

Данный проект демонстрирует пример использования ПЛК NLScon-RSB и ПЛК NLScon-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).

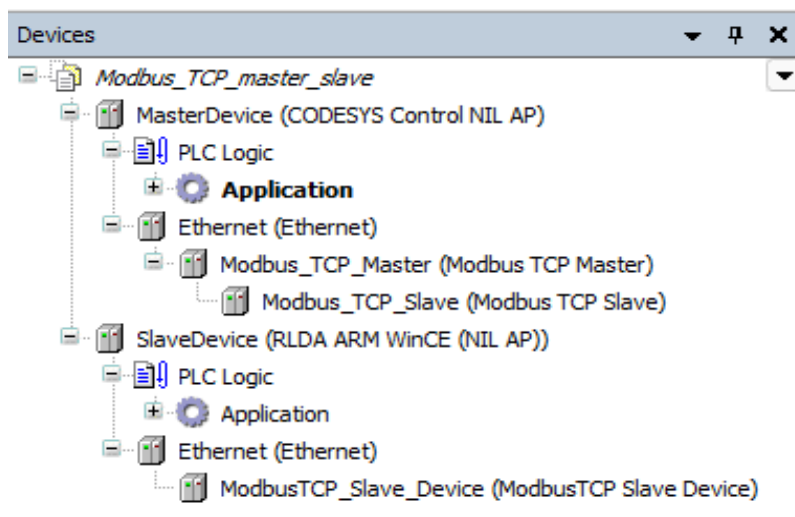


Рисунок 1 — Конфигурация дерева устройств для обмена данными между ПЛК RealLab! по сети Modbus TCP

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).

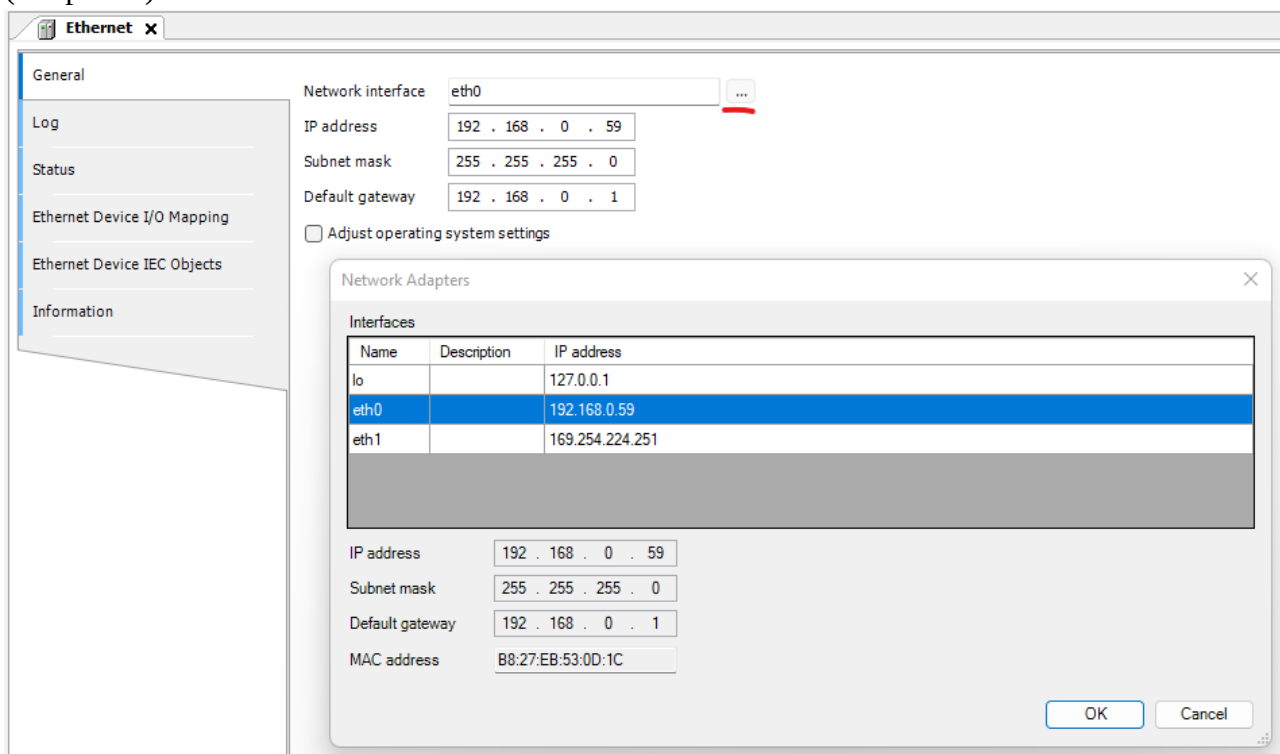


Рисунок 2 — Настройка параметров компонента Ethernet

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).

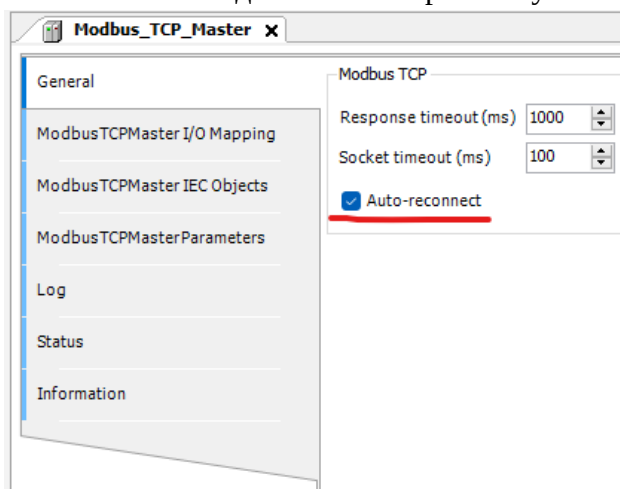


Рисунок 3 — Настройка параметров компонента Modbus_TCP_Master в дереве MasterDevice

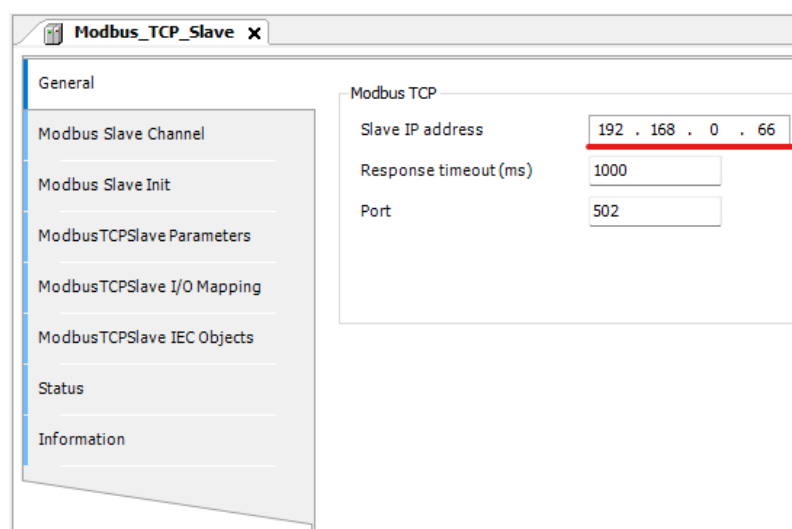


Рисунок 4 — Настройка параметров компонента Modbus_TCP_Slave в дереве MasterDevice

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)»).

Modbus_TCP_Slave								
General	Name	Access Type	Trigger	READ Offset	Length	Error Handling	WRITE Offset	Length
Modbus Slave Channel	0 Read_BOOL	Read Discrete Inputs (Function Code 02)	Cyclic, t#100ms	16#0000	1	Set to zero		
Modbus Slave Init	1 Read_WORD	Read Input Registers (Function Code 04)	Cyclic, t#100ms	16#0001	1	Set to zero		
ModbusTCPSlave Parameters	2 Write_REAL	Write Multiple Registers (Function Code 16)	Cyclic, t#100ms				16#0000	2
	3 Write_STRING	Write Multiple Registers (Function Code 16)	Cyclic, t#100ms				16#0005	3

Рисунок 5 — Настройка каналов компонента Modbus_TCP_Slave в дереве MasterDevice

Modbus_TCP_Slave							
General	Find	Filter	Show all	Add FB for IO Channel... Go to Instance			
Modbus Slave Channel	Variable	Mapping	Channel	Address	Type	Unit	Description
Modbus Slave Init			Read_BOOL	%IB0	ARRAY [0..0] OF BYTE		Read Discrete Inputs
ModbusTCPSlave Parameters			Read_BOOL[0]	%IB0	BYTE		Read Discrete Inputs
ModbusTCPSlave I/O Mapping	Application.Master_MAIN.b_slaveRead_var		Bit0	%B0-9	BOOL		0x0000
ModbusTCPSlave IEC Objects			Read_WORD	%IW1	ARRAY [0..0] OF WORD		Read Input Registers
Status	Application.Master_MAIN.w_slaveRead_var		Read_WORD[0]	%IW1	WORD		0x0001
Information	Application.Master_MAIN.real_word_var.word_var		Write_REAL	%QW0	ARRAY [0..1] OF WORD		Write Multiple Registers
			Write_REAL[0]	%QW0	WORD		0x0000
			Write_REAL[1]	%QW1	WORD		0x0001
			Write_STRING	%QW2	ARRAY [0..2] OF WORD		Write Multiple Registers
	Application.Master_MAIN.str_word_var[0].word_var		Write_STRING[0]	%QW2	WORD		0x0005
	Application.Master_MAIN.str_word_var[1].word_var		Write_STRING[1]	%QW3	WORD		0x0006
	Application.Master_MAIN.str_word_var[2].word_var		Write_STRING[2]	%QW4	WORD		0x0007

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).

ModbusTCP_Slave_Device	
General	Configured Parameters
Serial Gateway	<input type="checkbox"/> Watchdog 500 (ms)
Modbus TCP Slave Device I/O Mapping	Slave port 502 <input type="checkbox"/> Bind to Adapter
Modbus TCP Slave Device IEC Objects	Holding Registers 10 (%IW) <input type="checkbox"/> Writeable
Status	Input Registers 10 (%QW)
Information	<input checked="" type="checkbox"/> Discrete Bit Areas
	Coils 0 (%IX)
	Discrete Inputs 10 (%QX)
	Data Model
	StartAddresses
	Coils 0
	Discrete inputs 0
	Holding register 0
	Input register 0
	<input type="checkbox"/> Holding- and input register data areas overlay

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) — регистры 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.

Variable	Mapping	Channel	Address	Type	Unit	Description
Application.Slave_MAIN.real_word_var.word_var[0]		Holding Registers	%IW0	ARRAY [0..9] OF WORD		
Application.Slave_MAIN.real_word_var.word_var[1]		Holding Registers[0]	%IW0	WORD		
		Holding Registers[1]	%IW1	WORD		
		Holding Registers[2]	%IW2	WORD		
		Holding Registers[3]	%IW3	WORD		
		Holding Registers[4]	%IW4	WORD		
Application.Slave_MAIN.str_word_var[0].word_var		Holding Registers[5]	%IW5	WORD		
Application.Slave_MAIN.str_word_var[1].word_var		Holding Registers[6]	%IW6	WORD		
Application.Slave_MAIN.str_word_var[2].word_var		Holding Registers[7]	%IW7	WORD		
		Holding Registers[8]	%IW8	WORD		
		Holding Registers[9]	%IW9	WORD		
		Input Registers	%QW0	ARRAY [0..9] OF WORD		
Application.Slave_MAIN.w_SlaveRead_var		Input Registers[0]	%QW0	WORD		
		Input Registers[1]	%QW1	WORD		
		Input Registers[2]	%QW2	WORD		
		Input Registers[3]	%QW3	WORD		
		Input Registers[4]	%QW4	WORD		
		Input Registers[5]	%QW5	WORD		
		Input Registers[6]	%QW6	WORD		
		Input Registers[7]	%QW7	WORD		
		Input Registers[8]	%QW8	WORD		
		Input Registers[9]	%QW9	WORD		
		Discrete Inputs	%QB0	ARRAY [0..1] OF BYTE		
		Discrete Inputs[0]	%QB0	BYTE		
Application.Slave_MAIN.b_SlaveRead_var		Bit0	%QX20.0	BOOL		
		Bit1	%QX20.1	BOOL		
		Bit2	%QX20.2	BOOL		
		Bit3	%QX20.3	BOOL		
		Bit4	%QX20.4	BOOL		
		Bit5	%QX20.5	BOOL		
		Bit6	%QX20.6	BOOL		
		Bit7	%QX20.7	BOOL		
		Discrete Inputs[1]	%QB21	BYTE		
		Bit0	%QX21.0	BOOL		
		Bit1	%QX21.1	BOOL		

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).

