки ПЛК NLScon-RSB в качестве ведомого устройства, а ПЛК Codesys Control Win V3 — в качестве ведущего в сети Modbus TCP. Для разработки проекта использована среда Codesys 3.5.16 patch 4 и установочный пакет компонентов для модулей RealLab! CoDeSys Linux package.

Обмен данными между ПЛК Codesys Control Win V3 и ПЛК NLScon-RSB осуществляется по сети Modbus TCP. Для организации обмена ПЛК NLScon-RSB (порт Ethernet 0) и компьютер, на котором запущена служба Codesys Control Win V3 (виртуальный ПЛК), подключены к одной локальной сети.

С помощью стандартного функционала Codesys в дерево устройств добавлены устройства MasterDevice (Codesys Control Win V3) и SlaveDevice (NLScon-RSB). (рис. 1).

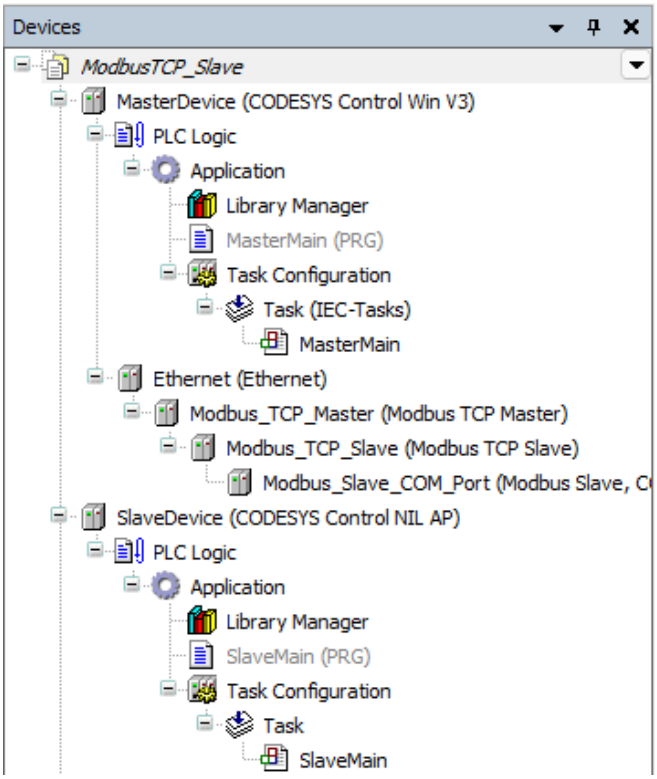


Рисунок 1 — Конфигурация дерева устройств для обмена данными в сети Modbus TCP

Для организации обмена с помощью функций библиотеки ModbusFB в Менеджере библиотек для ведомого устройства добавлены библиотеки ModbusFB, SysCom и Standard (рис. 2).

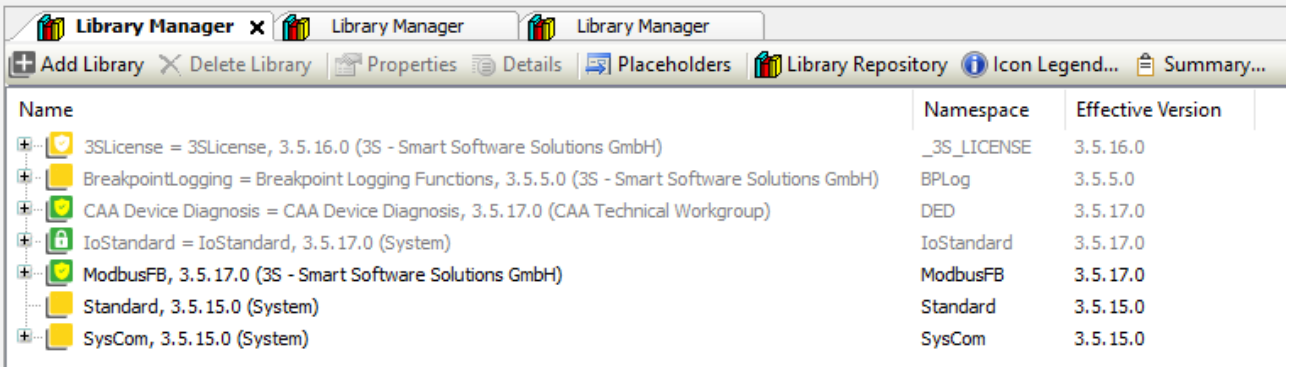


Рисунок 2 — Менеджер библиотек

Для настройки виртуального ПЛК Codesys Control Win V3 в качестве Modbus-мастера с помощью дерева устройств добавлены устройства (ПКМ на устройство MasterDevice, Add device):

1) Ethernet, настройка компонента приведена на рис. 3.

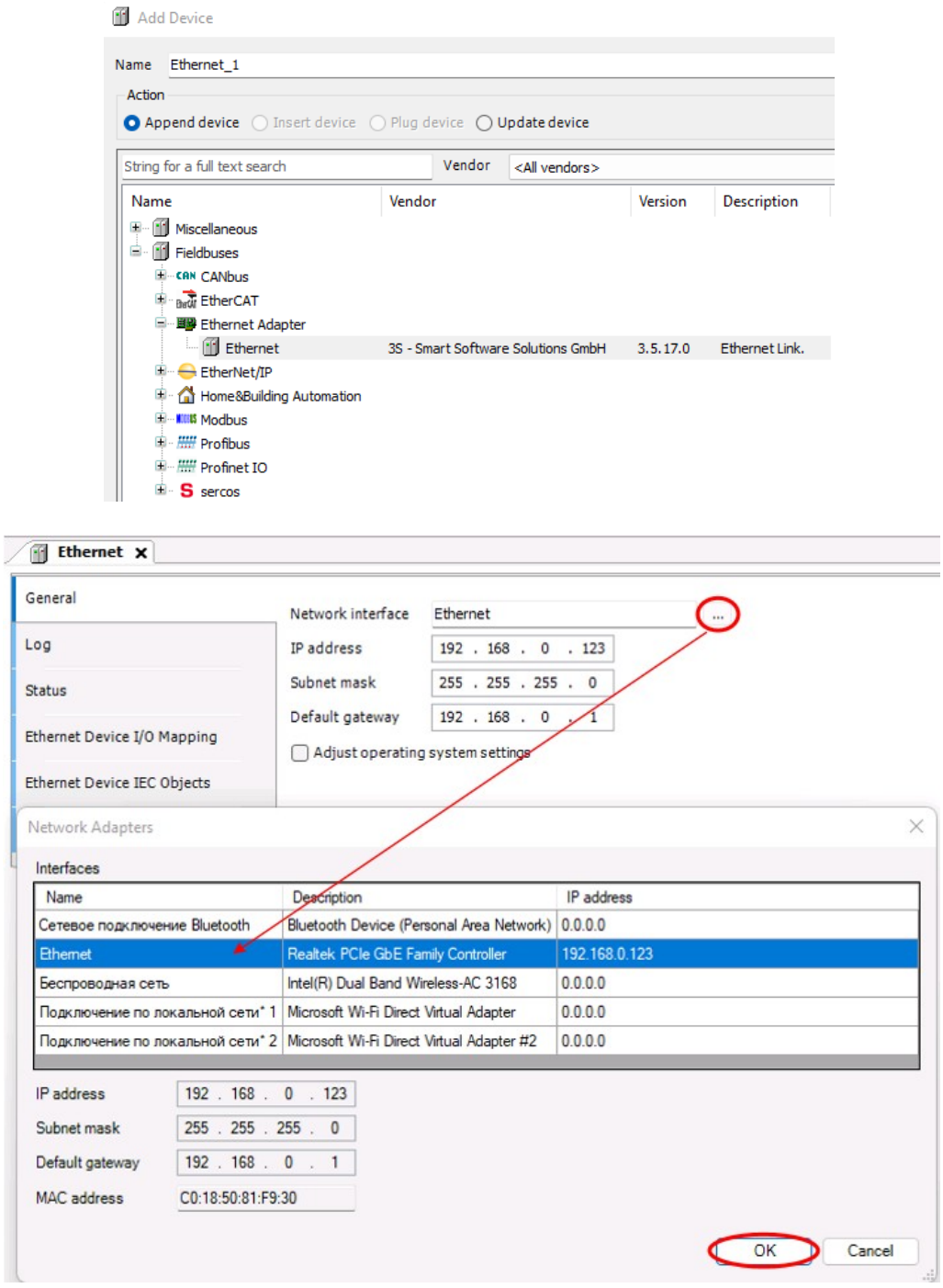


Рисунок 3 — Добавление и настройка компонента Ethernet дерева устройств

2) Modbus TCP Master. Для настройки компонента требуется выставить галочку Auto-reconnect (см. рис. 4).

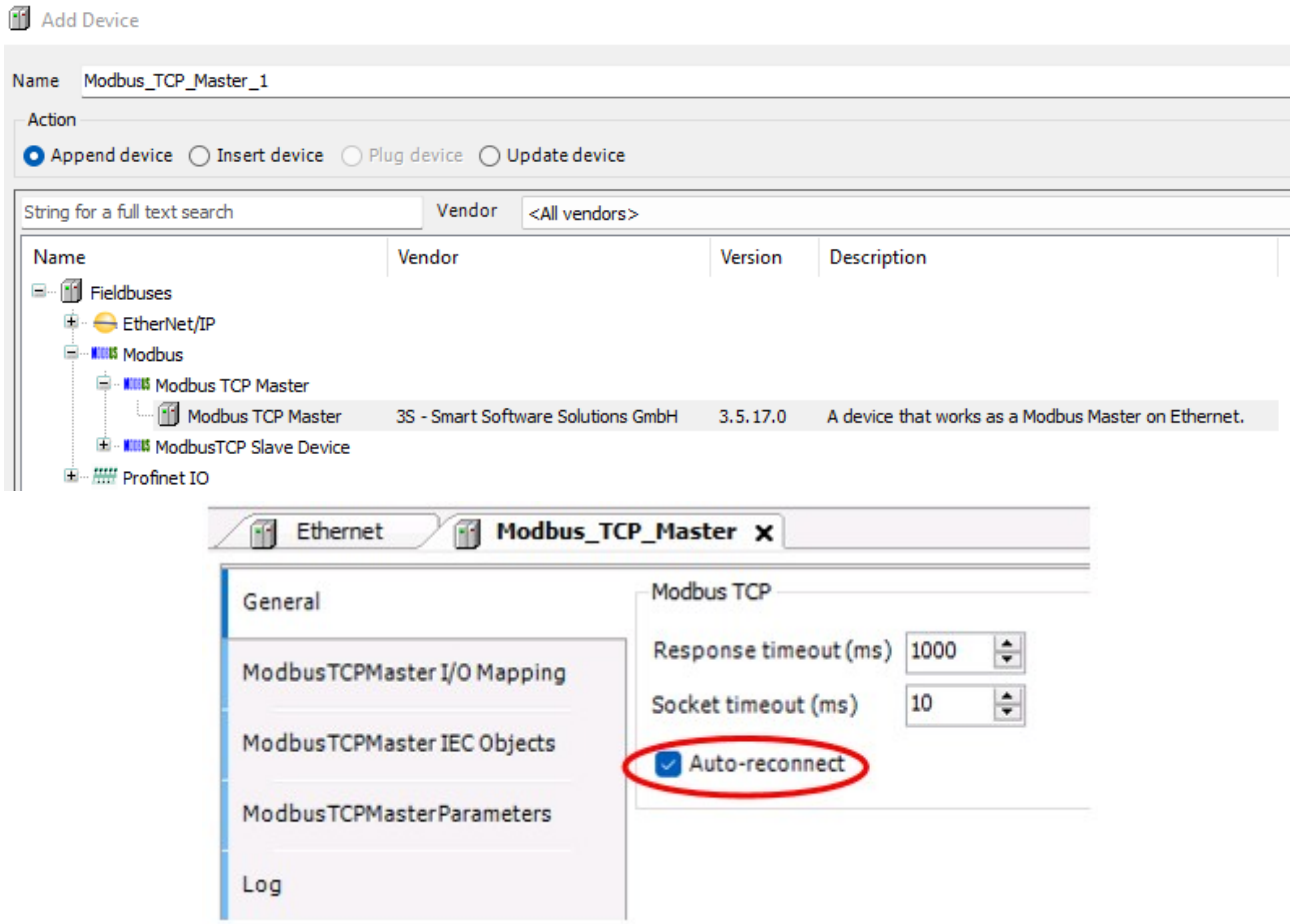


Рисунок 4 — Добавление и настройка компонента Modbus TCP Master дерева устройств

3) Modbus TCP Slave. Здесь в поле Slave IP address указываем IP-адрес Modbus TCP слева (см. рис. 5).

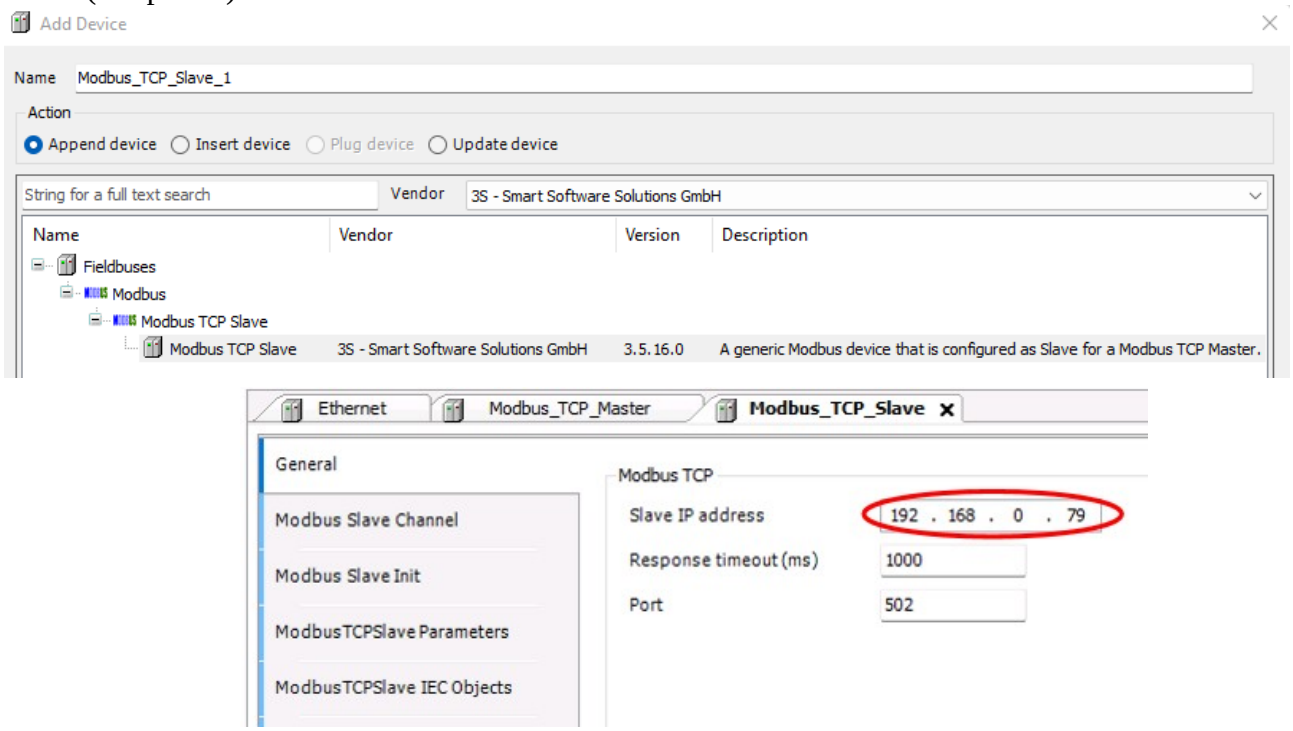


Рисунок 5 — Добавление и настройка компонента Modbus TCP Slave дерева устройств

4) Modbus_Slave_COM_Port. Адрес Slave address для ПЛК NLScon-RSB можно оставить равным 1. На вкладке Modbus Slave Channel нужно создать каналы опроса (см. рис. 6). После создания каналов нужно перейти на вкладку ModbusGenericSerialSlave I/O Mapping и в выпадающем списке Always update variables выбрать Enabled 2 (см. рис. 7).

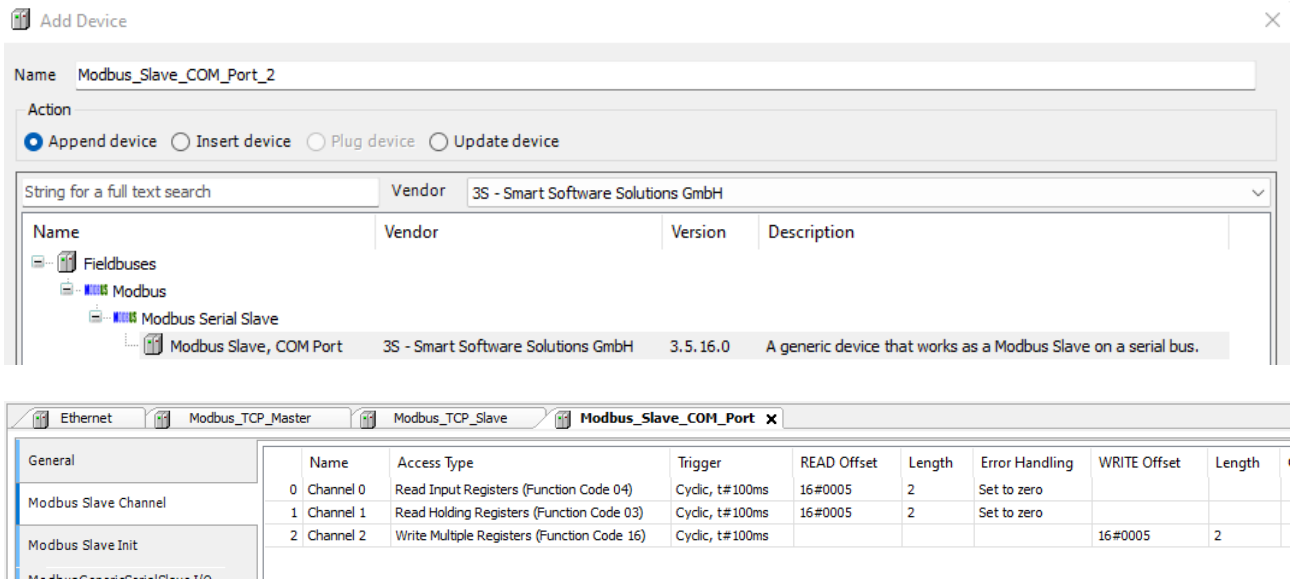


Рисунок 6 — Добавление и настройка компонента Modbus_Slave_COM_Port дерева устройств

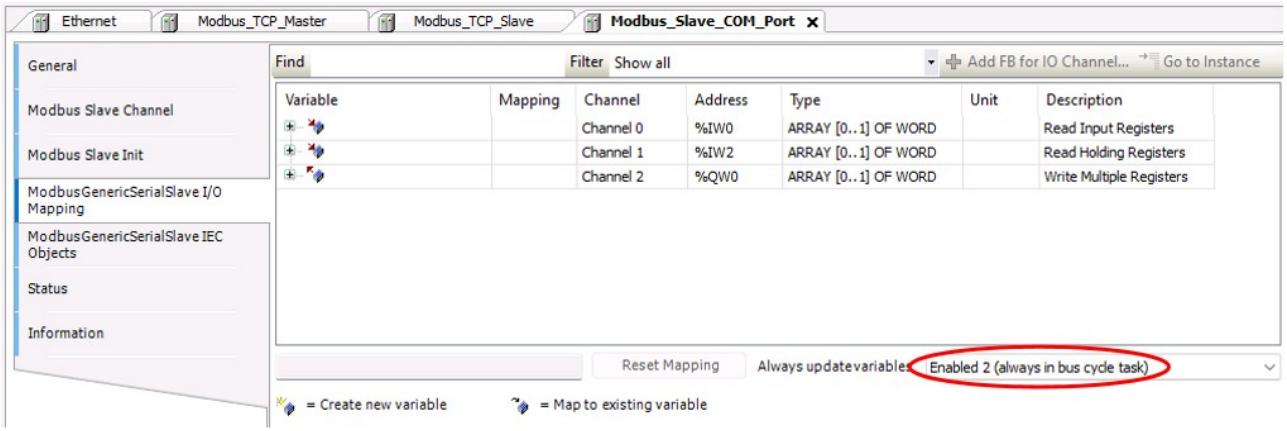


Рисунок 7 — Настройка компонента Modbus_Slave_COM_Port – обновление переменных

Для настройки ПЛК NLScon-RSB в качестве Modbus-слейва с помощью функционала библиотеки Modbus FB в программе SlaveMain выполнено следующее:

1) создан экземпляр функционального блока (ФБ) мастера (клиента) ModbusFB.ServerTCP;

```
serverTCP: ModbusFB.ServerTCP;
```

2) созданы экземпляр ФБ ModbusFB.ExampleDataModel для формирования модели данных слейва — набора регистров, доступных для обмена данными по Modbus RTU.

```
exampleDataModel : ModbusFB.ExampleDataModel;
```

Модель данных ModbusFB.ExampleDataModel является примером формирования набора регистров для слейва и включает следующие секции:
- discrete inputs;
- coils;
- input registers;
- holding registers.

Каждая секция включает по 2 поля данных и имеет начальный адрес 5 по умолчанию.

3) настройка разрешенных для слейва Modbus-функций:

```
fcsSupported : ModbusFB.SupportedFcs  := (
        ReadHoldingRegisters:=TRUE,
        ReadInputRegisters:=TRUE,
        WriteMultipleRegisters:=TRUE
        );
```

4) настройка слейва: настройка разрешенных Modbus-функций, привязка к модели данных слейва, символьное обозначение интерфейса Ethernet слейва, настройка номера Ethernet порта слейва; настройки вывода сообщений о ходе обмена данными в лог ПЛК. Настройки применятся по переднему фронту переменной xEnable

```
serverTCP(fcsSupported:=fcsSupported,
                dataModel:=exampleDataModel.tableDefs,
                xEnable:=FALSE,
                wsInterfaceName:="eth0",
                uiPort:=502,
                udiLogOptions:=(ModbusFB.LoggingOptions.ServerStartStop   OR
                ModbusFB.LoggingOptions.ServerReceivedValidRequests));
```

Узнать символьное имя сетевого интерфейса слейва можно, временно добавив компонент Ethernet в дереве устройств SlaveDevice и просмотрев параметры сетевого интерфейса (см. рис. 8). Сетевые параметры будут актуальны, если предварительно найти устройство SlaveDevice в сети (сканировать сеть). Здесь же можно узнать IP-адрес слейва, чтобы верно указать его в настройках компонента Modbus TCP Slave (см. рис. 5).

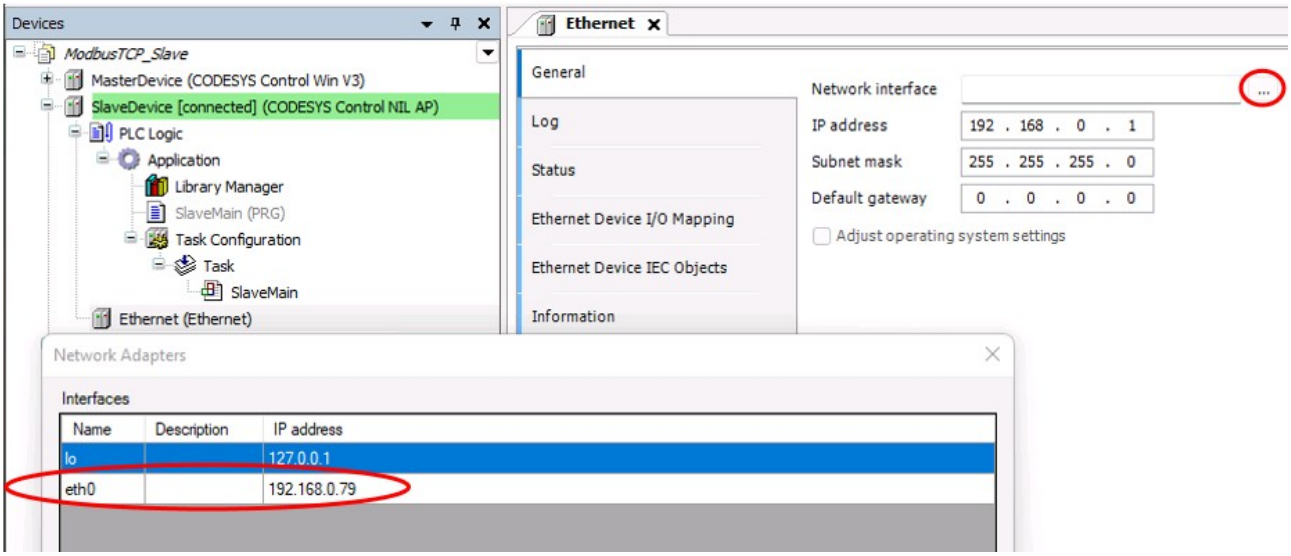


Рисунок 8 — Настройки сетевого интерфейса ПЛК NLScon-RSB

5) после того, как слейв настроен, требуется циклически вызывать экземпляр serverTCP ФБ ModbusFB.ServerTCP.

Вместо встроенной в библиотеку ModbusFB модели данных ModbusFB.ExampleDataModel можно настроить пользовательскую модель данных типа ModbusFB.TableDefinitions. Этот тип представляет собой структуру, включающую описание секций для четырех типов данных: discrete inputs, coils, input registers, holding registers.

В данном примере пользовательская модель данных реализована для устройства SlaveDevice_RegisterModel в программе SlaveMain (см. рис. 9).

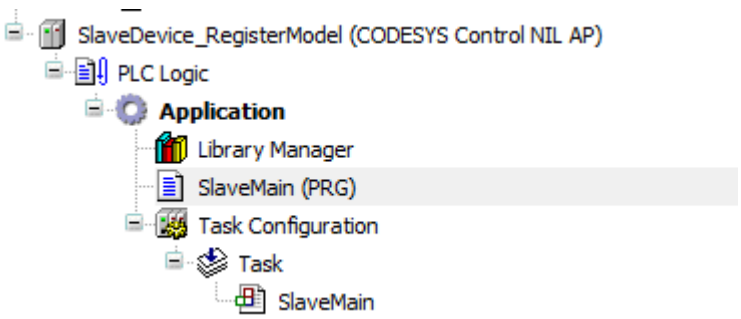


Рисунок 9 — Проект с пользовательской моделью данных в Modbus-слейв

В проекте создана пользовательская модель данных tableDefs типа ModbusFB.TableDefinitions (см. рис. 10). Эта модель в соответствии со своим типом состоит из четырех структур типа ModbusFB.TableDefinition:

- tableDiscreteInputs;
- tableCoils;
- tableInputRegisters;
- tableHoldingRegisters.

Каждая из перечисленных структур включает следующие поля: количество секций в структуре и указатель на первую секцию.

```
// пользовательская модель данных
tableDefs : ModbusFB.TableDefinitions := (
    tableDiscreteInputs := (
        uiNumSections := 1,
        pSections := ADR(aDiscreteInputsSections[0])
    ),
    tableCoils := (
        uiNumSections := 2,
        pSections := ADR(aCoilsSections[0])
    ),
    tableInputRegisters := (
        uiNumSections := 3,
        pSections := ADR(aInputRegistersSections[0])
    ),
    tableHoldingRegisters := (
        uiNumSections := 2,
        pSections := ADR(aHoldingRegistersSections[0])
    )
);
```

Рисунок 10 — Модель данных типа ModbusFB.TableDefinitions

Таким образом, для Modbus RTU слейва модель данных включает: 1 секцию «дискретных входов», 2 секции «катушек», 3 секции «входных регистров» и 2 секции «регистров хранения».

Секции задаются массивом из некоторого количества структур типа ModbusFB.TableSection, например, инициализация секций для «входных регистров» показана на рис. 11.

```
aInputRegistersSections : ARRAY [0..2] OF ModbusFB.TableSection := [
    // "input registers" 0..2
    (
        uiStart := 0,
        uiNumDataItems := 3,
        pStartAddr := ADR(aInputRegistersMemory1[0]),
        uiDataItemSize := 16
    ),
    // "input registers" 5..8
    (
        uiStart := 5,
        uiNumDataItems := 4,
        pStartAddr := ADR(aInputRegistersMemory1[3]),
        uiDataItemSize := 16
    ),
    // "input registers" 10..12
    (
        uiStart := 10,
        uiNumDataItems := 3,
        pStartAddr := ADR(aInputRegistersMemory2[0]),
        uiDataItemSize := 16
    )
];
```

Рисунок 11 — Таблица «входных регистров» типа ModbusFB.TableSection

В данном случае секции для «входных регистров» представляют собой 3 структуры типа ModbusFB.TableSection, которые описываются параметрами:

- начальный адрес uiStart;

- количество uiNumDataItems полей данных в секции (бит/байт/слов — в зависимости от назначенияч секции);

- указатель pStartAddr на область памяти в ПЛК, выделенную для этой секции;

- размер uiDataItemSize в битах поля данных. Для discrete inputs/coils значение uiDataItemSize может быть равно 1 или 8 бит, для input/holding registers значение uiDataItemSize равно 16 бит. Выбор размера для discrete inputs/coils зависит от того, как они привязаны к области памяти в ПЛК: если выделенная область памяти — это ARRAY [0..x] OF BOOL, то uiDataItemSize=8, если discrete inputs/coils адресуются побитово, то uiDataItemSize=1. Для input/holding registers область памяти должна быть массивом ARRAY [0..x] OF UINT.

В данном примере для секции «входных регистров» выделены области памяти:

aInputRegistersMemory1 : ARRAY [0..6] OF UINT;
aInputRegistersMemory2 : ARRAY [0..2] OF UINT;

В нашем случае «входные регистры» для ПЛК настроены следующим образом:

- первая секция:
  начальный адрес 0,
  количество полей в секции 3 штуки (3 регистра),
  их значения будут хранится в первых трех элементах массива aInputRegistersMemory1[0..2],
  размерность каждого «входного регистра» — 16 бит.
- вторая секция:
  начальный адрес 5,
  количество полей в секции 4 штуки (4 регистра),
  их значения будут хранится в последних четырех элементах массива aInputRegistersMemory1[3..6],
  размерность каждого «входного регистра» — 16 бит.
- третья секция:
  начальный адрес 10,
  количество полей в секции 3 штуки (3 регистра),
  их значения будут хранится в первых трех элементах массива aInputRegistersMemory2[0..2],
  размерность каждого «входного регистра» — 16 бит.

Подобным образом в программе SlaveMain объявлены секции для discrete inputs, coils и holding registers. Для всех секций в программе должны быть выделены области памяти (см. рис. 12).

```
VAR
    // области памяти, которые ниже будут привязаны к секциям модели данных слейва
    aDiscreteInputsMemory : ARRAY [0..9] OF BOOL;
    aCoilsMemory1 : ARRAY [0..4] OF BOOL;
    aCoilsMemory2 : ARRAY [0..4] OF BOOL;
    aInputRegistersMemory1 : ARRAY [0..6] OF UINT;
    aInputRegistersMemory2 : ARRAY [0..2] OF UINT;
    aHoldingRegistersMemory : ARRAY [0..6] OF UINT;
END_VAR
```

Рисунок 12 — Области памяти для пользовательской модели данных ПЛК

Чтобы запустить Modbus RTU слейв на ПЛК, потребуется настроить serverTCP (экземпляр ФБ ModbusFB.ServerTCP) и разрешить его работу:

1) настройка слейва: настройка разрешенных Modbus-функций, привязка к модели данных слейва, символьное обозначение интерфейса Ethernet слейва, настройка номера Ethernet порта слейва; настройки вывода сообщений о ходе обмена данными в лог ПЛК.

Настройки применятся по переднему фронту переменной xEnable

```
serverTCP(fcsSupported:=fcsSupported,
          dataModel:=exampleDataModel.tableDefs,
          xEnable:=FALSE,
          wsInterfaceName:="eth0",
          uiPort:=502,
          udiLogOptions:=(ModbusFB.LoggingOptions.ServerStartStop   OR
          ModbusFB.LoggingOptions.ServerReceivedValidRequests));
```

Узнать символьное имя сетевого интерфейса слейва можно, временно добавив компонент Ethernet в дереве устройств SlaveDevice_RegisterModel и просмотрев параметры сетевого интерфейса (см. рис. 8). Сетевые параметры будут актуальны, если предварительно найти устройство SlaveDevice_RegisterModel в сети (сканировать сеть). Здесь же можно узнать IP-адрес слейва, чтобы верно указать его в настройках компонента Modbus TCP Slave (см. рис. 5).

2) после того, как слейв настроен, требуется циклически вызывать экземпляр serverTCP ФБ ModbusFB.ServerTCP.

Для опроса слейва с пользовательской моделью данных нужно заменить в настройках каналов Modbus_Slave_COM_Port начальный адрес и количество регистров (при необходимости).