ООО «АСУ ПРО»

Модуль процессорный

КАПП2-00-000-1

Руководство по эксплуатации

73619730.26.20.30.000.020 РЭ

/Редакция 1.4/

Производитель:
ООО «АСУ ПРО»
460000, Оренбургская область, г.о. город Оренбург, г. Оренбург,
улица Черепановых, дом 7
Тел./факс: +7 (3532) 689-088, 689-241

E-mail: asupro@asupro.ru

г. Оренбург 2024 г.

СОДЕРЖАНИЕ

Согласовано

Подп. и дата

Инв. № подл.

Изм.	Кол.уч.	Лист	№ док.	Подпись	Дата
Разработал		Тимонов Е.С.			

73619730.26.20.30.000.020 РЭ

Модуль процессорный
КАПП2-00-000-1
Руководство по эксплуатации

Лит	Лист	Листов
	2	168

ООО «АСУ ПРО»

Согласовано

Подп. и дата

Инв. № подл.

Изм.	Кол.уч.	Лист	№ док.	Подпись	Дата
Разработал		Тимонов Е.С.			

73619730.26.20.30.000.020 РЭ

Модуль процессорный
КАПП2-00-000-1
Руководство по эксплуатации

Лит	Лист	Листов
	2	168

ООО «АСУ ПРО»

Руководство по эксплуатации содержит сведения, необходимые для обеспечения правильной эксплуатации и полного использования технических возможностей модуля процессорного КАПП2-00-000-1.

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

Изм.	Кол.уч.	Лист	№ док.	Подпись	Дата	73619730.26.20.30.000.020 РЭ	Лист 3

ФорматА4

1 Описание и работа изделия

1.1 Назначение

Настоящие технические условия распространяются на модуль процессорный КАПП2 00-000-1 (в дальнейшем - КАПП2), предназначен для обработки данных и выдачи сигналов управления в соответствии с прикладной программой, обмена данными с верхним уровнем АСУ ТП по Ethernet, подключенным к интерфейсам RS-232 и RS-485 оборудованием и приборам, а также между модулями КАПП2 по интерфейсу RS-485 (TBUS).

Модуль может применяться на объектах нефтяной, газовой и нефтехимической промышленности, а также в других областях промышленности для использования в составе автоматизированных измерительных и управляющих систем различной конфигурации.

1.2 Технические характеристики

1.2.1 Основные технические характеристики модуля приведены в таблицах 1-11.

Таблица 1 общие характеристики

№	Характеристика	Значение
1	Процессорное ядро	ARM Cortex M7
2	Количество бит на одно слово	32
3	Тактовая частота процессорного ядра, МГц	480
4	Тип памяти для хранения программ (ПЗУ)	Flash
5	Объем памяти для хранения программ (ПЗУ), кбайт	4096
6	Тип памяти для хранения данных (ПЗУ)	Карта microSD, microSDHC
7	Структура памяти для хранения данных (ПЗУ)	FAT32, с кластером 512 байт
8	Объем памяти для хранения данных (ПЗУ)	до 32 Гбайт
9	Тип памяти для хранения данных (ОЗУ)	SDRAM
10	Объем памяти для хранения данных (ОЗУ)	24 Мбайт
11	Тип памяти для сохранения данных состояния контроллера на случай отключения питания (сохранение в конце каждого цикла)	FRAM
12	Объем памяти для сохранения данных состояния контроллера на случай отключения питания (сохранение в конце каждого цикла)	128 Кбайт
13	Максимальное число подключаемых модулей	31
14	Среда разработки прикладных программ	CODESYS v3
15	Поддерживаемые протоколы	Modbus RTU
		Modbus TCP
		NTP v4.0
		FTP
		МЭК 60870-5-101
		МЭК 60870-5-104

16	Возможность повторного запуска		холодный	ДА
			теплый	НЕТ
			горячий	НЕТ
17	Время запуска модуля, с			8
18	Минимальное время цикла, мс			1
19	Световые индикаторы (двухцветные)	самодиагностика («ИСПР.»)	Нормальная работа	Постоянно горит зеленым светом
			Ошибка	Мигает красным светом
20	Световые индикаторы	Запуск приложения пользователя («ПУСК»)	Работает только операционная система	Индикатор выключен
			Приложение запущено	Мигает зеленым светом
21	Метод замены модуля			Замена при отключенном питании
22	Общая шина			Phoenix contact TBUS ME 22.5 (с питанием)
23	Связь с модулями ввода/вывода			Через общую шину RS-485
24	Источник питания			Внешний
25	Кнопка «СБРОС» (пленочная)		- при нажатии на кнопку	Происходит полная перезагрузка контроллера
			- при нажатии на кнопку с последующим удержанием кнопки «Старт/Стоп»	Происходит перезагрузка контроллера с удалением пользовательской программы
26	Кнопка «Старт/Стоп» (пленочная, без фиксации)		- при нажатии на кнопку	Приложение пользователя запускается или останавливается, при этом операционная система остаётся в работе.

Таблица 2 физические условия окружающей среды для рабочих условий эксплуатации

№	Характеристика		Значение
1	Температура окружающего воздуха, °С	максимальная	70
2		минимальная	минус 40
3	Относительная влажность окружающего воздуха, %	максимальная	95 (без конденсации)
4		минимальная	10

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

5	Атмосферное давление, кПа	максимальное	106,7
6		минимальное	79,5 (эквивалентно высоте над уровнем моря 2000 м)

Таблица 3 физические условия окружающей среды для транспортировки и хранения

№	Характеристика		Значение
1	Температура окружающего воздуха, °С	максимальная	70
2		минимальная	минус 40
3	Относительная влажность окружающего воздуха, %	максимальная	95 (без конденсации)
4		минимальная	10
5	Атмосферное давление, кПа	максимальное	106,7
6		минимальное	70 (эквивалентно высоте над уровнем моря 3000 м)

Таблица 4 нормальные условия эксплуатации

№	Характеристика		Значение
1	Температура окружающего воздуха, °С		23 ± 5
2	Относительная влажность окружающего воздуха, %	максимальная	80
3		минимальная	30
4	Атмосферное давление, кПа	максимальное	106,7
5		минимальное	84

Таблица 5 параметры защиты

№	Характеристика	Значение
1	Степень защиты корпуса модуля от проникновения твёрдых предметов, пыли и воды в соответствии с ГОСТ 14254-96	IP20
2	Степень загрязнения по ГОСТ IEC 61131-2-2012 при которой модуль работоспособен	1

Таблица 6 номинальные значения и рабочие диапазоны электропитания

№	Характеристика		Значение
1	Номинальное напряжение, В		24
2	Род тока		Постоянный
3	Предельное отклонение от номинального	максимальное Umax, %	+20 (28,8 В)
4		минимальное Umin, %	-15 (20,4 В)
5	Пиковая мощность потребления, Вт		0,9
6	Общая переменная составляющая с пиковым значением от номинального до, %		5

Таблица 6 резервное электропитание запоминающих устройств (ЗУ)

№	Характеристика		Значение
1	Обеспечение резервного	при рабочих условиях эксплуатации не менее, ч	100000

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

2	электропитания энергозависимой памяти для сохранения информации	при температуре не выше 25 °С в случае, если источник энергии номинальной мощности, ч	100000
3	Замена источника резервного электропитания	интервал замены, лет	10
4		рекомендуемый метод замены	Замена только при включенном питании
5		влияние замены на модуль	При включенном питании влияние отсутствует, при отключенном теряются данные в ЗУ

Таблица 7 характеристики интерфейса RS-232

№	Характеристика		Значение
1	Количество интерфейсов	Неизолированных	1 шт.: COM 1 (сигналы: RxD, TxD, RTS, CTS, GND)
2	Скорость передачи данных	максимальная	115,2 кбит/с
3		минимальная	2,4 кбит/с
4	Протокол связи		Modbus RTU
5	Характеристики кабеля	длина не более, м	15

Таблица 8 характеристики интерфейса RS-485

№	Характеристика		Значение
1	Количество интерфейсов	Изолированных	2 шт.: COM 2, X1
2	Встроенный резистор для согласования драйвера с кабелем с волновым сопротивлением 120 Ом		120 Ом
3	Подключение встроенного резистора		С помощью джампера
4	Режим передачи данных		Полудуплекс
5	Скорость передачи данных	максимальная	115,2 кбит/с
6		минимальная	2,4 кбит/с
7	Число абонентов (нагрузочная способность), шт		до 31
8	Протокол связи		Modbus RTU
9	Характеристики кабеля	длина не более, м	1200

Таблица 9 характеристики интерфейса RS-485 (TBUS)

№	Характеристика		Значение
1	Количество интерфейсов	Неизолированных	1 шт.
2	Встроенный резистор для согласования драйвера с кабелем с волновым сопротивлением 120 Ом		120 Ом
3	Подключение встроенного резистора		С помощью джампера
4	Режим передачи данных		Полудуплекс

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

5	Скорость передачи данных	максимальная	115,2 кбит/с
6		минимальная	2,4 кбит/с
7	Число абонентов (нагрузочная способность), шт		до 31
8	Протокол связи		Modbus RTU
9	Характеристики кабеля	длина не более, м	1200

Таблица 10 характеристики интерфейса Ethernet

№	Характеристика		Значение
1	Количество интерфейсов		1
2	Протокол связи		Modbus TCP
3	Скорость передачи данных		до 100 Мбит/с
4	Характеристики кабеля	тип кабеля	UTP
5		категория кабеля	5
6		длина не более, м	100

Таблица 11 массогабаритные характеристики

№	Характеристика	Значение
1	Габаритные размеры (длина×ширина×высота), мм	99×45×114,5
2	Масса, кг, не более	0,14

1.2.2 Показатели надежности (безотказности):

- средняя наработка на отказ в нормальных условиях с учетом технического обслуживания, предусмотренного настоящим руководством, не менее 130000 ч.
- срок службы не менее 10 лет.

1.3 Состав изделия

1.3.1 Модуль изготавливается в пластмассовом корпусе, предназначенном для крепления на DIN-рейку 35мм. Подключение всех внешних связей осуществляется через разъемные соединения, расположенные по двум сторонам модуля. Открытие корпуса для подключения внешних связей не требуется.

Разъемы модуля:

– TBUS – питание 24В, RS-485;
– X1– вход для подключения RS-485;
– Ethernet – для подключения TCP/IP (8P8C);
– COM1 – для подключения RS-232 (6P6C);
– COM2 – для подключения RS-485 (6P6C);

Индикация:

– Пуск;
– Исправность.

Кнопки:

– «Сброс»;
– «Старт/Стоп».

1.3.2 Комплект поставки модуля приведен в таблице 12.

Таблица 12

№	Наименование	Обозначение	Кол-во, шт.
1	Модуль процессорный	КАПП2-00-000-1	1
2	Руководство по эксплуатации	73619730.26.20.30.000.020 РЭ	1
3	Паспорт	73619730.26.20.30.000.020 ПС	1

1.4 Устройство и работа

Модуль процессорный выполняет функции исполнения прикладного программного обеспечения среды Codesys, связи с модулями ввода – вывода и другими устройствами.

Устройство изготавливается в пластмассовом корпусе, предназначенном для крепления на DIN-рейку 35мм. Подключение модулей ввода – вывода КАПП2 и источника питания осуществляется через шину TBUS. Внешние соединения по TCP/IP, RS-232, RS-485 осуществляется по соответствующим портам, расположенным на лицевой стороне модуля (Ethernet, COM1, COM2), а так же в нижней части модуля (X1). Открытие корпуса для подключения внешних связей не требуется.

Шина TBUS (рисунок 1) отвечает за питание и обмен данными между модулями и процессорным модулем, представлена 5-ти контактным клеммным соединителем, крепящимся на DIN-рейку, поверх которого устанавливается модуль.

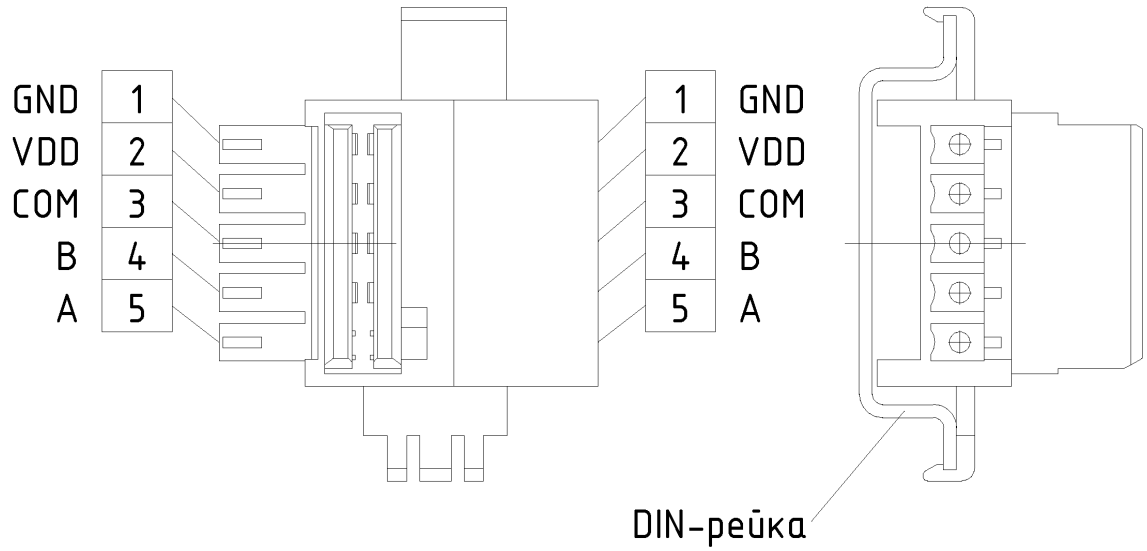


Рисунок 1а - Шина TBUS

Шина TBUS состоит из 3-х линий связи по интерфейсу RS-485 (Modbus RTU) и 2-х линий питания модулей (24 В постоянного напряжения).

Внешний вид устройства представлен в приложении А. Индикация режимов работы и кнопки «Сброс», «Старт/Стоп» располагаются на передней панели.

Индикатор «Испр.» постоянно горит зеленым светом, индицируя наличие питания на шине TBUS и корректном напряжении питания.

Программирование и конфигурирование модуля производится с помощью среды Codesys.

1.5 Маркировка и пломбирование

Состав и содержание основных маркировочных данных:
− функциональная схема модуля;
− номера разъемов;
− наименование страны происхождения;
− логотип производителя;
− наименование модуля: КАПП2-00-000-1;
− заводской номер, присвоенный модулю при изготовлении;
− дата изготовления;
− условия эксплуатации;
− IP.
Маркировочная табличка располагается на боковой стороне корпуса модуля.
Пломбирование не предусмотрено.

1.6 Упаковка

1.6.1 Упаковывание модуля производится в закрытом помещении при температуре окружающего воздуха от 15 °С до 40 °С и относительной влажности до 80 % по ГОСТ 23170-78. Модули, прошедшие консервацию, обернутые упаковочной бумагой по ГОСТ 8273-75, упаковываются в потребительскую тару (в коробки из гофрированного картона по ГОСТ Р 52901-2007). Пространство между устройствами и стенками потребительской тары должно быть уплотнено.

1.6.2 Принятые представителем заказчика модули должны быть упакованы отдельно в транспортную тару (коробки из гофрированного картона), плотно заполняя в них свободные места. В каждую коробку должен вкладываться упаковочный лист.

1.6.3 Сопроводительная документация (эксплуатационная (п. 2-4 таблицы 10) и товаросопроводительная) должна быть уложена в пакеты из полиэтиленовой пленки по ГОСТ 10354-82, которые помещают в транспортную тару.

2 Использование по назначению

2.1 Эксплуатационные ограничения

Модуль должен эксплуатироваться:
- в закрытых помещениях или шкафах электрооборудования, конструкция которых должна обеспечивать защиту модуля от попадания на контакты выходных разъемов и внутренних элементов влаги, грязи, пыли и посторонних предметов (см. таблицу 5);
- при физических условиях окружающей среды указанных в таблице 2, запрещается использование модуля при наличии в окружающей среде кислот, щелочей, масел и иных агрессивных веществ.

2.2 Подготовка изделия к использованию

2.2.1 Монтаж модуля

Подготовить место в шкафу электрооборудования. Укрепить модуль на DIN-рейку защелкой вниз.

Рекомендуемые расстояния при монтаже:

- между модулями в ряду: не имеет значения;
- между рядом модулей и кабельным каналом: не менее 30 мм.

При размещении модуля следует помнить, что при эксплуатации открытые контакты клемм могут находиться под напряжением, опасным для человеческой жизни. Доступ внутрь таких шкафов разрешен только квалифицированным специалистам.

2.2.2 Монтаж внешних связей

2.2.2.1 Питание модуля следует осуществлять от локального блока питания подходящей мощности, установленного совместно с модулем в шкафу электрооборудования. Во внешней цепи блока питания рекомендуется установить выключатель, обеспечивающий отключение модуля от сети. Подключение питания осуществляется через шину TBUS (см. рисунок 2).

2.2.2.2 Подключение модулей ввода - вывода выполняется по интерфейсу RS-485 шины TBUS по трехпроводной схеме. Подключение производить при отключенном напряжении питания всех устройств сети RS-485. Длина линии связи должна быть не более 1200 метров. Подключение следует осуществлять витой парой проводов, соблюдая полярность. Провод А подключается к выводу А шины TBUS, аналогично соединяются выводы В.

2.2.2.3 Подключение интерфейса Ethernet производится 8-ми жильным кабелем «витая пара» категории 5. На кабель установить оконечные разъемы типа RJ-45 (8P8C). Ответную часть кабеля подключить к Ethernet-концентратору, к сетевой плате компьютера или к иному оборудованию. При подключении к концентратору используется обычный (прямой) кабель, при подключении к сетевой плате или к иному оборудованию используется кабель с перекрестным монтажом, если устройства не поддерживают Auto MDI-X (автоматическое определение пар на прием и передачу).

2.2.2.4 Подключение интерфейса RS-232 выполняется через COM1, расположенный на лицевой панели контроллера, кабелем с оконечными разъемами типа RJ-25 6P6C.

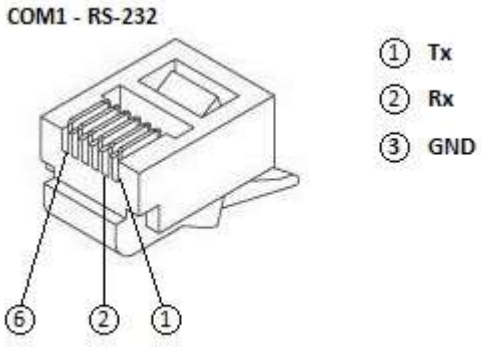


Рисунок 1б – RS-232

2.2.2.5 Подключение интерфейса RS-485 выполняется по трехпроводной схеме. Подключение производить при отключенном напряжении питания всех устройств сети RS-485. Длина линии связи должна быть не более 1200 метров. Подключение следует осуществлять витой парой проводов, соблюдая полярность. Провод А подключается к выводу А контроллера, аналогично соединяются выводы В. Подключение производить при отключенном питании всех устройств в линии RS-485. Для подключения через COM2 выполняется кабелем с оконечными разъемами типа RJ-25 6P6C.

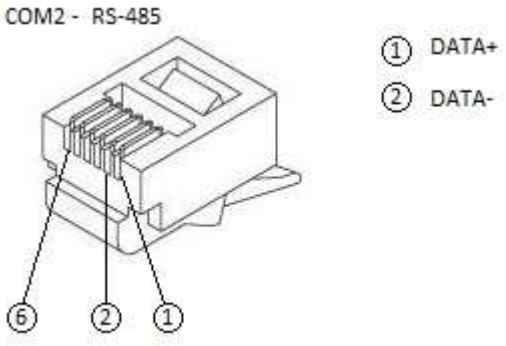


Рисунок 1в – RS-485

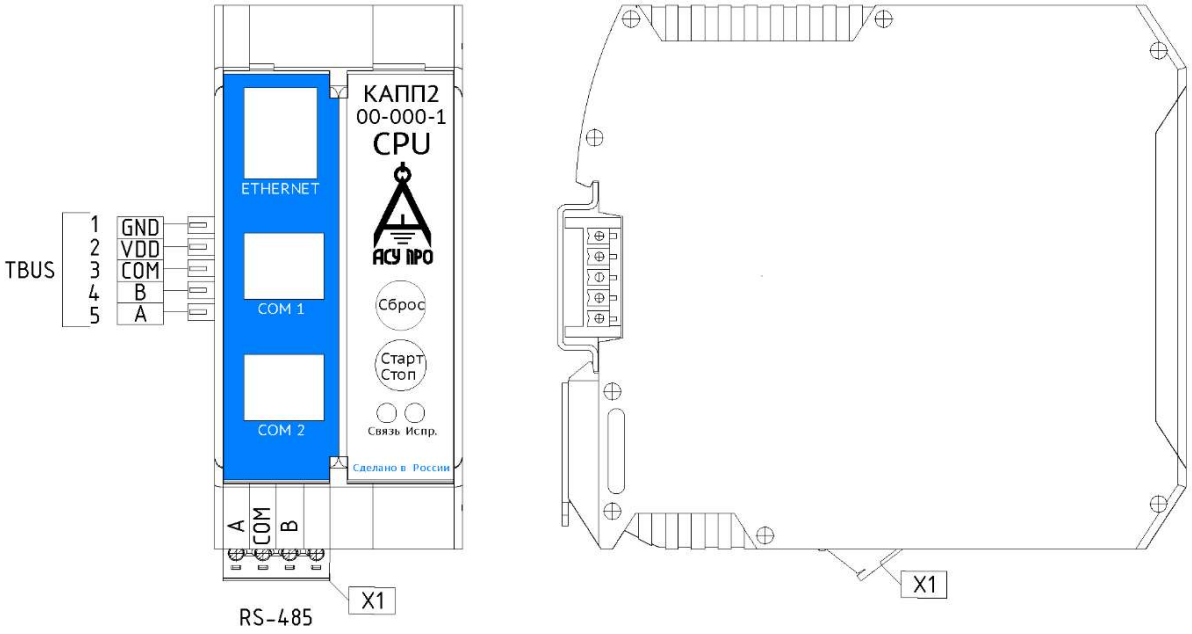


Рисунок 2 – Процессорный модуль КАПП2-00-000-1

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

Изм.	Кол.уч.	Лист	№ док.	Подпись	Дата	73619730.26.20.30.000.020 РЭ	Лист 12

ФорматА4

2.3 Использование изделия

2.3.1 Установка интегрированной среды разработки CODESYS V3.5

Контроллер программируется с помощью среды CODESYS V3.5 версии не ниже SP17.

Последнюю версию среды CODESYS, с которой совместим контроллер, можно скачать по ссылке: https://store-archive.codesys.com/ftp_download/3S/CODESYS/300000/3.5.17.30/CODESYS%203.5.17.30.exe.

Для установки нужно запустить скачанный дистрибутив. Первым делом установщик предложит установить необходимые компоненты (рисунок 4).

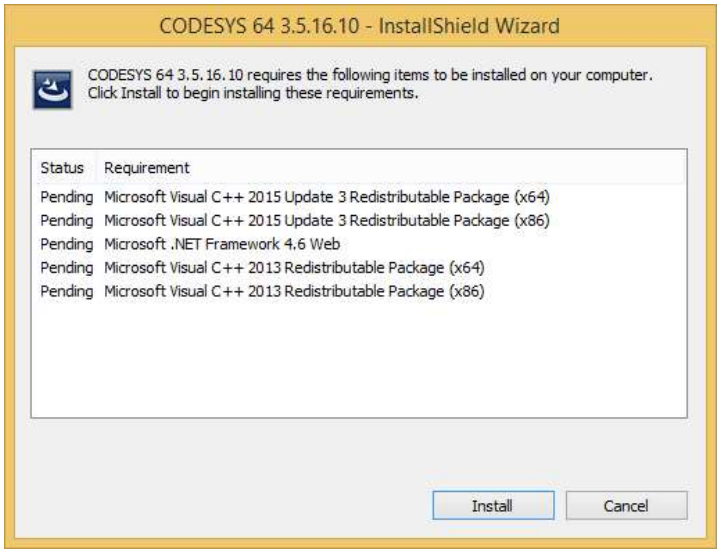


Рисунок 4 – Окно установки необходимых компонентов

Нужно нажать кнопку «Установить» («Install»). После чего компоненты начнут последовательно устанавливаться на ваш ПК.

Важно!!! Установка некоторых компонентов требует подключение к сети интернет.

Далее потребуется перезагрузка ПК. После перезагрузки инсталляция продолжится. Возможно, в появившемся окне, нужно будет снова нажать кнопку «Установить» («Install») для продолжения. После установки всех необходимых компонентов появится окно установки среды CODESYS (рисунок 5).

Рисунок 5 – Окно установки среды CODESYS

В окне необходимо нажать кнопку «Далее» («Next»), после чего появится окно с лицензионным соглашением. Для продолжения необходимо согласиться с соглашением, переключить радиокнопку в соответствующее положение (рисунок 6) и нажать ставшую активной кнопку «Далее» («Next»).

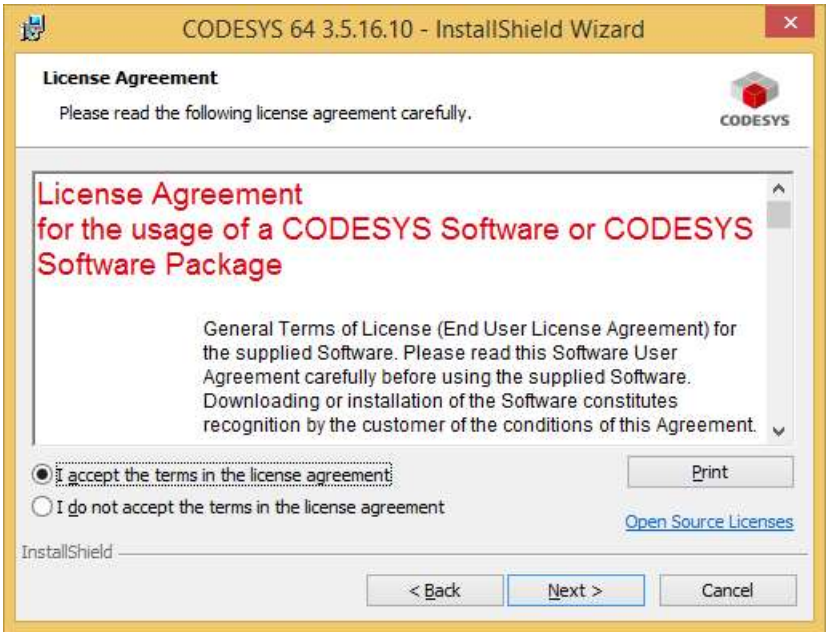


Рисунок 6 – Окно лицензионного соглашения

Далее установщик предложить указать путь установки (рисунок 7).

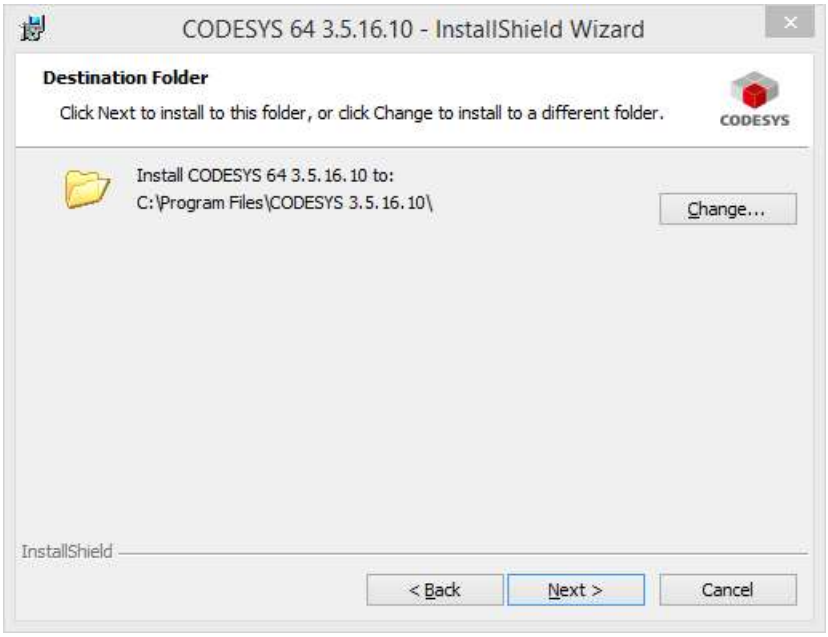


Рисунок 7 – Пусть установки среды CODESYS

Если нужно меняем его, нажав на кнопку «Изменить» («Change»), указываем новый путь. Для продолжения нажимаем на кнопку «Далее» («Next»). После этого установщик предложит тип установки (рисунок 8).

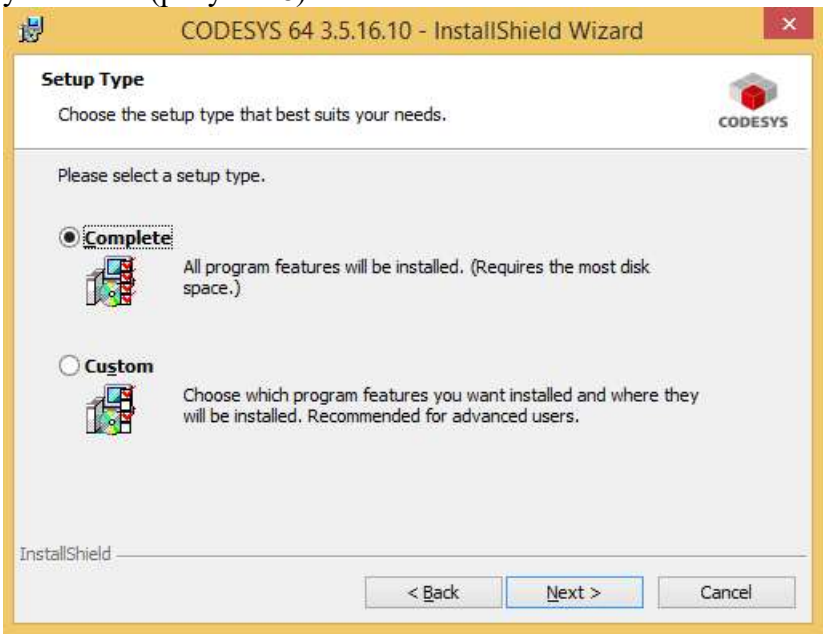


Рисунок 8 – Выбор типа установки

Для пользователей ранее не использовавших среду разработки CODESYS рекомендуется оставить радиокнопку в положении «Полная» («Complete»). Для продолжения необходимо нажать кнопку «Далее» («Next») и в новом окне нажать кнопку «Установить» («Install») для запуска инсталляции с выбранными опциями. После продолжительной установки появится сообщение об успешной установке среды CODESYS (рисунок 9) где нужно будет нажать кнопку «Закончить» («Finish»).

Рисунок 9 – Окно сообщение об успешной установке среды CODESYS

После чего на рабочем столе появится соответствующий ярлык (рисунок 10).

Рисунок 10 – Ярлык для запуска среды CODESYS

При наведении виден путь, по которому можно найти файл запуска среды CODESYS – CODESYS.exe. По умолчанию данный файл находится в папке C:\Program Files\CODESYS 3.5.16.10\CODESYS\Common, в соответствии с вашей ОС и версией среды CODESYS.

2.3.2 Установка пакета для программирования КАПП2

Перед тем, как приступить к программированию контроллера, необходимо установить пакет (package). Пакет содержит файлы целевой платформы (target - файлы) и библиотеки (library), необходимые для написания программ контроллера ERGON.

Чтобы установить пакет, выберите пункт «Менеджер пакетов…» в меню «Инструменты» (рисунок 11).

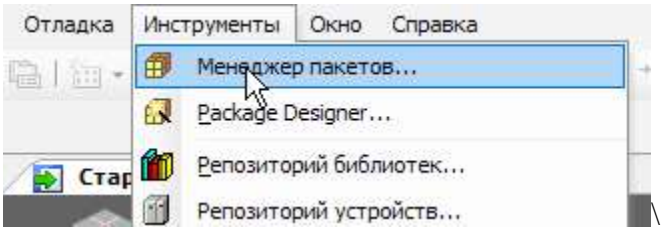


Рисунок 11 – Запуск менеджера пакетов

Откроется окно менеджера пакетов (рисунок 12).

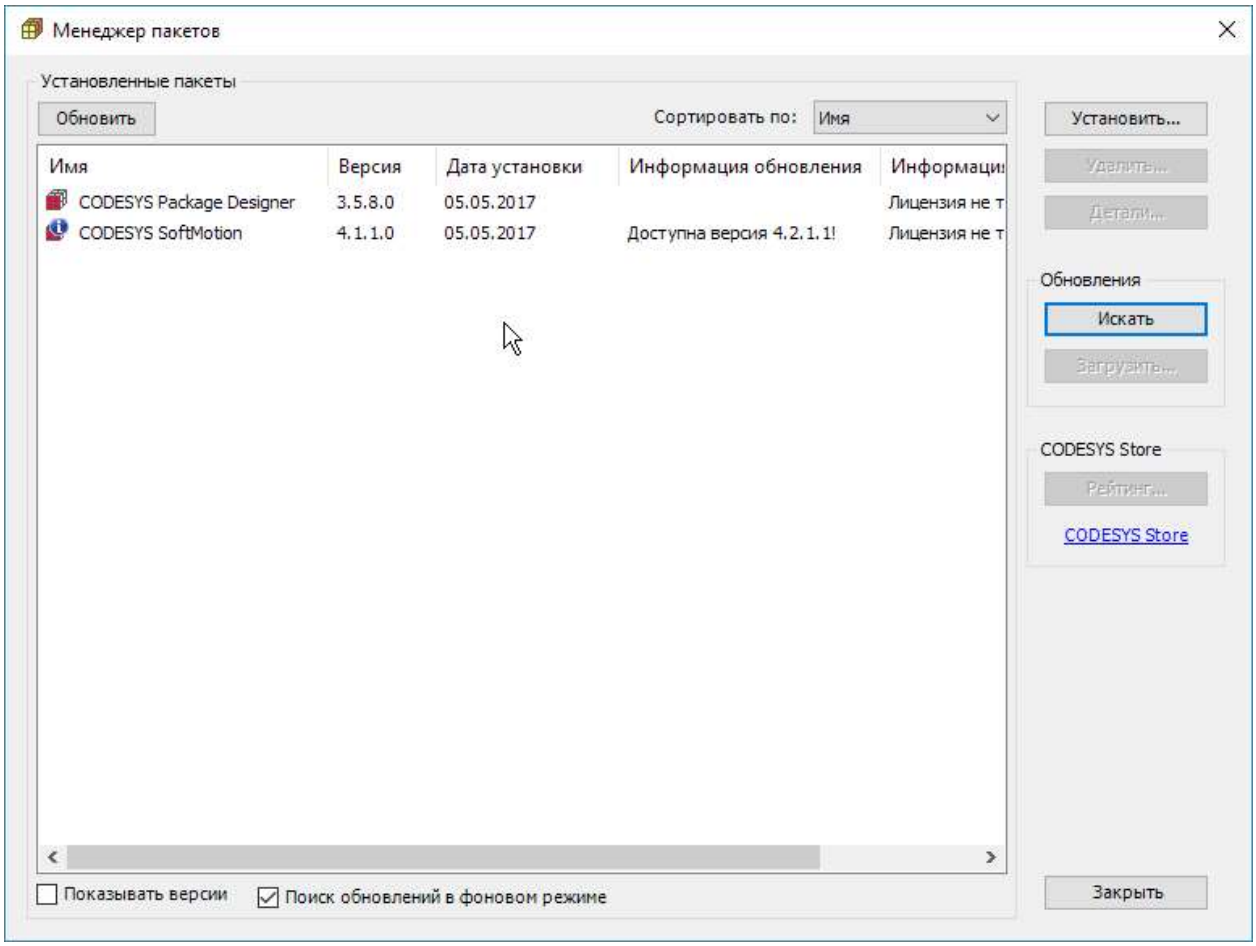


Рисунок 12 – Окно менеджера пакетов

Нажмите кнопку «Установить…» (рисунок 13).

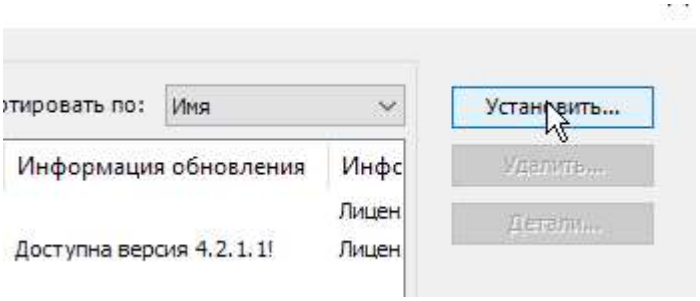


Рисунок 13 – Кнопка установки пакета

Выберите нужный файл с пакетом (рисунок 14). Для процессорного модуля КАПП2-00-000-1 имя файла KAPP2 CPU.package.

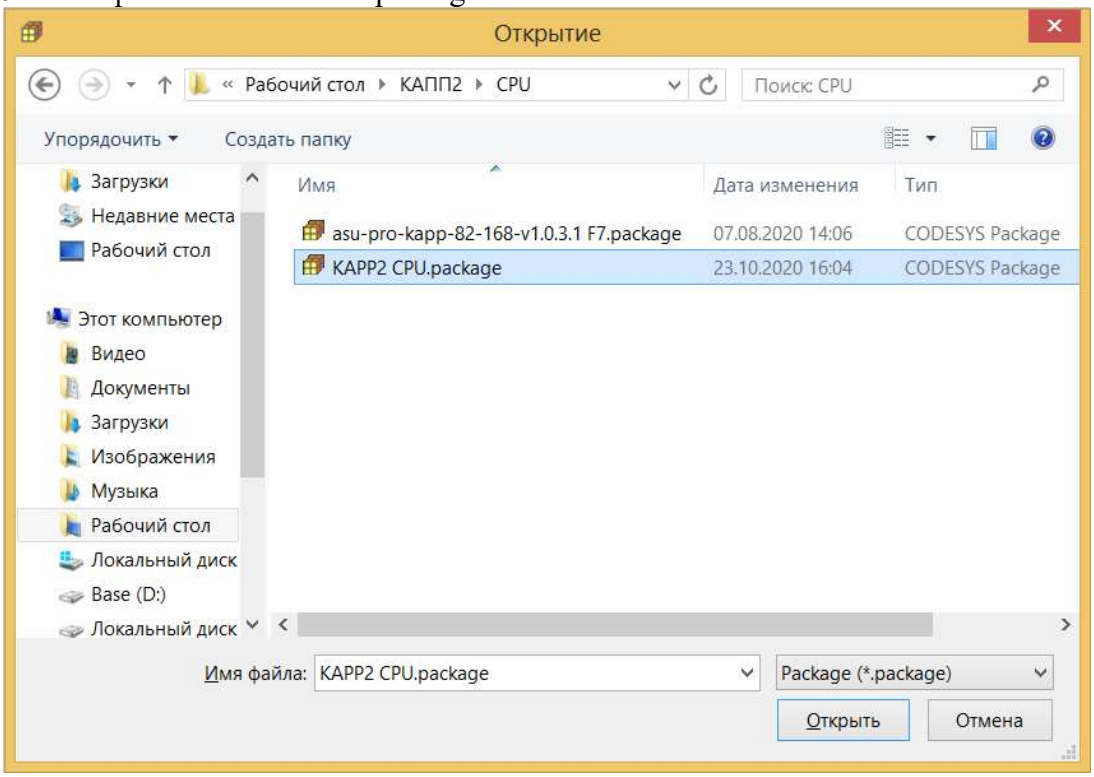


Рисунок 14 – Окно выбора файла пакета

Откроется окно установки пакета. Выберите пункт «Полная установка» и нажмите кнопку «Далее» (или «Next») (рисунок 15).

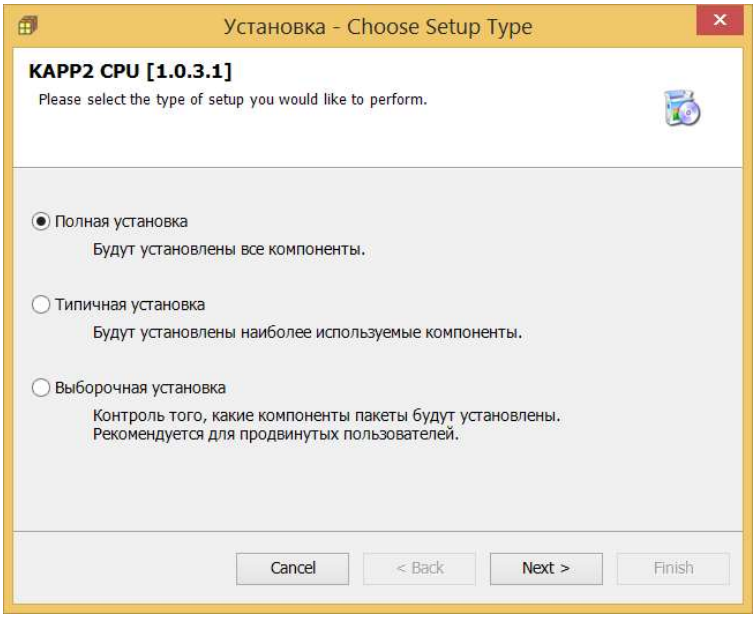


Рисунок 15 – Окно выбора типа установки пакета

Дождитесь установки пакета. Для того что бы посмотреть установленные компоненты нажмите кнопку «Далее» («Next») (рисунок 16).

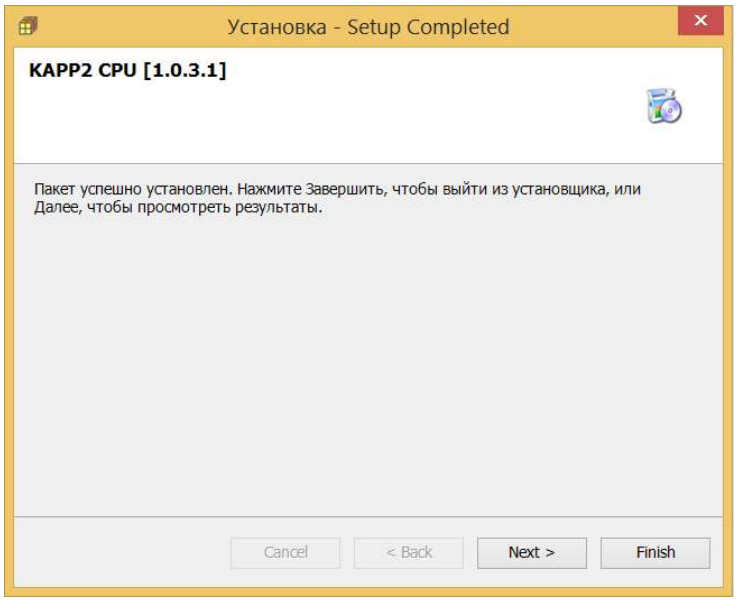


Рисунок 16 – Окно завершения установки

После чего откроется окно результатов установки (рисунок 17). Для более подробных результатов можно раскрыть вложенные ветки, нажав на значок «+», по каждому пункту.

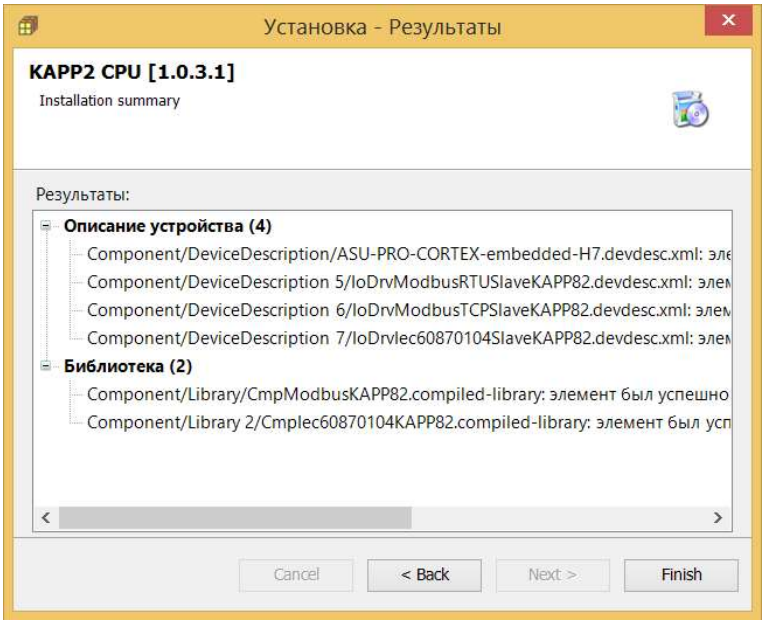


Рисунок 17 – Окно результатов установки

Для завершения установки необходимо нажать кнопку «Готово» («Finish»). После чего установленный пакет появится в списке менеджера пакетов (рисунок 18).

Имя	Версия	Дата установки
CODESYS Control for Raspberry PI	3.5.16.0	01.06.2020
CODESYS SoftMotion	4.5.1.0	16.04.2020
KAPP2 CPU	1.0.3.1	26.10.2020
OwenTargets	3.5.14.3003	28.05.2020
Softmotion Robotics HMI Example	1.0.0.0	02.06.2020

Рисунок 18 – Список пакетов

2.3.3 Установка новых файлов целевой платформы КАПП2 CPU

Если нужно поменять файлы целевой платформы при смене контроллера (например с КАПП-82-168 на КАПП2 CPU), для начала нужно удалить ранее установленное устройство КАПП-82-168. Для этого нужно выбрать пункт «Репозиторий устройств…» в меню «Инструменты» (рисунок 19).

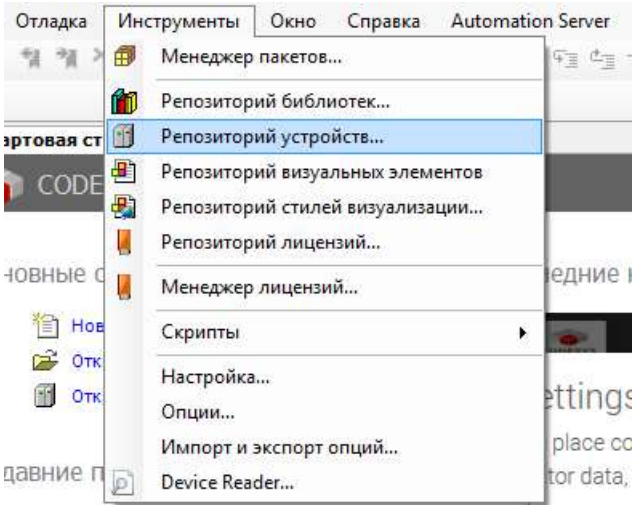


Рисунок 19 – Запуск репозитория устройств

Раскрыть список ПЛК, найти и выделить в нем КАПП-82-168, а затем нажать кнопку «Удалить» (рисунок 20).

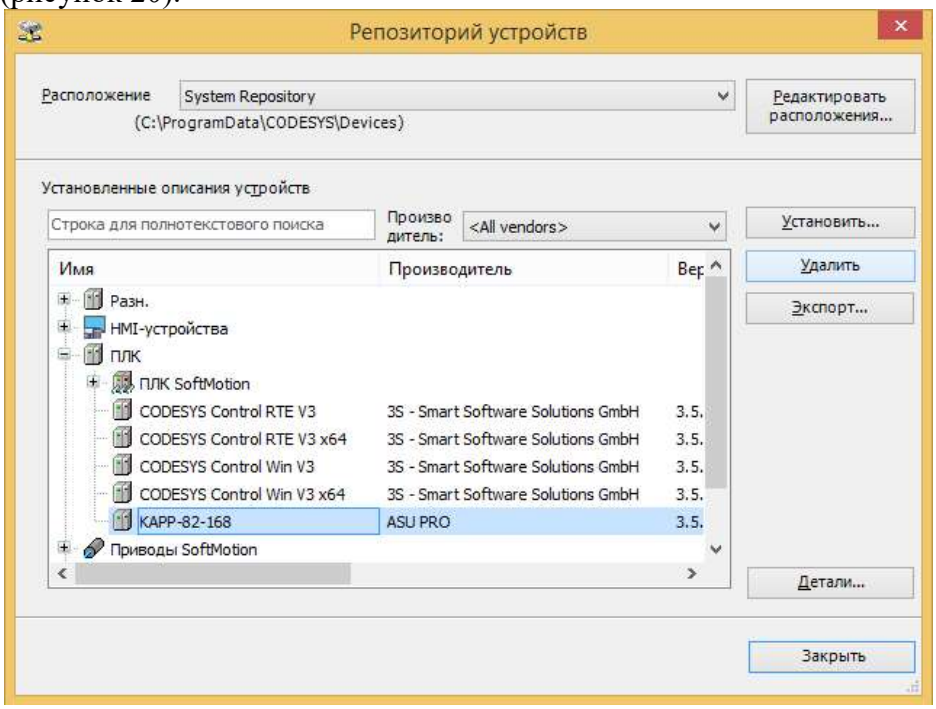


Рисунок 20 – Удаление устройства КАПП-82-168

Когда устройство будет удалено, в том же окне «Репозиторий устройств», необходимо нажать кнопку «Установить…». После чего откроется окно, в котором необходимо

указать путь к файлу описателю нового устройства и тип открываемого файла (рисунок 21).

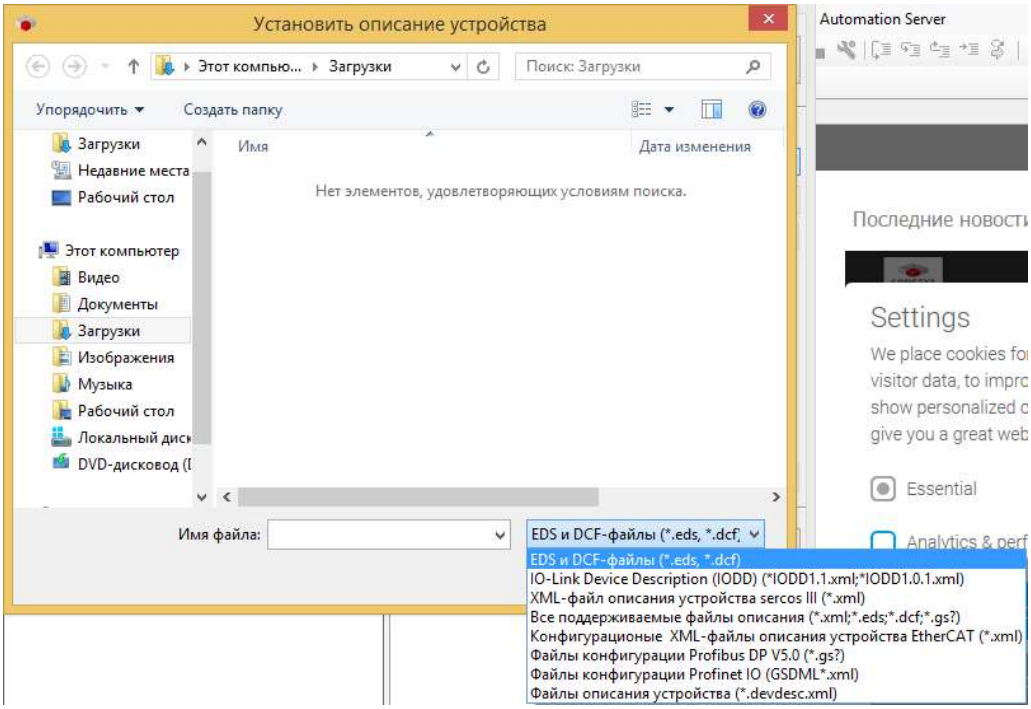


Рисунок 21 – Окно установки описателя устройства

Для работы с КАПП2-00-000-1 на основе процессорного ядра ARM Cortex M7, файл описателя устройства будет следующим - ASU-PRO-CORTEX-embedded-H7.devdesc.xml. При этом для того что бы в открывшемся окне были видны файлы данного типа необходимо выбрать соответствующий тип файлов. Например: Файлы описания устройства (*.devdesc.xml) (рисунок 22).

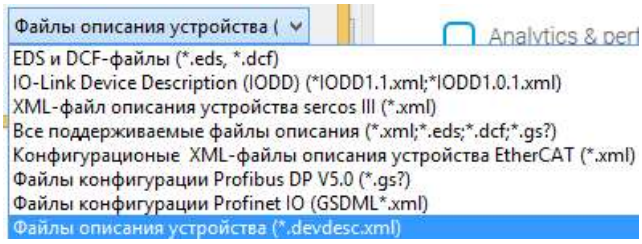


Рисунок 22 – Выбор типа файла для установки

Совет!!! Удобнее всего выбрать пункт «Все поддерживаемые файлы описания (*.xml; *.eds; *.dcf; *.gs?)». Тогда будут видны все поддерживаемые типы файлов.

После выбора нужно фала с описанием для установки необходимо нажать кнопку «Открыть» (рисунок 23).

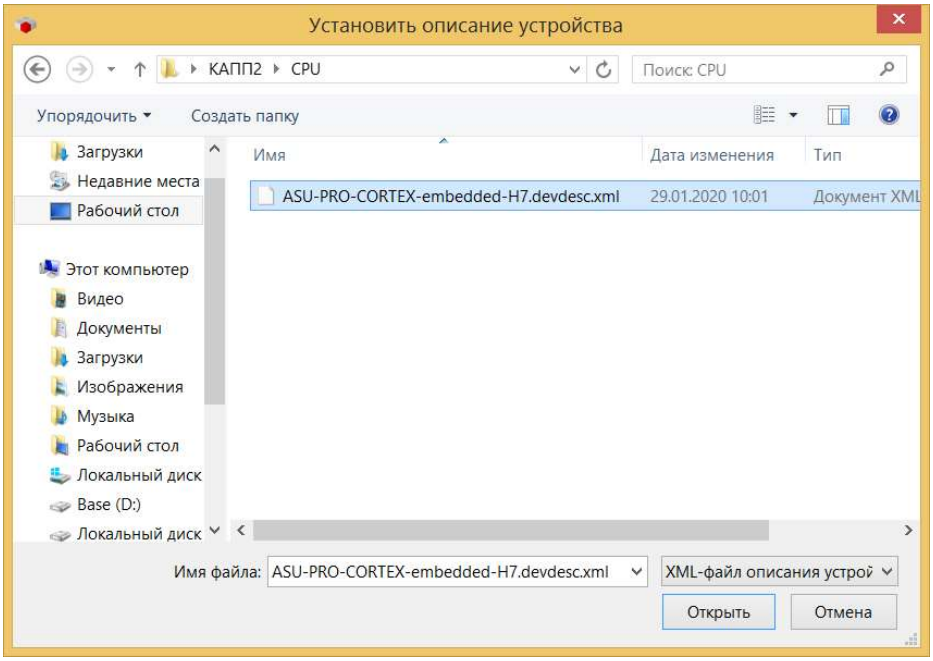


Рисунок 23 – Выбор описателя устройства

После чего в списке ПЛК появится новое устройство, в нашем случае КАПП2. Так же появится сообщение об установке данного устройства (рисунок 24).

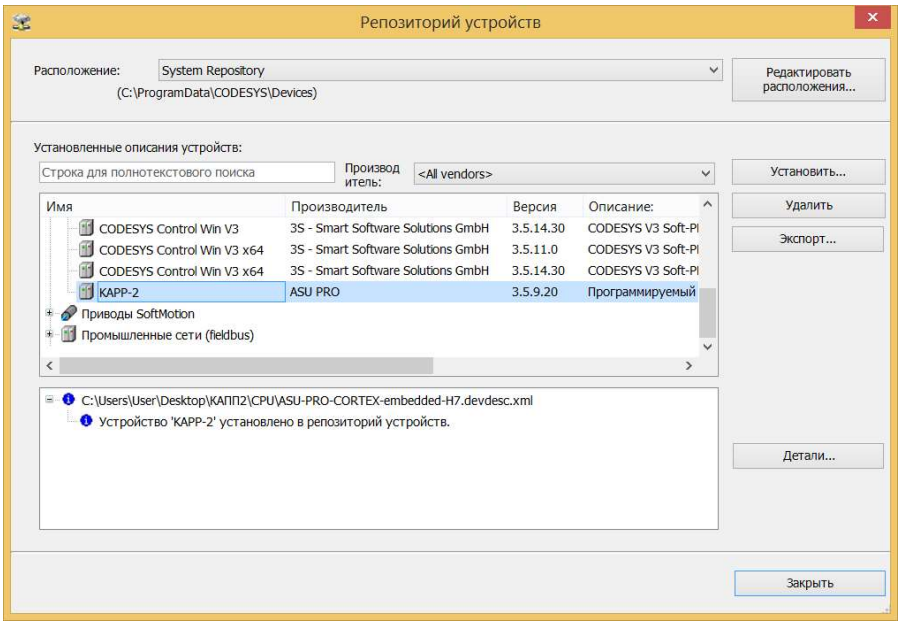


Рисунок 24 – Окно репозитория устройств после установки нового описателя устройства

2.3.4 Физическое подключение к ПК и определение IP-адреса контроллера

В качестве интерфейса связи с ПК с установленной средой программирования используется Ethernet (TCP-интерфейс). Можно использовать обычный патч-корд с коннекторами RJ-45.

Контроллер имеет следующие заводские настройки связи:
- IP-адрес: 192.168.20.99;
- Маска подсети: 255.255.255.0;
- Шлюз по умолчанию: 192.168.20.1.

Актуальный IP-адрес можно узнать с помощью файла CODESYSControl.cfg, находящегося в контроллере на флеш накопителе формата microSD. Для этого нужно вынуть флеш накопитель, вставить его в подходящий кардридер. В файловом менеджере при этом появится новый съемный носитель. Найти на нем вышеуказанный файл, нажать правой клавишей мыши для вызова меню и выбрать пункт «Открыть с помощью» (рисунок 25).

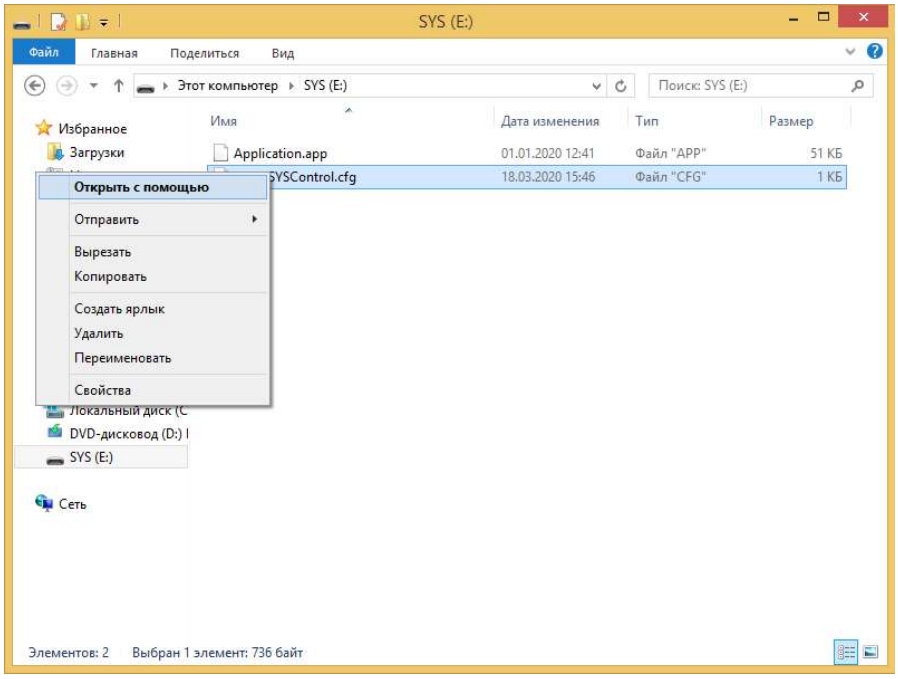


Рисунок 25 – Окно файлового менеджера съемного накопителя

После этого, в зависимости от вашей операционной системы, появится окно выбора приложения, с помощью которого требуется открыть указанный файл. В нем необходимо выбрать приложение «Блокнот». В Windows 8 например, первым делом появится окно с вопросом (рисунок 26).

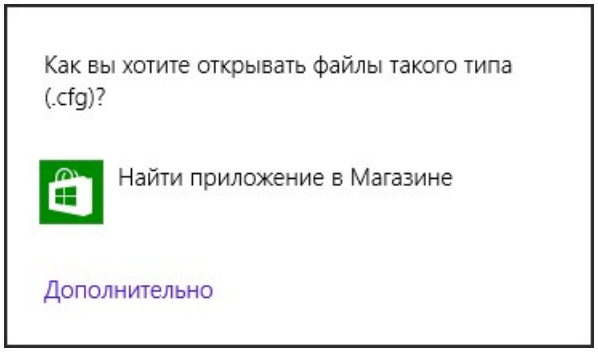


Рисунок 26 – Окно в Windows 8 после выбора пункта меню «Открыть с помощью»

Где необходимо сначала выбрать пункт « Дополнительно», после чего появятся другие приложения доступные для выбора. В том числе и приложение «Блокнот» (рисунок 27).

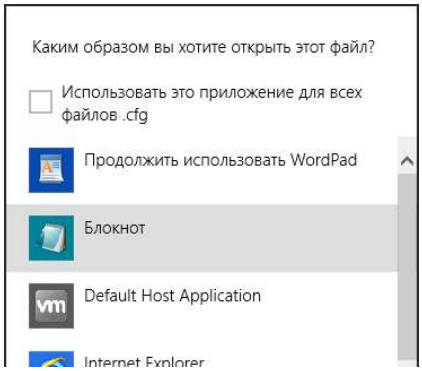


Рисунок 27 – Меню выбора приложения, с помощью которого необходимо открыть выбранный файл

Совет!!! В данном меню можно установить флаг «Использовать это приложение для всех файлов .cfg» (рисунок 28). После чего файлы с расширением по умолчанию будут открываться приложением «Блокнот», что очень удобно.

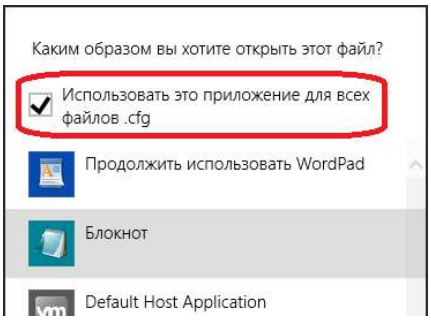


Рисунок 28 – Флаг «Использовать это приложение для всех файлов .cfg»

В открывшемся файле найти раздел «SysSocket», в котором будет указан актуальный IP-адрес (рисунок 29).

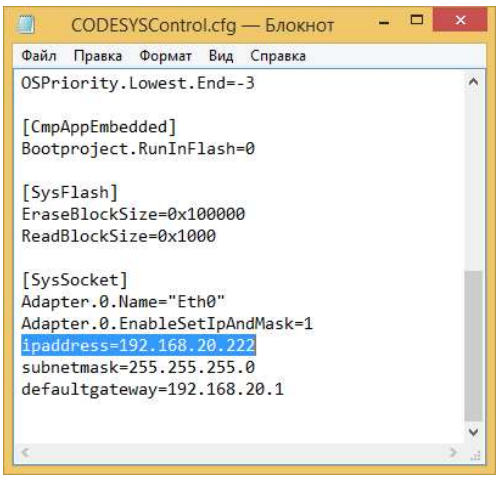


Рисунок 29 – Содержание конфигурационного файла CODESYSControl.cfg

В данном случае IP-адрес: 192.168.20.222. Этот адрес необходимо запомнить, в дальнейшем он понадобиться для настройки связи с контроллером в среде CODESYS.

Важно!!! Для установления связи между контроллером и ПК со средой разработки CODESYS необходимо чтобы подсети сетевой платы ПК и контроллера совпадали. В рассмотренном примере сетевая плата ПК должна IP-адрес: 192.168.20.XXX. Где – 192.168.20 подсеть, XXX – любой адрес устройства из диапазона 1..255. IP-адреса ПК и контроллера должны быть уникальными в сети.

Так же с помощью этого конфигурационного файла можно изменить настройки связи с контроллером. Процедура будет описана в следующем пункте раздела.

2.3.5 Установка произвольного сетевого адреса

Для того чтобы изменить настройки сети (например изменить IP-адрес контроллера), отредактируйте параметры файла «CODESYSControl.cfg» в блоке SysSocket:

```
[SysSocket]
Adapter.0.Name="Eth0"
Adapter.0.EnableSetIpAndMask=1
ipaddress=192.168.20.99
subnetmask=255.255.255.0
defaultgateway=192.168.20.1
```

Названия параметров обозначают следующее:
- ipaddress – IP-адрес;
- subnetmask – маска подсети;
- defaultgateway – шлюз по умолчанию.

Важно!!! Не изменяйте остальные параметры в конфигурационном файле. Сохраняйте резервную копию конфигурационного файла при каждом изменении, чтобы вернуть рабочую конфигурацию при установке неверных значений параметров.

2.3.6 Обновление внутреннего программного обеспечения

Контроллер имеет функцию обновления внутреннего программного обеспечения (прошивки). Для этого, необходимо поместить на карту памяти microSD файл с именем «KAPP2.ENC», содержащий новую версию прошивки.

Существует два способа поместить файл прошивки на карту памяти:

- подключить microSD карту к ПК или любому другому устройству, имеющему функцию работы с картами памяти, и скопировать файл на карту;
- подключиться к контроллеру при помощи среды CODESYS и скопировать файл через файловый менеджер CODESYS (рисунок 30) в корневой каталог.

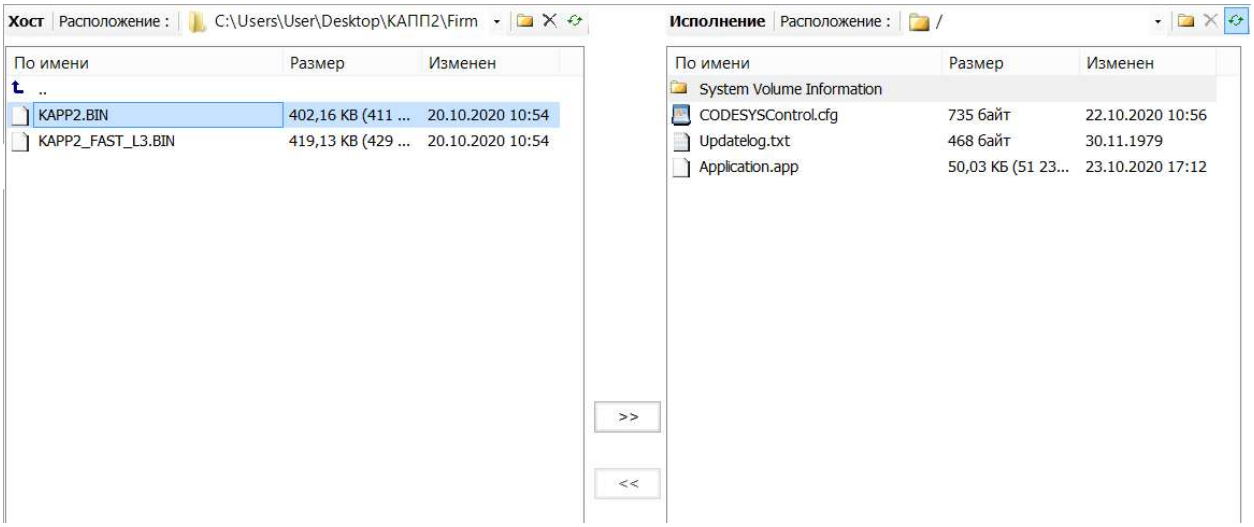


Рисунок 30 – Загрузка файла прошивки через среду CODESYS

После копирования файла прошивки на карту памяти и установки карты памяти в контроллер (если она вынималась), необходимо подать питание на контроллер или выполнить сброс, если он был включен. Обнаружив в момент запуска файл прошивки, контроллер запустит процесс обновления. Во время записи прошивки светодиодный индикатор «Испр.» будет светиться зеленым цветом, а индикатор «Пуск» мерцать зеленым цветом. После записи будет выполнена верификация записанных данных, во время которой светодиодные индикаторы «Испр.» и «Пуск» будут попеременно мерцать зеленым цветом. Вся процедура длится чуть менее 1 минуты.

По окончании процедуры прошивки необходимо подключить контроллер к ПК через среду CODESYS или извлечь карту памяти microSD, и вручную проверить содержимое карты. В случае успеха файл прошивки, скопированный ранее, будет удален, а также, на карте появится файл «Update.txt», в котором будет содержаться информация по ходу и результатам обновления с указанием версий.

Важно!!! После обновления внутреннего программного обеспечения контроллера (прошивки), пользовательская программа CODESYS может быть стерта!

Ниже приведен пример текста файла результатов процедуры обновления в случае успеха:

```
*** ASU-PRO KAPP2 Firmware UPDATER ***
***          version 1.2.1          ***
```

```
[ 12:00:00] Обнаружен файл прошивки!
[ 12:00:00] Стирание flash памяти...
[ 12:00:05] Готово!
[ 12:00:05] Запись данных...
[ 12:00:07] Запись данных завершена!
[ 12:00:07] Проверка записанных данных...
[ 12:00:31] Проверка завершена успешно!
[ 12:00:31] Обновлено с версии 1.0.1.4 на 1.0.1.4
[ 12:00:31] Обновление ПО завершено!
[ 12:00:31] Удаление файла прошивки с флеш-карты...
[ 12:00:31] Файл прошивки удалён!
[ 12:00:31] Перезагрузка...
```

Возможно появление следующих ошибок:

-«Ошибка чтения файла образа прошивки!» – убедитесь в исправности карты и целостности файловой системы. В качестве варианта решения проблемы можно отформатировать карту (файловая система должна быть FAT32) и заново записать туда необходимый файл, повторив процесс.

- «Некорректный файл прошивки, слишком большой размер, отмена...» – означает, что либо файл поврежден, либо файл не является файлом прошивки.

- «Ошибка записи во флэш!» – при появлении этой ошибки, а также если вам не удалось самостоятельно исправить другие ошибки, обратитесь в техническую поддержку.

2.3.7 Сброс пользовательской программы

Для удаления пользовательского кода CODESYS если, например, ошибки в программе пользователя привели к невозможности его запуска или доступа к контроллеру стандартным методом, необходимо удалить программу пользователя с контроллера.

Для удаления программы пользователя необходимо на панели контроллера, нажать кнопку «Сброс», после чего сразу нажать и удерживать кнопку «Старт / Стоп», до тех пор, пока индикатор «Пуск» не начнет мигать оранжевым, при этом индикатор «Исправность» будет постоянно гореть зелёным, после чего отпустить – всё, программа пользователя стёрта из памяти (microSD).

Убедиться в работоспособности контроллера путем подключения к нему через среду CODESYS.В случае, если это не решает проблему, необходимо обратится в техническую поддержку.

2.3.8 Создание первого проекта

Откройте среду CODESYS и выберите пункт «Новый проект…» в меню «Файл» (рисунок 32).

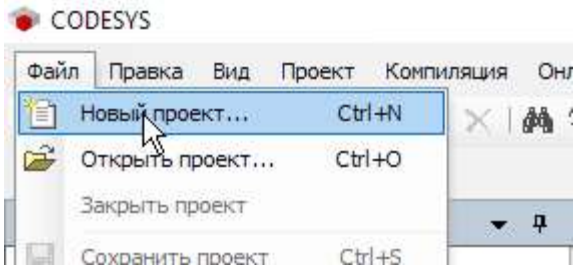


Рисунок 32 – Пункт меню создания проекта

В окне «Новый проект» выберите категорию «Проекты» и шаблон «Стандартный проект» (рисунок 33).

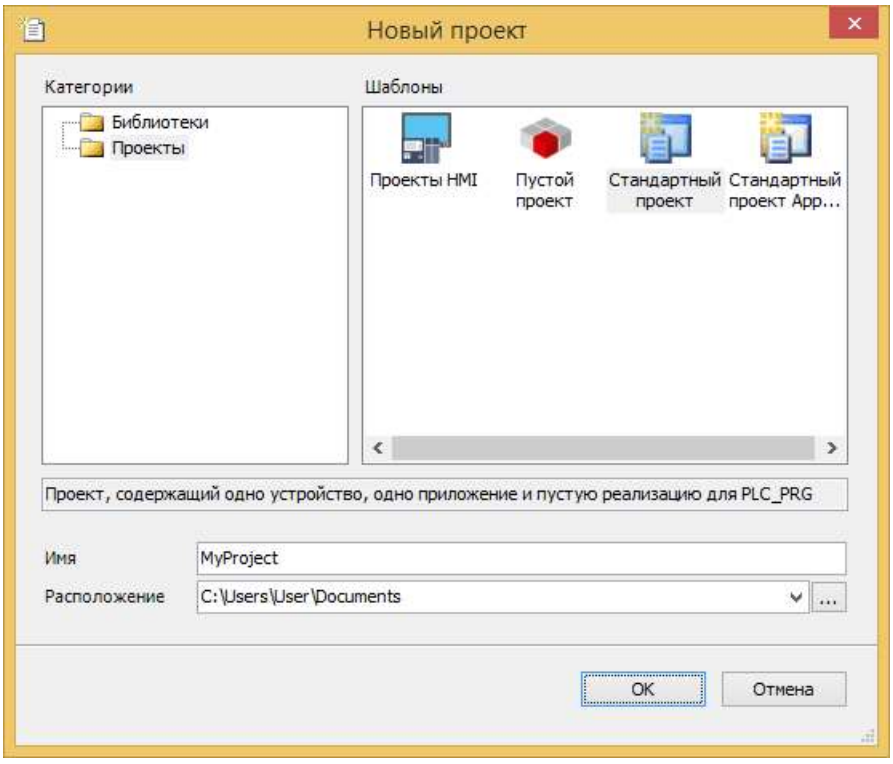


Рисунок 33 – Окно создания проекта

Введите желаемое имя проекта и нажмите кнопку «ОК».

В следующем окне в поле «Устройство» выберите «КАРР-2 (ASU PRO)» (рисунок 34).

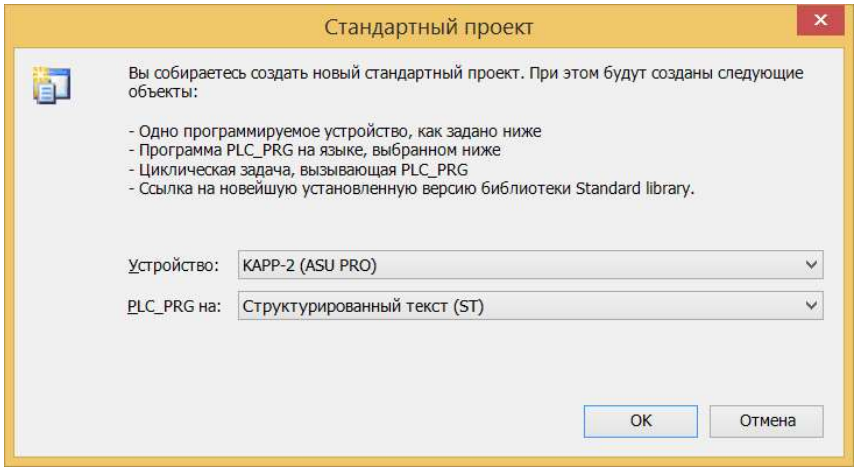


Рисунок 34 – Выбор устройства для создания проекта

В результате, вы получите новый проект со стандартной структурой и программой на языке ST (рисунок 35).

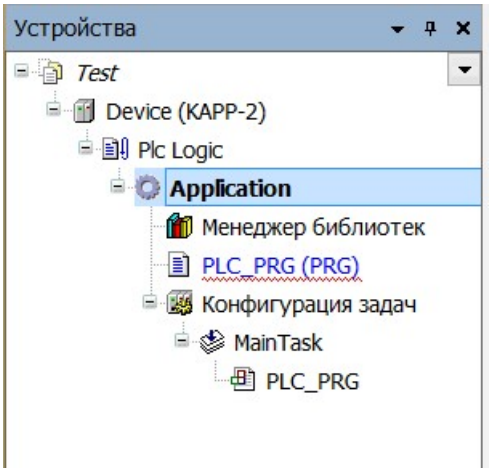


Рисунок 35 – Структура нового проекта

Как видно из рисунка пункт «Приложение» («Application») подчеркнут волнистой красной линией. Так же будут сообщения об ошибках (Библиотека не установлена в системе) в соответствующем окне (рисунок 36).

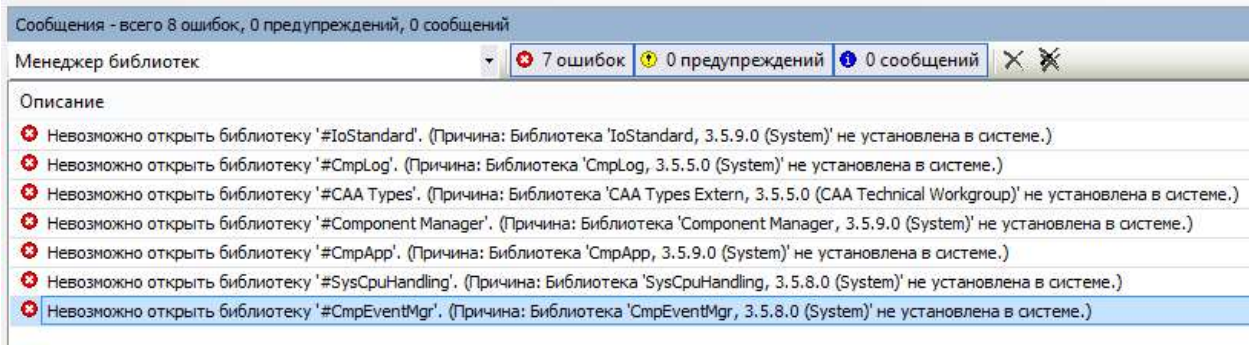


Рисунок 36 – Окно сообщений об ошибках

Для того что бы устранить данные ошибки необходимо запустить менеджер библиотек. Запустить менеджер можно двойным щелчком левой кнопкой мыши по соответствующему значку в дереве проекта (рисунок 37).

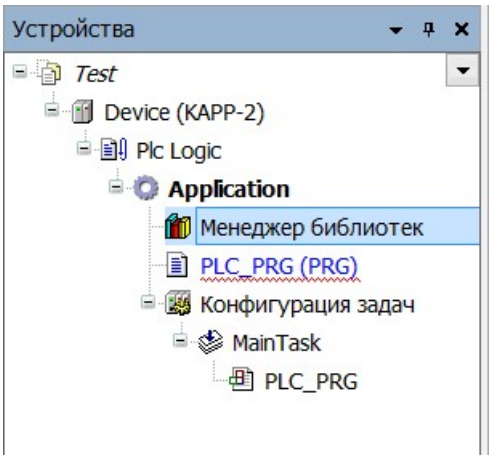


Рисунок 37 – Менеджер библиотек в дереве проекта

После этого в центральной части среды разработки CODESYS откроется вкладка менеджера библиотек. Для автоматической загрузки всех отсутствующих библиотек, в верхней части вкладки, необходимо нажать кнопку «Загрузка отсутствующих библиотек» (рисунок 38)

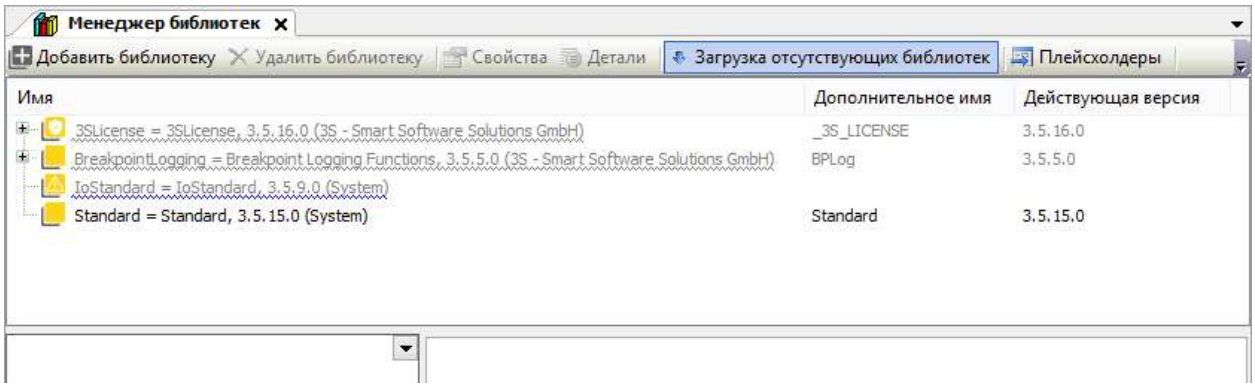


Рисунок 38 – Вкладка «Менеджер библиотек»

После чего откроется окно загрузки отсутствующих библиотек (рисунок 39), в котором необходимо нажать кнопку «Загрузка».

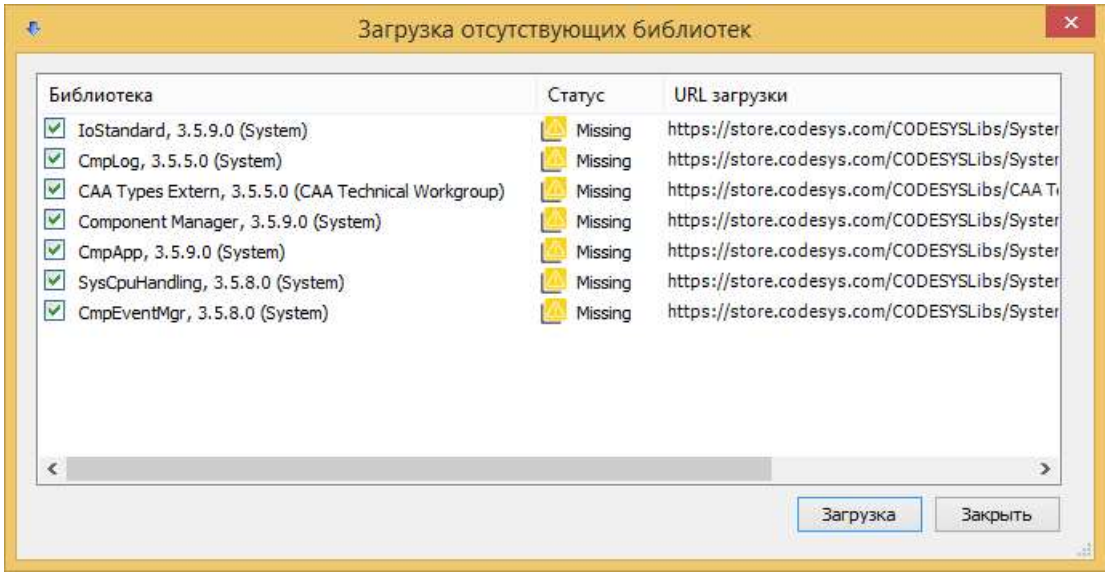


Рисунок 39 – Окно загрузки отсутствующих библиотек

Важно!!! Для загрузки отсутствующих библиотек необходимо подключение к сети интернет.

После успешной загрузки статус библиотек поменяется с «Отсутствует» («Missing») на «Установлено» (рисунок 40)

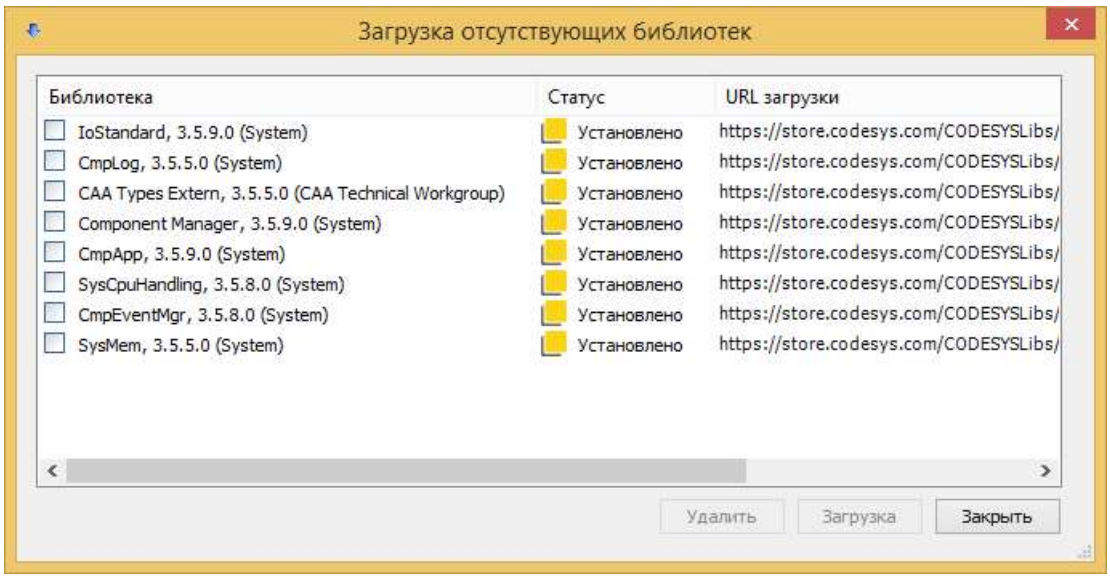


Рисунок 40 – Окно загрузки отсутствующих библиотек после установки

2.3.9 Установка связи с контроллером

Для установки связи с контроллером, дважды нажмите левой кнопкой мыши по строчке «Device»в структуре проекта в окне «Устройства» и выберите вкладку «Установка соединения» в появившемся окне. Появится окно установки соединения (рисунок 41).

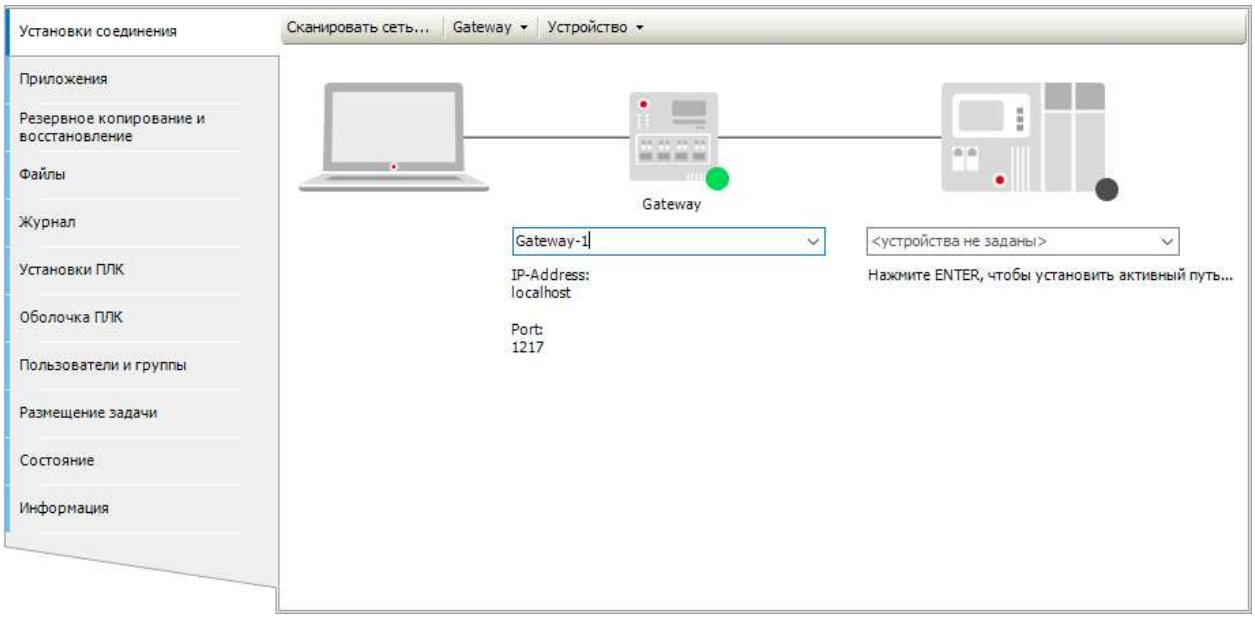


Рисунок 41 – Окно установки соединения

Далее, необходимо настроить шлюз (gateway). Для этого, выберите пункт «Конфигурация локального gateway…» в меню «Gateway» (рисунок 42).

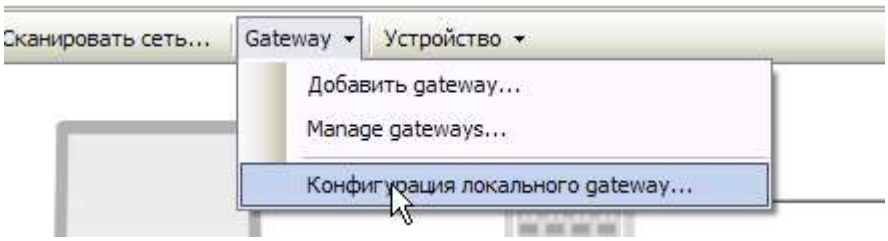


Рисунок 42 – Вызов окна конфигурации локального gateway

В появившемся окне, если у вас нет TCP – интерфейса, нажмите кнопку «Добавить» и выберите пункт «Добавить интерфейс верхнего уровня…» (рисунок 43).

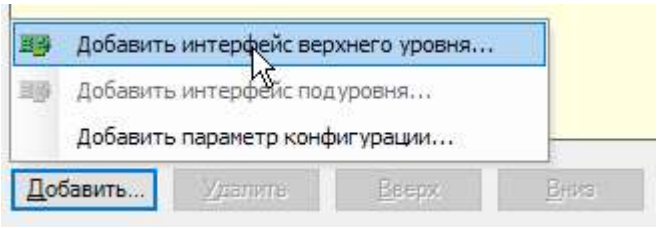


Рисунок 43 – Добавление интерфейса верхнего уровня

В появившемся выпадающем меню выберите пункт «TCP-интерфейс» и нажмите левой кнопкой мыши на свободную область. В поле «Порт» установите значение «11740» (рисунок 44).

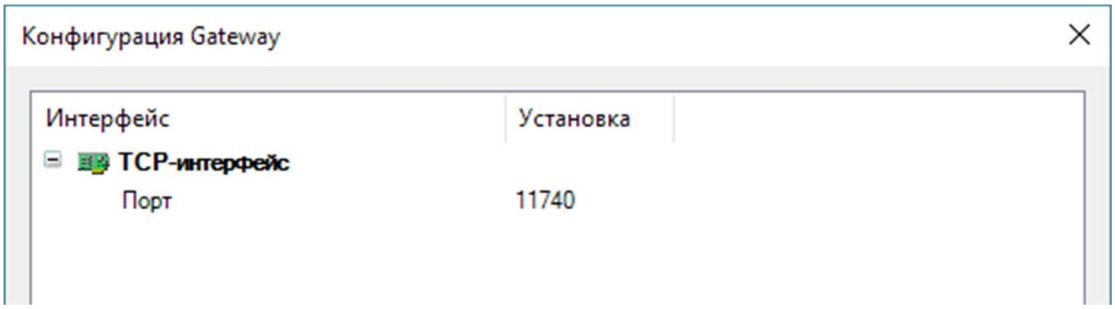


Рисунок 44 – Конфигурация шлюза

Нажмите кнопку «ОК», чтобы сохранить изменения.

Если у вас уже есть TCP – интерфейс, то необходимо просто в поле «Порт» установить значение «11740»

В окне установки соединения введите IP-адрес контроллера и нажмите клавишу «Ввод». Если соединение с контроллером прошло успешно, вы увидите общую информацию о вашем устройстве (рисунок 45).

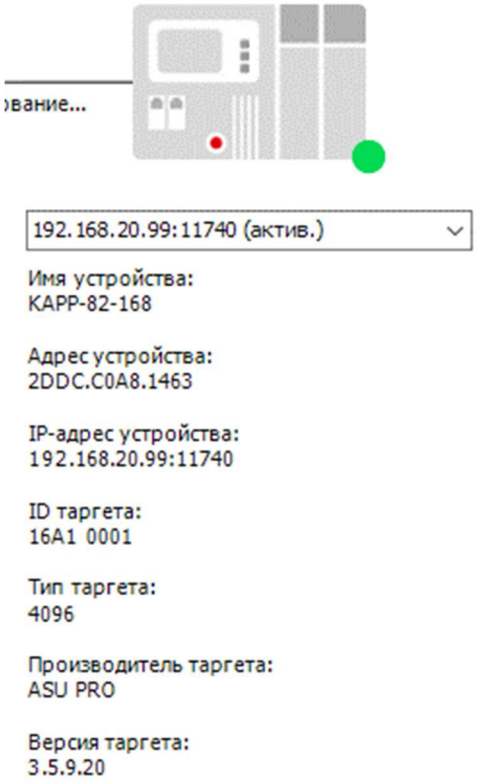


Рисунок 45 – Общая информация о контроллере

Контроллер готов для загрузки программы.

2.3.10 Загрузка программы в контроллер

Для демонстрации работы контроллера, создадим простую программу на языке ST. Для этого, откройте файл PLC_PRG в структуре проекта и наберите в нём текст, представленный на рисунке 46.

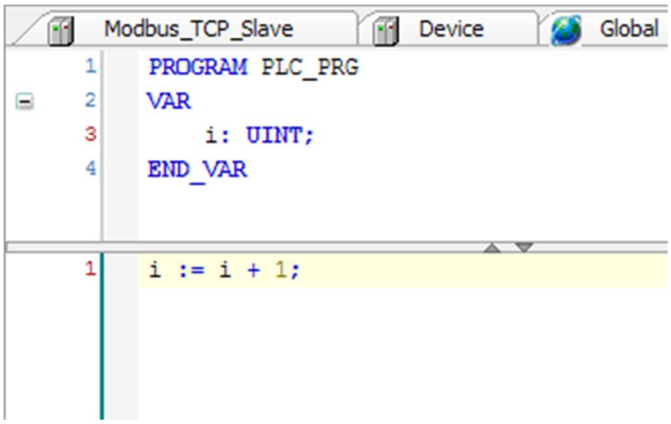


Рисунок 46 – Демонстрационная программа

Выберите пункт «Компиляция» в меню «Компиляция» или нажмите клавишу F11 (рисунок 47).

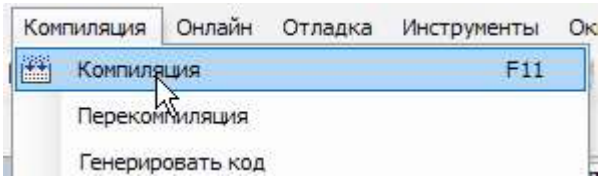


Рисунок 47 – Компиляция

При отсутствии ошибок, среда CODESYS сообщит об успешной компиляции (рисунок 48).

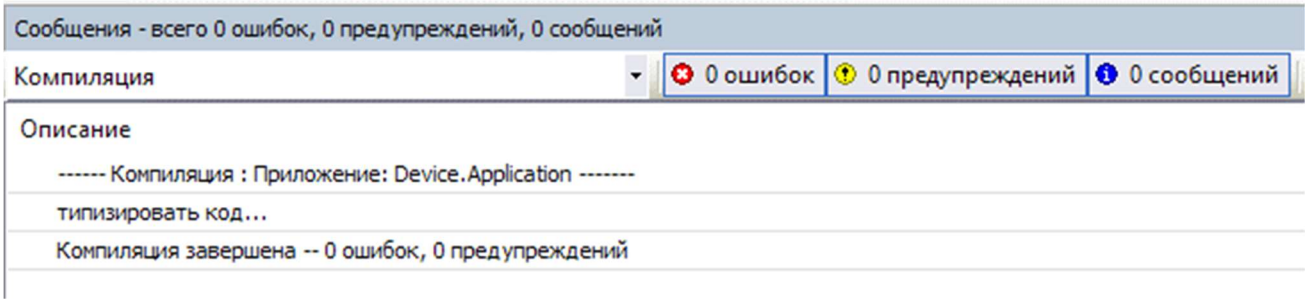


Рисунок 48 – Результаты компиляции

Выберите пункт «Логин» в меню «Онлайн» или нажмите сочетание клавиш Alt+F8 (рисунок 49).

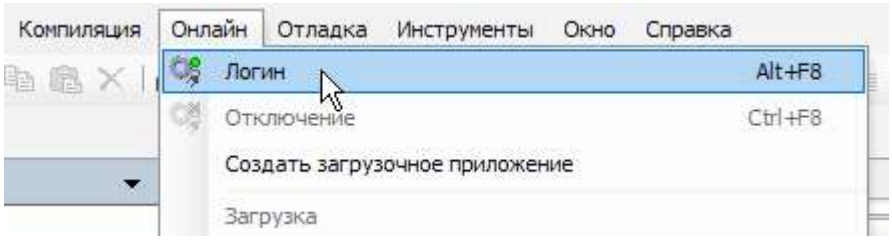


Рисунок 49 – Подключение к контроллеру

На предложение загрузить программу на контроллер нажмите «Да» (рисунок 50).

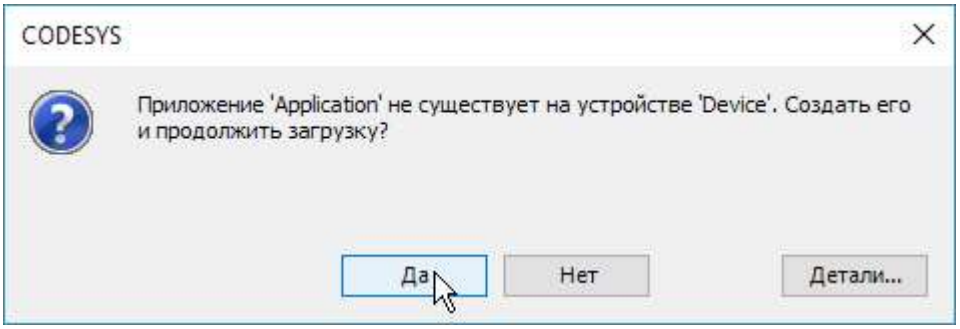


Рисунок 50 – Предложение загрузить программу на контроллер

CODESYS начнёт генерацию программы (рисунок 51) и загрузит её в контроллер. После чего появится сообщение о том, что программа загружена (рисунок 52)

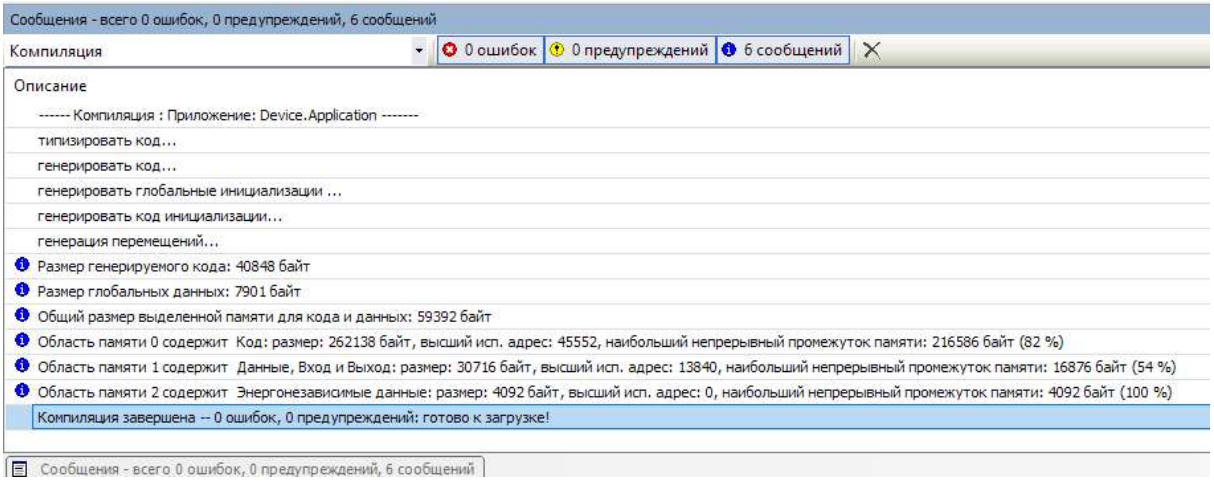


Рисунок 51 – Журнал генерации программы

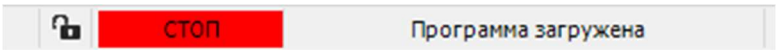


Рисунок 52 – Программа загружена

Для запуска программы, выберите пункт «Старт» в меню «Отладка» (рисунок 53).

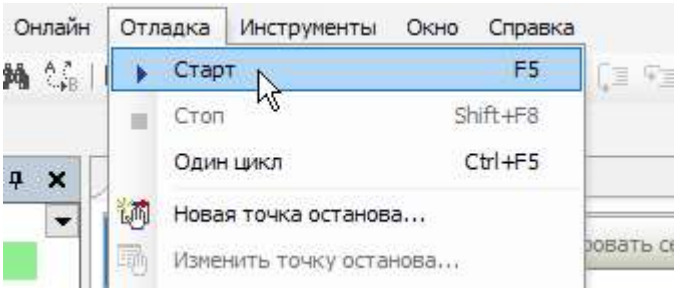


Рисунок 53 – Запуск программы

В статусной строке появится сообщение, что программа работает (рисунок 54).

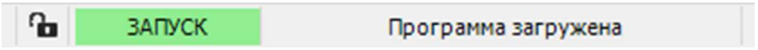


Рисунок 54 – Программа запущена

Откройте текст программы PLC_PRG, чтобы убедиться, что введённая ранее программа выполняется (рисунок 55).

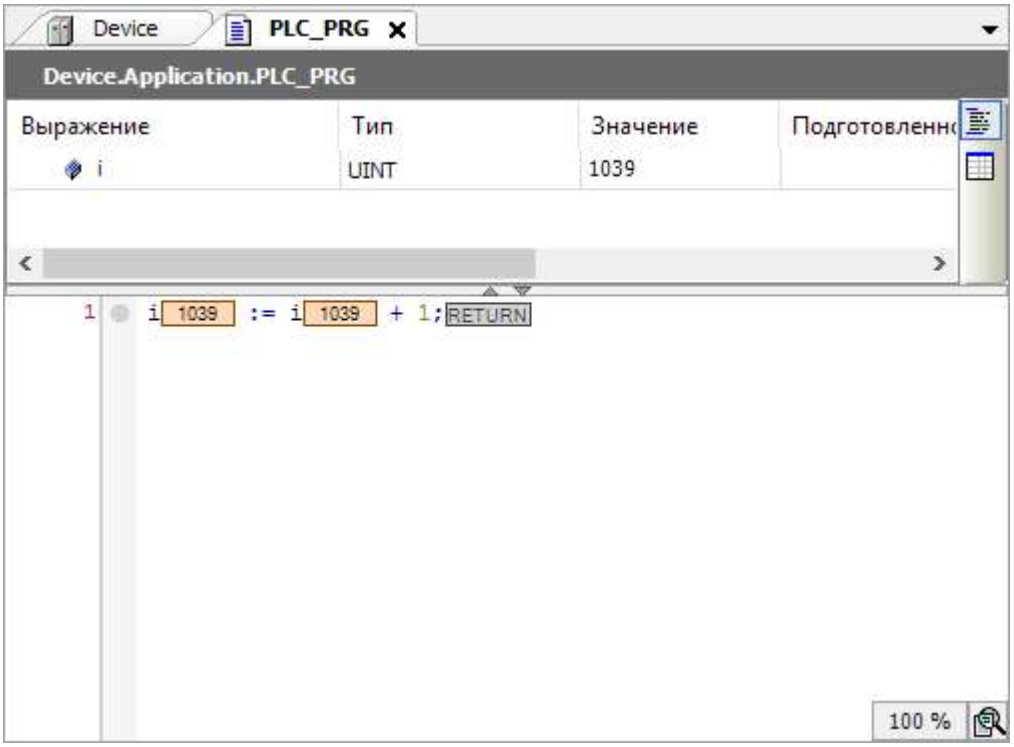


Рисунок 55 – Выполнение программы

Пользовательское приложение загружается во внутреннюю энергонезависимую память и автоматически начинает выполняться после перезагрузки контроллера.

На приложение пользователя действуют следующие ограничения:

- количество приложений – 1;
- количество задач – 32.

2.3.11 Загрузка и выгрузка исходного кода проекта

Функция загрузки и выгрузки исходного кода позволяет хранить архив исходного кода проекта в контроллере и выгружать его в среду CoDeSys при необходимости.

ВНИМАНИЕ! Функция загрузки/выгрузки работает в среде CoDeSys версии 3.5 SP17 и новее.

При загрузке исходного кода в контроллер в корневой папке на SD-карте, установленной в контроллере сохраняется проект в заархивированном виде в файле под именем Archive.prj. Если на карте памяти контроллера уже имеется файл с таким именем, он будет перезаписан новой версией.

Настройка загрузки и выгрузки исходного кода выполняется в разделе Загрузка исходного кода окна Установки проекта. Открывается окно установок через меню Проект → Установки проекта. Возможно установить выполнение загрузки кода в контроллер в автоматическом (с запросом или без запроса) или ручном режиме.

При установленной автоматической загрузке на контроллере всегда будет храниться актуальная версия проекта, полностью соответствующая скомпилированному приложению.

Для ручной загрузки архива проекта в контроллер необходимо в меню Файл выбрать команду Загрузка исходного кода…

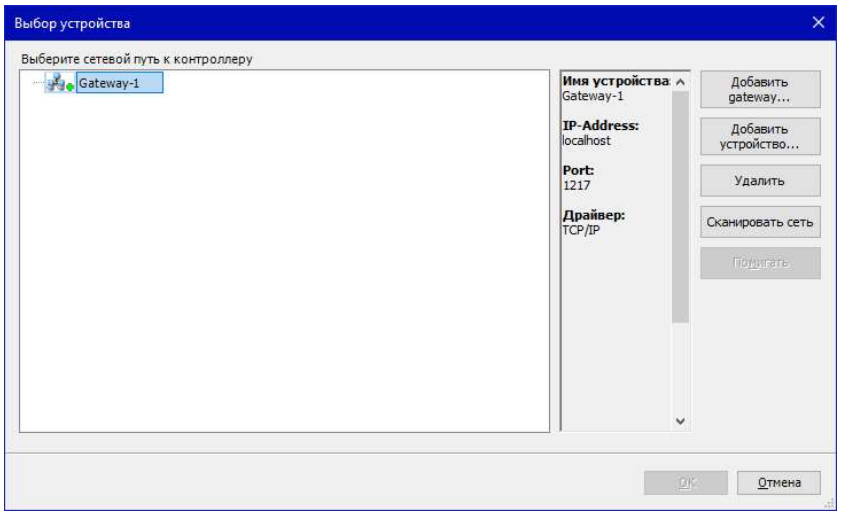


Рисунок 56 – Окно выбора устройства

В окне Выбор устройства (рисунок 56) нужно нажать кнопку Добавить устройство… Далее в окне Добавление устройства (рисунок 57) ввести IP-адрес контроллера, после чего будет выполнено сканирование шины и добавленное устройство будет отображено в списке окна Выбор устройства.

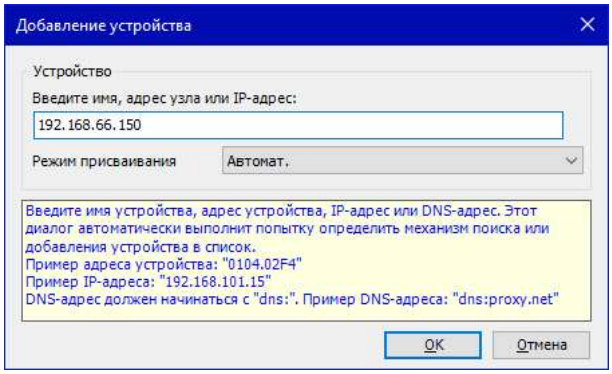


Рисунок 57 – Окно добавления устройства

Если этого не произошло, то нужно снова открыть окно добавления устройства и вместе с IP-адресом указать порт 11740 и протокол связи TCP/IP (рисунок 57).

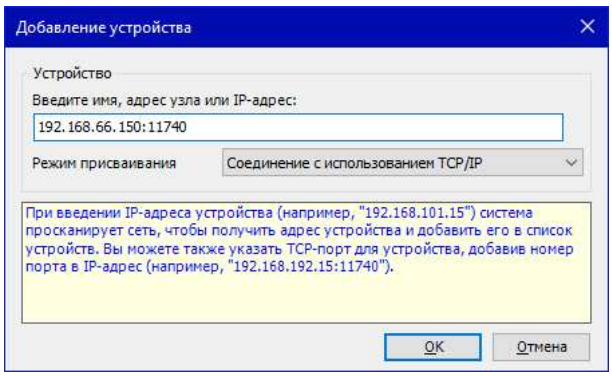


Рисунок 57 – Добавление устройства с заданием порта и протокола

Далее нужно выделить добавленное в список устройство и нажать OK (рисунок 58).

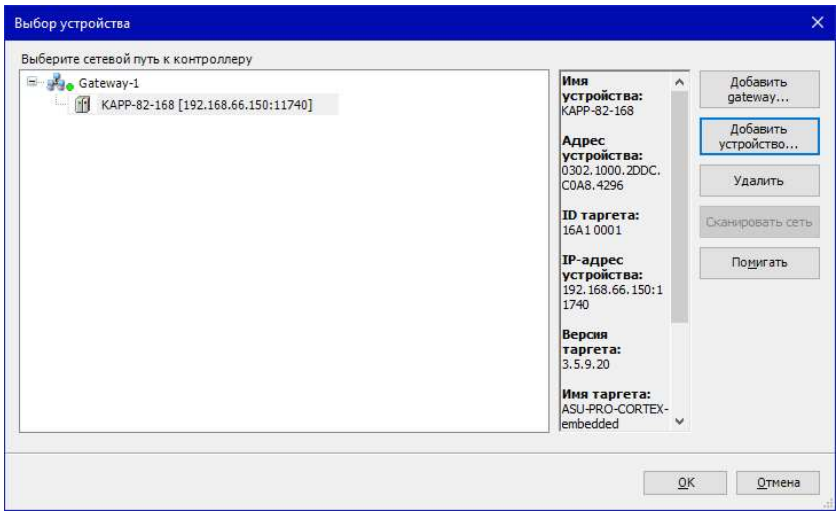


Рисунок 58 – Список устройств

Прогресс загрузки архива можно наблюдать в строке состояния.

Для выгрузки архива из контроллера в среду CoDeSys необходимо в меню Файл выбрать команду Выгрузка исходного кода… В окне Выбор устройства нужно выбрать

устройство и нажать OK. Если устройство в списке отсутствует, необходимо добавить его в список, как описано выше.

Далее начнется выгрузка проекта. Прогресс загрузки можно наблюдать в строке состояния. По окончании загрузки среда предложит указать путь для распаковки проекта (рисунок 59).

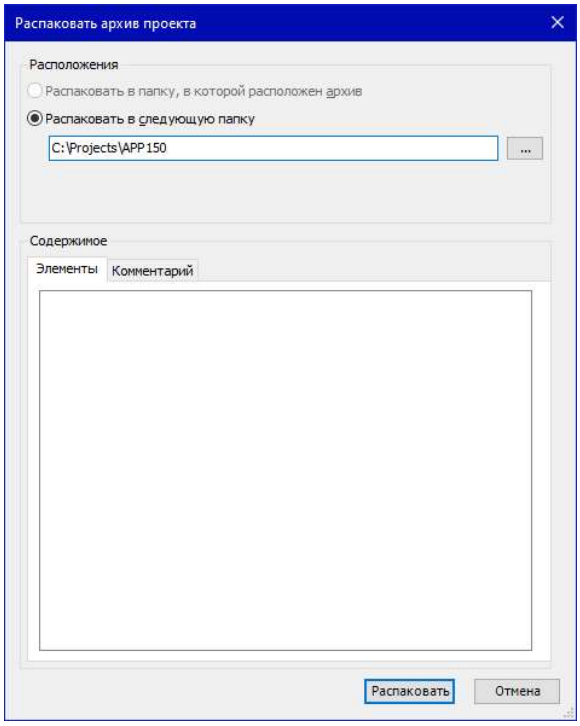


Рисунок 59 – Запрос при выгрузке и распаковке проекта

Если указанная папка не существует, то будет выдан запрос на создание папки. После сохранения на диск среда выдаст запрос, нужно ли открыть проект.

3 Работа с библиотеками CODESYS

3.1 Работа со стандартной библиотекой Standard.lib

Данная библиотека включена в проект по умолчанию, ее не нужно дополнительно добавлять в проект.

3.1.1 Строковые функции

LEN

Возвращает длину строки. Пример:

```
PROGRAM PLC_PRG
VAR
    STR: STRING;
    VarINT1 : INT ;
END_VAR
```

```
VarINT1:= LEN(STR);
```

Если `STR` = 'World', то значение переменной `VarINT1` = 5.

LEFT

Возвращает левую значимую часть строки заданной длины. Пример:

```
PROGRAM PLC_PRG
VAR
    STR: STRING;
    VarINT1 : INT ;
END_VAR
```

```
STR:= LEFT ('Hello World', VarINT1);
```

Если `VarINT1` = 5, то значение переменной `STR` = 'Hello'.

RIGHT

Возвращает правую значимую часть строки заданной длины. Пример:

```
PROGRAM PLC_PRG
VAR
    STR: STRING;
    VarINT1 : INT ;
END_VAR
```

```
STR:= RIGHT ('Hello World', VarINT1);
```

Если `VarINT1` = 5, то значение переменной `STR` = 'World'.

MID

Возвращает часть строки указанной длины с указанной позиции. Пример:

```
PROGRAM PLC_PRG
VAR
    STR: STRING;
END_VAR
```

```
STR:= MID ('Hello World',7,3);
```

В примере длинна 7, позиция 3, STR = 'llo Wor'.

CONCAT

Конкатенация (объединение) двух строк. Пример:

```
PROGRAM PLC_PRG
VAR
    STR: STRING;
END_VAR
```

```
STR:= CONCAT ('Hello ', 'World');
```

В результате выполнения строка STR примет значение 'Hello World'.

INSERT

Функция вставляет строку в указанную позицию другой строки. Пример:

```
PROGRAM PLC_PRG
VAR
    STR: STRING;
END_VAR
```

```
STR:= INSERT ('He World', 'llo', 2 );
```

В результате выполнения строка STR примет значение 'Hello World'.

DELETE

Функция удаляет часть строки заданной длины с указанной позиции. Пример:

```
PROGRAM PLC_PRG
VAR
    STR: STRING;
END_VAR
```

```
STR:= DELETE ('Hello World', 4, 3 );
```

В результате выполнения строка STR примет значение 'HeWorld'.

REPLACE

Функция заменяет часть строки другой строкой заданной длины с указанной позиции. Пример:

```
PROGRAM PLC_PRG
VAR
```

```
        STR: STRING;
    END_VAR
```

```
    STR:= REPLACE ('Hello World', 'i', 4, 2);
```

В результате выполнения строка `STR` примет значение 'Hi World'.

FIND

Функция возвращает позицию заданного контекста в строке. Нумерация позиций в строке начинается с 1. Если контекст в строке не найден, функция возвращает 0. Если в строке несколько совпадений, возвращает позицию первого. Пример:

```
    PROGRAM PLC_PRG
    VAR
        Pos: INT;
    END_VAR
```

```
    Pos:= REPLACE ('Hello World', 'l');
```

В результате выполнения переменная `Pos` примет значение 3.

3.1.2 Переключатели

Важно!!! Все функциональные блоки требуют объявления экземпляра.

SR

Переключатель с доминантой включения. Выход Q1 равен FALSE до тех пор, пока вход SET станет равным TRUE. Выход Q1 при этом принимает значение TRUE и сохраняет его до тех пор, пока вход SET не примет значение FALSE, а вход RESET1 значение TRUE. Пример:

```
PROGRAM PLC_PRG
VAR
    SRInst: SR;  // объявление экземпляра функционального
блока
    VarBool1, VarBool2, Out: BOOL;
END_VAR
```

```
SRInst(SET1:= VarBool1, RESET:= VarBool2, Q1=> Out);
```

RS

Переключатель с доминантой выключения. Выход Q1 равен FALSE до тех пор, пока вход SET станет равным TRUE. Выход Q1 при этом принимает значение TRUE и сохраняет его до тех пор, пока вход RESET1 не примет значение TRUE, вне зависимости от состояния входа SET. Пример:

```
PROGRAM PLC_PRG
VAR
    RSInst: RS;  // объявление экземпляра функционального
блока
    VarBool1, VarBool2, Out: BOOL;
END_VAR
```

```
RSInst(SET:= VarBool1, RESET1:= VarBool2, Q1=> Out);
```

3.1.3 Детекторы импульсов

R_TRIG

Функциональный блок R_TRIG генерирует импульс по переднему фронту входного сигнала. Пример:

```
PROGRAM PLC_PRG
VAR
    R_TRIG_Inst: R_TRIG;  // объявление экземпляра
    VarBool1: BOOL;
    VarInt1: INT := 0;  // значение по умолчанию будет равно
нулю
END_VAR
```

```
R_TRIG_Inst(CLK:= VarBool1);
IF R_TRIG_Inst.Q THEN
      VarINT1:= VarINT1+1;
END_IF
```

Выход Q равен FALSE до тех пор, пока вход CLK равен FALSE. Как только CLK получает значение TRUE, Q устанавливается в TRUE. При следующем вызове функционального блока выход сбрасывается в FALSE. Таким образом, блок выдает единичный импульс при каждом переходе CLK из TRUE в FALSE. При этом переменная VarINT1подсчитывает количество таких импульсов.

F_TRIG

Функциональный блок F_TRIG генерирует импульс по заднему фронту входного сигнала. Пример:

```
PROGRAM PLC_PRG
VAR
    F_TRIG_Inst: F_TRIG;  // объявление экземпляра
    VarBool1: BOOL;
    VarInt1: INT := 0;  // значение по умолчанию будет равно
нулю
END_VAR
```

```
F_TRIG_Inst(CLK:=VarBool1);
IF F_TRIG_Inst.Q THEN
      VarINT1:= VarINT1+1;
END_IF
```

Выход Q равен FALSE до тех пор, пока вход CLK равен TRUE. Как только CLK получает значение FALSE, Q устанавливается в TRUE. При следующем вызове функционального блока выход сбрасывается в FALSE. Таким образом, блок выдает единичный импульс при каждом переходе CLK из TRUE в FALSE. При этом переменная VarINT1подсчитывает количество таких импульсов.

3.1.4 Счетчики

CTU

Функциональный блок «инкрементный счетчик». По каждому переднему фронту на входе CU выход CV увеличивается на 1. Выход Q устанавливается в TRUE, когда счетчик достигнет значения заданного PV. Счетчик CV сбрасывается в 0 по входу RESET = TRUE. Пример:

```
PROGRAM PLC_PRG
VAR
    CTUInst: CTU;  // объявление экземпляра
    VarBool1, VarBool2, OUT: BOOL;
    Counter: WORD := 0;  // значение по умолчанию будет равно
нулю
END_VAR
```

```
CTUInst(CU:= VarBool1, RESET:=VarBool2 , PV:= 10,
CV=>Counter, Q=>OUT);
```

В данном примере ведется подсчет переходов переменной `VarBool1` из FALSE в TRUE. При достижении счетчиком `Counter` значения 10, переменная `OUT` примет значение TRUE. Сброс счетчика `Counter` и выхода `OUT` осуществляется состоянием TRUE переменной `VarBool2`.

Важно!!! Обратите внимание, что при вызове функциональных блоков нужно обязательно указать значения для входных параметров. В противном случае они могут принять случайное значение. Выходные параметры могут быть присвоены после вызова функционального блока. Например:

```
CTUInst(CU:= VarBool1, RESET:=VarBool2 , PV:= 10,);
Counter:= CTUInst.CV;
OUT:= CTUInst.Q;
```

CTD

Функциональный блок «декрементный счетчик». По каждому переднему фронту на входе CD выход CV уменьшается на 1. Когда счетчик достигнет 0, счет останавливается, выход Q переключается в TRUE. Счетчик CV загружается начальным значением, равным PV по входу LOAD = TRUE. Пример:

```
PROGRAM PLC_PRG
VAR
    CTDInst: CTD;  // объявление экземпляра
    VarBool1, VarBool2, OUT: BOOL;
    Counter: WORD := 0;  // значение по умолчанию будет равно
нулю
END_VAR
```

```
CTDInst(CD:= VarBool1, LOAD:=VarBool2 , PV:= 10);
Counter:= CTDInst.CV;
OUT:= CTDInst.Q;
```

Важно!!! В данном примере переменная `CV` **функционального блока** `CTDInst` **– беззнаковая. По этому, не может быть меньше 0, а значит, после достижения этого значения счетчик теряет свой смысл.**

Хорошим примером будет следующая реализация:

```
CTDInst(CD:= VarBool1, LOAD:= CTDInst.Q , PV:= 10);
Counter:= CTDInst.CV;
```

В нем при достижении переменной `CV` нуля выход `Q` сбросит счетчик в начальное состояние.

CTUD

Функциональный блок «инкрементный / декрементный счетчик». По входу RESET счетчик CV сбрасывается в 0, по входу LOAD загружается значением PV. По фронту на входе CU счетчик увеличивается на 1. По фронту на входе CD счетчик уменьшается на 1 (до 0). QU устанавливается в TRUE, когда CV больше или равен PV. QD устанавливается в TRUE, когда CV равен 0. Данный счетчик объединяет в себе два предыдущих. Пример:

```
PROGRAM PLC_PRG
VAR
    CTUDInst: CTUD;  // объявление экземпляра
    VarBool1, VarBool2, RES1, OUT: BOOL;
    Counter: WORD ;
END_VAR
```

```
CTUDInst(CD:= VarBool1, CU:= VarBool2, RESET:=RES1 , LOAD:=
CTUDInst.QD , PV:= 10);
Counter:= CTUDInst.CV;
OUT:= CTUDInst.QU;
```

В исходном состоянии такого счетчика значение CV будет равным 10. Так как при достижении им нуля будет срабатывать выход QD и загружать значение PV в CV. При достижении счетчика значения больше или равным 10, переменная `OUT` будет в состоянии TRUE. Переменной `RES1` счетчик сбрасывается в исходное состояние.

3.1.5 Таймеры

TP

Функциональный блок «таймер». Пока IN равен FALSE, выход Q = FALSE, выход ET = 0 (исходное состояние). При переходе IN в TRUE выход Q устанавливается в TRUE и таймер начинает отсчет времени (в миллисекундах) на выходе ET до достижения длительности, заданной PT. Далее счетчик не увеличивается. Таким образом, выход Q генерирует импульс длительностью PT по фронту входа IN. Пример:

```
PROGRAM PLC_PRG
VAR
    TPInst: TP;  // объявление экземпляра
    VarBool1, VarBool2, OUT: BOOL;
    ElpasedTime: TIME;
END_VAR
```

```
TPInst(IN := VarBool1, PT:= T#10S);
VarBool2 := TPInst.Q;
ElpasedTime:= TPInst.ET;
```

Временная диаграмма работы таймера TR изображена на рисунке 60.

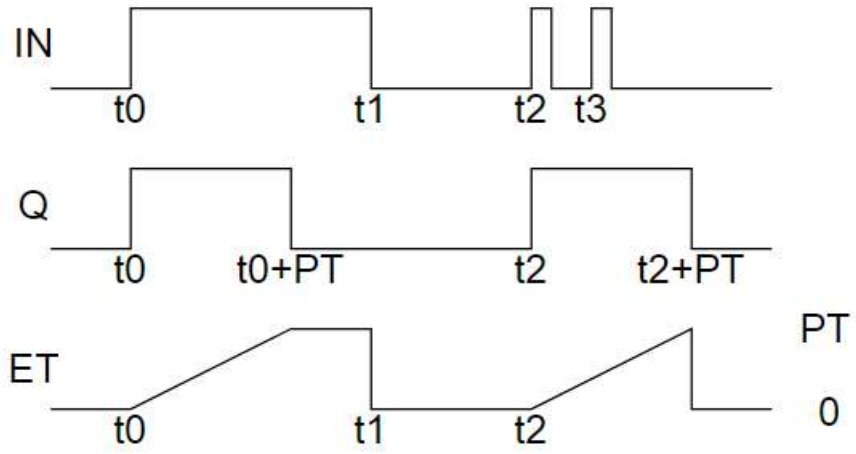


Рисунок 60 – Временная диаграмма работы таймера TR

Стоить заметить, что таймер возвращается в исходное состояние при условии ET=PT и IN=FALSE.

TON

Функциональный блок «таймер с задержкой включения». Пока IN равен FALSE, выход Q = FALSE, выход ET = 0 (исходное состояние). Как только IN становится TRUE, начинается отсчет времени (в миллисекундах) на выходе ET до значения, равного PT. Далее счетчик не увеличивается. Q равен TRUE, когда IN равен TRUE и ET равен PT, иначе FALSE. Таким образом, выход Q устанавливается с задержкой PT от фронта входа IN. Пример:

```
PROGRAM PLC_PRG
VAR
    TONInst: TON;  // объявление экземпляра
    VarBool1, VarBool2, OUT: BOOL;
    ElpasedTime: TIME;
END_VAR
```

```
TONInst(IN := VarBOOL1, PT:= T#3S);
VarBOOL2 := TONInst.Q;
ElpasedTime:= TONInst.ET;
```

Временная диаграмма работы таймера TON изображена на рисунке 61.

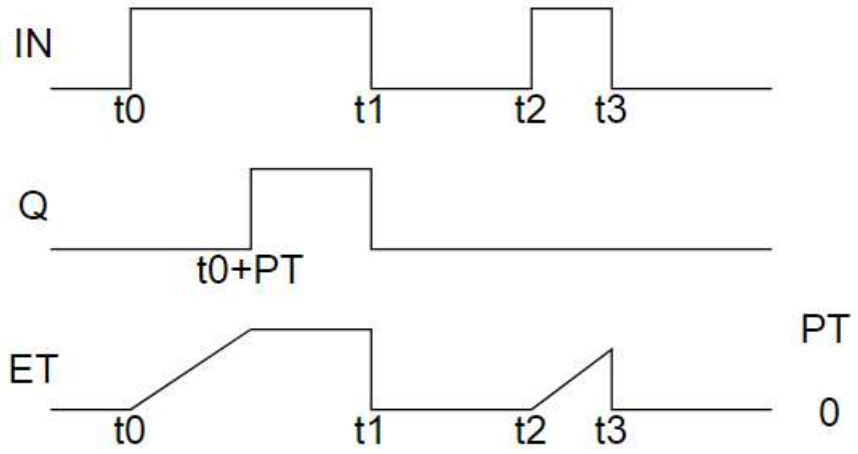


Рисунок 61 – Временная диаграмма работы таймера TON

Таймер возвращается в исходное состояние при условии IN=FALSE.

TOF

Функциональный блок «таймер с задержкой выключения». Если IN равен TRUE, то выход Q = TRUE и выход ET = 0 (исходное состояние). Как только IN переходит в FALSE, начинается отсчет времени (в миллисекундах) на выходе ET. При достижении заданной длительности отсчет останавливается. Выход Q равен FALSE, если IN равен FALSE и ET равен PT, иначе - TRUE. Таким образом, выход Q сбрасывается с задержкой PT от спада входа IN. Пример:

```
PROGRAM PLC_PRG
VAR
    TOFInst: TOF;  // объявление экземпляра
    VarBool1, VarBool2, OUT: BOOL;
    ElpasedTime: TIME;
END_VAR
```

```
TOFInst(IN := VarBOOL1, PT:= T#5S);
VarBOOL2 := TOFInst.Q;
ElpasedTime:= TOFInst.ET;
```

Временная диаграмма работы таймера TOF изображена на рисунке 62.

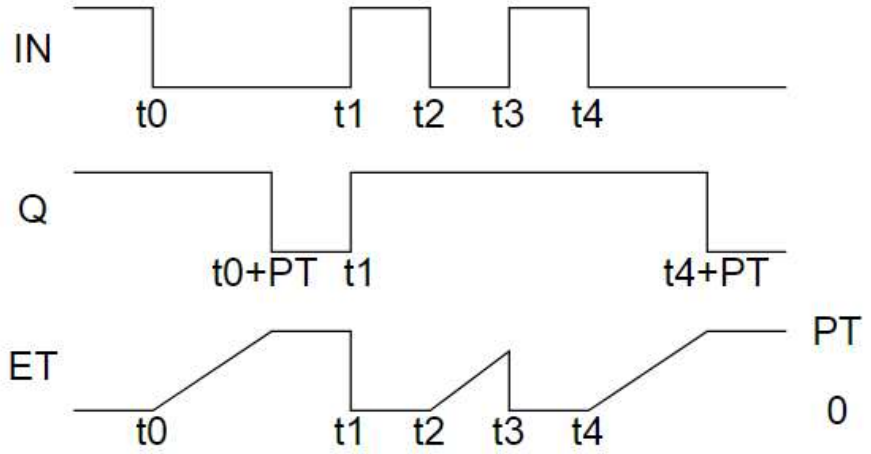


Рисунок 62 – Временная диаграмма работы таймера TOF

Таймер возвращается в исходное состояние при условии IN=TRUE.

3.2 Работа с библиотеками CmpModbusKAPP82 для реализации протокола Modbus

Для работы контроллера по протоколу Modbus необходимо добавить библиотеку CmpModbusKAPP82 (эта библиотека подходит как для КАПП-82-168 так и для КАПП2-00-000-1). Для этого откройте «Менеджер библиотек», нажмите кнопку «Добавить библиотеку» (рисунок 63).

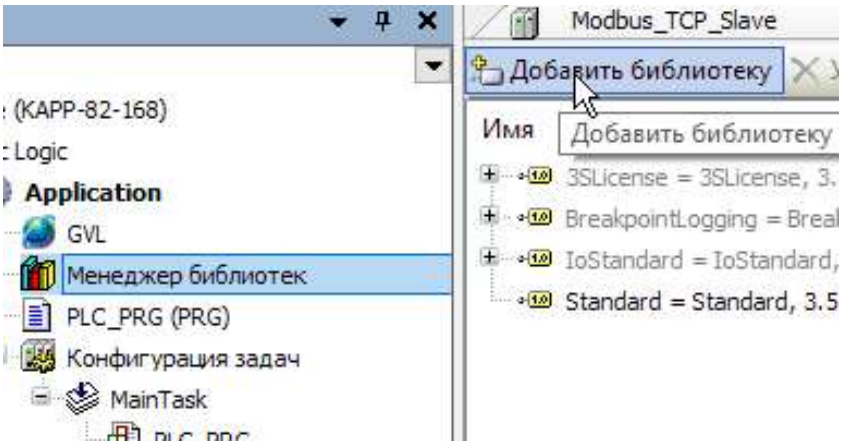


Рисунок 63 – Менеджер библиотек

В открывшемся окне «Библиотека» раскройте список «(Смешан.)», найдите необходимую библиотеку и нажмите кнопку «ОК» (рисунок 64).

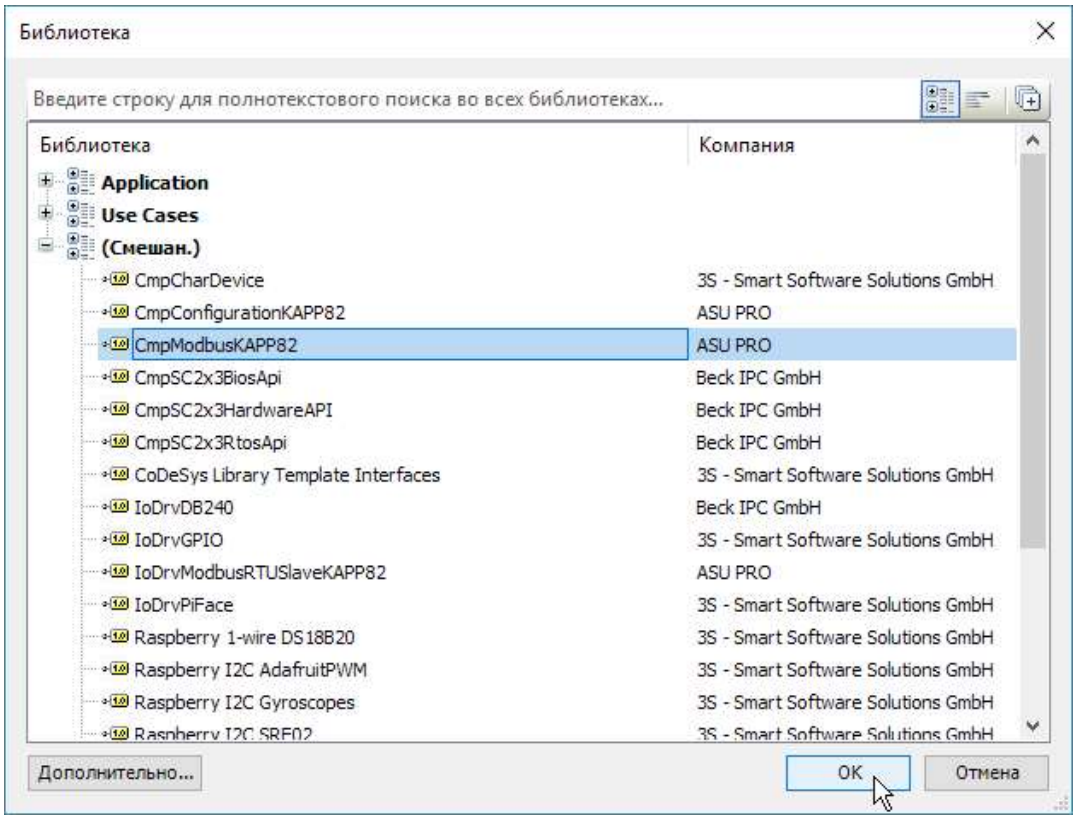


Рисунок 64 – Добавление библиотеки в проект

Список установленных библиотек в менеджере должен обновиться (рисунок 65).

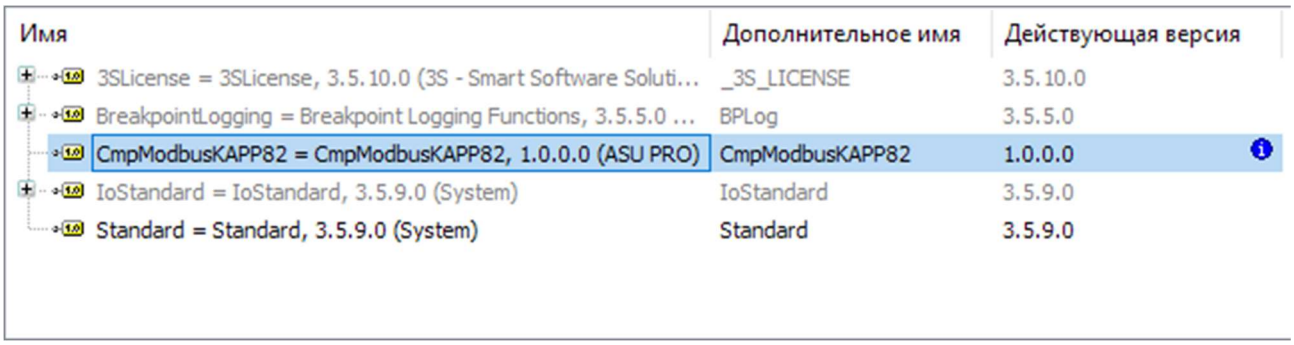


Рисунок 65 – Менеджер библиотек после добавления библиотеки CmpModbusKAPP82

3.2.1 Работа в качестве ведомого устройства по интерфейсам RS-485, RS-232 (Modbus RTU Slave)

Для работы контроллера в таком режиме необходимо добавить драйвер Modbus RTU Slave. Для этого в дереве проекта кликаем правой кнопкой по иконке «Устройства» («Device») для вызова контекстного меню, в котором нужно выбрать пункт «Добавить устройство» (рисунок 66).

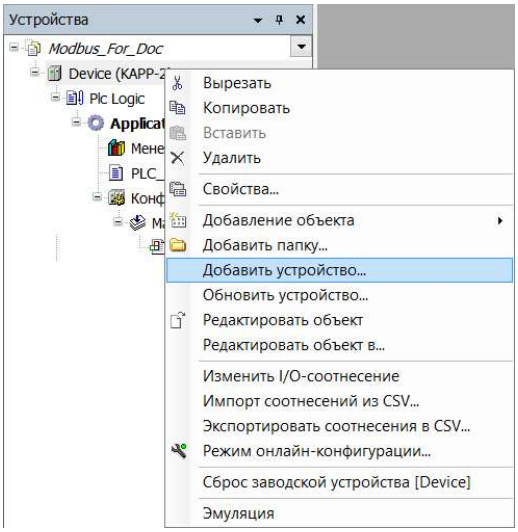


Рисунок 66 – Добавление нового устройства в менеджере проекта

Далее в верхней части открывшегося окна раскрываем дерево «Разн.» и находим Modbus RTU Slave производителя ASU PRO (рисунок 67).

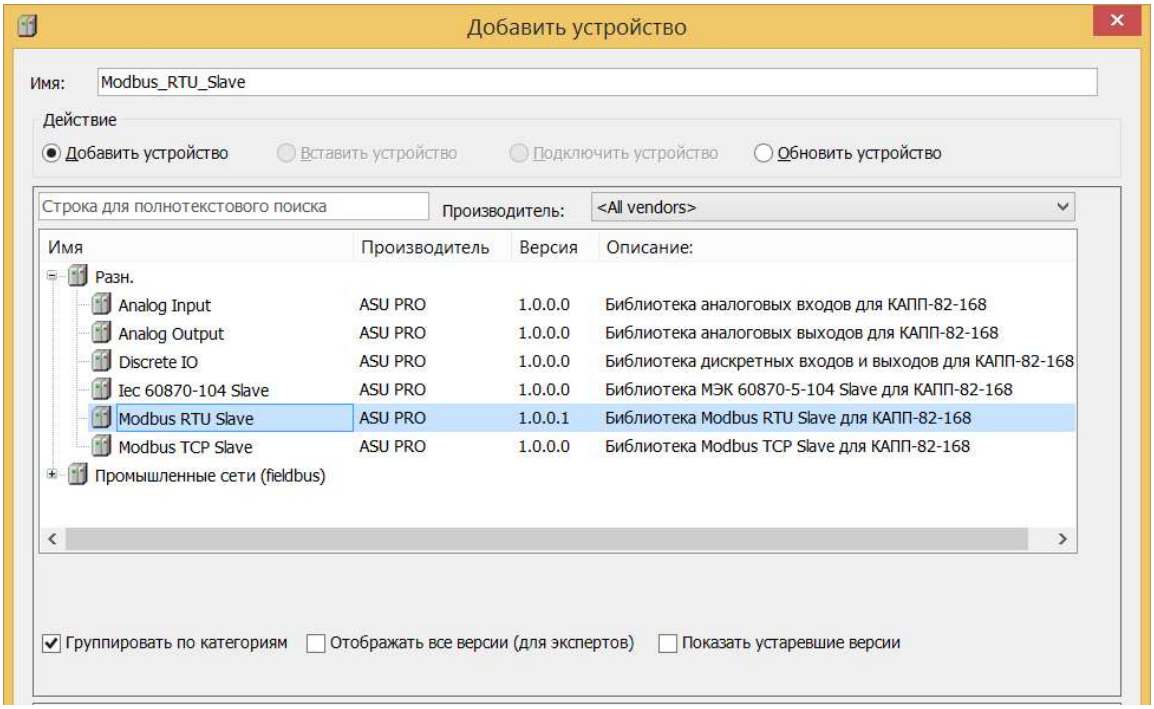


Рисунок 67 – Расположение устройства Modbus RTU Slave

Далее внизу, жмем кнопку «Добавить устройство» и закрываем окно «Добавить устройство». После чего в дереве проекта появится новое устройство Modbus RTU Slave (рисунок 68).

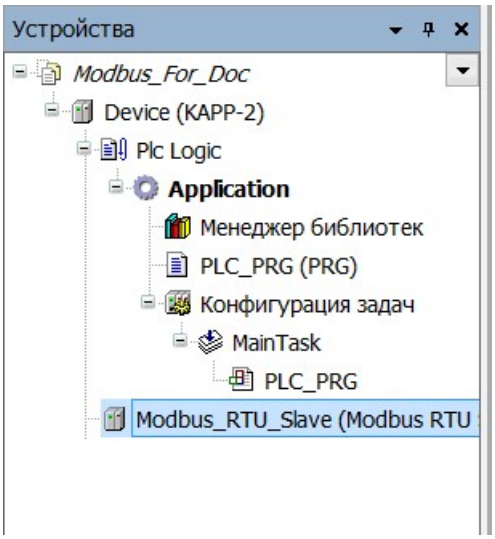


Рисунок 68 – Дерево проекта после добавления Modbus RTU Slave

Далее, необходимо выделить область памяти под регистры Modbus, создав список глобальных переменных. Для этого необходимо щелкнуть правой клавишей мыши по иконке «Приложение» («Application»), а далее в контекстном меню выбрать подпункт «Список глобальных переменных…» пункта «Добавить объект» (рисунок 69).

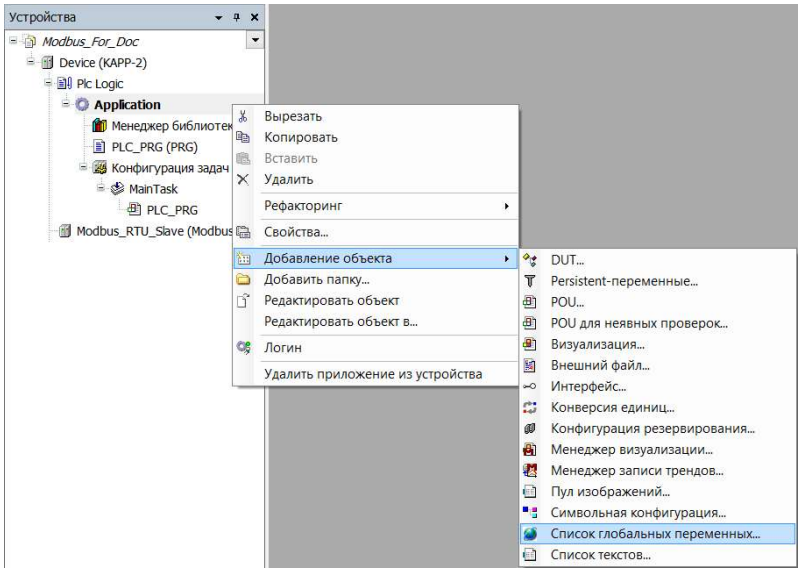


Рисунок 69 – Создание списка глобальных переменных

Назовите новый элемент «Global» и нажмите кнопку «Добавить».
Созданный список глобальных переменных должен содержать следующий код:

```
VAR_GLOBAL CONSTANT
    // Количество флагов
    numCoils: DINT := 200;
    // Количество дискретных входов
    numDiscreteInputs: DINT := 200;
    // Количество входных регистров
    numInputRegisters: DINT := 200;
    // Количество регистров хранения
    numHoldingRegisters: DINT := 200;
END_VAR

VAR_GLOBAL
    coils:              ARRAY[0..Global.numCoils-1]              OF
    CmpModbusKAPP82.ModbusCoil;
    discreteInputs:  ARRAY[0..Global.numDiscreteInputs-1]  OF
    CmpModbusKAPP82.ModbusDiscreteInput;
    inputRegisters:  ARRAY[0..Global.numInputRegisters-1]  OF
    CmpModbusKAPP82.ModbusInputRegister;
    holdingRegisters:  ARRAY[0..Global.numHoldingRegisters-1]
    OF CmpModbusKAPP82.ModbusHoldingRegister;
    mapping: CmpModbusKAPP82.ModbusMapping :=
        (NumberOfCoils              :=              Global.numCoils,
        NumberOfDiscreteInputs  :=  Global.numDiscreteInputs,
        NumberOfInputRegisters  :=  Global.numInputRegisters,
        NumberOfHoldingRegisters    :=    numHoldingRegisters,
        TabCoils    :=    ADR(coils),    TabDiscreteInputs    :=
        ADR(discreteInputs),        TabInputRegisters        :=
        ADR(inputRegisters),      TabHoldingRegisters      :=
        ADR(holdingRegisters)
```

```
        );
    END_VAR
```

Совет!!! Для ускорения процесса создания списка глобальных переменных можно просто скопировать текст из вышеуказанной таблицы. Так же это позволит избежать вероятных опечаток при наборе кода вручную.

После чего вкладка списка глобальных переменных примет следующий вид (рисунок 70)

Менеджер библиотек | Modbus_RTU_Slave | Global ×

```
{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
    // Количество флагов
    numCoils: DINT := 200;
    // Количество дискретных входов
    numDiscreteInputs: DINT := 200;
    // Количество входных регистров
    numInputRegisters: DINT := 200;
    // Количество регистров хранения
    numHoldingRegisters: DINT := 200;
END_VAR

VAR_GLOBAL
    coils: ARRAY[0..Global.numCoils-1] OF   CmpModbusKAPP82.ModbusCoil;
    discreteInputs: ARRAY[0..Global.numDiscreteInputs-1] OF   CmpModbusKAPP82.ModbusDiscreteInput;
    inputRegisters: ARRAY[0..Global.numInputRegisters-1] OF   CmpModbusKAPP82.ModbusInputRegister;
    holdingRegisters: ARRAY[0..Global.numHoldingRegisters-1] OF   CmpModbusKAPP82.ModbusHoldingRegister;
    mapping: CmpModbusKAPP82.ModbusMapping := (
        NumberOfCoils := Global.numCoils,
        NumberOfDiscreteInputs := Global.numDiscreteInputs,
        NumberOfInputRegisters := Global.numInputRegisters,
        NumberOfHoldingRegisters := numHoldingRegisters,
        TabCoils := ADR(coils),
        TabDiscreteInputs := ADR(discreteInputs),
        TabInputRegisters := ADR(inputRegisters),
        TabHoldingRegisters := ADR(holdingRegisters)
    );
END_VAR
```

Рисунок 70 – Код списка глобальных переменных

В данном примере размер массивов флагов (Coils), дискретных входов (Discrete Inputs), регистров хранения (Holding Registers), регистров ввода (Input Registers) равен двумстам элементам. Чтобы изменить количество необходимых регистров, изменяйте значения глобальных констант (раздел **VAR_GLOBAL CONSTANT**). Для правильной работы драйвера, не изменяйте текст объявлений глобальных массивов регистров и структуры таблицы регистров (mapping) в разделе **VAR_GLOBAL**.

Далее нужно настроить драйвер Modbus RTU Slave, нажав двойным кликом левой кнопкой мыши на соответствующий значок в дереве проекта. При этом в центральной части откроется вкладка Modbus_RTU_Slave (рисунок 71).

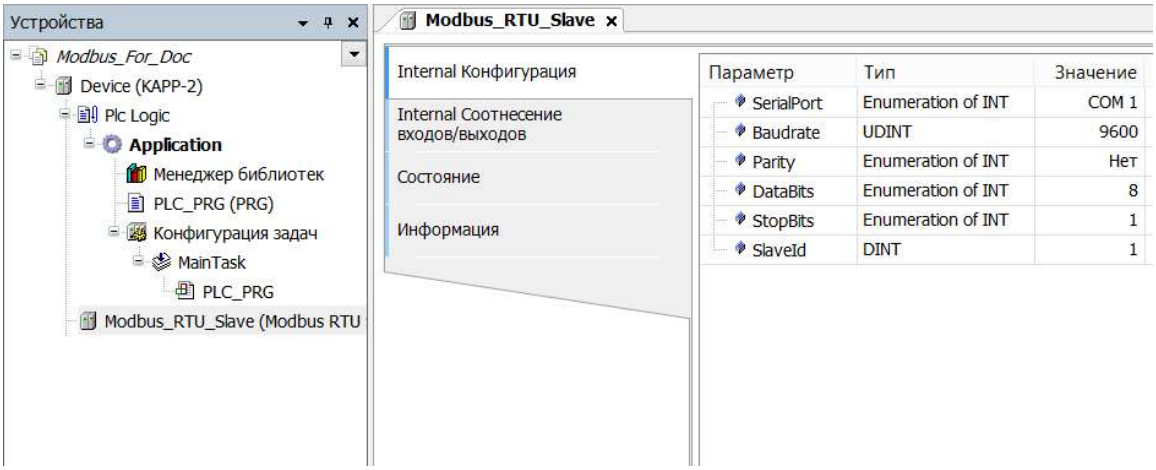


Рисунок 71 – Вкладка Modbus_RTU_Slave, раздел «Конфигурация»

В первом разделе «Конфигурация» вкладки Modbus RTU Slave можно настроить параметры порта, такие как номер порта, скорость, четность, размер данных, количество стоповых бит и адрес устройства.

Во втором разделе «Соотнесение входов/выходов» необходимо каналу ModbusMapping соотнести переменную `mapping`, объявленную ранее в глобальных переменных раздела **`VAR_GLOBAL.`** Проще всего это сделать, щелкнув дважды по строке «Переменная», после чего в правой части строки появятся кнопка ассистента ввода (рисунок 72)

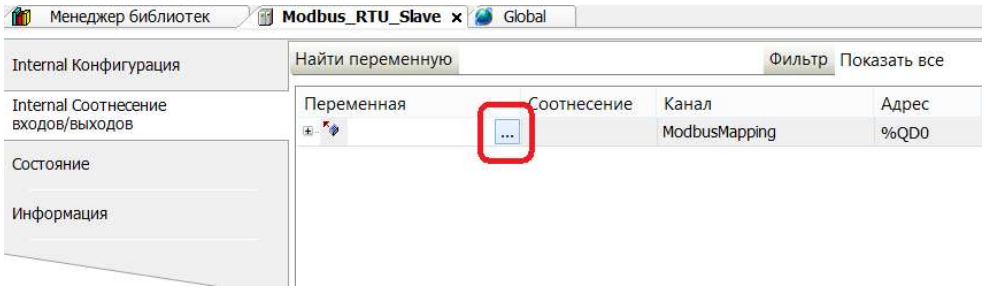


Рисунок 72 – Кнопка вызова ассистента ввода

Вызываем данной кнопкой окно ассистента ввода, ищем в нем структуру переменных `mapping` (расположение переменной Application/Global) и щелкаем двойным левым кликом мыши (рисунок 73).

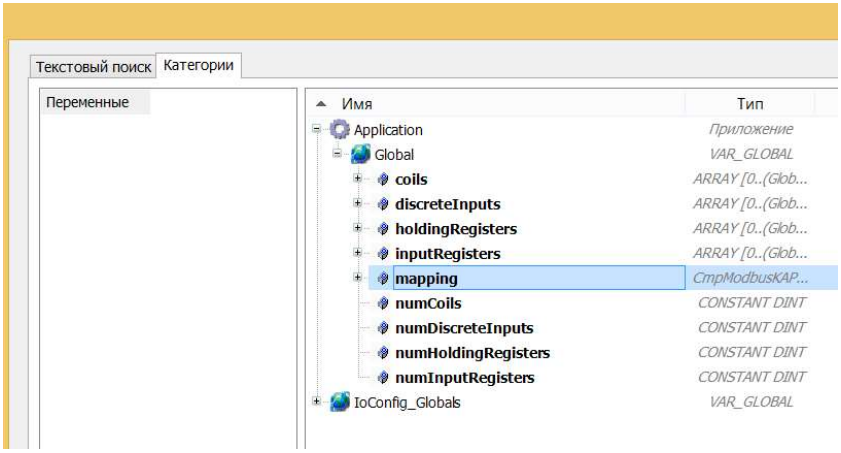


Рисунок 73 – Нахождение переменной mapping с помощью ассистента ввода

Далее, для того чтобы структура переменных `mapping` обновлялась всегда, выберите пункт «Вкл. 2 (всегда в задаче цикла шины)» списка «Всегда обновлять переменные» в окне «Соотнесение входов/выходов» устройства (рисунок 74).

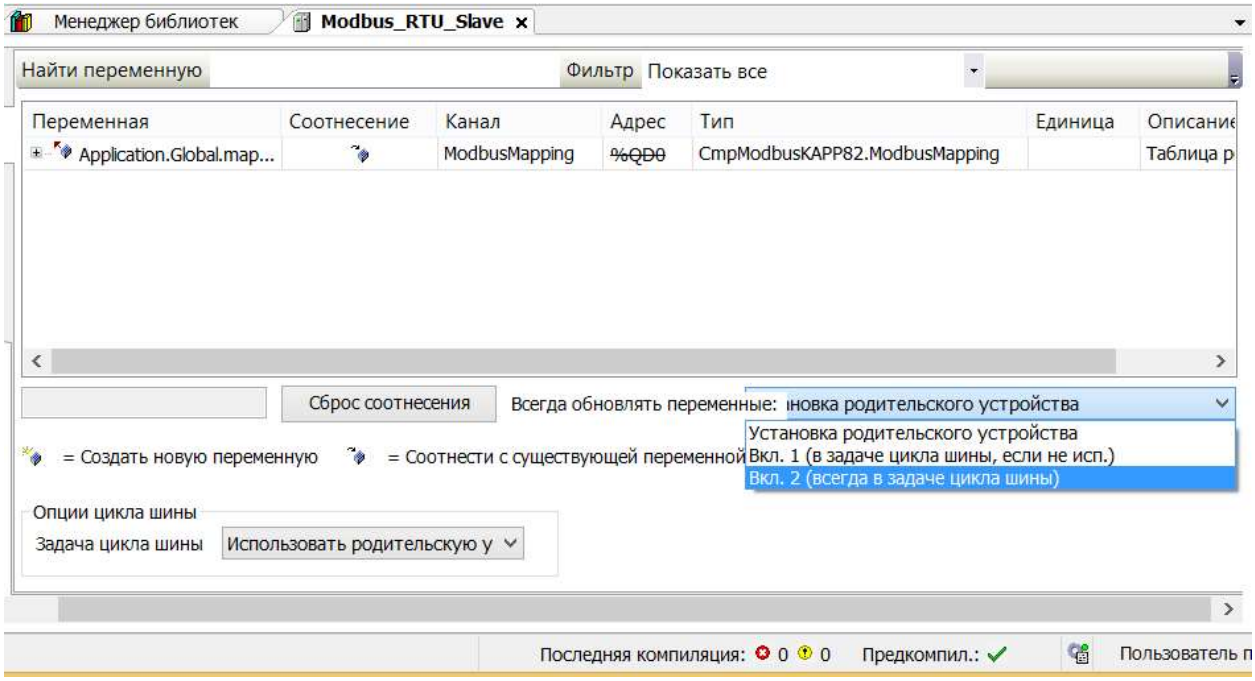


Рисунок 74 – Настройки обновления переменных

После загрузки приложения в контроллер, убедимся, что драйвер работает. Для этого необходимо обратить внимание на значок драйвера Modbus RTU Slave в дереве проекта (рисунок 75).

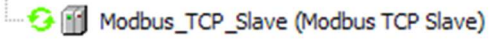


Рисунок 75 – Значок корректной работы драйвера

Возможно появление значка ошибки в работе драйвера (рисунок 76).

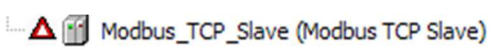


Рисунок 76 – Значок ошибки в работе драйвера

Возможные причины ошибки:
- не присвоена таблица регистров (`mapping`);
- размер всех типов данных в таблице регистров равен нулю;
- неверные размерности массивов под регистры;
- не настроено постоянное обновление переменных.

Для работы с регистрами предназначены функции библиотеки CmpModbusKAPP82 Get и Set для каждого типа регистров Modbus. Все функции можно посмотреть в Приложении В.

Так же все функции, описание и документацию можно найти в Менеджере библиотек, библиотека CmpModbusKAPP82 в папке Mapping (рисунок 77).

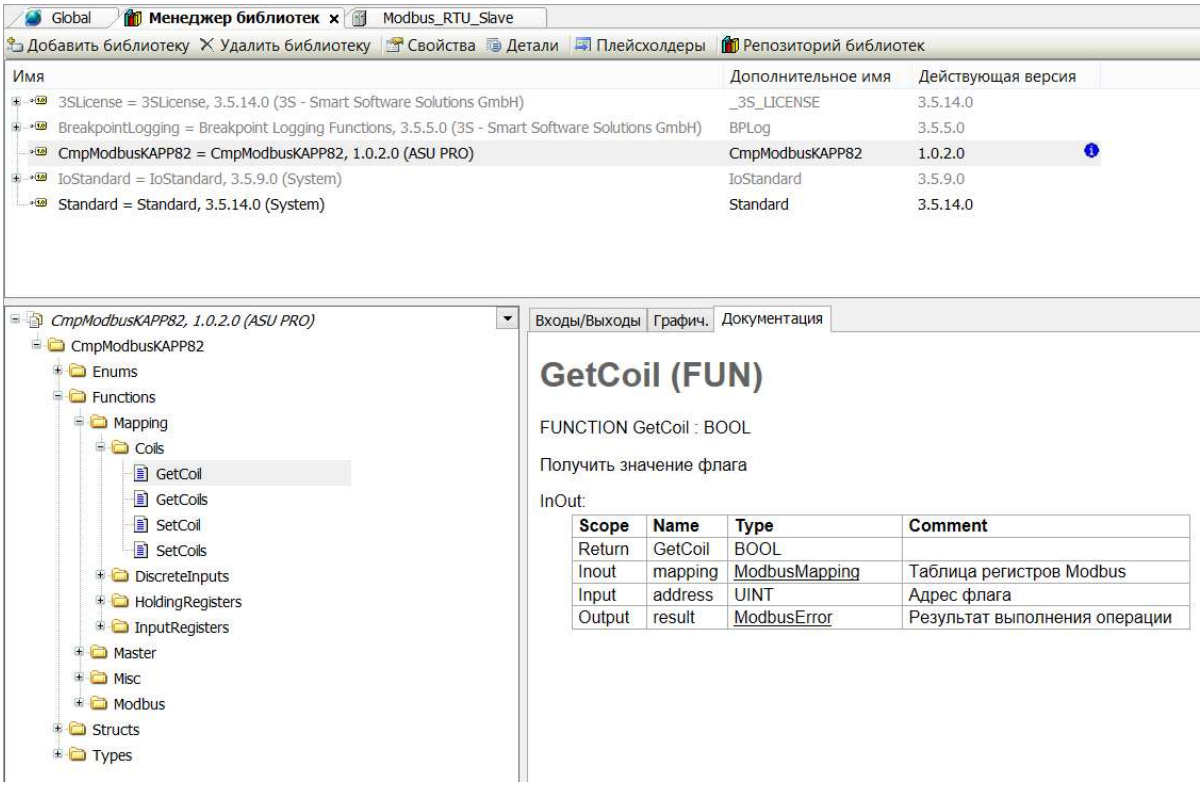


Рисунок 77 – Документация библиотеки CmpModbusKAPP82

Например, чтобы записать значение переменной I в нулевой регистр, используется следующий код:

```
CmpModbusKAPP82.SetHoldingRegister(Global.mapping, 0, i);
```

Совет!!! Для работы с регистрами Modbus необязательно пользоваться вышеуказанными функциями. С регистрами можно работать напрямую. При этом нужно учитывать, что регистры флагов (Coils) и дискретных входов (Discrete Inputs) в структуре `mapping` **имеют тип BYTE, по причине дискретности передаваемой информации равной одному байту, хотя и логически могут принимать значения 0 и**

1, что соответствует типу BOOL. По этому, для присвоения переменных с разным типом данных необходимо применять преобразования типа (`TO_BYTE`, `TO_BOOL`). Например:

```
PROGRAM PLC_PRG
VAR
    VarBool1, VarBool2: BOOL;
    VarUint1: UINT;
END_VAR
```

```
Global.coils[0]:= TO_BYTE(VarBool1);
VarBool2:= TO_BOOL(global.coils[2]);
Global.holdingRegisters[5]:= VarUint1;
```

Важно!!! Регистры флагов (Coils) и дискретных входов (Discrete Inputs) не должны принимать значения отличные от 0 и 1. Иначе при опросе ведущим устройством будет считываться недостоверная информация.

Несколько драйверов Modbus могут использовать одну и ту же таблицу регистров (mapping). Это полезно когда необходимо передавать одни и те же данные двум и более ведущим устройствам по разным портам в разных сетях. Например, по интерфейсу RS-232 на панель оператора через драйвер Modbus RTU Slave по порту COM1, другому контроллеру по интерфейсу RS-485 через драйвер Modbus RTU Slave по порту COM2, SCADA – системе по интерфейсу Ethernet через драйвер Modbus TCP Slave. При этом не нужно создавать несколько глобальных структур типа `mapping`, нужно лишь каждому соответствующему каналу ModbusMapping соотнести одну структуру `mapping`.

3.2.2 Работа в качестве ведомого устройства по интерфейсу Ethernet (Modbus TCP Slave)

Для работы контроллера в таком режиме необходимо добавить драйвер Modbus TCP Slave. Добавление драйвера происходит аналогично драйверу Modbus RTU Slave (рисунок 78).

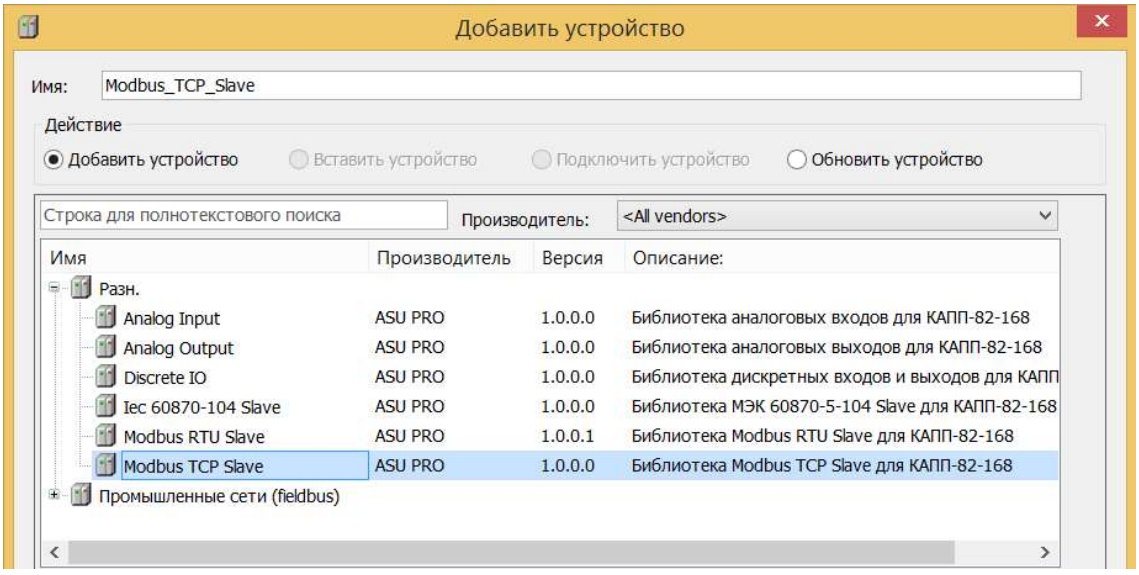


Рисунок 78 – Расположение устройства Modbus TCP Slave

Если ранее область памяти под регистры Modbus не выделялась (структура `mapping`) или нужна другая структура `mapping`, выделяем новую область памяти точно так же как и ранее, для драйвера Modbus RTU Slave в пункте 2.4.3. Аналогично предыдущему драйверу соотносим каналу ModbusMapping переменную `mapping`, не забывая при этом настроить обновление переменных всегда в задаче цикла шины (рисунок 79).

По умолчанию компонент Modbus TCP Slave выполняется в задаче с наименьшим временем цикла. Поэтому для корректной работы компонента задача, в которой реализованы функции библиотеки CmpModbusKAPP82 для работы с Modbus TCP Slave, должна иметь наименьшее время цикла.

Либо возможно привязать выполнение компонента Modbus RTU Slave к задаче, в которой реализованы функкции библиотеки CmpModbusKAPP82 принудительно, установив в настройках «Опции цикла шины» параметр «Задача цикла шины» нужную задачу, например MainTask (рисунок 79).

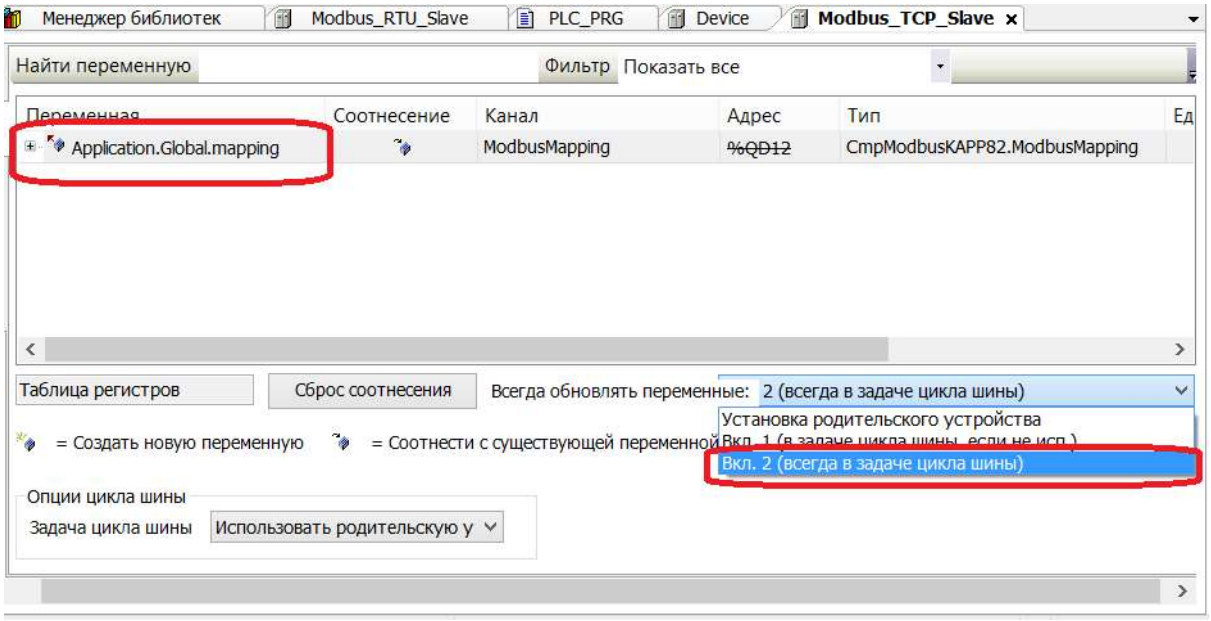


Рисунок 79 – Вкладка Modbus_TCP_Slave, раздел «Соотнесение входов/выходов»

В первом разделе «Конфигурация» вкладки Modbus TCP Slave можно настроить номер порта TCP и максимальное количество одновременных клиентов (рисунок 80).

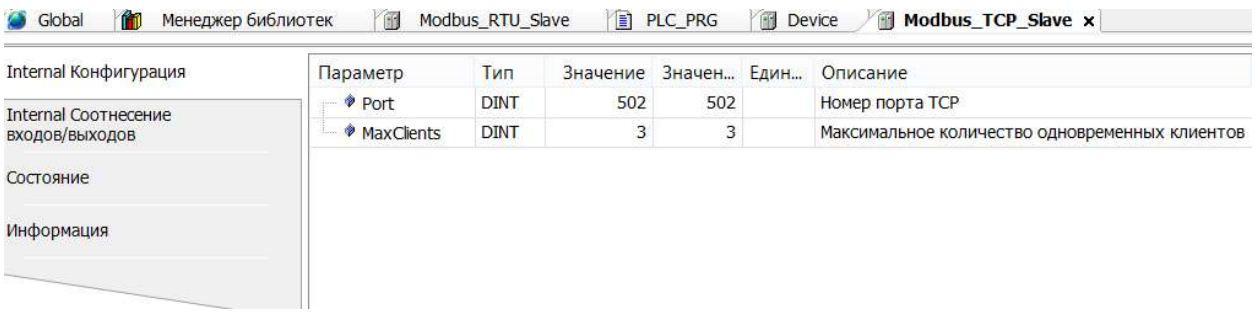


Рисунок 80 – Вкладка Modbus_TCP_Slave, раздел «Конфигурация»

При этом IP – адрес устройства не меняется и определяется конфигурационным файлом CODESYSControl.cfg (рисунок 81)

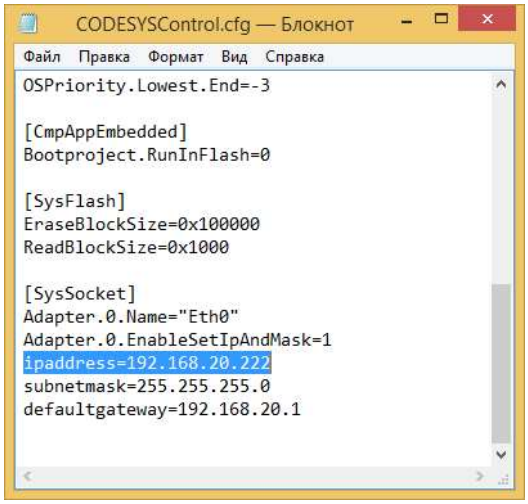


Рисунок 81 – Содержание конфигурационного файла CODESYSControl.cfg

Так же стоит учитывать, что у модуля КАПП2-00-000-1 один порт Ethernet.

Работы с регистрами Modbus осуществляется так же, как и в случае драйвера Modbus RTU Slave пункт 2.4.3.

3.2.3 Работа в качестве ведущего устройства по интерфейсам RS-485, RS-232 (Modbus RTU Master)

Для работы в качестве ведущего устройства драйверов подобным Modbus RTU Slave и Modbus TCP Slave для КАПП2-00-000-1 нет. Для такой работы алгоритм опроса ведомых устройств необходимо описывать самому. Алгоритм Modbus Master должен исполняться в отдельной задаче. Для этого необходимо щелкнуть правой кнопкой мыши по значку «Конфигурация задач» и в контекстном меню «Добавление объекта» выбрать «Задача…» (рисунок 82).

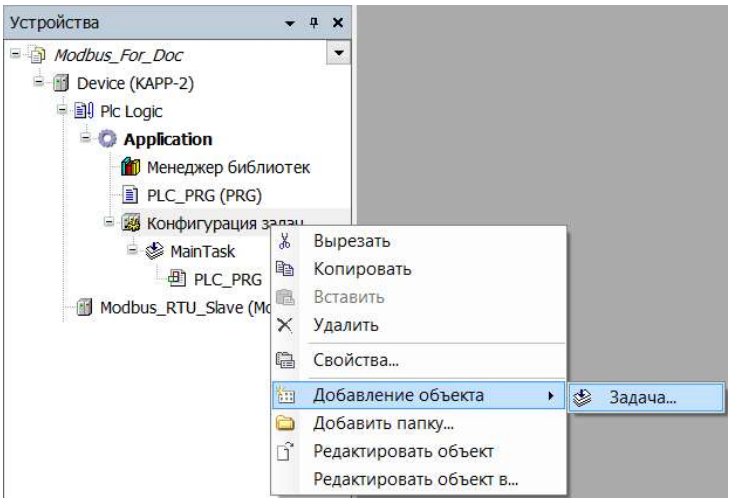


Рисунок 82 – Добавление новой задачи

В диалоговом окне «Добавить Задача» укажем имя новой задачи ModbusMaster и нажмем кнопку «Добавить» (рисунок 83).

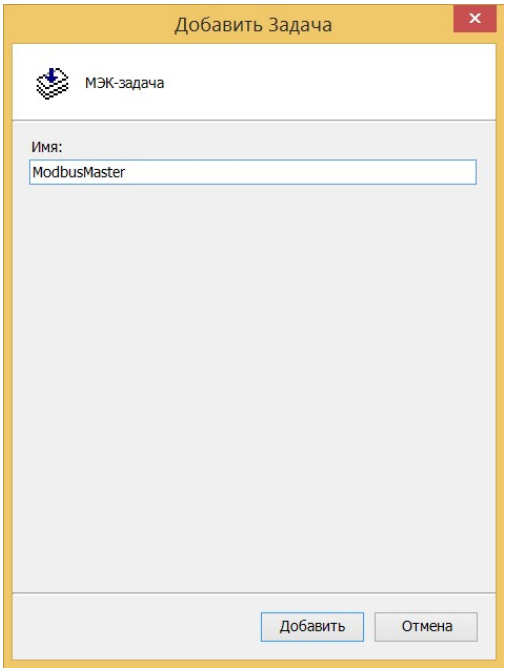


Рисунок 83 – Создание имени новой задачи

После чего откроется окно «Конфигурация» вкладки задачи ModbusMaster (рисунок 84).

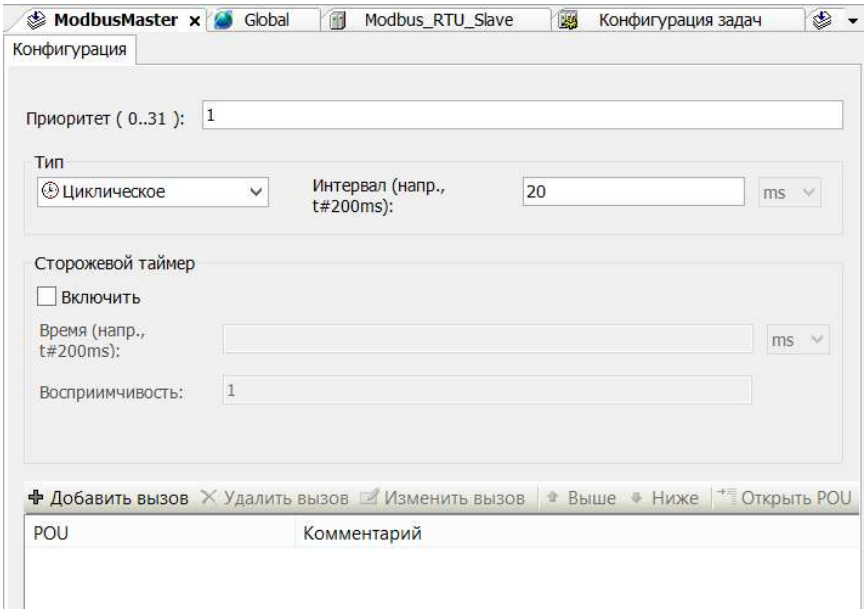


Рисунок 84 – Окно конфигурации задачи

Как видно из рисунка задача будет выполняться циклически каждые 20 мс.

Далее необходимо создать новую программу. Для этого щелкаем правой кнопкой мыши по значку «Приложение» («Application») и выбираем в подменю пункт «POU…» меню «Добавление объекта» (рисунок 85)

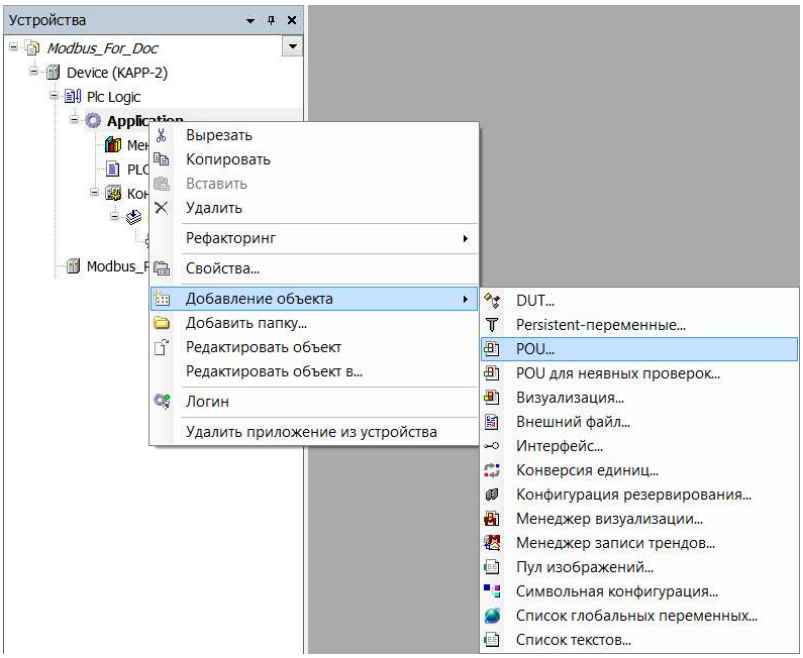


Рисунок 85 – Добавление объекта POU

В открывшемся окне «Добавить POU» выбираем тип «Программа», язык реализации «Структурированный текст (ST)» и даем имя RTU и жмем кнопку «Добавить» (рисунок 86)

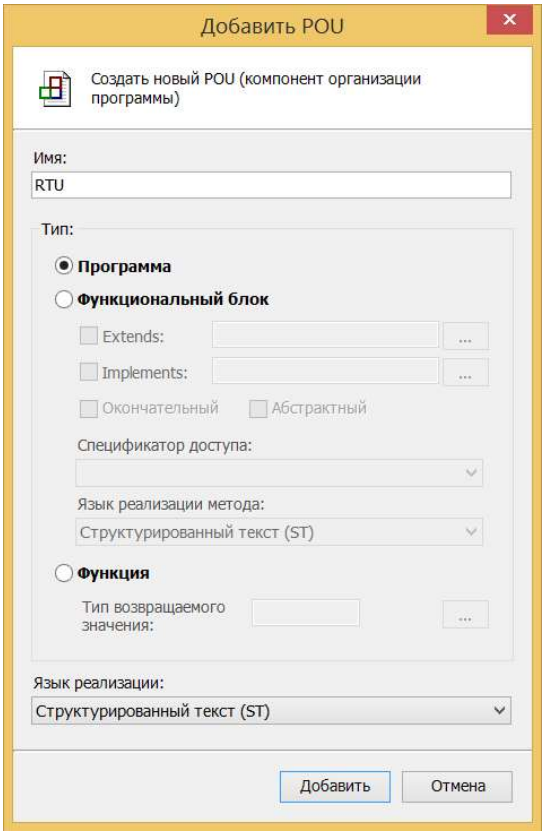


Рисунок 86 – Окно настроек добавления нового POU

Далее нужно добавить вызов созданной программы RTU в задаче ModbusMaster. Для этого в конфигурации задачи ModbusMaster жмем кнопку «Добавить вызов POU» (рисунок 87).

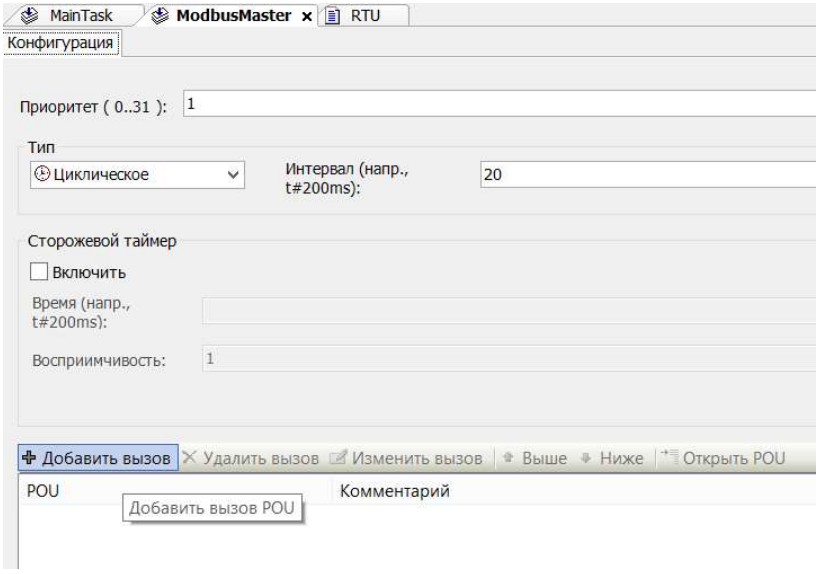


Рисунок 87 – Добавление вызова POU

В открывшемся окне ассистента ввода найти и выделить программу RTU, а затем нажать кнопку «ОК» (рисунок 88)

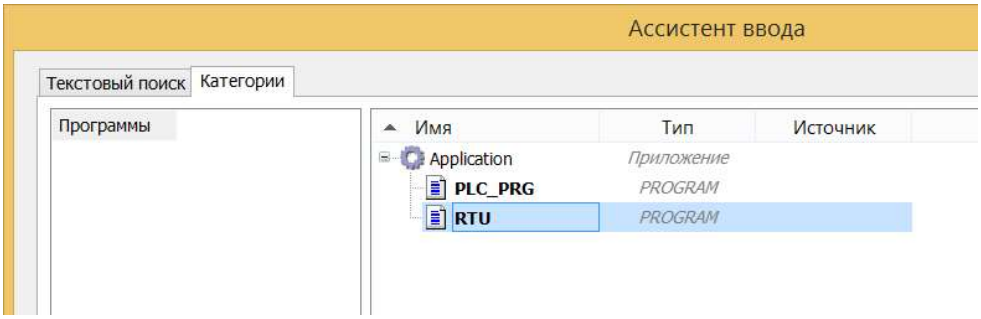


Рисунок 88 – Добавление программы в задачу с помощью ассистента ввода

Все необходимая подготовка программы выполнена, после чего необходимо написать саму программу опроса подчиненных устройств.

Задача: написать код для опроса первых 100 регистров хранения подчиненного устройства с адресом 10. Параметры соединения: Номер последовательного порта – COM1; Скорость последовательного порта – 9600 Бод; Чётность – нет; Количество битов данных – 8; Количество стоп-битов – 1; Таймаут опроса – 1 с.

Пример кода для выполнения вышеуказанной задачи:

```
VAR_GLOBAL
    buffer : ARRAY [0..99] OF WORD;
END_VAR
```

```
PROGRAM RTU
VAR
    created: BOOL := FALSE;
    ctx: CmpModbusKAPP82.ModbusContext;
    result: CmpModbusKAPP82.ModbusError;
    errors, success: UDINT;
END_VAR
```

```
IF NOT created THEN

      created := TRUE;

 // Создаём экземпляр драйвера Modbus RTU (выставляем
ему параметры соответствующие slave устройству)
 CmpModbusKAPP82.ModbusNewRtu(
      4,
      9600,
      CmpModbusKAPP82.ModbusParity.NONE,
      CmpModbusKAPP82.ModbusDataBits.BITS_8,
      CmpModbusKAPP82.ModbusStopBits.ONE,
      context => ctx
 );
 // Устанавливаем таймаут в 1 секунду
 CmpModbusKAPP82.ModbusSetResponseTimeout(ctx, 1, 0);
 // Устанавливаем Slave ID 10
 CmpModbusKAPP82.ModbusSetSlave(ctx, 10);
ELSE
 // Читаем первые 100 регистров
```

```
    result := CmpModbusKAPP82.ReadHoldingRegisters(ctx, 0, 10,
   ADR(Global.buffer));
    IF result <> CmpModbusKAPP82.ModbusError.ERROR_OK THEN
        errors := errors + 1;
    ELSE
        success := success + 1;
    END_IF
   END_IF
```

В рассмотренном примере, при запуске контроллера переменная `created = FALSE` по этому, в первом цикле выполнится настройка порта COM1 и определится его контекст. После чего переменная `created` станет `TRUE`, и данная процедура выполнятся больше не будет до перезапуска контроллера, а начнет выполнятся процедура опроса ведомого устройства. В рассмотренном примере Буфер объявляется в глобальном списке переменных, для возможности работы с ним из всех программ проекта. В функции `CmpModbusKAPP82.ReadHoldingRegisters` для записи в буфер `Global.buffer` используется не сама переменная, а указатель к ней **`ADR`**`(Global.buffer)`. При этом происходит заполнение элементов `Global.buffer` (с нулевого элемента) считанными регистрами, в количестве указанном в функции ReadHoldingRegisters (в данном случае 100 элементов). Функция ModbusFlush является не обязательной, однако очищение входного буфера перед приемом нового сообщения, обезопасит его от влияния предыдущего мусора на конечный результат. Иными словами уменьшит количество ошибок при обмене.

Важно!!! При включении контроллера процедура настройки порта для каждого подключения должна выполняться один раз, иначе множество созданных контекстов могут переполнить память устройства, контроллер не сможет работать корректно.

Важно!!! Размеры массивов (буферов) для регистров должны быть не меньше, чем количество запрашиваемых или записываемых регистров. В противном случае, возможны ошибки работы с памятью. Контроллер при этом не будет работать корректно.

Совет!!! Для возможности изменения адреса функцией CmpModbusKAPP82.ModbusSetSlave(ctx, SlaveID) можно использовать вместо канстанты перменную SlaveID. Данную функцию использовать перед функцией чтени/записи.

Полный список функций для настройки порта можно посмотреть в Приложении Г.

Полный список функций для чтения и записи можно посмотреть в Приложении Д.

Так же все функции, описание и документацию можно найти в Менеджере библиотек, настройка портов в папке Modbus, функции чтения и записи в папке Master (рисунок 89).

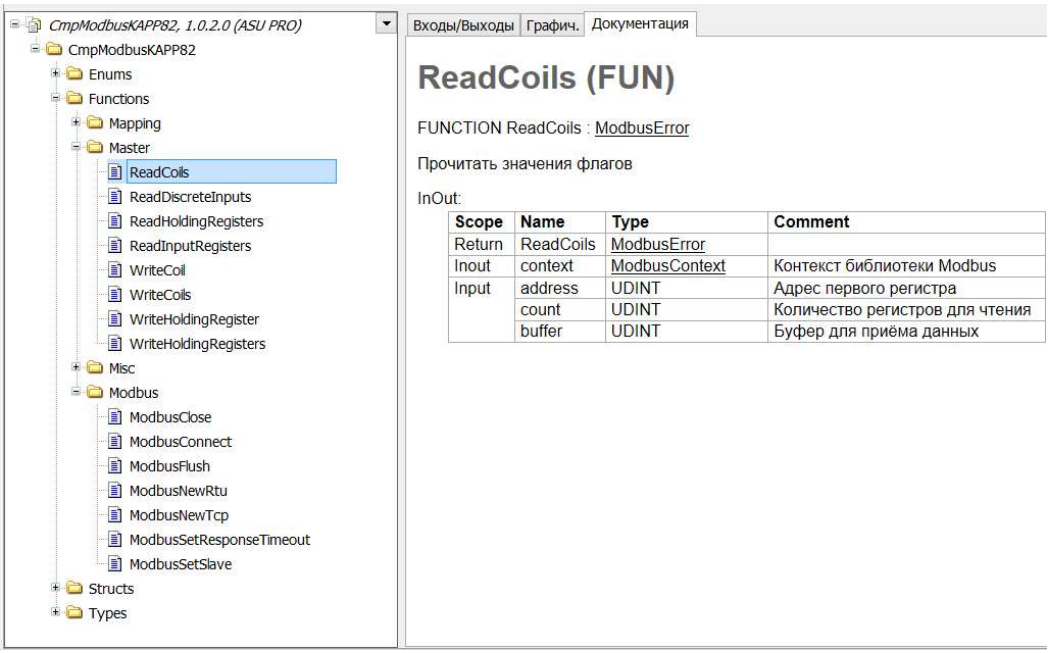


Рисунок 89 – Документация функций для чтения/записи и настройки портов

3.2.4 Работа в качестве ведущего устройства по интерфейсу Ethernet (Modbus TCP Master)

Для работы в качестве ведущего по интерфейсу Ethernet необходимо так же, как и в случае интерфейса RS-485 написать похожий код. Алгоритм тоже должен исполняться в отдельной задаче. Если у вас уже есть отдельная задача ModbusMaster, создадим новую программу TCP и добавим ее в эту задачу. Создание отдельной задачи, новой программы и добавление программы в задачу описано в пункте 2.4.5.

Задача: написать код для опроса первых 100 регистров хранения подчиненного устройства с IP – адресом 192.168.20.130. Параметры соединения: Номер порта TCP – 502; Таймаут опроса – 1 с.

Пример кода для выполнения вышеуказанной задачи:

```
VAR_GLOBAL
    Buffer2 : ARRAY [0..99] OF WORD;
END_VAR
```

```
PROGRAM RTU
VAR
    created, started: BOOL := FALSE;
    ctx: CmpModbusKAPP82.ModbusContext;
    result: CmpModbusKAPP82.ModbusError;
    errors, success: UDINT;
END_VAR
```

```
IF NOT created THEN
 created := TRUE;
    // Создаем экземпляр драйвера Modbus TCP
    CmpModbusKAPP82.ModbusNewTcp('192.168.20.130',        502,
```

```
        context => ctx);
        // Устанавливаем таймаут в 1 сукунду
        CmpModbusKAPP82.ModbusSetResponseTimeout(ctx, 1, 0);
    END_IF

    IF NOT started THEN
        // Подключаемся к серверу
        result := CmpModbusKAPP82.ModbusConnect(ctx);
        IF result <> CmpModbusKAPP82.ModbusError.ERROR_OK THEN
            started := FALSE;
        ELSE
            started := TRUE;
        END_IF
    END_IF

    IF started THEN
        // Читаем первые 100 регистров
        result  :=  CmpModbusKAPP82.ReadHoldingRegisters(ctx,  0,
        100,    ADR(Global.buffer2));
        IF result <> CmpModbusKAPP82.ModbusError.ERROR_OK THEN
            // В случае ошибки, отключаемся и подключаемся заново
            errors := errors + 1;
            started := FALSE;
            CmpModbusKAPP82.ModbusClose(ctx);
        ELSE
            success := success + 1;
        END_IF
    END_IF
```

В рассмотренном примере, при запуске контроллера переменная `created = FALSE` по этому, в первом цикле выполнится настройка Ethernet порта и определится его контекст. После чего переменная `created` станет `TRUE`, и данная процедура выполнятся больше не будет до перезапуска контроллера. После чего устанавливается соединение с подчиненным устройством. Когда соединение установлено, происходит чтение первых 100 регистров с записью их в `buffer2`. В случае возникновении ошибки при чтении, соединение закрывается. Далее опять устанавливается и чтение регистров вновь возобновляется.

Важно!!! Так же как и при работе по RS-485, RS-232 контекст должен быть определен 1 раз для каждого подключения, а размер буфера должен быть не меньше количества запрашиваемых регистров.

Полный список функций для настройки порта можно посмотреть в Приложении Г.

Полный список функций для чтения и записи можно посмотреть в Приложении Д.

Так же все функции, описание и документацию можно найти в Менеджере библиотек, настройка портов в папке Modbus, функции чтения и записи в папке Master.

3.2.5 Преобразование чисел с плавающей точкой для передачи по Modbus

Стандарт Modbus предусматривает только два типа данных, участвующих в обмене – BOOL и WORD (с CODESYS V3.5 полный аналог UINT). Достаточно часто возникает потребность передать данные типа REAL (чисел с плавающей точкой). В этом случае на устройстве, которое отравляет данные, необходимо преобразовать их в последовательность WORD регистров. Соответственно, на устройстве, получающем данные, должно быть выполнено обратное преобразование.

Передача REAL по протоколу Modbus не стандартизирована – значение с плавающей точкой передаются в виде двух регистров (переменных типа WORD), но порядок этих WORD переменных (или даже их байт) может отличаться. В данном случае следует привести их к нужному для конкретного устройства виду.

Для решения подобных задач в CODESYS есть два базовых способа для подобных преобразований: объединения и указатели. Но удобнее всего воспользоваться функциями библиотеки CmpModbusKAPP82.

Для преобразования числа с плавающей точкой в два регистра без перестановки в, библиотеке CmpModbusKAPP82, используется функция SetFloatABCD. Пример:

```
PROGRAM PLC_PRG
VAR
    VarReal: REAL;
    VarArray: ARRAY [0..1] OF UINT;
END_VAR
```

```
    CmpModbusKAPP82.SetFloatABCD (VarReal,VarArray);
    CmpModbusKAPP82.SetHoldingRegisters(Global.mapping, 0, 2,
    VarArray);
```

Рассмотрим пример, когда необходимо обратное преобразование – два регистра без перестановки в число с плавающей точкой. Для этого можно использовать функцию GetFloatABCD. Пример:

```
VAR
    VarReal: REAL;
    VarArray: ARRAY [0..1] OF UINT;
END_VAR
```

```
PROGRAM PLC_PRG
VAR
    CmpModbusKAPP82.GetHoldingRegisters(Global.mapping, 0, 2,
    VarArray);
    VarReal2:= CmpModbusKAPP82.GetFloatABCD (VarArray);
END_VAR
```

Полный список функций преобразования чисел с плавающей точкой в регистры Modbus можно посмотреть в Приложении Е.

Так же все функции, описание и документацию можно найти в Менеджере библиотек, в папке Misc (рисунок 90).

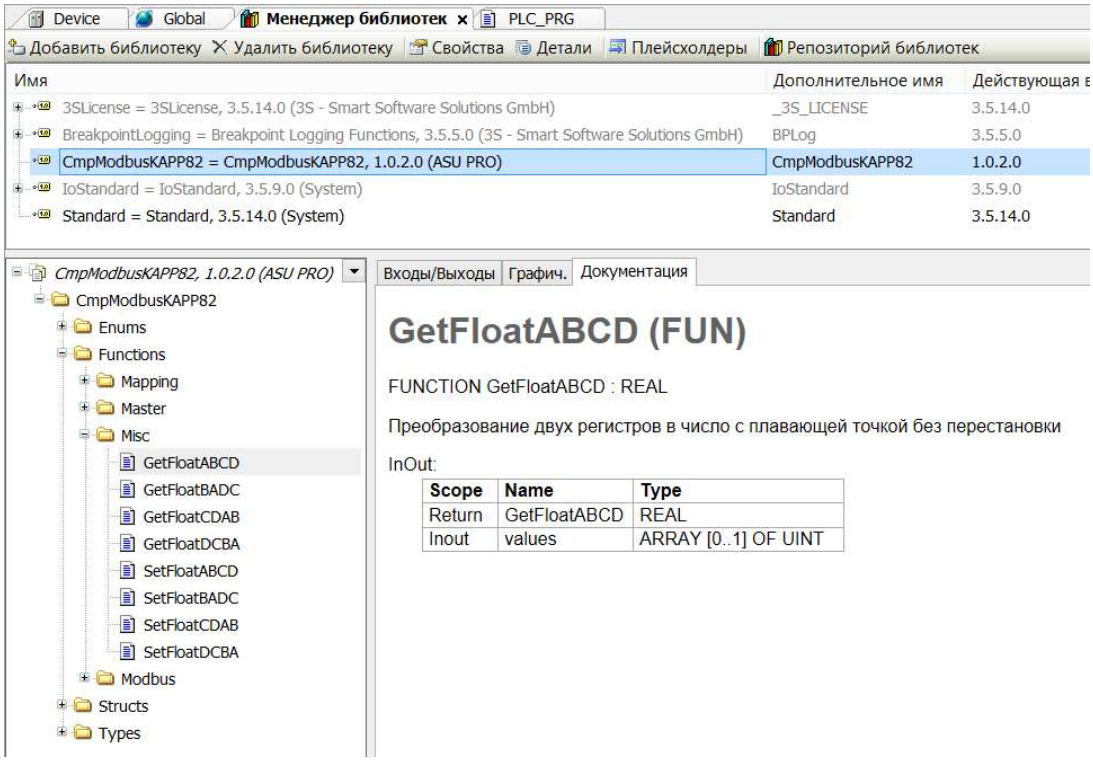


Рисунок 90 – Документация функций преобразования чисел с плавающей точкой

Для преобразования данных так же можно использовать объединения (UNION). Объединение представляет собой пользовательский тип данных, все переменные которого расположены в одной области памяти. Таким образом, переменные различных типов будут представлять различную интерпретацию одних и тех же данных. Для конвертации достаточно записать значение в одну из переменных объединения и считать его из другой.

Для конвертации значения с плавающей точкой из двух переменных типа WORD в переменную типа REAL способом объединения необходимо нажать правую кнопку мыши по значку «Приложение» («Application») и добавить объект DUT (рисунок 91).

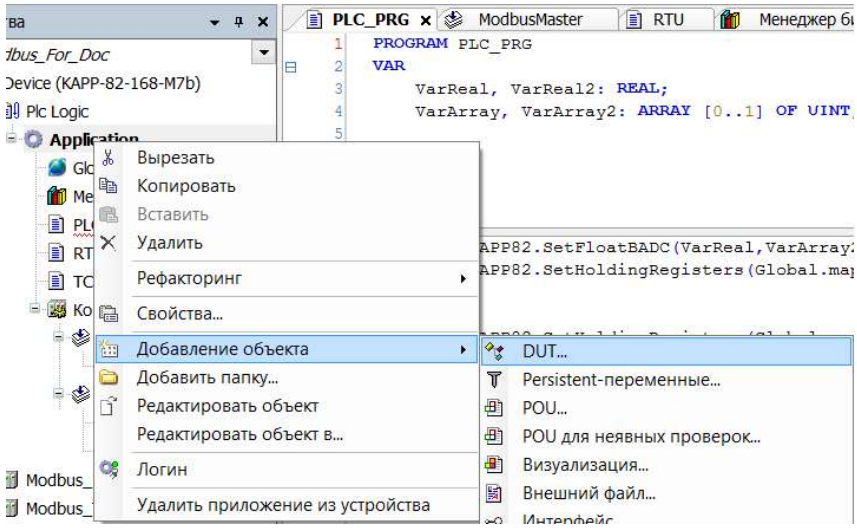


Рисунок 91 – Добавление объекта DUT

Далее задаем имя (например Real_Word), указываем тип объекта – «Объединение» и нажимаем кнопку «Добавить» (рисунок 92).

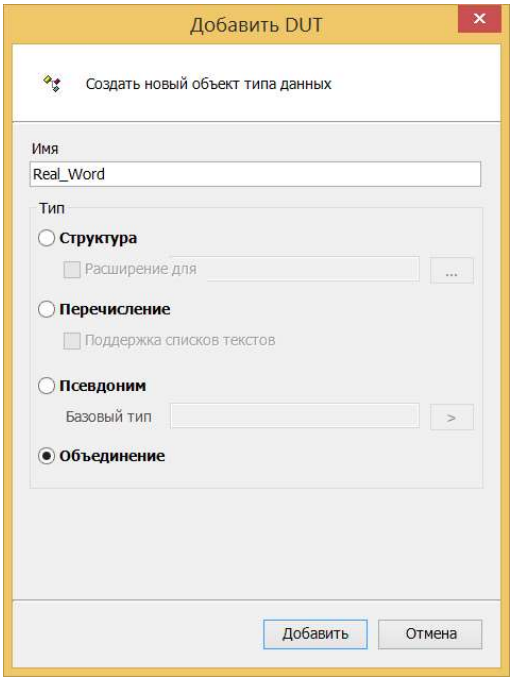


Рисунок 92 – Создание объекта типа данных Объединение

В описании объекта объявляем переменные и типы данных, которые требуется объединить. Пример:

```
TYPE Real_Word :
UNION
    RealValue: REAL;                    // число с плавающей
    точкой
    ModbusReal: ARRAY[0..1] OF UINT;    //массив слов для
    Modbus
END_UNION
END_TYPE
```

Далее присваиваем элементам массива `ModbusReal` соответствующим регистры Modbus и получаем число с плавающей точкой в переменной `RealValue`. Пример:

```
VAR
    VarReal: REAL;
    Real_2Word: Real_Word;
END_VAR
```

```
PROGRAM PLC_PRG
VAR
    CmpModbusKAPP82.GetHoldingRegisters(Global.mapping, 0, 2,
    Real_2Word.ModbusReal);
    VarReal:= Real_2Word.RealValue;
END_VAR
```

Для конвертации значения с плавающей точкой в два регистра Modbus способом объединения необходимо записать его в переменную `RealValue`, а затем массив `ModbusReal` записать в регистры Modbus. Пример:

```
VAR
    VarReal: REAL;
    Real_2Word: Real_Word;
END_VAR
```

```
PROGRAM PLC_PRG
VAR
    Real_2Word.RealValue:=VarReal ;
    CmpModbusKAPP82.SetHoldingRegisters(Global.mapping, 0, 2,
    Real_2Word.ModbusReal);
END_VAR
```

Важно!!! Такой подход соответствует функциям библиотеки CmpModbusKAPP82 с порядком байт BADC. Первым старший байт соответствует дефолтному порядку байт Modbus. В данном случае младшее слово первым. Такой подход может более удобным, чем использования функций CmpModbusKAPP82, если нужен именно такой порядок байт и слов. Порядок хранение в памяти слов, байтов и битов обусловлен особенностью архитектуры контроллеров.

Совет!!! В случае необходимости изменения порядка байтов можно создать два объединения – в первом будет происходить конвертация полученных по Modbus значений WORD в массив байтов, а во втором – конвертация нового массива байтов (переставленных в нужном порядке) в переменную типа REAL. Пример конвертации 2 WORD в REAL с перестановкой байт (аналог функции GetFloatABCD):

```
TYPE Word_Byte :
UNION
    ModbusReal: ARRAY[0..1] OF UINT;      //массив слов для
    Modbus
    Bytes: ARRAY [0..3] OF BYTE;        // массив байтов
END_UNION
END_TYPE
```

```
TYPE Byte_Real :
UNION
    Bytes: ARRAY [0..3] OF BYTE;     // массив байтов
    RealValue: REAL;                    // число с плавающей
    точкой
END_UNION
END_TYPE
```

```
VAR
    Word_to_Bytes: Word_Byte;
    Bytes_to_Real: Byte_Real;
    VarReal: REAL;
END_VAR
```

```
PROGRAM PLC_PRG
VAR
    CmpModbusKAPP82.GetHoldingRegisters(Global.mapping, 0, 2,
    Word_to_Bytes.ModbusReal);
    Bytes_to_Real.Bytes[0]:= Word_to_Bytes.Bytes[1];
    Bytes_to_Real.Bytes[1]:= Word_to_Bytes.Bytes[0];
    Bytes_to_Real.Bytes[2]:= Word_to_Bytes.Bytes[3];
    Bytes_to_Real.Bytes[3]:= Word_to_Bytes.Bytes[2];
    VarReal:= Bytes_to_Real.RealValue;
END_VAR
```

Такой подход может использоваться при необходимости поменять порядок бит в байтах. Тогда нужно будет вместо массивов байтов использовать массивы битов.

Еще один подход к данной задаче использование указателей.

Указатели содержат адреса переменных. Обращаясь к переменной по указателю, пользователь работает непосредственно с областью памяти, в которой хранится эта переменная, что позволяет производить любую обработку находящихся в ней данных.

Важно!!! Использование указателей подразумевает соответствующую квалификацию программиста. Некорректное использование указателей может привести к «зависанию» программы и контроллера.

Пример получения числа с плавающей точкой из двух регистров мадбас:

```
VAR
    VarReal: REAL :=2.54;
    VarArray: ARRAY [0..1] OF UINT;
    PrReal: POINTER TO REAL;
END_VAR
```

```
PROGRAM PLC_PRG
VAR
    CmpModbusKAPP82.GetHoldingRegisters(Global.mapping, 0, 2,
    VarArray);
    PrReal:= ADR(VarArray);
    VarReal:=PrReal^;
END_VAR
```

Такой подход тоже соответствует функциям библиотеки CmpModbusKAPP82 с порядком байт BADC.

Совет!!! Для перестановки байтов или битов такой подход будет хоть и осуществимым, но очень трудоемким. Рекомендуется использовать для этого первые два способа (функции библиотеки CmpModbusKAPP82 или Объединения).

3.2.6 Пример работы с модулями ввода-вывода КАПП2

Для работы с модулями ввода-вывода необходимо настроить контроллер на работу в качестве ведущего устройства по интерфейсу RS-485 (COM4) шины TBUS. Так же необходимо внимательно изучить руководства на модули ввода-вывода проектируемой системы.

Для проектируемой системы необходимо учитывать следующие моменты:

1. Количество регистров на чтение/запись влияет на минимальное возможное время обмена между контроллером и модулем. Для минимального времени, например для модуля AI КАПП2-80-000-0, необходимо считать только измеренное значение восьми каналов (16 регистров).
2. Для правильной работы за цикл задачи Modbus Master необходимо выполнять только одно действие чтения или записи в один модуль. Так же в данной задаче не должны вызываться другие программы. Это приведет к увеличению времени выполнения задачи и как следствие время обмена увеличится.
3. Интервал выполнения задачи Modbus Master влияет на время обмена между контроллером и модулем, но не должен быть менее 10 мс. В противном случае может ухудшиться качество связи, что приведет к ожиданию ответа от ведомого до таймаута и как следствие время обмена увеличится. Так же стоить отметить, что некоторые слейв устройства не могут выполнять запросы чаще определенного интервала времени.
4. В модулях ввода-вывода КАПП2 есть параметр «Задержка ответа по Modbus» принимающий значения от 0 до 5 мс.
5. Параметр время ожидания ответа от слейв устройства функции ModbusSetResponseTimeout библиотеки CmpModbusKAPP82 в случае ошибок может сильно увеличить время обмена. Его необходимо подбирать опытным путем, выбирая наименьшее значение, при котором количество успешно обработанных запросов составляет 100%. Желательно так же модули связь с которыми потеряна (3 раза подряд результат выполнения запроса - ошибка) исключать из опроса и выставлять флаг ошибки модуля.

Рассмотрим задачу разработки алгоритма опроса модулей контроллера КАПП2 и обработки аналоговых сигналов.

Первым делом создадим структуры данных модулей:

```
    TYPE ModStruc :
    STRUCT
     addr: INT;                 // адрес модуля по модбас
     EnModule: BOOL :=0;        //флаг отключения из опроса
     AlmModule: BOOL;           // Флаг ошибки обмена с модулем
     CountIO: UINT;             // количество точек В/В
     IO: ARRAY [1..16] OF REAL;    // токовое значение каналов
модуля
     TypeIO: UINT;              // тип модуля: например AI-1; AO-2;
DI-3; DO-4
     OffsetIO: UINT;               // смещение (начальный адрес
первого канала)
     errors, success: UINT;     // колчиство ответов и ошибок
     ErrCount: UINT;               // Количество ошибок подряд
    END_STRUCT
    END_TYPE
```

Размер массива IO определяется максимально возможным количеством каналов на модулях ввода вывода.

Далее создаем структуру для настройки, обработки и хранения считанных аналоговых значений с модулей аналогового ввода:

```
TYPE Analog :
STRUCT
 Value: REAL;                    // обработаное значение
 iValue: REAL;                   // входное значение с модуля
 MinEU: REAL    :=0;             // миниммум инж. единицы
 MaxEU: REAL    :=100;           // максимум инж. единицы
 SimEn: BOOL;                    // флаг симуляции
 SimVal:REAL;                    // значение симуляции
 UnRange: BOOL;                  // флаг обрыв
 OvRange: BOOL;                  // флаг КЗ
END_STRUCT
END_TYPE
```

По необходимости в структуру можно добавлять и обрабатывать другие переменные такие как: верхний/нижний аварийный/предупредительный; верхняя/нижняя обрезка минимума для отображения (Hi/Low CutOff); единицы измерения и т.д.

Создадим структуру для настройки, обработки и хранения считанных дискретных значений с модулей дискретного ввода:

```
TYPE Discrete :
STRUCT
 Value: BOOL;              // обработаное значение
 iValue: BOOL;             // входное значение с модуля
 SimEn: BOOL;              // включение симуляции
 SimVal: BOOL;             // значение симуляции
 Inverse: BOOL;            // флаг инверсии входного значения
END_STRUCT
END_TYPE
```

Далее создадим список глобальных переменных:

```
VAR_GLOBAL CONSTANT
 ModuleCount: INT := 10;           // количество модулей В/В
END_VAR

VAR_GLOBAL
 Module: ARRAY [1..ModuleCount] OF ModStruc; //массив с
данными по каждому модулю
 AI: ARRAY [1..10] OF Analog;                //массив с
входными аналоговыми значениями
 DI: ARRAY [1..10] OF Discrete;                  //массив с
входными дискретными значениями
END_VAR
```

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

Изм.	Кол.уч.	Лист	№ док.	Подпись	Дата

73619730.26.20.30.000.020 РЭ

Лист 76

ФорматА4

Создадим функциональный блок обработки аналоговых сигналов и переменных, заданных в структуре Analog:

```
FUNCTION_BLOCK FB_AI
VAR_INPUT
 iVal: REAL;
END_VAR
VAR_IN_OUT
 AI:Analog;
END_VAR
```

```
AI.iValue:= iVal;
IF AI.SimEn THEN
 AI.Value:= AI.SimVal;
ELSE
 AI.Value:= (iVal-4)/(20-4)*(AI.MaxEU-AI.MinEU)+AI.MinEU;
 IF iVal<4 THEN
      AI.UnRange:= TRUE;
 ELSE
      AI.UnRange:= FALSE;
 END_IF
 IF iVal>20 THEN
      AI.OvRange:= TRUE;
 ELSE
      AI.OvRange:= FALSE;
 END_IF
END_IF
```

Функциональный блок выполняет масштабирование аналогового сигнала в инженерные единицы и обработку флагов обрыва и короткого замыкания, если не установлен флаг симуляции значения. В противном случае обработанное значение принимает значение симуляции.

Создадим функциональный блок обработки дискретных сигналов и переменных, заданных в структуре Discrete:

```
FUNCTION_BLOCK FB_DI
VAR_INPUT
 iVal: REAL;
END_VAR
VAR_IN_OUT
 DI:Discrete;
END_VAR
```

```
DI.iValue:= TO_BOOL(iVal);
IF DI.SimEn THEN
 DI.Value:= DI.SimVal;
ELSE
 IF DI.Inverse THEN
      DI.Value:=NOT(TO_BOOL(iVal));
 ELSE
      DI.Value:=TO_BOOL(iVal);
```

```
    END_IF
  END_IF
```

Обратите внимание, что в данной программе используется преобразование типов. С модуля будет считываться 16 битный регистр по каждому каналу, содержащий значения 0 (Ложь) и 1 (Истина). Такой подход позволяет использовать одну универсальную структуру практически для всех модулей ввода-вывода.

Для дискретных модулей ввода так же можно считывать 1 регистр – слово состояния всех входов или состояния всех дискретных входов, при этом в структуру ModStruc необходимо добавить переменные соответствующие типу считываемых данных либо делать подобные структуры на каждый тип модуля.

Далее необходимо создать программу для обмена с модулями:

```
  PROGRAM ModbusMaster
  VAR
    // Настройки шины TBUS
    created: BOOL := FALSE;       //флаг создания порта
    TimeOut: UDINT := 100000;          // время таймуата в
микросекундах
    baudrate: UDINT := 115200;         //скорость шины TBUS
    PortN: BYTE := 1;
    //-------------------------------
    buffer : ARRAY [0..124] OF WORD;        // буфер для данных
шины TBUS
    ctx: CmpModbusKAPP82.ModbusContext;     //контекст шины TBUS
    result: CmpModbusKAPP82.ModbusError;    // результат
выполнения функций CmpModbusKAPP82
    TempArray: ARRAY [0..1] OF UINT;        // временный массив
для преобразования чисел с плавающей точкой
    i, j: INT :=1;                          //вспомогательные индексы
    Reset: BOOL;                            // флаг сброса модуля
    NumModRes: UINT := 1;                        //номер модуля
который необходимо сбросить
  END_VAR
```

```
  IF NOT created THEN
    // Создаём экземпляр драйвера Modbus RTU  (выставляем ему
параметры соответствующие slave устройству)
    CmpModbusKAPP82.ModbusNewRtu(PortN, baudrate,
    CmpModbusKAPP82.ModbusParity.NONE,
    CmpModbusKAPP82.ModbusDataBits.BITS_8,
    CmpModbusKAPP82.ModbusStopBits.ONE,
    context => ctx
    );
    // Устанавливаем таймаут в 1 секунду
    CmpModbusKAPP82.ModbusSetResponseTimeout(ctx, 0, TimeOut);
    // устанвливаем флаг создания порта
    created:= TRUE;
    //Описываем обмен с модулями В/В
  //----module 1--------
    Global.Module[1].EnModule:=TRUE;
```

```
     Global.Module[1].addr:= 5;
     Global.Module[1].TypeIO:= 1 ; // AI
     Global.Module[1].CountIO := 8;
     Global.Module[1].OffsetIO := 11;
     //Global.IO[1].success:=0;
     //Global.IO[1].errors:= 0;

    //----module 2--------
     Global.Module[2].EnModule:=TRUE;
     Global.Module[2].addr:= 4;
     Global.Module[2].TypeIO:= 2; // AO
     Global.Module[2].CountIO := 4;
     Global.Module[2].OffsetIO := 10;

    //----module 3--------
     Global.Module[3].EnModule:=TRUE;
     Global.Module[3].addr:= 3;
     Global.Module[3].TypeIO:= 3; // DI
     Global.Module[3].CountIO := 16;
     Global.Module[3].OffsetIO := 10;

    //----module 4--------
     Global.Module[4].EnModule:=TRUE;
     Global.Module[4].addr:= 6;
     Global.Module[4].TypeIO:= 4; // DO
     Global.Module[4].CountIO := 8;
     Global.Module[4].OffsetIO := 10;

    //----module n--------

    //----------------------
    END_IF

    IF Global.Module[j].EnModule THEN
          // Устанавливаем Slave ID
          result:=   CmpModbusKAPP82.ModbusSetSlave(ctx,
Global.Module[j].addr);
          CASE Global.Module[j].TypeIO  OF

               1:   // Модули AI
                    // Читаем нужное количество каналов с адреса
смещения
                    result :=
CmpModbusKAPP82.ReadHoldingRegisters(ctx,
Global.Module[j].OffsetIO, Global.Module[j].CountIO*2,
ADR(buffer));
                    IF result <>
CmpModbusKAPP82.ModbusError.ERROR_OK THEN
                         Global.Module[j].errors :=
Global.Module[j].errors + 1;
                         Global.Module[j].ErrCount:=
```

```
Global.Module[j].ErrCount+1;
                    ELSE
                         Global.Module[j].success :=
Global.Module[j].success + 1;
                         Global.Module[j].ErrCount:= 0;
                         FOR i:=0 TO
TO_INT(Global.Module[j].CountIO)-1 DO
                              TempArray[0]:= buffer[i*2];
                              TempArray[1]:= buffer[i*2+1];
                              Global.Module[j].IO[i+1]:=
CmpModbusKAPP82.GetFloatBADC(TempArray);
                         END_FOR
                    END_IF

               2:   // Модули AO
                    FOR i:=1 TO TO_INT(Global.Module[j].CountIO)
DO

                         result:=  CmpModbusKAPP82.SetFloatBADC
(Global.Module[j].IO[i], TempArray);
                         buffer[(i-1)*2]:= TempArray[0];
                         buffer[(i-1)*2+1]:= TempArray[1];

                    END_FOR

                    result :=
CmpModbusKAPP82.WriteHoldingRegisters(ctx,
Global.Module[j].OffsetIO, Global.Module[j].CountIO*2,
ADR(buffer));
                    IF result <>
CmpModbusKAPP82.ModbusError.ERROR_OK THEN
                         Global.Module[j].errors
:=Global.Module[j].errors + 1;
                         Global.Module[j].ErrCount:=
Global.Module[j].ErrCount+1;
                    ELSE
                         Global.Module[j].success :=
Global.Module[j].success + 1;
                         Global.Module[j].ErrCount:= 0;
                    END_IF

               3:   // Модули DI
                    // Читаем нужное количество каналов с адреса
смещения
                    result :=
CmpModbusKAPP82.ReadHoldingRegisters(ctx,
Global.Module[j].OffsetIO, Global.Module[j].CountIO,
ADR(buffer));
                    IF result <>
CmpModbusKAPP82.ModbusError.ERROR_OK THEN
                         Global.Module[j].errors
```

```
:=Global.Module[j].errors + 1;
                        Global.Module[j].ErrCount:=
Global.Module[j].ErrCount+1;
                    ELSE
                        Global.Module[j].success :=
Global.Module[j].success + 1;
                        Global.Module[j].ErrCount:= 0;
                        FOR i:=0 TO
TO_INT(Global.Module[j].CountIO)-1 DO
                            Global.Module[j].IO[i+1] :=
WORD_TO_REAL(buffer[i]);
                        END_FOR
                    END_IF

            4:   // Модули DO
                    FOR i:=1 TO TO_INT(Global.Module[j].CountIO)
DO
                        buffer[i-1]:=
TO_WORD(Global.Module[j].IO[i]);
                    END_FOR

                    result :=
CmpModbusKAPP82.WriteHoldingRegisters(ctx,
Global.Module[j].OffsetIO, Global.Module[j].CountIO,
ADR(buffer));
                    IF result <>
CmpModbusKAPP82.ModbusError.ERROR_OK THEN
                        Global.Module[j].errors
:=Global.Module[j].errors + 1;
                        Global.Module[j].ErrCount:=
Global.Module[j].ErrCount+1;
                    ELSE
                        Global.Module[j].success :=
Global.Module[j].success + 1;
                        Global.Module[j].ErrCount:= 0;
                    END_IF
        END_CASE
    END_IF

    // индексируем номер модуля
    j:=j+1;
    IF j> Global.ModuleCount THEN
     j:=1;
    END_IF
    // если модуль не ответил больше 2 раз, исключаем его из
опроса
    // и устанавливаем флаг ошибки модуля
    FOR i:=1 TO Global.ModuleCount DO
     IF Global.Module[i].ErrCount>2 THEN
           Global.Module[i].EnModule := FALSE;
           Global.Module[i].AlmModule := TRUE;
```

```
    END_IF
END_FOR
// при устранении отказа модуля сбрасываем флаг ошибки
// и включаем его в цикл опроса
IF Reset THEN
 Global.Module[NumModRes].EnModule:= TRUE;
 Global.Module[NumModRes].AlmModule:= FALSE;
 Global.Module[NumModRes].ErrCount:= 0;
 Reset := FALSE;
END_IF
```

Важно!!! За каждый интервал выполнения данной программы опрашивается только один модуль. Время ожидания ответа от модуля установлено 100 мс. При отсутствие связи с модулем цикл выполнения увеличится на 100 мс. за каждый такой модуль, по этому если модуль корректно не отвечает на запросы несколько раз подряд то он исключается из опроса. Это позволяет уменьшить среднее время опроса всех модулей. Время ожидания в 1 секунду является избыточным и ведет неоправданному увеличению времени опроса при ошибках модулей ввода-вывода.

Далее программу `ModbusMaster` помещаем в отдельную задачу (рисунок 93)

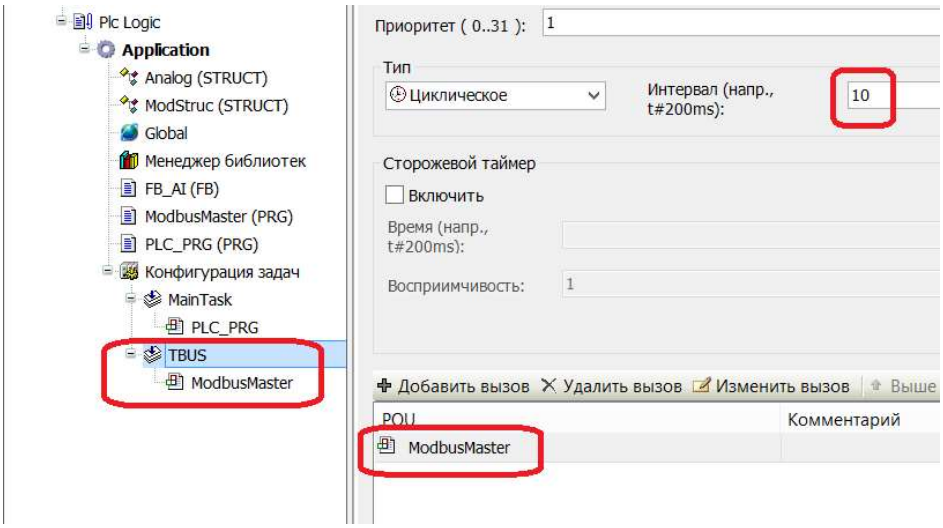


Рисунок 93 – Размещение программы ModbusMaster в задаче TBUS

Важно!!! Интервал выполнения данной задачи равен 10 мс. За это время задача при успешном выполнении функций чтения и записи будет успевать выполняться и среднее время опроса всех модулей ввода-вывода будет равняться произведению интервала задачи (10мс) на количество модулей (задается переменной `ModuleCount` в списке глобальных констант). Ни каких других программ в данной задаче быть не должно, иначе это приведет к увеличению времени цикла опроса всех модулей.

В другой задаче размещаем программу с обработкой аналоговых сигналов, например в задаче MainTask (рисунок 94):

```
PROGRAM ProgAI
VAR
 AI: FB_AI;
```

```
END_VAR
```

```
//обработка аналогового сигнала №1 модуль №1 канал №1
AI(iVal:= Global.Module[1].IO[1], AI:= Global.AI[1]);
// обработка аналогового сигнала №2 модуль №1 канал №2
AI(iVal:= Global.Module[1].IO[2], AI:= Global.AI[2]);
// обработка аналогового сигнала №3 модуль №2 канал №1
AI(iVal:= Global.Module[2].IO[1], AI:= Global.AI[3]);
```

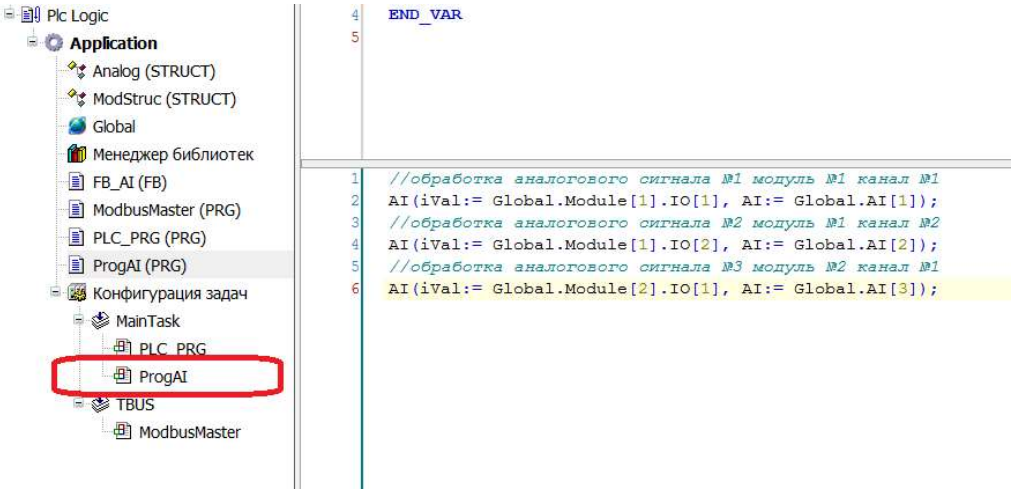


Рисунок 94 – Расположение программы ProgAI в задаче MainTask

Программу с обработкой дискретных сигналов можно так же разместить в той же задаче что и программу обработки аналоговых сигналов (рисунок 95):

```
PROGRAM ProgDI
VAR
 DI: FB_DI;
 r1: real;
END_VAR
```

```
//обработка дискретного сигнала №1 модуль №3 канал №1
DI(iVal:= Global.Module[3].IO[1], DI:= Global.DI[1]);
//обработка дискретного сигнала №2 модуль №3 канал №2
DI(iVal:= Global.Module[3].IO[2], DI:= Global.DI[2]);
```

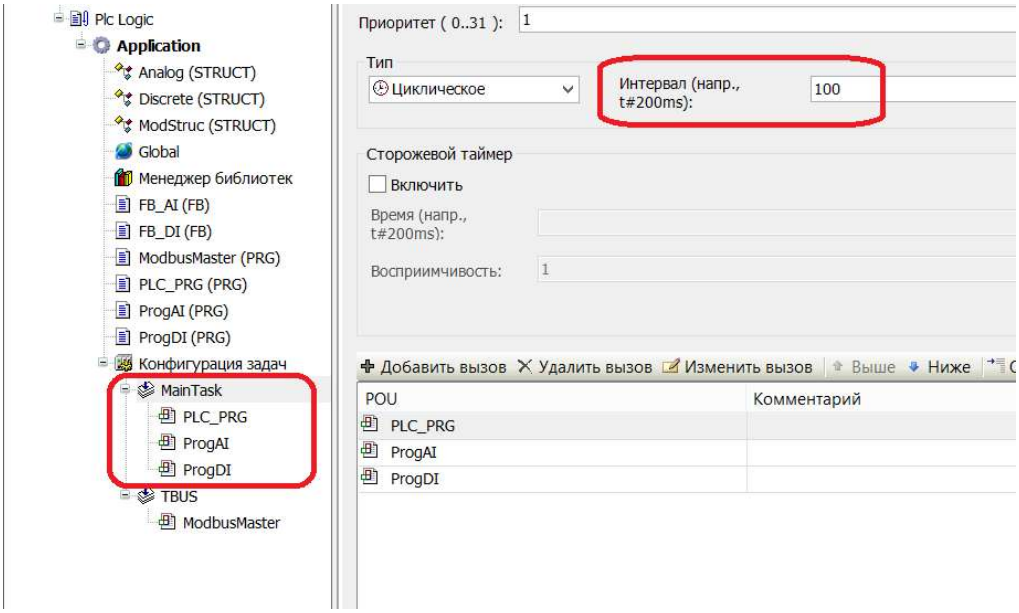


Рисунок 95 – Состав и настройки задачи MainTask

Важно!!! В каждой задаче вызываемые программы выполняются разово по очереди в каждый интервал времени указанный в настройках. Важно следить за тем что бы время выполнения задачи не превышало этот интервал, иначе интервалы выполнения задачи не будут соответствовать заложенному времени выполнения.

Проверить время выполнения задачи можно во вкладке проекта «Конфигурация задач» в режиме онлайн (рисунок 96).

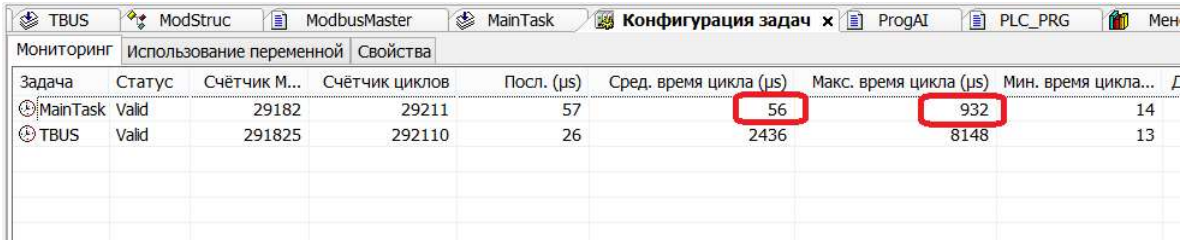


Рисунок 96 – Мониторинг выполнения задач

Из рисунка видно, что среднее время выполнения задачи составляет 56 микросекунд, а максимальное время выполнения не превышает 1 миллисекунду.

Совет!!! Если максимальное время превышает интервал, заложенный в настройках задачи, стоит перераспределить программы в этой задаче по разным задачам, или увеличить интервал выполнения, так как меньший интервал в таком случае не имеет смысла.

3.3 Работа с драйверами МЭК 60870-5

Контроллер поддерживает драйвер контролируемой станции, работающей по протоколу, определяемому стандартом ГОСТ Р МЭК 60870-5-101/104. Формуляр согласования реализации протокола расположен в приложении Ж.

Для понимания работы протокола пользователь должен быть знаком со стандартом ГОСТ Р МЭК 60870-5-101/104.

Настройка параметров для обмена по протоколам осуществляется путем добавления устройств «**IoDrvIEC60870_101_Slave**» и «**Iec 60870-104 Slave**» в **Дерево устройств** проекта, для протокола МЭК 60870-5-101 и МЭК 60870-5-104 соответственно.

Прикладные процессы пользователя (обработка запроса Общий опрос, циклическая передача данных, спорадическая передача предача данных и т.д.) осуществляются с помощью библиотеки **CmpIec60870104KAPP82**.

Библиотека CmpIec60870104KAPP82 содержит следующие типы данных:

- Iec60870MeasuredValueShort – значение измеряемой величины, короткий формат с плавающей запятой (M_ME_NC_1) , идентификатор типа **<13>** по ГОСТ Р МЭК 870-5-104;
- Iec60870MeasuredValueShortWithCP56Time2a – значение измеряемой величины, короткий формат с плавающей запятой с меткой времени CP56Время2a (M_ME_TF_1), идентификатор типа **<36>** по ГОСТ Р МЭК 870-5-104;
- Iec60870SinglePointInformation – одноэлементная информация (M_SP_NA_1), идентификатор типа **<1>** по ГОСТ Р МЭК 870-5-104;
- Iec60870SinglePointWithCP56Time2a – Одноэлементная информация с меткой времени CP56Время2a (M_SP_TB_1), идентификатор типа **<30>** по ГОСТ Р МЭК 870-5-104;

Библиотека содержит функции для реализации следующих процессов пользователя:

- обработка запроса «Общий опрос станции»
- циклическая передача данных
- спорадическая передача данных
- прием команд от пункта управления
- функции для работы с меткой времени

Для того чтобы контролируемая станция могла обрабатывать команды, необходимо реализовать функции-обработчики: для обработки принятого ASDU и обработки команды опроса, описаны далее, пп. 3.3.2 – 3.3.4.

Команда синхронизации времени обрабатывается без участия пользовательского кода.

Количество одновременных подключений контролирующих станций – 1.

Добавить в проект библиотеку CmpIec60870104KAPP82 можно с помощью менеджера библиотек (рисунок 97).

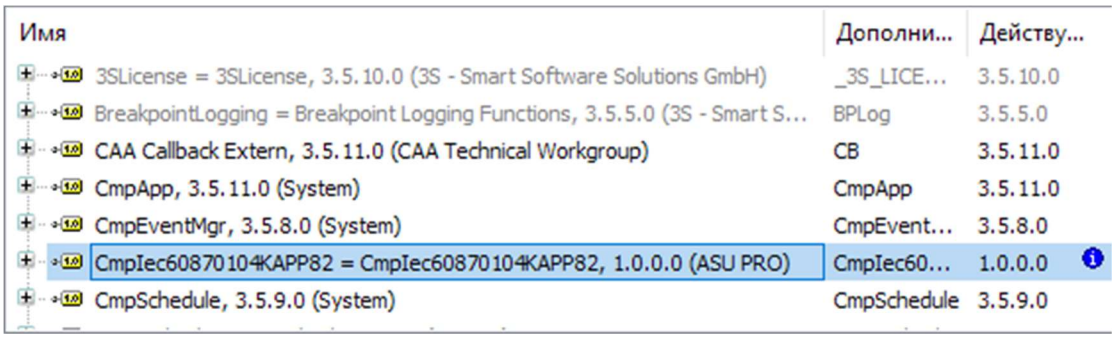


Рисунок 97 – Добавление библиотеки CmpIec60870104KAPP82

3.3.1 Настройка проекта МЭК 60870-5-104

Для включения работы драйвера необходимо добавить устройство «Iec 60870-104 Slave»в проект (рисунок 98).

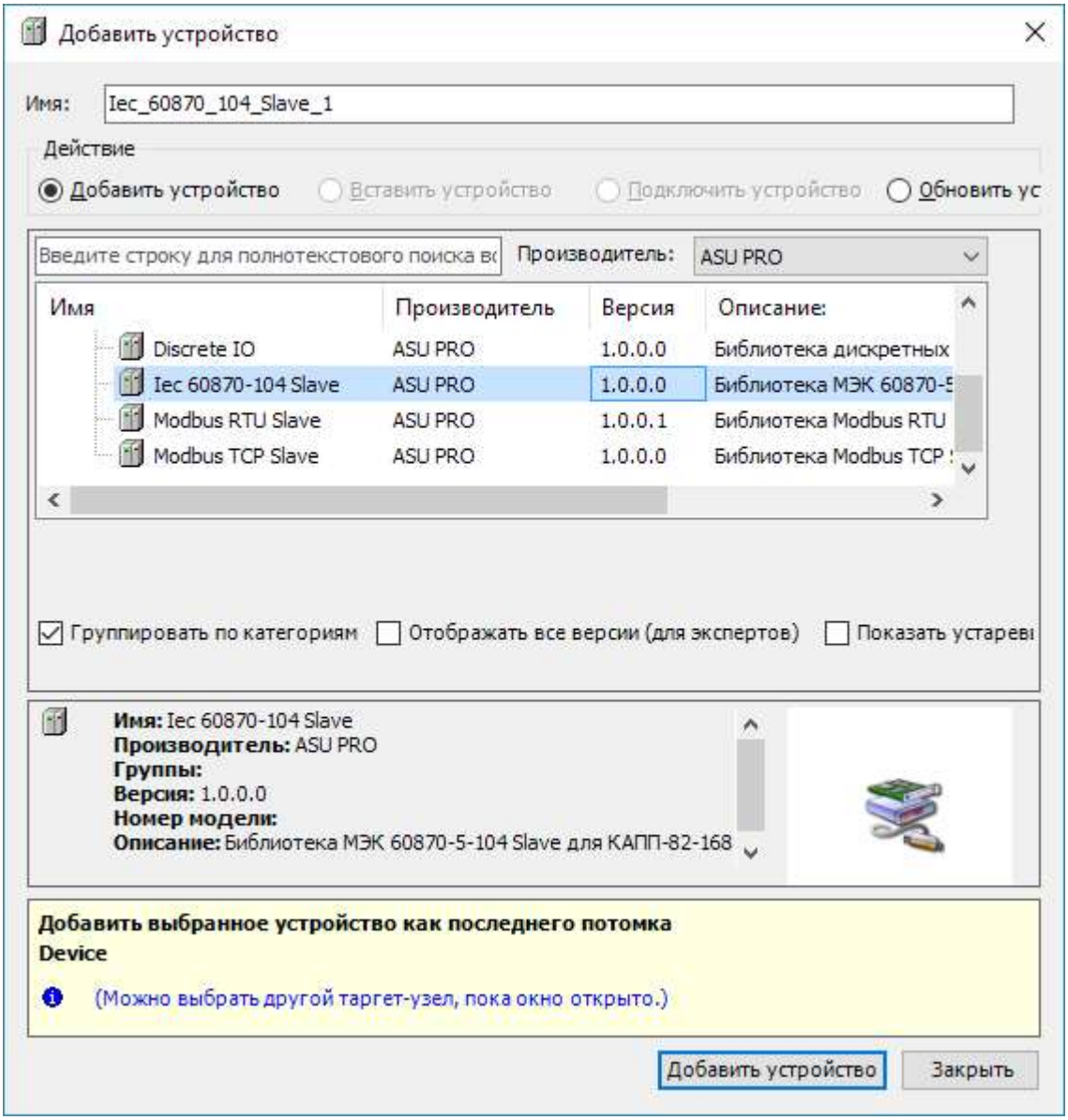


Рисунок 98 – Добавление устройства Iec 60870-104 в проект

После добавления устройства в проект, можно установить необходимые настройки протокола (рисунок 99).

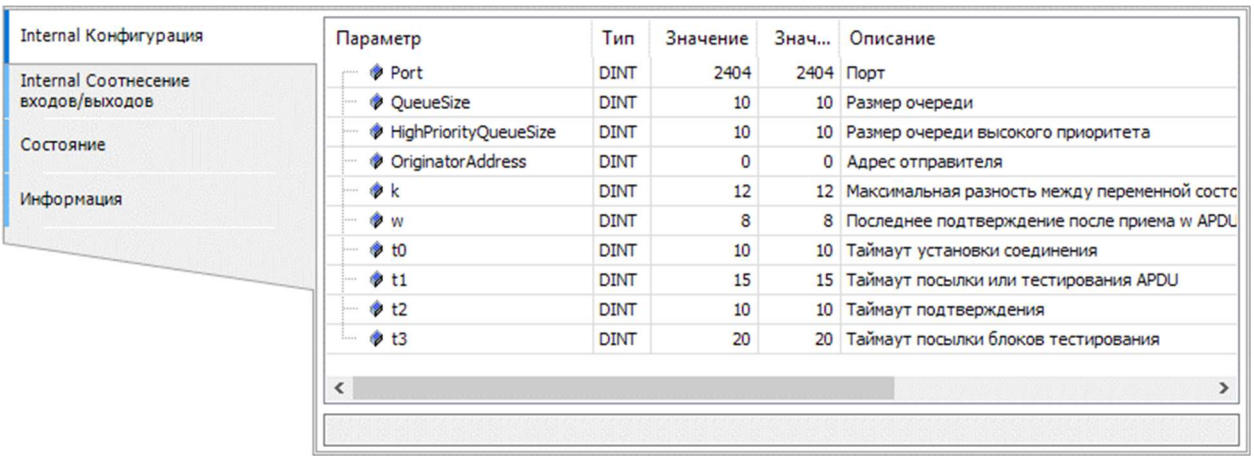


Рисунок 99 – Настройка протокола

На этом настройка проекта для работы с драйвером протокола МЭК 60870-5-104 закончена. После запуска приложения, драйвер будет принимать подключения к указанному в настройках порту (по умолчанию 2404).

3.3.2 Настройка проекта МЭК 60870-5-101

Для включения работы драйвера необходимо добавить устройство «IoDrvIEC60870_101_Slave»в проект (рисунок 100).

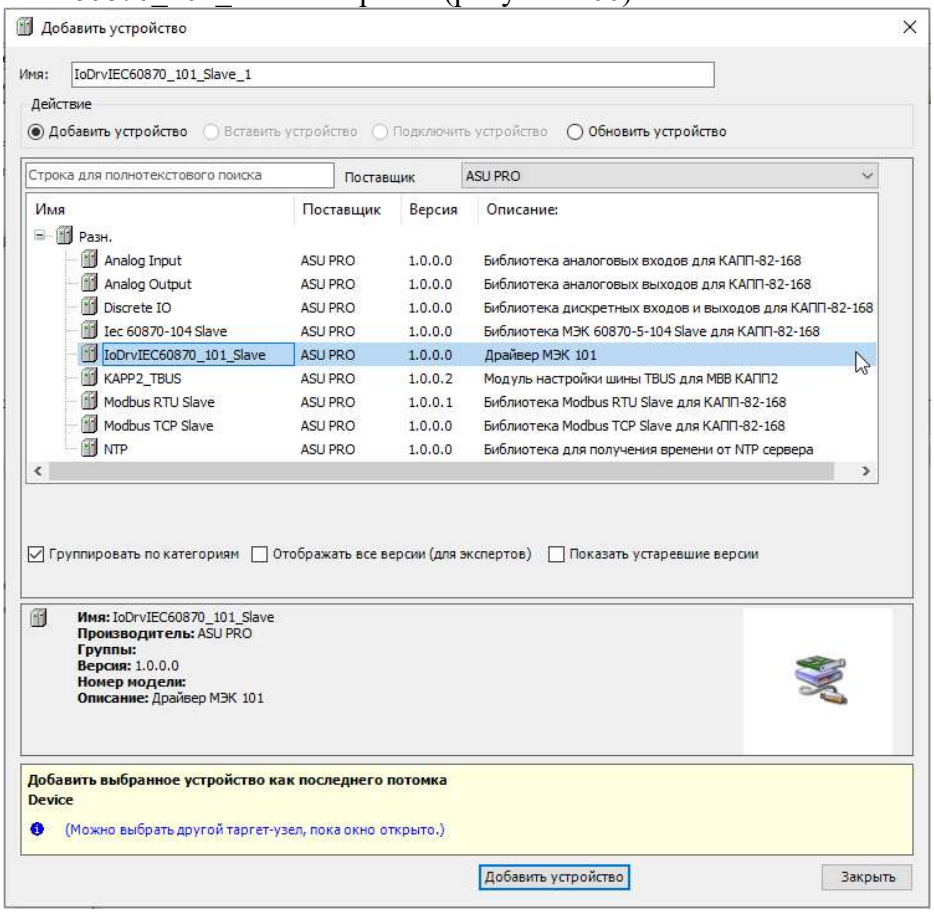


Рисунок 100. Добавление устройства IoDrvIEC60870_101_Slave в проект.

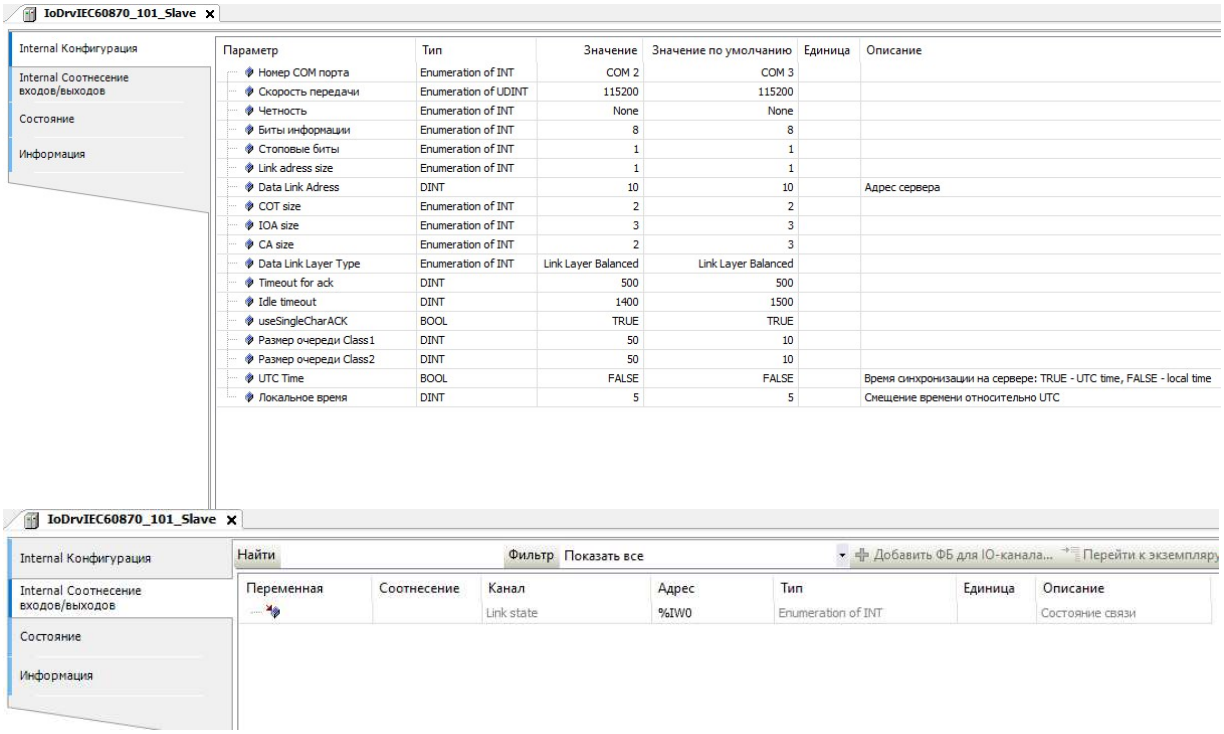


Рисунок 101. Настройки IoDrvIEC60870_101_Slave в проекте.

Таблица 3.3.2 Описание параметров протокола МЭК 60870-5-101.

Канал	**Тип**	**Описание**
Вкладка **Конфигурация**		
Номер COM-порта	ENUM of INT	Номер COM-порта контроллера, по которому будет осуществляется обмен
Скорость передачи	ENUM of UDINT	Скорость передачи данных, кбит/с
Число бит информации	ENUM of INT	Число бит данных (8 или 9)
Четность	ENUM of INT	Режим контроля четности (нет/четный/нечетный)
Число стоп-бит	ENUM of INT	Число стоп бит (1 или 2)
Link address size	ENUM of INT	Размер адреса станции
Data Link address	ENUM of INT	Адрес станции в сети
COT size	ENUM of INT	Размер поля причина передачи (1 или 2 байта)
IOA size	ENUM of INT	Размер адреса объекта информации (1, 2 или 3) байта
CA size	ENUM of INT	Размер общего адреса ASDU (1 или 2 байта)
Data Link Layer Type	ENUM of INT	Режим канального уровня: балансный, небалансный
TimeoutForACK	DINT	Время ожидания подтверждения
Idle timeout	DINT	Время для повторной посылки сообщения
UseSingleCharACK	BOOL	Использовать кадр канального уровня «ACK»
Размер очереди Class1	DINT	Глубина очереди сообщений класса 1
Размер очереди Class2	DINT	Глубина очереди сообщений класса 2
UTC Time	BOOL	Время на сервере: TRUE – UTC time, FALSE – local time
Локальное время	DINT	Смещение времени относительно UTC

Вкладка **Соотнесение входов/выходов**		
Состояние канала передачи данных	ENUM of INT	0 - Канальный уровень простаивает, связь отсутствует 1 - На канальном уровне произошла ошибка, возможно, канал недоступен 2 - Канальный уровень занят и недоступен для использования 3 - Канал доступен для передачи и приема пользовательских данных

3.3.2 Спорадическая передача информации

Контролируемая станция может отправлять данные по своей инициативе, например, при изменении значений переменных.

Для этого объявим объект информации типа M_ME_NC_1 (Значение измеряемой величины, короткий формат с плавающей запятой) и зададим ему адрес.

```
VAR
  io1: CmpIec60870104KAPP82.Iec60870MeasuredValueShort:=(ObjectAddress:=
123);
   //Объявляем блок данных (ASDU), который будет содержать объект
информации.
  asdu: CmpIec60870104KAPP82.Iec60870ASDU;
END_VAR
```

Реализация программы.

В коде программы создаём блок данных:

```
asdu := CmpIec60870104KAPP82.Iec60870AsduCreate(
     CmpIec60870104KAPP82.Iec60870Type.M_ME_NC_1,
     FALSE,
     CmpIec60870104KAPP82.Iec60870CauseOfTransmission.Periodic,
     0, 1, FALSE, FALSE);
```

Помещаем объект информации в блок данных:

```
CmpIec60870104KAPP82.Iec60870AsduAddInformationObjectBoolTS(
          asdu, ADR(io1), 1);
```

Помещаем блок данных в очередь на отправку:

```
CmpIec60870104KAPP82.Iec60870104SlaveEnqueueAsdu(asdu);
```

В случае протокола МЭК 60870-5-101 блок данных помещаем в очередь функцией:

```
CmpIec60870104KAPP82.CS101_Slave_enqueueUserDataClass1(asdu);
```

После помещения в очередь, блок данных уничтожается, и память, занимаемая им, освобождается. Поэтому, не следует создавать блоки данных, не отправляя их в очередь отправки, иначе может произойти переполнение памяти.

3.3.3 Обработка ASDU

Для обработки ASDU, принятых от контролирующей станции (команды, уставки), необходимо реализовать функцию обратного вызова. Для этого выберите пункт меню «Добавление объекта» – «POU…» (рисунок 102).

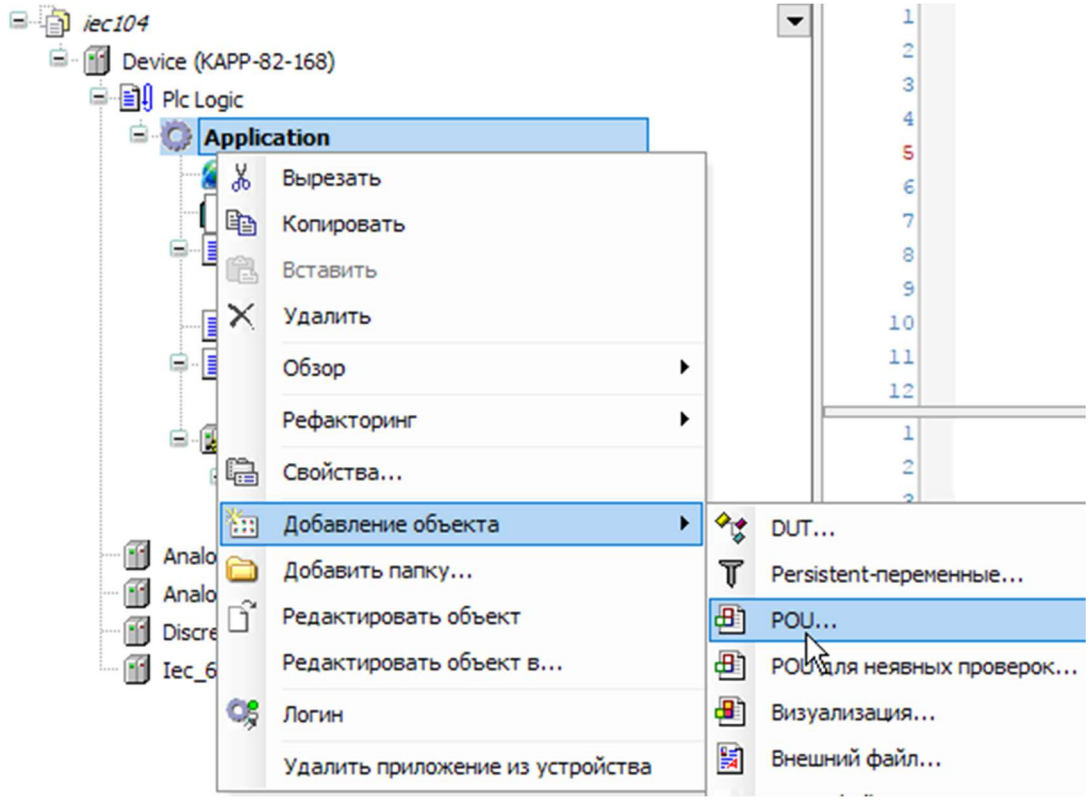


Рисунок 102 – Добавление функции обратного вызова

В окне «Добавить POU» (рисунок 103) введите имя создаваемого компонента, например, «AsduReceivedCallback» и выберите тип «Функциональный блок» и язык «ST».

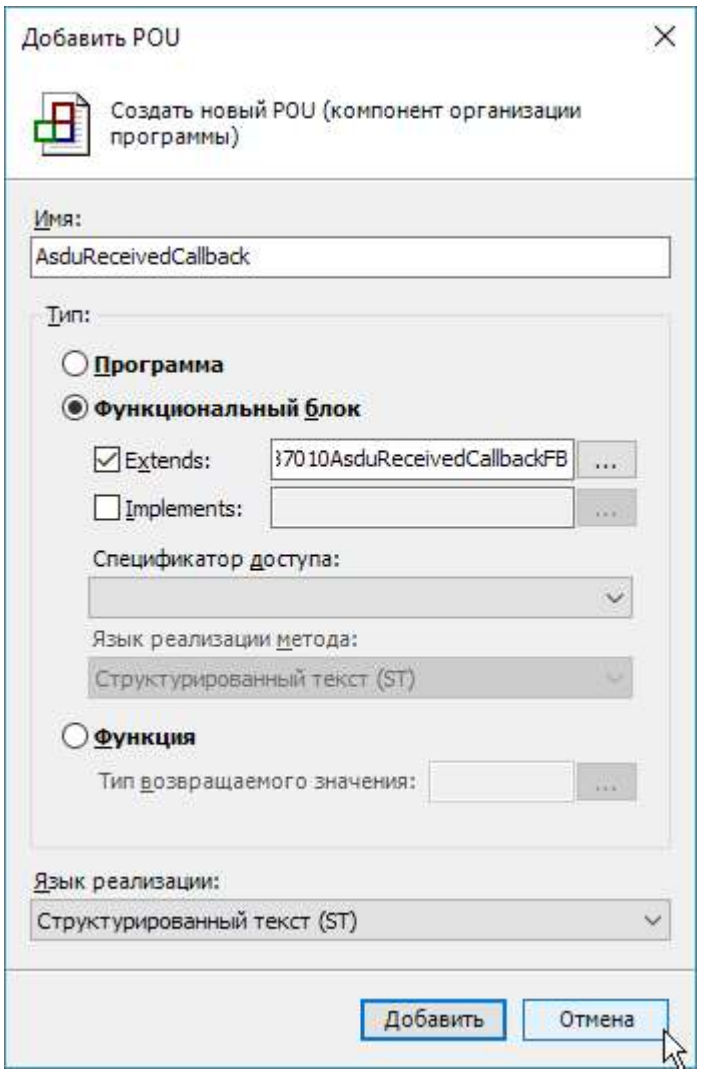


Рисунок 103 – Создание функционального блока

Поставьте галочку «Extends:» и введите в поле напротив неё значение
- «CmpIec60870104KAPP82.Iec6087010AsduReceivedCallbackFB» - для МЭК-104
- «CmpIec60870104KAPP82.Iec60870_101AsduReceivedCallbackFB» - для МЭК-101

или выберите его в «ассистенте ввода», нажав на кнопку «…» (рисунок 104).

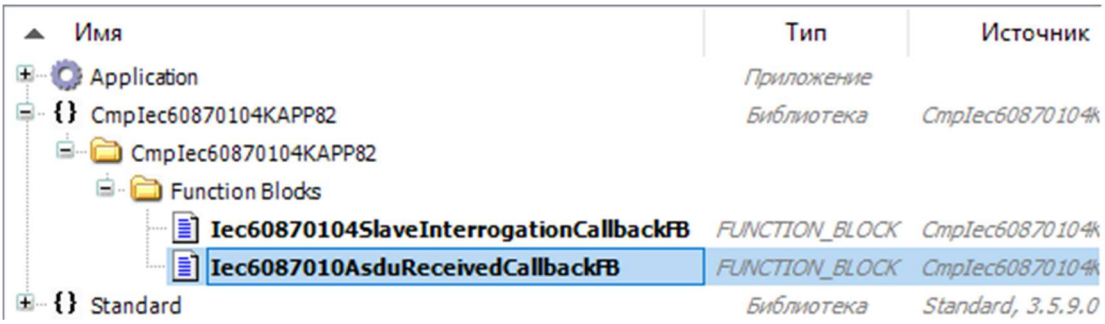


Рисунок 104 – Выбор шаблона функционального блока

При возникновении события приёма блока данных, будет вызываться функция EventCallback созданного функционального блока.

Далее, необходимо добавить создать экземпляр созданного функционального блока в список переменных (глобальный список или список программы):

```
asduReceivedCallback: AsduReceivedCallback;
```

Перейдём в метод EventCallback блока AsduReceivedCallback. Для примера, реализуем обработку команды уставки (короткое число с плавающей запятой, C_SE_NC_1).

Для этого добавим внутренние переменные для содержимого команды, объекта информации и причины передачи:

```
METHOD EventCallback : CmpEventMgr.RTS_IEC_RESULT
VAR_INPUT
     pEventParam: POINTER TO CmpEventMgr.EventParam;
END_VAR
VAR
     pEvtParam: POINTER TO
CmpIec60870104KAPP82.EVTPARAM_IEC60870104AsduReceived;
     newAsdu: CmpIec60870104KAPP82.Iec60870ASDU;
     cot: CmpIec60870104KAPP82.Iec60870CauseOfTransmission;
     io: CmpIec60870104KAPP82.Iec60870InformationObject;
     setpointCommandShort:
CmpIec60870104KAPP82.Iec60870SetpointCommandShort;
END_VAR
```

Реализация программы.

В тела метода необходимо сначала извлечь параметры события. Для этого, нужно привести переменную параметра к типу EVTPARAM_IEC60870104AsduReceived:

```
pEvtParam := pEventParam^.pParameter;
```

Для обработки команды, сначала нужно определить тип ASDU (функция Iec60870AsduGetType). Далее, определить причину передачи (функция Iec60870AsduGetCOT). Если тип ASDU и причина передачи совпадают с теми, которые нам необходимо обработать, извлекаем объект информации из ASDU с помощью функции Iec60870AsduGetElement, после чего преобразуем его в желаемый тип (функция Iec60870GetSetpointCommandShort). Значение уставки находится в поле Value объекта информации.

После обработки команды, необходимо отправить результат, установив причину передачи ActivationCon и отправив ASDU.

В конце, переменной result параметров функции присваивается значение TRUE при условии успешной обработки команды.

Код программы будет выглядеть следующим образом:

```
// Эта функция вызывается при приёме ASDU
// При успешной обработке ASDU необходимо отправить
// подтверждение и установить результат в TRUE
// Иначе результат устанавливается в FALSE
```

```
IF pEventParam^.EventId =
CmpIec60870104KAPP82.EventIds.EVT_IEC60870104ASDUReceived THEN
//Для МЭК-101 следует проверять на событие -
//CmpIec60870104KAPP82.EventIds.EVT_IEC60870_101ASDUReceived
pEvtParam := pEventParam^.pParameter;

// Определяем тип ASDU
asduType :=
CmpIec60870104KAPP82.Iec60870AsduGetType(pEvtParam^.asdu);

CASE asduType OF
 CmpIec60870104KAPP82.Iec60870Type.C_SE_NC_1:
  cot :=
CmpIec60870104KAPP82.Iec60870AsduGetCOT(pEvtParam^.asdu);

  IF cot =
CmpIec60870104KAPP82.Iec60870CauseOfTransmission.Activation THEN
   io :=
CmpIec60870104KAPP82.Iec60870AsduGetElement(pEvtParam^.asdu, 0);
   setpointCommandShort :=
     CmpIec60870104KAPP82.Iec60870GetSetpointCommandShort(io);

    IF setpointCommandShort.ObjectAddress =
Global.shortValue1.ObjectAddress THEN
     // Копируем значение уставки в объект информации
     Global.shortValue1.Value := setpointCommandShort.Value;
     CmpIec60870104KAPP82.Iec60870AsduSetCOT(pEvtParam^.asdu,

     CmpIec60870104KAPP82.Iec60870CauseOfTransmission.ActivationC
on);

CmpIec60870104KAPP82.Iec60870104SlaveSendASDU(pEvtParam^.connect
ion,
          pEvtParam^.asdu);
     pEvtParam^.result := TRUE;
    END_IF

  END_IF
END_CASE;
END_IF
```

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

Изм.	Кол.уч.	Лист	№ док.	Подпись	Дата

3.3.4 Обработка команды опроса

Для обработки команды опроса необходимо реализовать функцию обратного вызова. Для этого выберите пункт меню «Добавление объекта» – «POU…».

В окне «Добавить POU» (рисунок 105) введите имя создаваемого компонента, например, «InterrogationCallback» и выберите тип «Функциональный блок» и язык «ST».

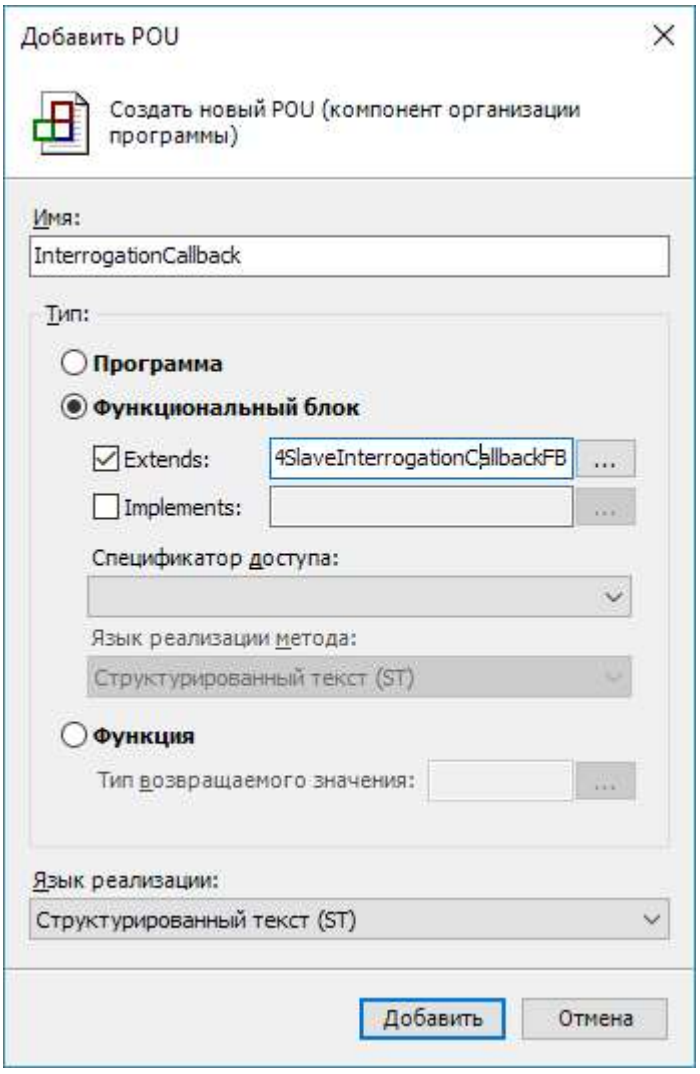


Рисунок 105 – Создание функционального блока

Поставьте галочку «Extends:» и введите в поле напротив неё значение

- «CmpIec60870104KAPP82.Iec60870104SlaveInterrogationCallbackFB» для МЭК-104
- «CmpIec60870104KAPP82.Iec60870_101_SlaveInterrogationCallbackFB» для МЭК-101

или выберите его в «ассистенте ввода», нажав на кнопку «…» (рисунок 106).

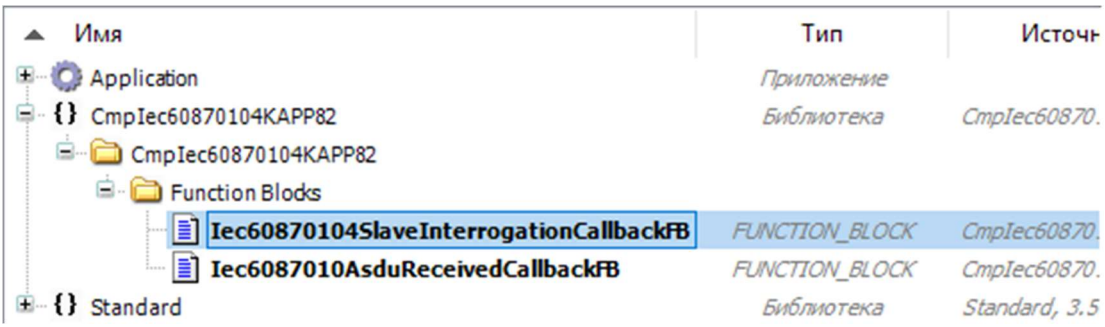


Рисунок 106 – Выбор шаблона функционального блока

При возникновении события опроса станции, будет вызываться функция EventCallback созданного функционального блока. Группа опроса помещается в параметры функции.

Далее, необходимо добавить создать экземпляр созданного функционального блока в список переменных (глобальный список или список программы):

```
interrogationCallback: InterrogationCallback;
```

Перейдём в метод EventCallback блока InterrogationCallback. Для примера, реализуем обработку команды общего опроса, отправив все имеющиеся на станции объекты информации.

Для этого добавим внутренние переменные для содержимого команды и нового ASDU:

```
METHOD EventCallback : CmpEventMgr.RTS_IEC_RESULT
VAR_INPUT
     pEventParam: POINTER TO CmpEventMgr.EventParam;
END_VAR
VAR
     PEvtParam: POINTER TO

     CmpIec60870104KAPP82.EVTPARAM_IEC60870104InterrogationReques
t;
     newAsdu: CmpIec60870104KAPP82.Iec60870ASDU;
END_VAR
```

Реализация:

В тела метода необходимо сначала извлечь параметры события. Для этого, нужно привести переменную параметра к типу IEC60870104InterrogationRequest:

```
pEvtParam := pEventParam^.pParameter;
```

Для обработки опроса, нужно отправить на контролирующую станцию подтверждение начала опроса, создать новый ASDU, поместить в него объекты информации, отправить ASDU и подтвердить завершение опроса.

Примерный код программы выглядит следующим образом:

```
// Эта функция вызывается при получении команды за опрос станции
// Это может быть как общий опрос, так и опрос определённой
```

```
группы
IF pEventParam^.EventId =
     CmpIec60870104KAPP82.EventIds.EVT_IEC60870104InterrogationRe
quest THEN
//Для МЭК-101 следует проверять на событие -
//CmpIec60870104KAPP82.EventIds.EVT_IEC60870_101InterrogationReq
uest
  pEvtParam := pEventParam^.pParameter;
  // pEvtParam^.qoi - группа опроса
    IF qoi = pEvtParam^.qoi THEN
    // Отправляем подтверждение начала опроса

CmpIec60870104KAPP82.Iec60870104SlaveSendActCon(pEvtParam^.conne
ction,    pEvtParam^.asdu, FALSE);

    // Создаём новый ASDU
    newAsdu := CmpIec60870104KAPP82.Iec60870AsduCreate(
     CmpIec60870104KAPP82.Iec60870Type.M_ME_NC_1, FALSE,

     CmpIec60870104KAPP82.Iec60870CauseOfTransmission.Interrogate
dByStation,
     0, 1, FALSE, FALSE);

    // Добавляем объекты информации в созданный ASDU

CmpIec60870104KAPP82.Iec60870AsduAddInformationObjectRealTS(newA
sdu,      ADR(IEC104.io1), 0);

CmpIec60870104KAPP82.Iec60870AsduAddInformationObjectRealTS(newA
sdu,      ADR(IEC104.io2) , 0);

CmpIec60870104KAPP82.Iec60870AsduAddInformationObjectRealTS(newA
sdu,      ADR(Global.shortValue1) , 0);

    // Отправляем ASDU
    // ASDU автоматически уничтожается после отправки
    // поэтому не следует создавать ASDU, не отправляя их
    // (это грозит переполнением памяти)
    // Можно создавать и отправлять несколько ASDU
последовательно,
    // если в одном не хватает места для всех объектов
информации

CmpIec60870104KAPP82.Iec60870104SlaveSendASDU(pEvtParam^.connect
ion,      newAsdu);

    // Отправляем завершение опроса
    CmpIec60870104KAPP82.Iec60870104SlaveSendActTerm(
    pEvtParam^.connection, pEvtParam^.asdu);

    // Возвращаем TRUE, если опрос станции успешно обработан
```

```
    pEvtParam^.result := TRUE;
END_IF
END_IF
```

3.3.5 Пример опроса контроллера по протоколам МЭК 60870-5-101/104

Приведенные ниже данные носят справочный характер и предназначены для ознакомления с возможностями контроллера в части передачи данных по протоколу МЭК 60870-5-101/104. Приведенные значения могут отличаться от указанных, в зависимости от сложности проекта пользователя, качества и загруженности каналов связи и т.д.

3.3.5.1 Время получения данных по протоколу МЭК 60870-5-101

При испольовании посылки команды «Общий опрос» на контроллер, с заданными на нем параметрами протокола:

- Порт COM3 (X1), 115200 8N1;
- Размер общего адреса ASDU (CA size) - 1 байт;
- Размер адреса объекта информации (IOT size) - 3 байта;
- Размер поля причина передачи (COT size) - 2 байт;
- Режим – балансный;
- Время ожидания подтверждения (TimeoutForACK) – 500 мс;
- Время для повторной посылки сообщения (Idle timeout) – 1500 мс;
- Использовать кадр канального уровня «ACK» (UseSingleCharACK) – TRUE;
- Размер очереди Class1 – 100;
- Размер очереди Class2 – 100,

среднее время цикла задачи IEC101 в контроллере составляет 0,2 мс.

Минимальное время получения данных на стороне ПК с момента посылки команды «Общий опрос» до момента получения последней переменной каждого типа в отдельности:

№	Тип переменной	Количество	Время, мс	
			10 запросов	100 запросов
1	M_SP_NA_1 (bool)	10	90	90
2	M_ME_NC_1 (real)	10	193	178

Количество переменных для соотношения 0,8 мс для дискретных, 0,2 мс для аналоговых переменных.

№	Тип переменной	Количество	Время, мс	
			10 запросов	100 запросов
1	M_SP_NA_1 (bool)	**1400**	830	811
2	M_ME_NC_1 (real)	**40**	222	247

Количество переменных для соотношения 0,5 мс для дискретных, 0,5 мс для аналоговых переменных.

№	Тип переменной	Количество	Время, мс	
			10 запросов	100 запросов
1	M_SP_NA_1 (bool)	**840**	505	480
2	M_ME_NC_1 (real)	**330**	504	499

Количество переменных для соотношения 0,2 мс для дискретных, 0,8 мс для аналоговых переменных.

№	Тип переменной	Количество	Время, мс	
			10 запросов	100 запросов
1	M_SP_NA_1 (bool)	**250**	209	214
2	M_ME_NC_1 (real)	**650**	806	831

3.3.5.2 Время получения данных по протоколу МЭК 60870-5-104

При испольовании посылки команды «Общий опрос» на контроллер, с заданными на нем параметрами протокола:

- Порт TCP 2404;
- Размер очереди сообщений (QueueSize) – 200;
- Размер очереди сообщений (HighPrioQueueSize) – 200;
- Параметр k – 1200;
- Параметр w – 800;
- Параметр t0 – 10;
- Параметр t1 – 15;
- Параметр t2 – 10;
- Параметр t3 – 20;

среднее время цикла задачи IEC104 в контроллере составляет 4 мс.

Минимальное время получения данных на стороне ПК с момента посылки команды «Общий опрос» до момента получения последней переменной каждого типа в отдельности:

№	Тип переменной	Количество	Время, мс	
			10 запросов	100 запросов
1	M_SP_NA_1 (bool)	10	2	2
2	M_ME_NC_1 (real)	10	3	3

Количество переменных для соотношения 0,8 мс для дискретных, 0,2 мс для аналоговых переменных.

№	Тип переменной	Количество	Время, мс	
			10 запросов	100 запросов
1	M_SP_NA_1 (bool)	**38000**	724	766
2	M_ME_NC_1 (real)	**7000**	198	196

Количество переменных для соотношения 0,5 мс для дискретных, 0,5 мс для аналоговых переменных.

№	Тип переменной	Количество	Время, мс	
			10 запросов	100 запросов
1	M_SP_NA_1 (bool)	**22500**	523	468
2	M_ME_NC_1 (real)	**16000**	451	467

Количество переменных для соотношения 0,2 мс для дискретных, 0,8 мс для аналоговых переменных.

№	Тип переменной	Количество	Время, мс	
			10 запросов	100 запросов
1	M_SP_NA_1 (bool)	**10000**	189	191
2	M_ME_NC_1 (real)	**28000**	767	768

3.4 Работа с функциями времени контроллера КАПП2 CPU (библиотека SysTime)

Для работы с часами реального времени, необходимо добавить библиотеку SysTime в проект CODESYS. Для этого в менеджере библиотек жмем кнопку «Добавить библиотеку» (рисунок 107).

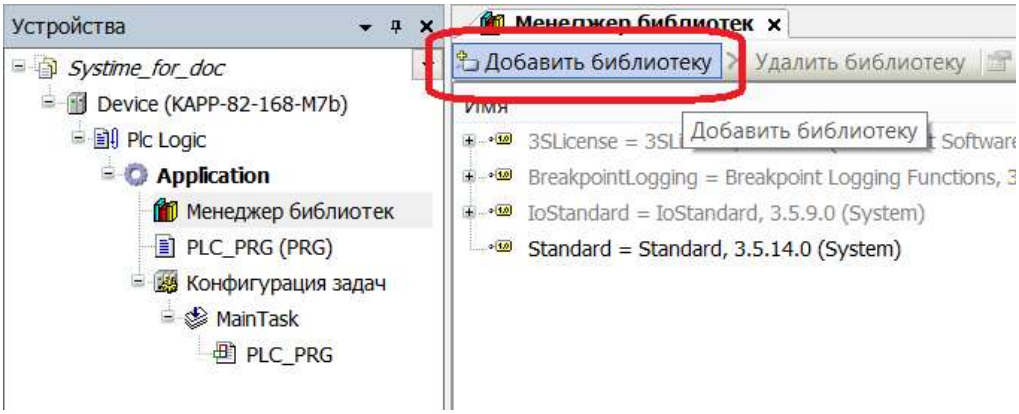


Рисунок 107 – Менеджер библиотек

В открывшемся окне «Библиотека» в нижнем правом углу жмем кнопку «Дополнительно…» и в новом окне, в строке поиска набираем ключевое слово «Systime», выбираем нужную библиотеку и нажимаем кнопку «ОК»(рисунок 108).

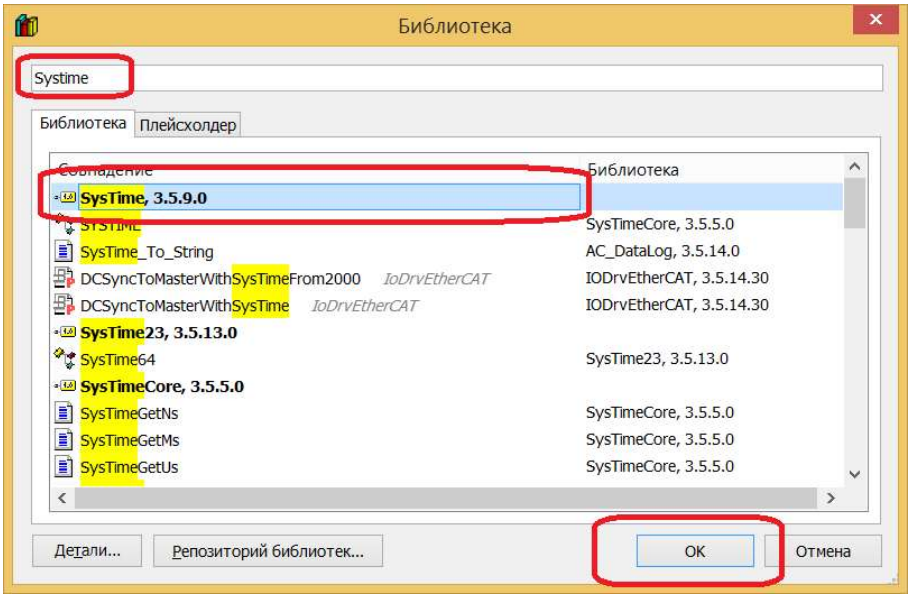


Рисунок 108 – Поиск и добавление библиотеки

Аналогичным способом необходимо добавить библиотеку SysTypes для дополнительных системных типов переменных (например, нам понадобится тип RTS_IEC_RESULT – результат выполнения функции), а так же библиотеку CmpErrors, в которой содержатся коды ошибок выполнения функций.

Данная библиотека состоит из двух: SysTimeCore и SysTimeRtc.

3.4.1 Функции SysTimeCore

Функции SysTimeCore предназначены для определения относительного времени:

SysTimeGetMs – возвращает монотонно возрастающее число, миллисекундные такты, число (UDINT) меняется в диапазоне от 0 до 4294967295. Такты можно использовать для измерения таймаута и относительного времени. **Примечание: часы реального времени не влияют на этот показатель!**

SysTimeGetNs – записывает в аргумент функции типа InOut монотонно возрастающее число, наносекундные такты. Можно использовать для измерения времени с очень высоким разрешением. **Примечание: часы реального времени не влияют на этот показатель! Возвращает код ошибки системы времени выполнения (см. CmpErrors.library)**

SysTimeGetUs – аналогичная функции описанной выше, только в данном случае 1 такт это 1 микросекунда.

Пример использования данных функций:

```
PROGRAM PLC_PRG
VAR
    TicMs: UDINT  ;
    TicNs: SYSTIME;
    TicUs: SYSTIME;
    ResultNs, ResultUs : RTS_IEC_RESULT;
END_VAR
```

```
TicMs:=SysTimeGetMs();
ResultNs:=SysTimeGetNs(TicNs);
ResultUs:=SysTimeGetUs(TicUs);
```

Обратите внимание, при онлайн просмотре выполнения кода переменная `TicNs` не изменяется, а результат выполнения (перменная `ResultNs`) равен 12 (рисунок 109).

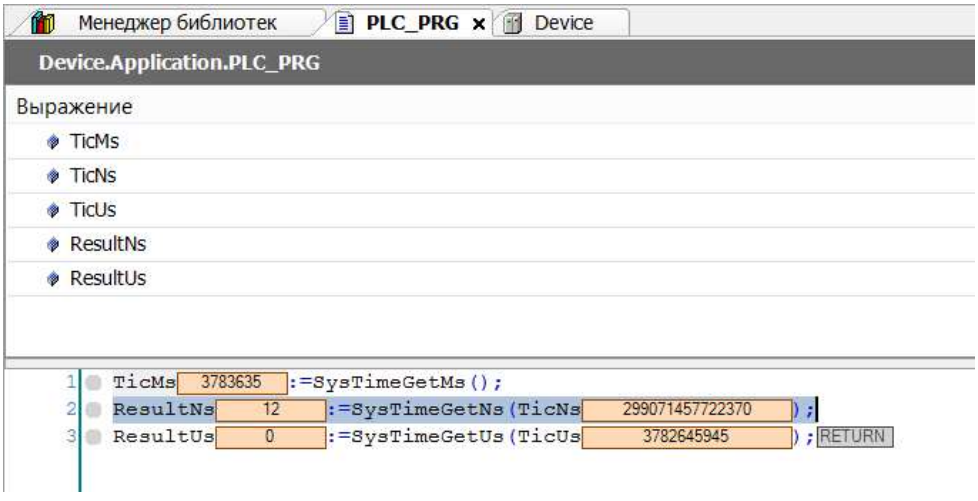


Рисунок 109 – Онлайн просмотр выполнения кода

По коду ошибки можно определить причину вызвавшую ее. В этом может помочь библиотека CmpErrors. Для этого необходимо открыть менеджер библиотек. Далее выделим данную библиотеку и найдем пункт «Errors» (рисунок 110).

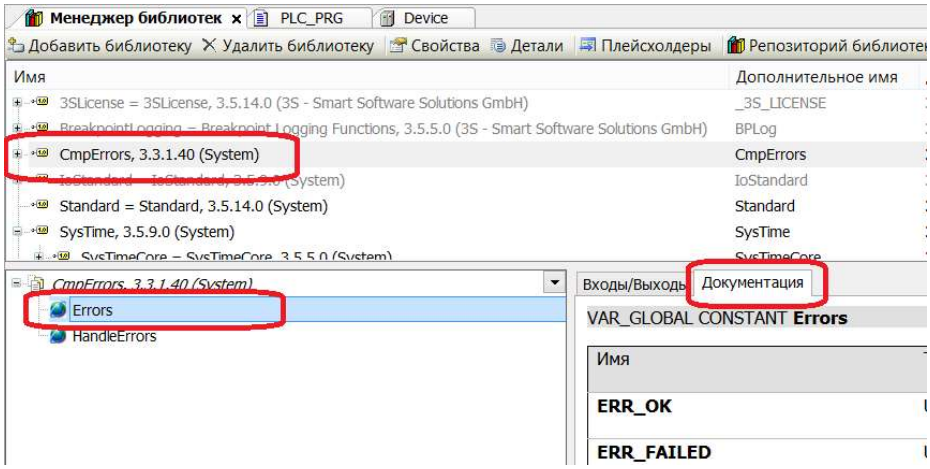


Рисунок 110 – Расположение пункта «Errors» в библиотеке CmpErrors

Далее во вкладке «Документация» найдем ошибку по ее коду. Необходимо учитывать, что коды ошибок в документации указаны в шестнадцатеричной системе счисления (на что указывает символ 16#). Режим отображения по умолчанию в Codesys десятичный. Убедиться в этом можно щелкнув правой кнопкой по полю с исполняемым кодом, пункт «Режим отображения», при необходимости можно переключить режим (рисунок 111).

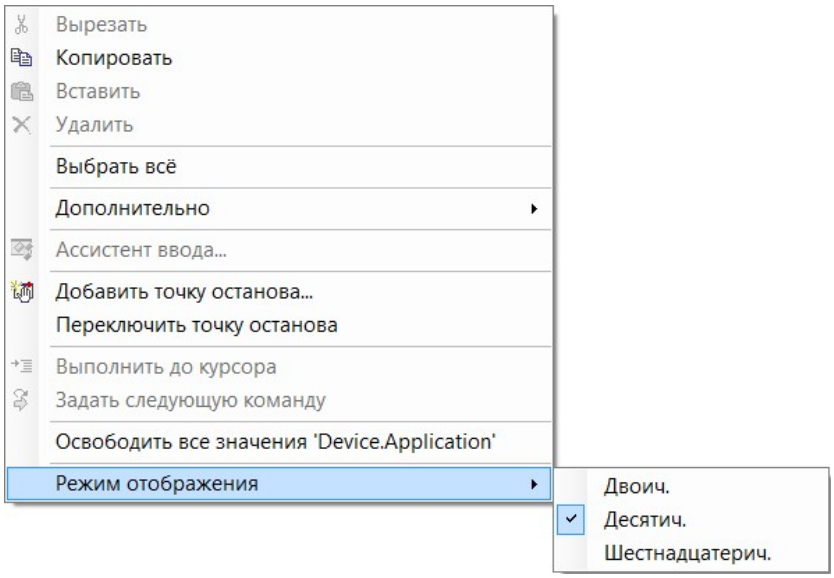
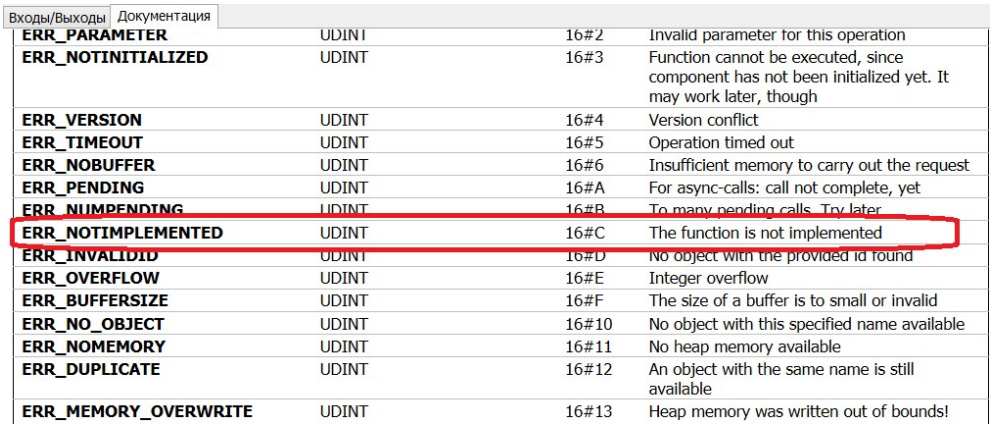


Рисунок 111 – Меню режима отображения переменных

В нашем случае код ошибки (12 равно 16#C) соответствует ошибке **ERR_NOTIMPLEMENTED**(рисунок 112)**.**



Входы/Выходы | Документация

ERR_PARAMETER	UDINT	16#2	Invalid parameter for this operation
ERR_NOTINITIALIZED	UDINT	16#3	Function cannot be executed, since component has not been initialized yet. It may work later, though
ERR_VERSION	UDINT	16#4	Version conflict
ERR_TIMEOUT	UDINT	16#5	Operation timed out
ERR_NOBUFFER	UDINT	16#6	Insufficient memory to carry out the request
ERR_PENDING	UDINT	16#A	For async-calls: call not complete, yet
ERR_NUMPENDING	UDINT	16#B	To many pending calls. Try later
ERR_NOTIMPLEMENTED	UDINT	16#C	The function is not implemented
ERR_INVALIDID	UDINT	16#D	No object with the provided id found
ERR_OVERFLOW	UDINT	16#E	Integer overflow
ERR_BUFFERSIZE	UDINT	16#F	The size of a buffer is to small or invalid
ERR_NO_OBJECT	UDINT	16#10	No object with this specified name available
ERR_NOMEMORY	UDINT	16#11	No heap memory available
ERR_DUPLICATE	UDINT	16#12	An object with the same name is still available
ERR_MEMORY_OVERWRITE	UDINT	16#13	Heap memory was written out of bounds!

Рисунок 112 – Документация по ошибкам библиотеки CmpErrors.

Данная ошибка сообщает о том, что функция не поддерживается аппаратно, что обусловлено архитектурой контроллера.

Данные функции можно использовать, например, для замера времени исполнения участка кода в микросекундах. Для этого в цикле от 1 до 100, будем делить переменную `VarReal` на величину итерации цикла и результат помещать в эту же переменную (`VarReal`). Пример:

```
PROGRAM PLC_PRG
VAR
    VarReal: REAL := 10000.123456;
    I: INT;
    CycleWorkTime, CycleStartTick: ULINT;
END_VAR
```

```
 // определяем начальное значение счетчика
SysTimeGetUs(CycleStartTick);

// выполняем необходимые операции
FOR i:=1 TO 100 DO
    VarReal:= VarReal / i;
END_FOR

// определяем значение счетчика после выполнения операций
SysTimeGetUs(CycleWorkTime);

//количество микросекунд затраченных на выполнение необходимых операций
CycleWorkTime:= CycleWorkTime - CycleStartTick;
```

Полученные результаты (рисунок 113) можно сравнить с временем выполнения задачи. Интуитивно понятно, что они примерно будут равны, разница будет лишь в выполнении функций SysTimeGetUs, SysTimeGetUs и определении разницы значений счетчика. Время выполнения задачи можно посмотреть во вкладке «Конфигурация задач» (рисунок 114).

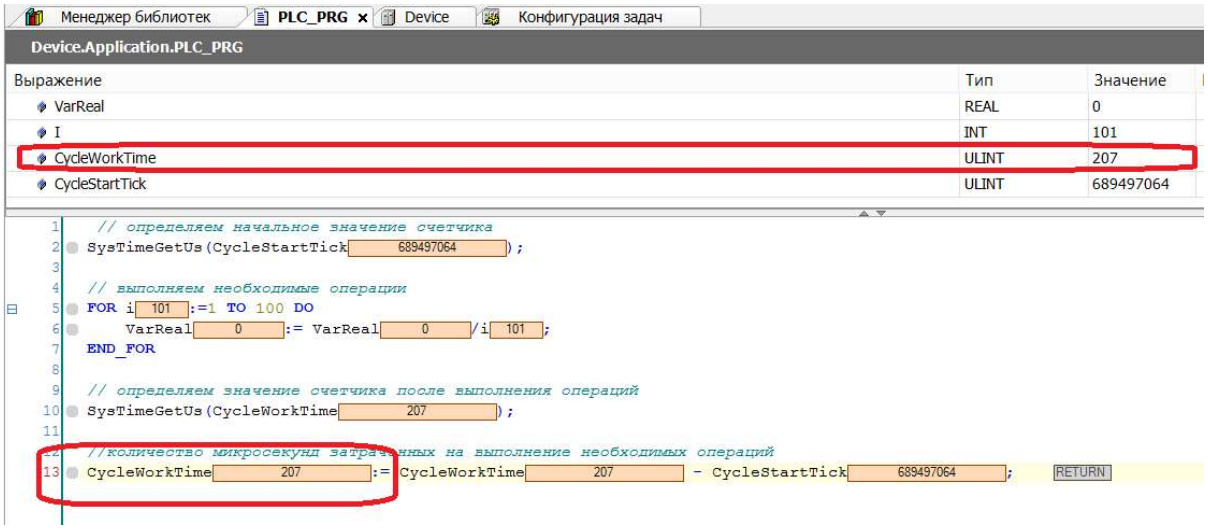


Рисунок 113 – Выполнение кода в режиме онлайн

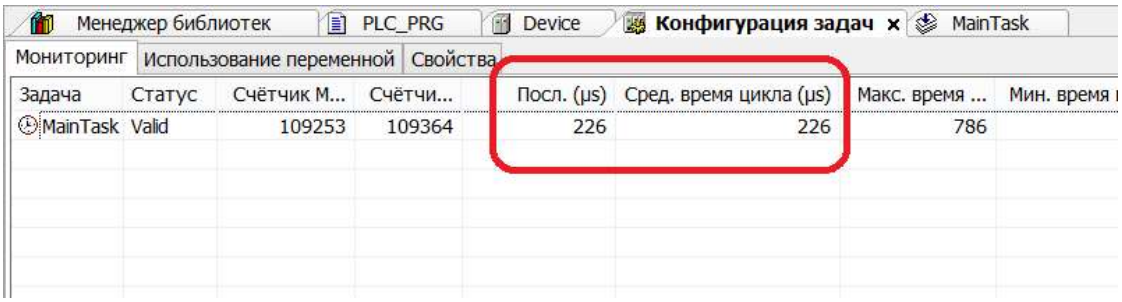


Рисунок 114 – Вкладка конфигурация задач в режиме онлайн

Согласовано

Взаим. инв.

№Взаим. инв.

Подп. и дата

Инв. № подл.

Изм.	Кол.уч.	Лист	№ док.	Подпись	Дата

73619730.26.20.30.000.020 РЭ

Лист

103

ФорматА4

3.4.2 Функции SysTimeRtc

Функции SysTimeRtc предназначены для работы с часами реального времени контроллера.

Библиотека SysTimeRtc не поддерживает часовые пояса и связанные с этим функции.

Функции SysTimeRtc группы High Resolution предназначены для работы со временем высокого разрешения:

SysTimeRtcHighResGet – возвращает текущее время в формате UTC с учётом миллисекунд. Возвращаемое значение имеет тип SYSTIME (64-разрядное целое число).

SysTimeRtcConvertHighResToDate – преобразует время из типа SYSTIME в структурированный формат SYSTIMEDATE (с учётом миллисекунд).

SysTimeRtcConvertDateToHighRes – обратное преобразование времени из структурированного формата SYSTIMEDATE в время SYSTIME (с учётом миллисекунд).

Функция установки текущего времени **SysTimeRtcHighResSet** не поддерживается из-за отсутствия аппаратной возможности задать миллисекунды системных часов. Для установки текущего времени с точностью до секунд используйте функцию **SysTimeRtcSet.**

Функции SysTimeRtc папки Standart поддерживаемые контроллером:

SysTimeRtcGet – возвращает текущее время в формате UTC. Тип возвращаемого числа DWORD.

SysTimeRtcSet – установка системных часов, время передается в формате UTC, типа DWORD.

SysTimeRtcConvertUtcToDate – преобразует время в формате UTC (DWORD) в структурированный формат «SYSTIMEDATE».

SysTimeRtcConvertDateToUtc – обратное преобразование из структурированного формата в UTC (DWORD).

Для непосредственной работы со временем используется структура «RTS_SYSTIMEDATE» – псевдоним «SYSTIMEDATE». Далее приведена таблица, описывающая поля данной структуры.

Таблица 17 – Структура RTS_SYSTIMEDATE:

Имя	тип	комментарий
wYear	UINT	Год (например 2006)
wMonth	UINT	Месяц (1..12: Январь = 1, Декабрь = 12)
wDay	UINT	День месяца / число (1..31)
wHour	UINT	Часы (0..23)
wMinute	UINT	Минуты (0..59)
wSecond	UINT	Секунды (0..59)
wMilliseconds	UINT	Миллисекунды (0..999). Опционально! (не поддерживаются)
wDayOfWeek	UINT	День недели (1..7: Воскресенье = 1, Суббота=7)
wYday	UINT	День года (1..365): 1ое Января = 1, 31ое декабря = 364/365

UTC – Всемирное координированное время (англ. Coordinated Universal Time) – стандарт, по которому общество регулирует часы и время. Отличается на целое количество секунд от атомного времени и на дробное количество секунд от всемирного времени UT1. UTC было введено вместо устаревшего среднего времени по Гринвичу (GMT).

Пример использования функций:

```
PROGRAM PLC_PRG
VAR
    Result: RTS_IEC_RESULT;
    TimeUTC, TimeLoc: DWORD;
    SysDate: SYSTIMEDATE;
END_VAR
```

```
TimeUTC:=SysTimeRtcGet(Result);
Result:=SysTimeRtcConvertUtcToDate(TimeUTC, SysDate);
```

В данном примере получаем значение системного времени контроллера в формате UTC, а затем переводим его в понятную нам структуру SYSTIMEDATE, описанную ранее в таблице 17.

С помощью функции SysTimeRtcSet можно изменить часы реального времени контроллера. Модифицируем ранее описанный код для возможности изменения текущего часа:

```
PROGRAM PLC_PRG
VAR
    change: BOOL :=FALSE;
    Result: RTS_IEC_RESULT;
    TimeUTC, TimeLoc: DWORD;
    SysDate: SYSTIMEDATE;
    Hour: UINT;
END_VAR
```

```
TimeUTC:=SysTimeRtcGet(Result);
SysTimeRtcConvertUtcToDate(TimeUTC, SysDate);
IF change THEN
    SysDate.wHour:=Hour;
    SysTimeRtcConvertDateToUtc(SysDate, TimeUTC);
    SysTimeRtcSet(TimeUTC);
    change:= false;
END_IF
```

В данном примере по событию `change=TRUE` меняется текущий час на значение указанное в переменной `Hour`. Ели значение `Hour` отлично от указанного в таблице 17 диапазона изменения не произойдет.

Важно!!! Все переменные структуры RTS_SYSTIMEDATE могут принимать значения только из указанного в таблице 17 диапазона.

Совет!!! Часы реального времени контроллера лучше всего менять с помощь сервисного ПО Caltester.exe, функцией «Синхронизация времени с ПК» (см. Программа «АТ-КАПП» для настройки и проверки работоспособности ПЛК КАПП-82-168 и КАПП2-00-000-1. Руководство пользователя. 73619730.425200.005 34).

Так же данные функции хорошо подойдут для создания стекового исторического архива. Задача: есть некая переменная `VarReal`, при выходе ее значение за 10 необходимо в исторический архив размером 30 последних записей, записывать время и дату возникновения данного события. Пример реализации задачи:

```
PROGRAM PLC_PRG
VAR
    err: BOOL :=FALSE;     // выход за предел наблюдаемого
    параметра
    Result: RTS_IEC_RESULT;
    TimeUTC: DWORD;
    SysDate: SYSTIMEDATE;
    History: ARRAY [1..30] OF SYSTIMEDATE;    // исторический
    архив
    VarReal: REAL;                              // наблюдаемый
    параметр
    R_TRIG_Inst: R_TRIG;   // детектор импульса по переднему
    фронту
    i: usint :=1;                    // номер записи в архиве
    History
END_VAR
```

```
// определяем выход за допустимый диапазон наблюдаемого
параметра
IF VarReal>10 THEN
    err:= TRUE;
ELSE
    err:= FALSE;
END_IF

// наблюдаем за возникновением выхода за диапазон
R_TRIG_Inst(CLK:= err);

// по переднему фронту считываем время и дату с контроллера и
записываем в соответствующий элемент исторического архива,
инкрементируем индекс события
IF R_TRIG_Inst.Q THEN
    TimeUTC:=SysTimeRtcGet(Result);
    SysTimeRtcConvertUtcToDate(TimeUTC, SysDate);
    History[i]:= SysDate;
    i:=i+1;
    IF i>30 THEN // при выходе за архив возвращаемся к 1
    элементу
        i:=1;
    END_IF
END_IF
```

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

Изм.	Кол.уч.	Лист	№ док.	Подпись	Дата

3.5 Работа с SD картой контроллера КАПП2 CPU

Для просмотра, загрузки новых и переноса файлов с SD–карты на ПК без извлечения ее из контроллера можно воспользоваться менеджером файлов среды CODESYS. Чтобы открыть менеджер файлов, нужно выполнить двойной щелчок мышью по значку «Устройство» («Device») в дереве проекта, после чего откроется соответствующая вкладка (рисунок 115).

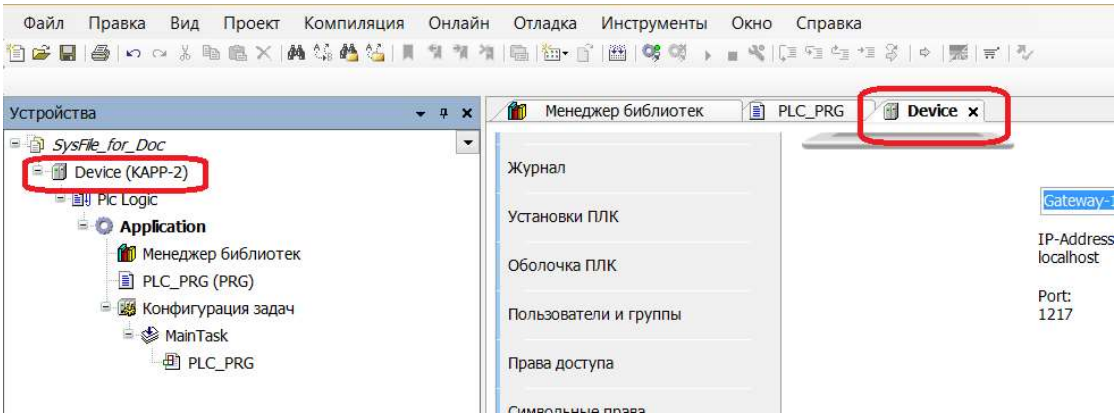


Рисунок 115 Расположение значка «Device» и открытая соответствующая вкладка

Далее обязательно проверить соединение с контроллером KAPP-2, для чего убедиться, что индикаторы подключения отображаются зеленым цветом, а имя устройства имеет значение «KAPP-82-168» (рисунок 116).

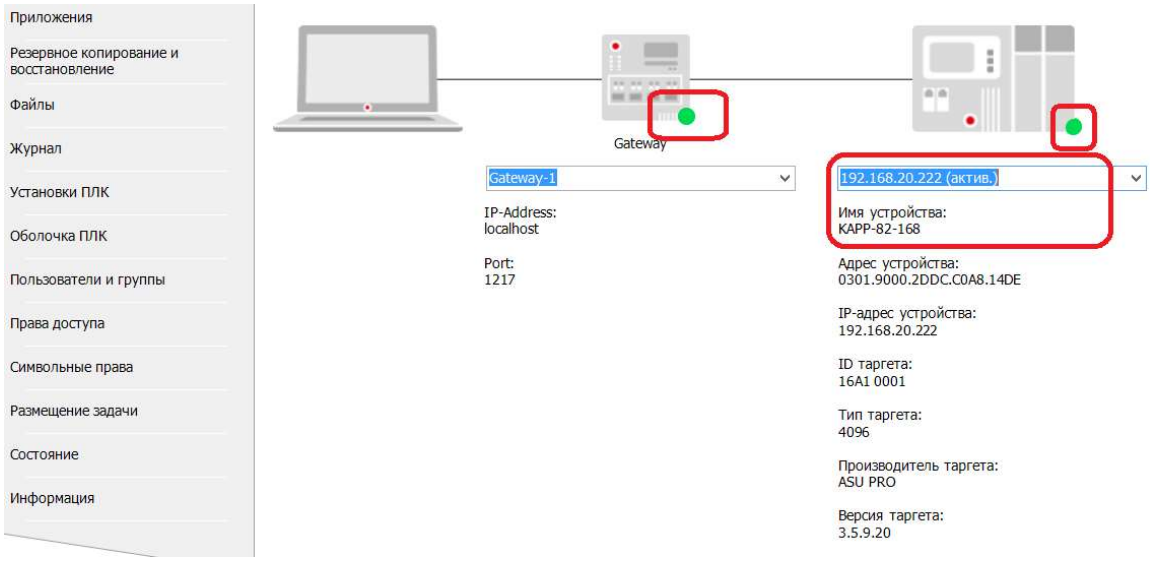


Рисунок 116 – Активное соединение с контроллером на вкладке «Device» Установка соединения

В случае отсутствия соединения, необходимо установить его, путем ввода IP-адреса контроллера KAPP-2, настроенного в конфигурационном файле на SD-карте и нажать клавишу «Enter». Более подробно можно почитать в главе 2.3.9 Установка связи с контроллером.

Затем необходимо перейти на вкладку «Файлы». Для получения списка файлов на SD-карте необходимо нажать кнопку «Обновить». При помощи кнопок «>>» и «<<» можно перенести фалы с ПК на SD – карту и обратно.

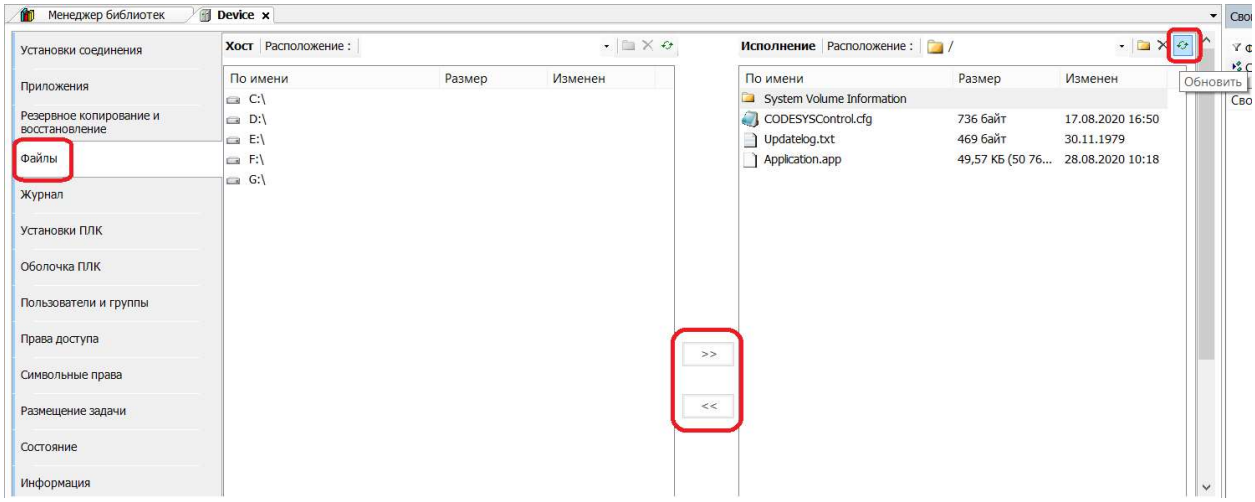


Рисунок 117 – Файловый менеджер среды CODESYS

Следует учитывать, что действуют ограничения конкретной файловой системы SD-карты (FAT, FAT32 или EFAT). По умолчанию максимальный размер фала 4 Гбайт. Файл «Application.app» является программой пользователя, его не следует изменять или открывать из программы пользователя, так как в этом случае программа пользователя повредит сама себя. Ниже приведена таблица с ограничениями для поддерживаемых файловых систем:

Таблица 18 – Ограничения для поддерживаемых файловых систем:

ФС	Максимальная длина имен	Допустимые символы	Максимальная длина пути	Максимальный размер файла
FAT	8 3 ANSI	Любые символы ANSI (Unicode для VFAT), кроме NUL.	Нет ограничений	2 Гбайта
FAT32	255 UTF	Любые символы Юникода, кроме NUL.	Нет ограничений	4 Гбайта
ExFAT	255 UTF	Любые символы Юникода, кроме NUL.	Нет ограничений	16 Эбайта

Внимание! В текущей версии прошивки (v2.0.2.8) возможна загрузка файлов размером не более 2 Мбайт.

Контроллер поддерживает карты форматов SD и SDHC до Class 10. Использование иных типов карт не гарантирует его нормальной работы.

Стоит отметить, что для того, чтобы скопировать при помощи оболочки Codesys файл созданный в контроллере (к примеру, текстовый файл содержащий значения аналогового/дискретного сигнала и время контроллера), файл должен быть гарантированно закрыт в момент копирования!!! См. пункт 3.5.1.

3.5.1 Работа с SD картой контроллера КАПП2 CPU с помощью библиотеки SysFile

Для работы с файловой системой microSD карты, необходимо добавить следующие библиотеки в проект CODESYS: SysFile; SysTypes; CmpErrors. Как добавить библиотеку можно посмотреть в пункте 3.4 Работа с функциями времени контроллера КАПП2 CPU (библиотека SysTime).

Для работы с файлами необходима переменную типа «RTS_IEC_HANDLE», которая будет являться указателем на структуру файла (дескриптор). Так же понадобиться переменная типа «RTS_IEC_RESULT» для хранения результата операции (описание кодов ошибок можно найти в документации к библиотеке «CmpErrors»). Подробнее по кодам ошибок можно посмотреть в главе 3.4.1 Функции SysTimeCore.

Для примера напишем код приложения, который по команде `run` создает файл «file.txt» в корне SD карты и помещает в него текст «Hello World!!!», а затем в строковую переменную запишем этот текст из файла. Пример:

```
PROGRAM PLC_PRG
VAR
    run: bool := false;
    FILE_X:  RTS_IEC_HANDLE;
    F_RESULT:  RTS_IEC_RESULT;
    APPEND_DATA: STRING := 'Hello World!!!';
    READ_DATA : ARRAY [0..99] OF BYTE;
    VarSTR: STRING;
    LenFile: DWORD;
    i: INT;
    pSTR: POINTER TO STRING;
END_VAR
```

```
IF run THEN
    run:= FALSE;

    //запись строки APPEND_DATA в файл
    FILE_X:=SysFileOpen('file.txt',         SysFile.AM_WRITE,
    ADR(F_RESULT));
    IF F_RESULT = CmpErrors.Errors.ERR_OK THEN
        SysFileWrite(FILE_X,                ADR(APPEND_DATA),
        TO_DWORD(LEN(APPEND_DATA)), ADR(F_RESULT));
        SysFileClose(FILE_X);
    END_IF

    //чтение в буфер READ_DATA из файла
    LenFile:=SysFileGetSize('file.txt', F_RESULT);
    FILE_X:=     SysFileOpen('file.txt',     SysFile.AM_READ,
    ADR(F_RESULT));
    IF F_RESULT = CmpErrors.Errors.ERR_OK THEN
        SysFileRead(FILE_X,      ADR(READ_DATA),      LenFile,
        ADR(F_RESULT));
        SysFileClose(FILE_X);
    END_IF
```

```
        //преобразование буфера READ_DATA в строку VarSTR
        pSTR:= ADR(READ_DATA);
        VarSTR:= pSTR^;
        // обнуление (очистка) буфера
        FOR i:=0 TO 99 DO
            READ_DATA[i]:=0;
        END_FOR
    END_IF
```

Обратите внимание на то, что в примере для чтения используется не строка, а массив байтов, который после преобразования и записи данных из него в строку полностью обнуляется. При использовании строки без промежуточного массива, при повторном чтении данных меньших, чем считанные ранее, длинна строки будет равна максимальной длине считанных данных. И соответственно оставшиеся ранее считанные данные остаются и меняются только данные считанные повторно. Например: считываем строку «`Hello World!!!`», затем считываем из файла строку с меньшей длиной «`World`», а получаем строку «`World World!!!`».

В примере файл открывается с помощью функции «SysFileOpen». Подставив в функцию имя файла, режим открытия файла и указатель на переменную типа «RTS_IEC_RESULT» функция возвращает результата открытия файла. Если результат соответствует «ERR_OK», файл открылся успешно, описатель помещен в переменную типа «RTS_IEC_HANDLE». Таблица 19 содержит имена и описания режимов открытия файлов.

Таблица 19 – Режимы открытия файлов:

Наименование режима	Описание
AM_READ	Открывает существующий файл с доступом только для чтения. Если файла не существует, будет возвращена ошибка. Указатель устанавливается в начало файла.
AM_WRITE	Создает новый файл с доступом только на запись. Если файл уже существует, его содержимое отбрасывается. Указатель устанавливается в начало файла.
AM_APPEND	Открывает существующий файл с доступом только на запись. Если файла не существует, будет возвращена ошибка. Указатель устанавливается в конец файла.
AM_READ_PLUS	Открывает существующий файл с доступом на чтение и запись. Если файла не существует, будет возвращена ошибка. Указатель устанавливается в начало файла.
AM_WRITE_PLUS	Создает новый файл с доступом на чтение и запись. Если файл уже существует, его содержимое отбрасывается. Указатель устанавливается в начало файла.
AM_APPEND_PLUS	Открывает существующий файл с доступом на чтение и запись. Если файл не существует, то создает новый файл. Указатель устанавливается в конец файла.

Важно!!! После завершения работы с файлом его необходимо закрыть, в противном случае есть вероятность потери или искажения его содержимого. Для этого используется соответствующая функция «SysFileClose».

Совет!!! Если необходимо использовать функцию чтения и записи в одном действии (последовательно), лучше использовать режим открытия PLUS и закрыть файл в конце этого действия. Так же не забывайте контролировать текущее положение указателя функциями SysFileGetPos и SysFileSetPos.

Реализованные в КАПП2 CPU функции библиотеки SysFile можно посмотреть в таблице 20.

Таблица 20 – Реализованные функции библиотеки SysFile:

Название	Комментарий
SysFileOpen	открыть файл
SysFileClose	закрыть файл
SysFileTruncate	усечь файл
SysFileRead	прочесть из файла
SysFileWrite	записать в файл
SysFileDelete	удалить файл
SysFileRename	переименовать файл
SysFileGetPos	получить текущую позицию в файле
SysFileGetSize	получить размер файла
SysFileSetPos	установить позицию файла
SysFileCopy	скопировать файл
SysDirOpen	открыть папку
SysDirClose	закрыть папку
SysDirRead	прочитать содержимое папки
SysDirCreate	создать папку
SysDirDelete	удалить папку
SysDirRename	переименовать папку
SysDirGetCurrent	получить активную папку
SysDirSetCurrent	установить активную папку
SysFileEOF	функция проверяет достигнут ли конец файла (при чтении)
SysFileFlush	Запись данных в файл, при этом файл остается открытым.

Более подробно про функции можно почитать в описании и документации Менеджера библиотек.

3.5.2 Доступ к SD карте контроллера КАПП2 CPU через FTP сервер.

Для доступа к файлам на SD накопителе, в контроллере реализован FTP сервер. Служба FTP сервера запущена постоянно. Доступ к фалам осуществляется по IP адресу контроллера, порт 21, по умолчанию логин = USER пароль = USER. Изменить логин и пароль можно в файле KAPP_Config.ini на SD накопителе.

Не рекомендуется копировать файлы с контроллера в режиме ONLINE CODESYS. Это может привести к повреждению файла. Для копирования файлов из контроллера предпочтительно использовать Total Commander.

3.6 Работа с портами USART (RS232 / 485) контроллера КАПП2 CPU с помощью библиотеки SysCom

Для работы с портами COM1, COM2, COM3 (X1), COM4 необходимо добавить следующие библиотеки в проект CODESYS: SysCom; SysTypes; CmpErrors.

Важно!!! Реализация работы с портами должна выполняться в отдельной задаче, так как время выполнения такой задачи может превышать указанный в настройках конфигурации задач интервал ее выполнения, в следствии ожидания таймаутов приема/передачи.

В контроллере реализован весь функционал библиотеки SysCom. В таблице 21 приведен перечень доступных функций.

Таблица 21 – Перечень функций библиотеки SysCom:

Название	Комментарий
SysComClose	закрывает устройство последовательной связи
SysComGetSettings	получает настройки интерфейса по переданному указателю
SysComOpen	открывает устройство последовательной связи. **Не рекомендуется использовать данную функцию, так как открытие производится без настройки!**
SysComOpen2	открывает устройство последовательной связи, и настраивает согласно переданной в «pSettings» структуре настроек. **ВАЖНО!!! Параметр «pSettingsEx» должен быть равен нулю**
SysComPurge	производит очистку буфера чтения (используется FIFO кольцевой буфер, размер фиксирован и не зависит от настройки в структуре «pSettings», равен 1024 байт)
SysComRead	производит чтение из устройства последовательной связи в неблокирующем режиме. Если в кольцевом FIFO буфере имеется затребованное кол-во байт, то возврат произойдет немедленно. В противном случае задача, из которой вызвана данная функция, будет остановлена до того момента пока в буфере не наберется требуемое количество байт, или не выйдет период времени (таймаут) указанный в структуре «pSettings»
SysComSetSettings	изменение /установка настроек в переданное устройство последовательной связи
SysComSetTimeout	отдельно изменяет таймаут операций чтения и записи в устройство последовательной связи
SysComWrite	производит запись в устройство последовательной связи в неблокирующем режиме

Структура конфигурации порта «COM_Settings», которая задает настройки, таймаут операций чтения и записи приведена в таблице 22.

Таблица 22 – Структура конфигурации COM_Settings:

Имя	Тип	Комментарий
sPort	COM_Ports	Номер порта: 0=запрещен, 1=COM1, 2=COM2, N=COMN
byStopBits	COM_StopBits	Число стоп бит, см. таблицу 23 – Перечисление COM_StopBits
byParity	COM_Parity	Паритет, см. таблицу 24 – Перечисление «COM_Parity»

ulBaudrate	COM_Baudrate	Скорость порта, см. таблицу 25 – Перечисление «COM_Baudrate»
ulTimeout	UDINT	Ограничение времени на операции чтения и записи в миллисекундах, «SYS_NOWAIT» (0) - без таймаута, «SYS_INFINITE» (16#FFFFFFFF) - ожидание без ограничений по времени
ulBufferSize	UDINT	Размер кольцевого буфера FIFO, в данном случае указывать его не имеет смысла, так как в ПЛК он фиксирован и равен 1024 байт.

Таблица 23 – Перечисление COM_StopBits:

Имя	Описание
SYS_ONESTOPBIT	Один стоп бит
SYS_ONE5STOPBITS	Полтора стоп бита
SYS_TWOSTOPBITS	Два стоп бита

Таблица 24 – Перечисление COM_Parity:

Имя	Описание
SYS_NOPARITY	Без паритета
SYS_ODDPARITY	Нечётный паритет
SYS_EVENPARITY	Чётный паритет

Таблица 25 – Перечисление COM_Baudrate

Имя	Описание
SYS_BR_4800	Скорость 4800 Бод
SYS_BR_9600	Скорость 9600 Бод
SYS_BR_19200	Скорость 19200 Бод
SYS_BR_38400	Скорость 38400 Бод
SYS_BR_57600	Скорость 57600 Бод
SYS_BR_115200	Скорость 115200 Бод

Рассмотрим пример использования библиотеки для частичной реализации протокола Modbus. В частности реализуем функцию чтения ста регистров хранения в режиме подчиненного с ограничениями. В примере, что бы сильно не усложнять код, при приеме посылки будет только проверка SlaveID (посылка, адресуемая нашему контроллеру) и кода функции (код функции 03 – чтение регистров хранения). Результат выполнения будет выдаваться ведущему с контрольной суммой CRC и кодом ошибки в случае кода функции отличного от 03 – чтение регистров ранения. Проверка контрольной суммы CRC проверка соответствия запрашиваемых регистров и прочее, реализовываться не будет. Для начала добавим функциональный блок AddCRC, где входными данными будут массив и количество байтов, для которых требуется вычислить контрольную сумму. Выходная переменная – слов контрольная сумма CRC:

```
FUNCTION_BLOCK AddCRC
VAR_INPUT
      INPUT:ARRAY [0..250] OF BYTE; (*входной буфер*)
      size: UINT;(*длина буфера*)
END_VAR
VAR_OUTPUT
```

```
        CRC: WORD; (*вычисленная контрольная сумма*)
    END_VAR
    VAR
        i, j: UINT;
        VarBool: BOOL;
    END_VAR
```

```
    CRC:= 16#FFFF;
    FOR i:=0 TO size-1 DO                          // т.к начало с 0
        crc := crc XOR INPUT[i];
        FOR j:=0 TO 7 DO                           // 8 бит в байте
            VarBool:=crc.0;
            crc:= SHR(crc,1);                      // сдвиг вправо
            IF VarBool THEN
                crc:=crc XOR 16#A001;              // полином 16#A001 для Modbus
            END_IF
        END_FOR
    END_FOR
```

Для изменения порядка байтов в слове понадобится создать объединение Word_Byte (как ранее указывалось в главе 3.2.5 в модуле КАПП2-00-000-1 порядок байтов ВА):

```
    TYPE Word_Byte :
    UNION
        VarWord: WORD;
        ArrByte: ARRAY [0..1] OF BYTE;
    END_UNION
    END_TYPE
```

После этого добавим функциональный блок Processing, в котором будет производиться обработка полученного запроса. Входными данными будут принятые данные от ведущего устройства и массив регистров хранения. Выходными данными будут массив данных для передачи и его длина:

```
    FUNCTION_BLOCK Processing
    VAR_INPUT
        RxData: ARRAY [0..250] OF BYTE;
        HoldReg: ARRAY [0..99] OF WORD;
    END_VAR
    VAR_OUTPUT
        TxData: ARRAY [0..250]  OF BYTE;
        TxLen: UDINT;
    END_VAR
    VAR
        AddrData, Quantity, Temp: Word_Byte;
        i, j: UINT;
        CRC: WORD:= 16#FFFF;
        FunCRC: AddCRC;
    END_VAR
```

```
    CASE  RxData[1]OF          // действие в зависимости от кода
```

```
функции
 3:                                    //обработка функции 03
      TxData[0]:= RxData[0];           //адрес ведомого
      TxData[1]:= RxData[1];           //код функции

      //перекладываем в нужный порядок байтов начальный адрес
и количество регистров
      AddrData.ArrByte[1]:= RxData[2];
      AddrData.ArrByte[0]:= RxData[3];
      Quantity.ArrByte[1]:= RxData[4];
      Quantity.ArrByte[0]:= RxData[5];

      //количество байтов для передачи (1 регистр – 2 байта)
      TxData[2]:= TO_BYTE(Quantity.VarWord*2);
      j:=3;

      // перекладываем запрошенные регистры в посылку для
передачи с учетом порядка байтов BA, через временную
переменную
      FOR i:=AddrData.VarWord TO (Quantity.VarWord -1) DO
           Temp.VarWord:= HoldReg[i];
           TxData[j]:= Temp.ArrByte[1];
           j:= j+1;
           TxData[j]:= Temp.ArrByte[0];
           j:= j+1;
      END_FOR

      // вычисляем контрольную сумму и получившуюся длину
посылки
      FunCRC(INPUT:=TxData, size:=(j), CRC=>CRC);
      Temp.VarWord:= CRC;
      TxData[j]:= Temp.ArrByte[0];
      TxData[j+1]:= Temp.ArrByte[1];
      TxLen:= j+2;
 ELSE                                        // нет кода в списке
      TxData[0]:= RxData[0];
      TxData[1]:= RxData[1]+128; // по протоколу  в случае
ошибки старший бит = 1 (вес 8-го бита 128)
      TxData[2]:= 1;  // при неверной функции код ошибки =1
      FunCRC(INPUT:=TxData, size:=3, CRC=>CRC);
      Temp.VarWord:= CRC;
      TxData[3]:= Temp.ArrByte[0];
      TxData[4]:= Temp.ArrByte[1];
      TxLen:= 5;
END_CASE
```

Далее пишем основной код настройки порта и приема/передачи данных по нему:

```
PROGRAM PLC_PRG
VAR
      StartPort: BOOL := FALSE;
```

```
        ComSet: COM_Settings;              // Настройки порта
        COM_P1: RTS_IEC_HANDLE;            // Указатель на порт
        F_Result: RTS_IEC_RESULT;              //     Результат
    функций
        // буферы чтения, ответа и временный
        RxData, TxData, TempData: ARRAY [0..250] OF BYTE;
        RxLen: UDINT := 8; //Количество байт в посылке при
    приеме
        TxLen: UDINT; //Количество байт в посылке при передаче
        HoldReg: ARRAY [0..99] OF WORD; // массив регистров
    хранения
        Proc: Processing;        //   основной функциональный
    блок
        SlaveID:   UINT:=1;   //адрес   подчиненного   (данного
    контроллера)
END_VAR
```

```
// Инициализация настроек порта
IF NOT StartPort THEN
      // Режим паритета - без него
      ComSet.byParity := SYS_NOPARITY;
      // Число стоп-бит - 1 стоп          бит
      ComSet.byStopBits := SYS_ONESTOPBIT;
      // Номер порта                  (1..4) - 4й порт
      ComSet.sPort := SYS_COMPORT1;
      // Скорость 115200  ComSet.ulBufferSize := 0;
      ComSet.ulBaudrate := SYS_BR_115200;
      // Таймаут приёма и отправки в миллисекундах
      ComSet.ulTimeout := 100;

      // Открываем порт с этими настройками
      COM_P1     :=     SysCom.SysComOpen2(ADR(ComSet),     0,
    ADR(F_Result));
      // Проверяем открылся ли
      IF F_Result = CmpErrors.Errors.ERR_OK THEN
            // Если порт открылся успешно, переход в основной
    цикл
            StartPort:= TRUE;
      END_IF
END_IF

// Основной цикл работы COM порта
IF  StartPort THEN
      SysCom.SysComRead(COM_P1,  ADR(TempData),  RxLen,  100,
    F_Result);
      IF TempData[0]= SlaveID THEN      //проверка SlaveID
            RxData:=TempData;
            SysComPurge(COM_P1);         //очистка кольцевого
    буфера
            Proc(RxData:=    RxData,    HoldReg:=    HoldReg,
    TxData=>TxData, TxLen=>TxLen); //формирование ответа
```

```
                SysCom.SysComWrite(COM_P1,   ADR(TxData),   TxLen,
        100, F_Result);       // передача в порт сформированного
        ответа
          END_IF
    END_IF
```

В рассмотренном примере обязательно следить за тем, чтобы не вылезти за диапазон массивов регистров хранения (запрашивать регистры от 0 до 99). Так же важно заметить, что размеры массивов заведомо больше чем принятое и передаваемое количество байтов информации. Прием команды чтения регистров хранения – 8 байтов: 1байт – адрес ведомого; 1 байт – код функции; 2 байта – адрес начального регистра; 2 байта – количество запрашиваемых регистров; 2 байта – контрольная сумма CRC. Передача информации: 1байт – адрес ведомого; 1 байт – код функции; 1 байт – количество байтов передаваемой информации + количество передаваемых регистров умноженное на 2 + контрольная сумма CRC. Что в сумме не должно превышать 205 байтов информации.

Проверить выше описанный код можно любым ведущим устройством. В качестве ведущего может выступать ПК с программой Modbus Poll (рисунок 118) или другой контроллер.

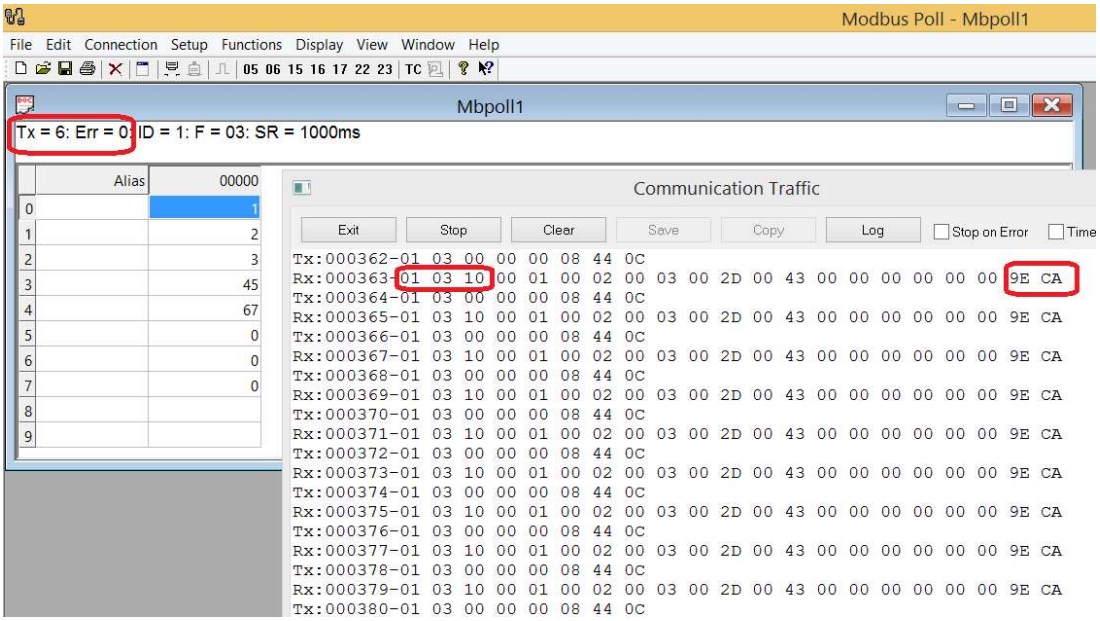


Рисунок 118 – Общий вид ПО Modbus Poll с открытой вкладкой Communication Traffic

На рисунке выделено количество переданных пакетов (Tx=6), количество ошибок (Err=0), адрес ведомого устройства, код функции, количество байтов информации в ответе и контрольная сумма, принятых от ведомого устройства.

На рисунке 119 показан вариант запроса входных регистров (код функции 04).

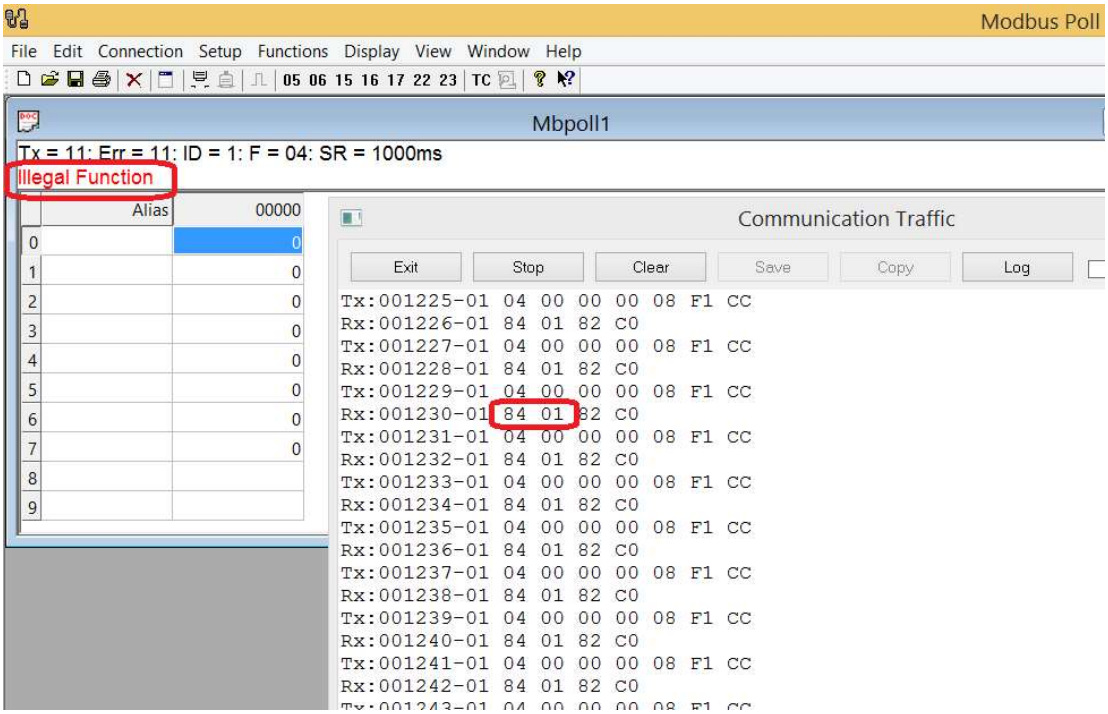


Рисунок 119 – Общий вид ПО Modbus Poll с открытой вкладкой Communication Traffic при использовании функции, не поддерживаемой ведомым устройством

Важно!!! Данный код приведен исключительно в качестве примера для понимания принципов работы и области применения библиотеки. В данном примере важно следить за тем, чтобы не вылезти за диапазон массивов, что может произойти также при приеме искаженных данных. Для использования в реальных проектах обязательна доработка – обработка ошибок при приеме (проверка контрольной суммы CRC, проверка соответствие запрашиваемых регистров массиву HoldReg и т.д.).

Использование данной библиотеки может найти применение так же для произвольного протокола последовательной связи.

3.7 Работа с Ethernet портом с помощью библиотеки SysSocket

Со́кет (англ. socket — разъём) — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одном устройстве, так и на различных, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения. Следует различать клиентские и серверные сокеты. Сокет Клиента может работать только с одним сокетом Сервера. Сокет Сервера может работать сразу с несколькими сокетами Клиентов.

Для взаимодействия между устройствами с помощью стека протоколов TCP/IP используются адреса и порты. Адрес представляет собой 32-битную структуру для протокола IPv4. Номер порта — целое число в диапазоне от 0 до 65535 (для протокола TCP). Эта пара определяет сокет («гнездо», соответствующее адресу и порту).

Каждый процесс (в нашем случае задача) может создать «слушающий» сокет (серверный сокет) и привязать его к какому-нибудь физическому порту устройства (в нашем случае единственный порт Ethernet).

Слушающий процесс обычно находится в цикле ожидания, то есть просыпается при появлении нового соединения.

Обычно клиент явно «подсоединяется» к слушателю, после чего любое чтение или запись через его файловый дескриптор будут передавать данные между ним и сервером.

Для работы с портом Ethernet необходимо добавить следующие библиотеки в проект CODESYS: SysSocket; SysTypes; CmpErrors. Для тестирования можно воспользоваться ПО SocketTest v 3.0.

В общем случае структурная схема работы Клиент – Сервера с помощью библиотеки SysSocket выглядит следующим образом (рисунок 120)

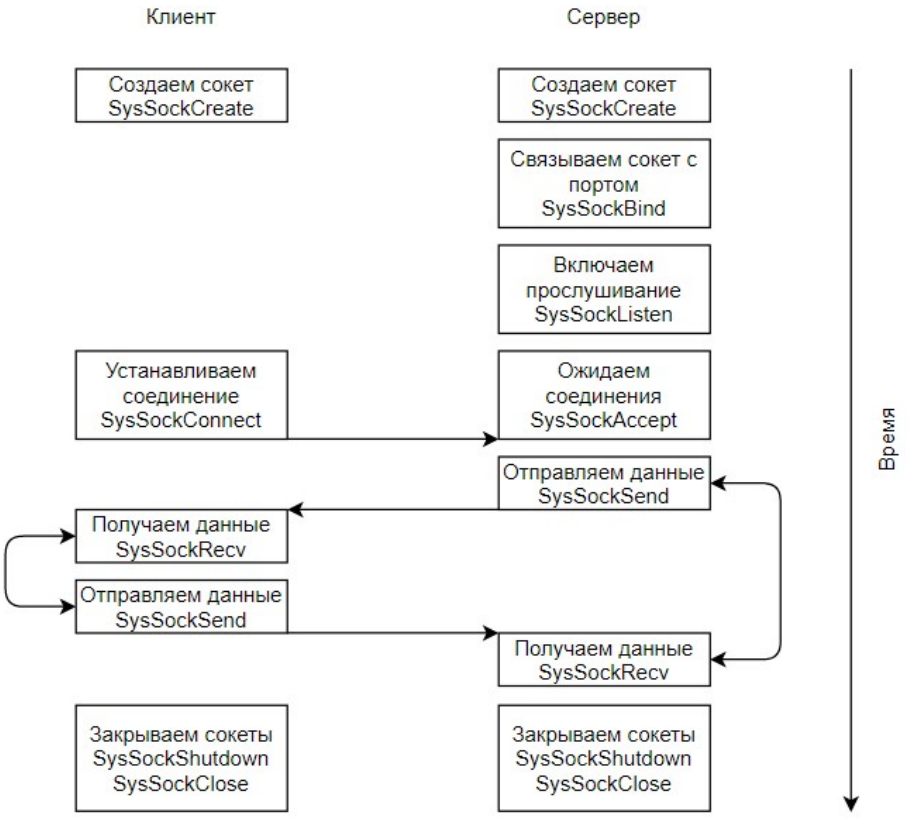


Рисунок 120 – Структурная схема работы Клиент – Сервера с помощью библиотеки SysSocket

3.7.1 Работа Сервера

Первое что понадобится это создать сокет функцией SysSockCreate. Пример:

```
Global.hServSoc := SysSockCreate(SOCKET_AF_INET,
SOCKET_STREAM, SOCKET_IPPROTO_TCP, ADR(F_Result));
```

Важно!!! Обратит внимание, что дескриптор `hServSoc` **объявляется в глобальных переменных Global RETAIN блока. При таком подходе дескриптор к созданному серверу не будет теряться в процессе изменения проекта, а так же при неожиданной перезагрузке контроллера. Так же в RETAIN блок лучше помещать все переменные, которые должны сохранять свои значения (дескриптор клиента, этап выполнения программы).**

Функция создает новый сокет и возвращает указатель (дескриптор) на него (таблица 26).

Таблица 26 – Описание функции SysSockCreate

Область	Имя	Тип	Комментарий
Return	SysSockCreate	RTS_IEC_HANDLE	Функция возвращает дескриптор (указатель) созданного сокета, который требуется в качестве входного параметра для других функций библиотеки, таких как SysSockBind, SysSockConnect и т. д.
Input	AddressFamily	INT	Семейство адресов сокетов
	Type	DINT	Тип сокета
	Protocol	DINT	Протокол сокета
	Result	POINTER TO RTS_IEC_RESULT	Указатель на код ошибки системы Runtime (см. VAR_GLOBAL CONSTANT, Errors библиотеки CmpErrors)

Возможные значения семейства адресов сокетов можно посмотреть в списке глобальных констант GVL библиотеки SysSocket (таблица 27).

Таблица 27 – Перечисления семейства адресов сокетов

Имя	Семейство адресов сокета
SOCKET_AF_UNSPEC	Неопределенное
SOCKET_AF_LOCAL	Для локального хоста (LocalHost – 127.0.0.1)
SOCKET_AF_UNIX	Обратной совместимости
SOCKET_AF_INET	DINTernetwork: UDP, TCP и т.д.
SOCKET_AF_IMPLINK	ARPANET протокола IMP
SOCKET_AF_PUP	PUP протоколов, таких как BSP
SOCKET_AF_CHAOS	CHAOS протоколов MIT
SOCKET_AF_NS	NS протоколов XEROX
SOCKET_AF_ISO	ISO протоколов
SOCKET_AF_OSI	OSI протоколов
SOCKET_AF_ECMA	Европейских производителей компьютеров
SOCKET_AF_DATAKIT	Datakit протоколов

SOCKET_AF_CCITT	CCITT протоколов, X.25 и подобные
SOCKET_AF_SNA	SNA протоколов IBM
SOCKET_AF_DECnet	DECnet протоколов
SOCKET_AF_DLI	интерфейса прямого канала передачи данных (DEC Direct data link DINTerface)
SOCKET_AF_LAT	LAT
SOCKET_AF_HYLINK	NSC Hyperchannel
SOCKET_AF_APPLETALK	AppleTalk
SOCKET_AF_ROUTE	Routing протоколов
SOCKET_AF_LINK	Интерфейсов канального уровня
SOCKET_pseudo_AF_XTP	eXpress Transfer Protocol (без AF)
SOCKET_AF_COIP	IP с установлением соединения, также известный как ST II
SOCKET_AF_CNT	Computer Network Technology
SOCKET_pseudo_AF_RTIP	Вспомогательных идентификаторов RTIP пакетов
SOCKET_AF_IPX	Novell протокол на IP
SOCKET_AF_SIP	Simple Internet Pritocol
SOCKET_pseudo_AF_PIP	Вспомогательных идентификаторов PIP пакетов
SOCKET_AF_MAX	Максимального разрешения
SOCKET_AF_INET_BSD	специфичных для BSD INET AF
SOCKET_AF_INET_STREAMS	специфичных для STREAMS INET AF

Возможные значения типа сокета можно посмотреть в списке глобальных констант GVL библиотеки SysSocket (таблица 28).

Таблица 28 – Перечисления типа сокета

Имя	Тип сокета
SOCKET_STREAM	Сокет потока
SOCKET_DGRAM	Сокет дейтаграмм
SOCKET_RAW	Сокет низкого уровня
SOCKET_RDM	Сокет надежно доставленного сообщения
SOCKET_SEQPACKET	Сокет пакетов

Возможные значения протокола сокета можно посмотреть в списке глобальных констант GVL библиотеки SysSocket (таблица 29).

Таблица 29 – Перечисления протокола сокета

Имя	Протокол сокета
SOCKET_IPPROTO_IP	IP уровень
SOCKET_IPPROTO_ICMP	Протокол межсетевых управляющих сообщений
SOCKET_IPPROTO_IGMP	Протокол управления групповой передачей данных
SOCKET_IPPROTO_GGP	Протокол межшлюзового взаимодействия (не рекомендуется!!!)
SOCKET_IPPROTO_TCP	Протокол управления передачей
SOCKET_IPPROTO_PUP	Протокол универсальных пакетов
SOCKET_IPPROTO_UDP	Протокол пользовательских дейтаграмм
SOCKET_IPPROTO_IDP	Протокол межсетевых дейтаграмм
SOCKET_IPPROTO_ND	НЕОФИЦИАЛЬНЫЙ протокол сетевого диска
SOCKET_IPPROTO_TLS	НЕОФИЦИАЛЬНЫЙ протокол защиты транспортного

	уровня
SOCKET_IPPROTO_RAW	Протокол сетевой печати
SOCKET_IPPROTO_MAX	Протокол максимального разрешения

Далее необходимо настроить адресацию сокета и привязать настройки к созданному сокету функцией SysSockBind. Пример:

```
//настройка адресации сокета и привязка их к сокету
SocAddrServer.sin_family      := SOCKET_AF_INET;
SocAddrServer.sin_port             := SysSockHtons(Port);
SocAddrServer.sin_addr.ulAddr  := SOCKET_INADDR_ANY;
F_Result  := SysSockBind(Global.hServSoc, ADR(SocAddrServer),
SIZEOF(SOCKADDRESS));
```

Тип переменной `SocAddrServer` является SOCKADDRESS. Структура SOCKADDRESS представлена в таблице 30.

Таблица 30 – Структура SOCKADDRESS

Имя	Тип	Комментарий
sin_family	INT	Семейство входящих адресов
sin_port	UINT	Идентификационный номер порта. **Должен быть преобразован в порядок соответствующему порядку шины с помощью SysSockHtons()!**
sin_addr	INADDR	IP адрес входящего соединения. Является объединением и содержит IP-адрес в трех разных форматах: 1) S_un_b (тип: UDINT_IN_BYTES) – Адрес для побайтового доступа 2) S_un_w (тип: UDINT_IN_WORDS) – Адрес для пословного доступа 3) ulAddr (тип: UDINT) – IP – адрес в целочисленном виде
sin_zero	ARRAY [0..7] OF BYTE	Рудимент (по причинам совмещения).

В данном примере `SocAddrServer.sin_addr.ulAddr` равен `SOCKET_INADDR_ANY`, что указывает серверу принимать подключения от любого клиента. В таблице 31 перечислены другие возможные варианты.

Таблица 31 – Перечисления sin_addr

Имя	Комментарий
SOCKET_INADDR_ANY	Любой адрес
SOCKET_INADDR_LOOPBACK	Устройство обратной петли
SOCKET_INADDR_BROADCAST	Широковещательная передача
SOCKET_INADDR_NONE	Без указания

Для указания конкретного адреса необходимо с помощью строковой переменной (`strIP_Addres :STRING (16)`) указать этот адрес в привычном формате (например – 192.168.0.100), а затем с помощью функции SysSockInetAddr преобразовать в формат UDINT необходимый переменной `sin_addr.ulAddr`. Пример использования функции:

```
F_Result:=                    SysSockInetAddr(strIP_Addres,
ADR(SocAddrServer.sin_addr.ulAddr));
```

SysSockBind вызывает функцию операционной системы, которая связывает локальный адрес (`SocAddrServer`)) и сокет (`Global.ServerSocket`), который уже был создан с помощью SysSockCreate. Это делается до вызова таких функций, как SysSockListen или SysSockAccept (таблица 31).

Таблица 31 – Описание функции SysSockBind

Область	Имя	Тип	Комментарий
Return	SysSockBind	RTS_IEC_RESULT	Код ошибки системы Runtime (см. VAR_GLOBAL CONSTANT, Errors библиотеки CmpErrors)
Input	hSocket	RTS_IEC_HANDLE	Указатель (дескриптор) сокета
	pSockAddr	POINTER TO SOCKADDRESS	Указатель к локальному адресу с настройками (SOCKADDRESS)
	diSockAddrSize	DINT	Длина структуры SOCKADDRESS

Далее запускаем прослушивание сокета для определения попытки подключения к серверу с помощью функции SysSockListen (таблица 32).

Таблица 32 – Описание функции SysSockListen

Область	Имя	Тип	Комментарий
Return	SysSockListen	RTS_IEC_RESULT	Код ошибки системы Runtime (см. VAR_GLOBAL CONSTANT, Errors библиотеки CmpErrors)
Input	hSocket	RTS_IEC_HANDLE	Указатель (дескриптор) сокета
	diMaxConnections	DINT	Максимальное разрешенное количество подключений

Пример использования функции SysSockListen:

```
F_Result := SysSockListen(Global.hServSoc, maxConn);
```

Где переменная `maxConn` целочисленного типа указывает максимальное количество клиентов в очереди.

После этого принимаем следующее входящее соединение от клиента функцией SysSockAccept (таблица 33).

Таблица 33 – Описание функции SysSockAccept

Область	Имя	Тип	Комментарий
Return	SysSockAccept	RTS_IEC_HANDLE	Указатель (дескриптор) сокета Клиента
Input	hSocket	RTS_IEC_HANDLE	Указатель (дескриптор) сокета Сервера
	pSockAddr	POINTER TO SOCKADDRESS	Указатель на структуру SOCKADDRESS Клиента
	pdiSockAddrSize	POINTER TO DINT	Указатель длины структуры SOCKADDRESS
	pResult	POINTER TO	Указатель на код ошибки системы

		RTS_IEC_RESULT	Runtime (см. VAR_GLOBAL CONSTANT, Errors библиотеки CmpErrors)

Пример использования функции SysSockAccept:

```
Global.hCliSoc := SysSockAccept(Global.hServSoc,
ADR(SocAddrClient), SIZEOF(SOCKADDRESS), ADR(F_Result));
```

Как только функцией SysSockAccept принято подключение от клиента, в зависимости от необходимой нам логики начинаем обмен данными между клиентом и сервером. В нашем случае передаем приветственное сообщение от сервера с помощью функции SysSockSend (таблица 34).

Таблица 34 – Описание функции SysSockSend

Область	Имя	Тип	Комментарий
Return	SysSockSend	__XINT	Количество переданных байтов. 0 при неудаче.
Input	hSocket	RTS_IEC_HANDLE	Указатель (дескриптор) сокета Клиента
	pbyBuffer	POINTER TO BYTE	Указатель к буферу содержащего сообщение для клиента
	diBufferSize	__XINT	Длина данных для передачи
	diFlags	DINT	Определяет способ вызова функции; зависит от опций сокета. Более подробно можно посмотреть в глобальных переменных библиотеки в категории TCP flags.
	pResult	POINTER TO RTS_IEC_RESULT	Указатель на код ошибки системы Runtime (см. VAR_GLOBAL CONSTANT, Errors библиотеки CmpErrors)

Пример использования:

```
NumBytesTx := SysSockSend(Global.hCliSoc, ADR(ServMessages),
LEN(ServMessages)+1, 0, ADR(F_Result));
```

Важно!!! Обратите внимание, что длина сообщения больше самого сообщения на 1 байт, для посылки ноль терминатора.

Далее в нашем случае принимаем данные функцией SysSockRecv (таблица 35).

Таблица 35 – Описание функции SysSockRecv

Область	Имя	Тип	Комментарий
Return	SysSockRecv	__XINT	Количество принятых байтов. 0 при неудаче.
Input	hSocket	RTS_IEC_HANDLE	Указатель (дескриптор) сокета Клиента
	pbyBuffer	POINTER TO BYTE	Указатель к буферу принимающему сообщения от клиента
	diBufferSize	__XINT	Максимальная длина принимаемых

			данных
	diFlags	DINT	Определяет способ вызова функции; зависит от опций сокета. Более подробно можно посмотреть в глобальных переменных библиотеки в категории TCP flags
	pResult	POINTER TO RTS_IEC_RESULT	Указатель на код ошибки системы Runtime (см. VAR_GLOBAL CONSTANT, Errors библиотеки CmpErrors)

Пример использования:

```
NumBytesRx:=  SysSockRecv(Global.hCliSoc,  ADR(CliMessages),
255, 0, ADR(F_Result));
```

В данном примере максимальная длина принимаемых данных, равна 255 байт, что соответствует максимальной длине строки STRING(255).

Далее если необходимо отключить и/или закрыть сокет клиента/сервера необходимо воспользоваться функциями SysSockShutdown (таблица 36) и SysSockClose (таблица 37).

Таблица 36 – Описание функции SysSockShutdown

Область	Имя	Тип	Комментарий
Return	SysSockShutdown	RTS_IEC_RESULT	Указатель на код ошибки системы Runtime (см. VAR_GLOBAL CONSTANT, Errors библиотеки CmpErrors)
Input	hSocket	RTS_IEC_HANDLE	Указатель (дескриптор) сокета который необходимо отключить
	diHow	DINT	Указывает, какие операции больше не разрешены. Более подробно можно посмотреть в глобальных переменных библиотеки в категории shutdown flags

Таблица 37 – Описание функции SysSockClose

Область	Имя	Тип	Комментарий
Return	SysSockClose	RTS_IEC_RESULT	Указатель на код ошибки системы Runtime (см. VAR_GLOBAL CONSTANT, Errors библиотеки CmpErrors)
Input	hSocket	RTS_IEC_HANDLE	Указатель (дескриптор) сокета который необходимо закрыть

Пример использования (отключение и закрытие сокета сервера):

```
SysSockShutdown(Global.hServSoc, SOCKET_SD_BOTH);
SysSockClose(Global.hServSoc);
```

Рассмотрим теперь пример написания кода сервера, который после подключения клиента отсылает ему приветственное сообщение «Hello From Server!!!», далее принимает сообщение от клиента, разрывает соединение и снова ждет подключение клиента.

Для начала напишем простейшую функцию обработки ошибок:

```
FUNCTION ErrPerf : STRING
VAR_INPUT
    Str: STRING;
    F_Result: RTS_IEC_RESULT;
END_VAR
VAR
END_VAR
```

```
IF F_Result <> CmpErrors.Errors.ERR_OK THEN
    ErrPerf := Concat(Str, TO_STRING(F_Result));
ELSE
    ErrPerf := 'Ok';
END_IF
```

Функция запрашивает название команды и результат ее выполнения. Если результат выполнения успешный возвращает строку «Ok». При ошибке возвращает название команды и код ошибки. Лучше всего вычислять еще текущее время (3.4 Работа с функциями времени контроллера КАПП2-00-000-1) и записывать результат с меткой времени в лог – файл на SD – карту контроллера (3.5 Работа с SD картой контроллера КАПП2-00-000-1).

В глобальных переменных сохраняемой области памяти помещаем переменные указатели (дескрипторы) сокетов клиента и сервера, а так же текущий шаг алгоритма. При таком подходе в процессе изменения алгоритмов с последующей загрузкой в контроллер эти переменные не будут теряться.

```
VAR_GLOBAL RETAIN
    hServSoc, hCliSoc: RTS_IEC_HANDLE;
    Step:INT;
END_VAR
```

Основной цикл программы:

```
PROGRAM PLC_PRG
VAR
    SocAddrServer, SocAddrClient: SOCKADDRESS;
    SocAddrSize: DINT;
    Port: WORD:= 1315;
    maxConn: UINT:=10;
    F_Result: RTS_IEC_RESULT := CmpErrors.Errors.ERR_FAILED;
    Err: STRING:='Ok';
    ServMessages: STRING:='Hello From Server!!!$R$N';
    RxData: ARRAY [0..254] OF BYTE;
    CliMessages: STRING(255);
    pStr:POINTER TO STRING;
    NumBytesTx, NumBytesRx: DINT;
    ResServ: BOOL:=FALSE;
```

```
        i: UINT;
    END_VAR
```

```
    CASE Global.Step OF
     0:
        Global.hServSoc:=           SysSockCreate(SOCKET_AF_INET,
        SOCKET_STREAM, SOCKET_IPPROTO_TCP, ADR(F_Result));
        Err:=   ErrPerf('SysSockCreate   Err:   ',F_Result);   //
        обработка ошибок
        IF F_Result = CmpErrors.Errors.ERR_OK THEN
            SocAddrServer.sin_family:= SOCKET_AF_INET;
            SocAddrServer.sin_port:= SysSockHtons(Port);
            SocAddrServer.sin_addr.ulAddr:= SOCKET_INADDR_ANY;
            F_Result:=SysSockBind(Global.hServSoc,
            ADR(SocAddrServer), SIZEOF(SOCKADDRESS));
            Err:=  ErrPerf('SysSockBind   Err:   ',F_Result);   //
            обработка ошибок
            IF F_Result = CmpErrors.Errors.ERR_OK THEN
                F_Result:=         SysSockListen(Global.hServSoc,
                maxConn);
                Err:= ErrPerf('SysSockListen Err: ',F_Result); //
                обработка ошибок
                IF F_Result = CmpErrors.Errors.ERR_OK THEN
                    Global.Step:= 1;
                END_IF
            END_IF
        ELSE
            Global.Step:= 5;
        END_IF

     1:
        Global.hCliSoc:=           SysSockAccept(Global.hServSoc,
        ADR(SocAddrClient), SIZEOF(SOCKADDRESS), ADR(F_Result));
        Err:=   ErrPerf('SysSockAccept   Err:   ',F_Result);   //
        обработка ошибок
        IF Global.hCliSoc <> RTS_INVALID_HANDLE AND F_Result =
        CmpErrors.Errors.ERR_OK THEN
            Global.Step := 2;
        ELSE
            Global.Step := 4;
        END_IF

     2:
        NumBytesTx         :=         SysSockSend(Global.hCliSoc,
        ADR(ServMessages),         LEN(ServMessages)+1,         0,
        ADR(F_Result));
        Err   :=   ErrPerf('SysSockSend   Err:   ',F_Result);   //
        обработка ошибок
        IF NumBytesTx <> 0 AND F_Result = CmpErrors.Errors.ERR_OK
        THEN
            Global.Step := 3;
```

```
        ELSE
            Global.Step := 4;
        END_IF

    3:
        FOR i:=0 TO 254 DO
            RxData[i]:=0;
        END_FOR
        NumBytesRx:=   SysSockRecv(Global.hCliSoc,   ADR(RxData),
        255, 0, ADR(F_Result));
        Err:= ErrPerf('SysSockRecv Err: ',F_Result); // обработка
        ошибок
        pStr := ADR(RxData);
        CliMessages := pStr^;
        Global.Step := 4;

    4:
        SysSockShutdown(Global.hCliSoc,    SOCKET_SD_BOTH);    //
        выключаем в ОС сокет, прием и передачу
        SysSockClose(Global.hCliSoc); // закрываем сокет
        IF ResServ THEN
            Global.Step := 5;
            ResServ:= false;
        ELSE
            Global.Step := 1;
        END_IF

    5:
        SysSockShutdown(Global.hServSoc,    SOCKET_SD_BOTH);    //
        выключаем в ОС сокет, прием и передачу
        SysSockClose(Global.hServSoc); // закрываем сокет
        Global.Step := 0;

    ELSE
        Global.Step   :=   0;   //   если   шаг   Global.Step   принял
        ошибочное значение, обнуляем его и переходим к шагу 0
    END_CASE
```

В приведенном примере весь цикл программы разбит на шесть шагов. На нулевом шаге (`Global.Step=0`) создается и настраивается сокет сервера, а так же включается прослушивание входящих соединений. Этот шаг выполняется только 1 раз при включении или при получении команды сброса сервера (`ResServ = TRUE`). Шаги с первого по четвертый являются основным циклом. На первом шаге принимается входящее соединение, на втором шаге отсылается приветственное сообщение клиенту, на третьем шаге принимается от него входящее сообщение и наконец, на четвёртом шаге выключается и закрывается сокет клиента. Далее цикл начинается заново с первого шага, при условии, отсутствия команды на сброс сервера. Если была команда на сброс сервера, выполняется шаг пятый, в котором закрывается сокет сервера и далее программа выполняется с нулевого шага.

Важно!!! Приемный буфер (`RxData`) представляет из себя массив байтов (`ARRAY [0..254] OF BYTE`). Перед приемом данных массив обнуляется, после чего по указателю данные переносятся в строку (`CliMessages`). Размер буфера и размер строки совпадают.

Обратите внимание, что в конце приветственного сообщения сервера содержатся символы перевода строки (`$R`) и возврата каретки (`$N`), для более наглядного отображения обмена в диалоговом окне клиента.

В качестве клиента для примера, можно использовать программное обеспечение SocketTest (рисунок 121).

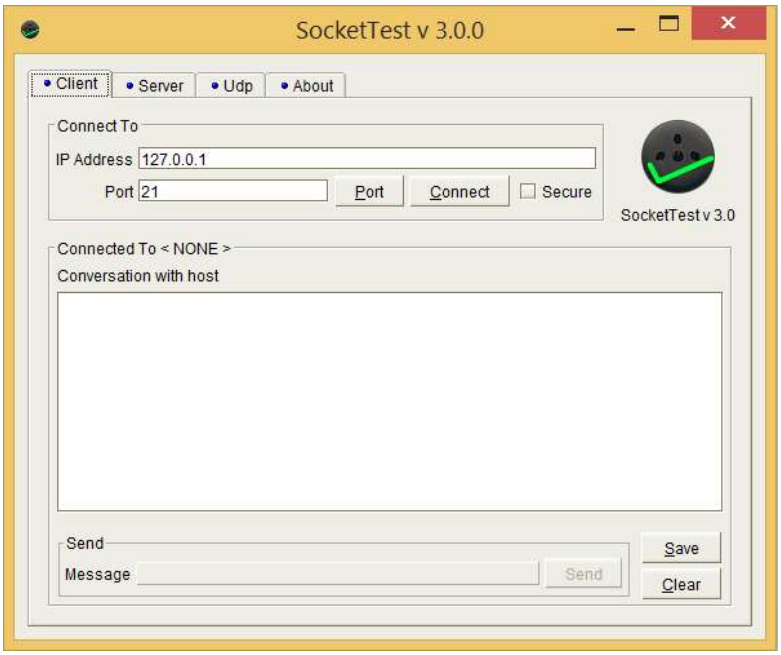


Рисунок 121 – Общий вид ПО SocketTest, вкладка «Клиент»

Для его использования в качестве клиента достаточно указать в поле IP Address вместо 127.0.0.1 адрес сервера (в нашем случае 192.168.20.222), а в поле Port – порт сервера (в нашем случае 1315). После этого необходимо нажать кнопку «Connect» (Подключить) (рисунок 122).

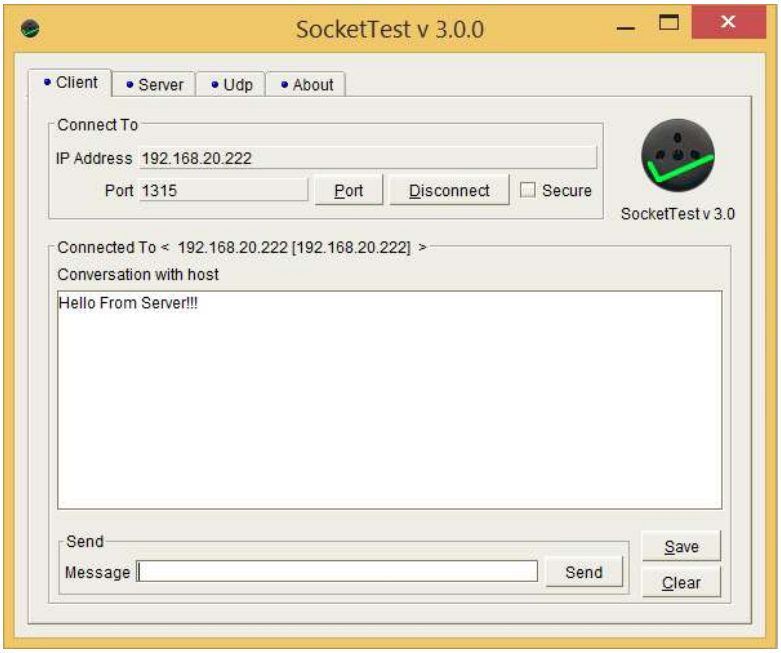


Рисунок 122 – Общий вид ПО SocketTest после подключения к серверу

Обратите внимание в поле Connected To (Подключено к), появился IP – адрес сервера. В диалоговое окно Conversation with host (Диалог с сервером) пришло сообщение от сервера «Hello From Server!!!». Для передачи сообщения серверу необходимо набрать его в поле Message (Сообщение) и нажать кнопку «Send» (Отправить). После этого сервер разорвет соединение согласно алгоритму.

3.7.1 Работа Клиента

Работа клиента несколько отличается от работы сервера. Так же как и в случае сервера, клиенту необходимо создать сокет командой SysSockCreate. Далее указать параметры сервера, к которому необходимо подключиться. Для определения IP – адрес сервера в формате UDINT структуры SOCKADDRESS, можно воспользоваться функцией SysSockInetAddr (таблица 38).

Таблица 38 – Описание функции SysSockInetAddr

Область	Имя	Тип	Комментарий
Return	SysSockInetAddr	RTS_IEC_RESULT	Указатель на код ошибки системы Runtime (см. VAR_GLOBAL CONSTANT, Errors библиотеки CmpErrors)
Input	szIPAddress	REFERENCE TO STRING	IP – адрес в формате STRING, который требуется преобразовать
	pInAddr	POINTER TO UDINT	Указатель к переменной типа UDINT (преобразованный IP – адрес для структуры SOCKADDRESS)

Пример использования:

```
F_Result:= SysSockInetAddr(strIPadrServ,
ADR(SocAddrServer.sin_addr.ulAddr));
```

Далее необходимо соединиться с сервером с помощью функции SysSockConnect (таблица 39).

Таблица 39 – Описание функции SysSockConnect

Область	Имя	Тип	Комментарий
Return	SysSockConnect	RTS_IEC_RESULT	Указатель на код ошибки системы Runtime (см. VAR_GLOBAL CONSTANT, Errors библиотеки CmpErrors)
Input	hSocket	RTS_IEC_HANDLE	Указатель (дескриптор) на сокет созданный на стороне клиента
	pSockAddr	POINTER TO SOCKADDRESS	Указатель к структуре SOCKADDRESS с настройками сервера
	diSockAddrSize	DINT	Размер структуры SOCKADDRESS сервера

Пример использования:

```
F_Result:= SysSockConnect(Global.hCliSoc, ADR(SocAddrServer),
SIZEOF(SocAddrServer));
```

Обмен информацией между клиентом и сервером осуществляется так же, как и в случае сервера, с помощью функций SysSockRecv и SysSockSend.

Отключение и закрытие сокета клиента осуществляется аналогично серверу, с помощью функций SysSockShutdown и SysSockClose.

Напишем простейший код клиента. Для простоты понимания обойдемся без функции проверки ошибок. Клиент будет подключаться к серверу, принимать от него данные и выдавать ответное сообщение содержащее строку «Принятое сообщение от сервера: » и принятые данные. В случае разрыва соединения клиент будет закрывать сокет, создавать его заново и пытаться подключатся к серверу.

Аналогично серверу переменные дескриптор сокета (`hCliSoc`) и шаг (`Step`) поместим в список глобальных переменных сохраняемой области памяти:

```
VAR_GLOBAL RETAIN
    hCliSoc: RTS_IEC_HANDLE;
    Step:INT;
END_VAR
```

Основной цикл клиента:

```
PROGRAM PLC_PRG
VAR
    SocAddrServer: SOCKADDRESS;
    Port: WORD:= 1315;
    strIPadrServ: STRING:='192.168.20.130';
    F_Result: RTS_IEC_RESULT := CmpErrors.Errors.ERR_FAILED;
    CliMessages: STRING;
    RxData: ARRAY [0..254] OF BYTE;
    ServMessages: STRING(255);
    pStr:POINTER TO STRING;
    NumBytesTx, NumBytesRx: DINT;
    i: UINT;
END_VAR
```

```
CASE Global.Step OF

 0:
    Global.hCliSoc:=            SysSockCreate(SOCKET_AF_INET,
    SOCKET_STREAM, SOCKET_IPPROTO_TCP, ADR(F_Result));
    SocAddrServer.sin_family:= SOCKET_AF_INET;
    SocAddrServer.sin_port:= SysSockHtons(Port);
    F_Result:=                  SysSockInetAddr(strIPadrServ,
    ADR(SocAddrServer.sin_addr.ulAddr));
    F_Result:=                 SysSockConnect(Global.hCliSoc,
    ADR(SocAddrServer), SIZEOF(SocAddrServer));
    Global.Step := 1;

 1:
    FOR i:=0 TO 254 DO
        RxData[i]:=0;
    END_FOR
    NumBytesRx:=   SysSockRecv(Global.hCliSoc,   ADR(RxData),
    255, 0, ADR(F_Result));
    pStr := ADR(RxData);
```

```
        ServMessages := pStr^;
        IF NumBytesRx <> 0 AND F_Result = CmpErrors.Errors.ERR_OK
        THEN
            Global.Step := 2;
        ELSE
            Global.Step := 3;
        END_IF

    2:
        CliMessages:= CONCAT('Принятое сообщение от сервера: ',
        ServMessages);
        NumBytesTx:=                    SysSockSend(Global.hCliSoc,
        ADR(CliMessages), LEN(CliMessages)+1, 0, ADR(F_Result));
        Global.Step := 1;

    3:
        SysSockShutdown(Global.hCliSoc,   SOCKET_SD_BOTH);      //
        выключаем в ОС сокет, прием и передачу
        SysSockClose(Global.hCliSoc);         // закрываем сокет
        Global.Step := 0;

    ELSE
        Global.Step   :=   0;   //   если   шаг   Global.Step   принял
        ошибочное значение, обнуляем его и переходим к шагу 0

    END_CASE
```

В рассмотренном примере наше номер один проверяется количество принятых данных и успешность выполнения функции приема. В случае успеха далее выполняется второй шаг, иначе переходим к шагу номер 3, где сокет выключается и закрывается, а затем создается новый. Во втором шаге обрабатывается принятое сообщение, отправляется ответная посылка. После этого клиент снова ожидает данные от сервера.

В качестве сервера для примера, можно так же использовать программное обеспечение SocketTest (рисунок 123).

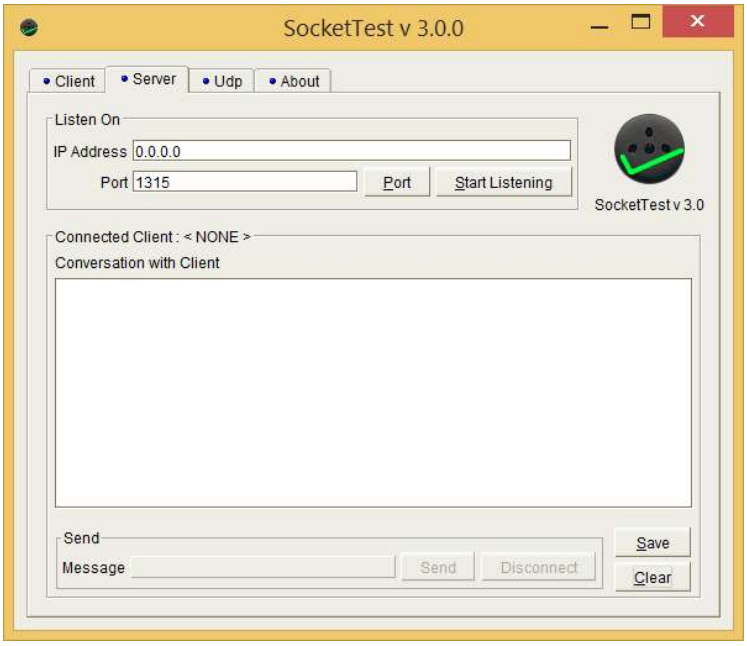


Рисунок 123 – Общий вид ПО SocketTest, вкладка «Сервер»

На данной вкладке необходимо указать порт сокета (в нашем случае 1315) и нажать кнопку «Start Listening» (Начать прослушивание). После запуска сервера и подключения к нему клиента в диалоговом окне появятся соответствующие сообщения (рисунок 124).

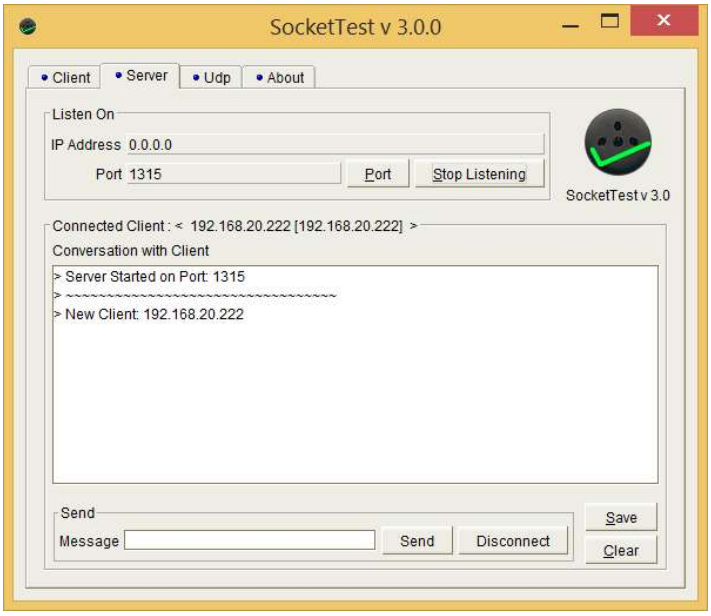


Рисунок 124 – Общий вид ПО SocketTest, вкладка «Сервер» после подключения клиента

Так же после подключения клиента разблокируется меню «Send» (Отправка сообщений). Для передачи сообщения клиенту необходимо набрать его в поле Message (Сообщение) и нажать кнопку «Send» (Отправить). После этого клиент, согласно алгоритму, обработает полученное сообщение и пришлет ответ содержащий строку «Принятое сообщение от сервера: » и принятые данные (рисунок 125).

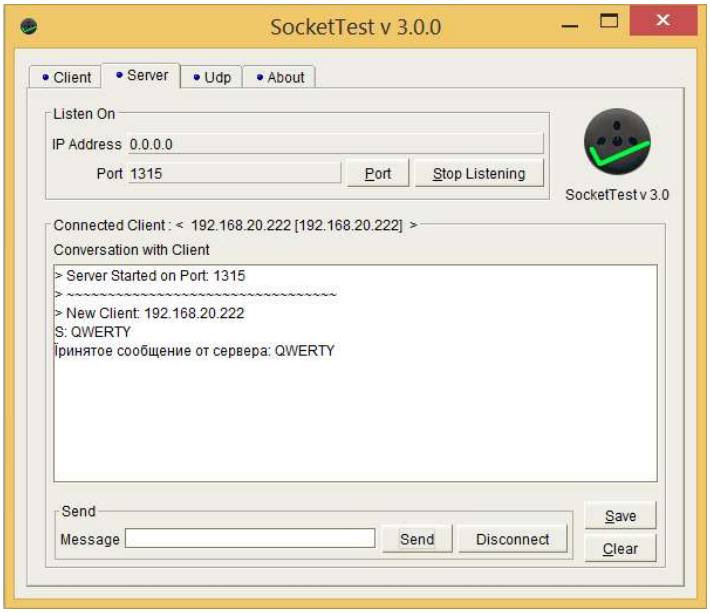


Рисунок 125 – Общий вид ПО SocketTest, вкладка «Сервер» после отправки сообщения «QWERTY» клиенту

3.8 Синхронизация часов контроллера по протоколу NTP

Компонент предназначен для синхронизации часов контроллера с сервером точного времени по протоколу NTP. Обновление времени происходит в автоматическом режиме в соответствии с заданными настройками. Сервер NTP должен работать на машине в локальной сети, к которой подключен контроллер.

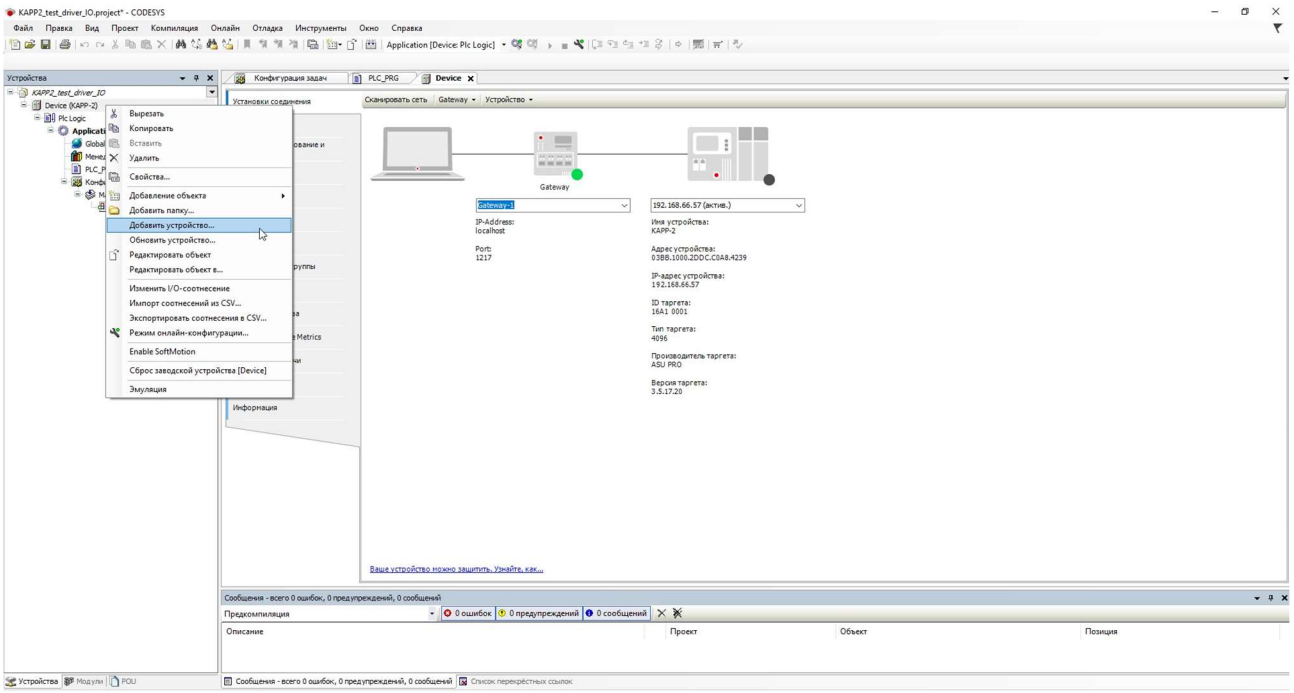
Установка компонента в среду CODESYS

Компонент устанавливается вместе с пакетом таргет-файлов для контроллера КАПП2, см. данное руководство п. 2.3.2 Установка пакета для программирования КАПП2.

Добавление компонента в проект

Для добавления компонента в проект CODESYS следует в дереве устройств открыть окно свойств узла Device(KAPP-2) и выбрать команду **Добавить устройство**. В появившемся окне следует открыть папку **Разн.** и выбрать компонент **NTP**, после чего нажать кнопку **Добавить устройство.**

В результате компонент будет добавлен в дерево проекта:



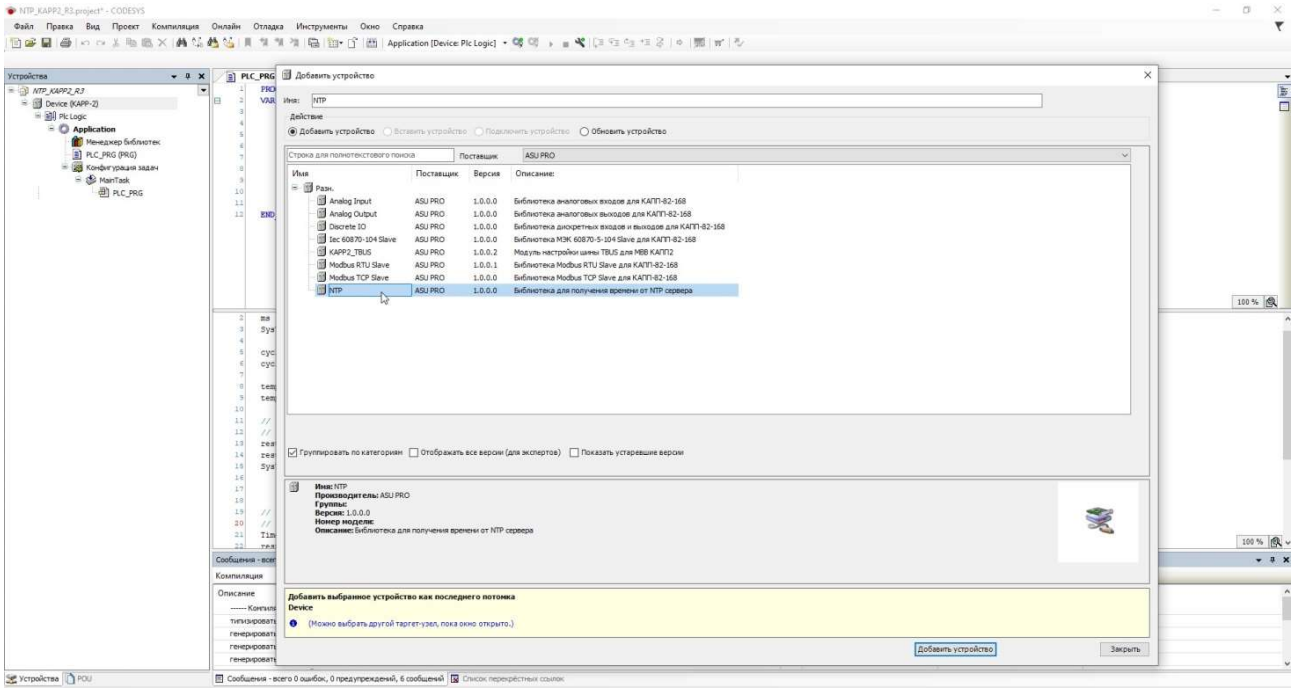


Рис 3.8.1 Добавление компонента NTP в дерево проекта

Удаление компонента из проекта

Для удаления любого модуля нужно открыть его свойства и в появившемся окне выбрать команду **Удалить.** После чего следует скомпилировать проект и загрузить его в контроллер.

Обновление компонента в проекте

В случае загрузки обновленной версии пакета таргет файлов в CODESYS, устройства в проекте следует обновить. Для этого нужно открыть окно свойств модуля и выбрать команду **Обновить устройство.**

Настройка компонента в проекте

Каждый компонент содержит вкладку **Конфигурация**. На вкладке **Конфигурация** задаются настройки компонента – IP адрес сервера, интервал обновления времени и т. д. При загрузке проекта заданные здесь настройки будут записаны в контроллер.

Описание каналов компонента

Для корректной работы компонента следует выполнить его настройку перед загрузкой проекта в контроллер. Для этого нужно открыть окно компонента на вкладке **Конфигурация** установить параметры:

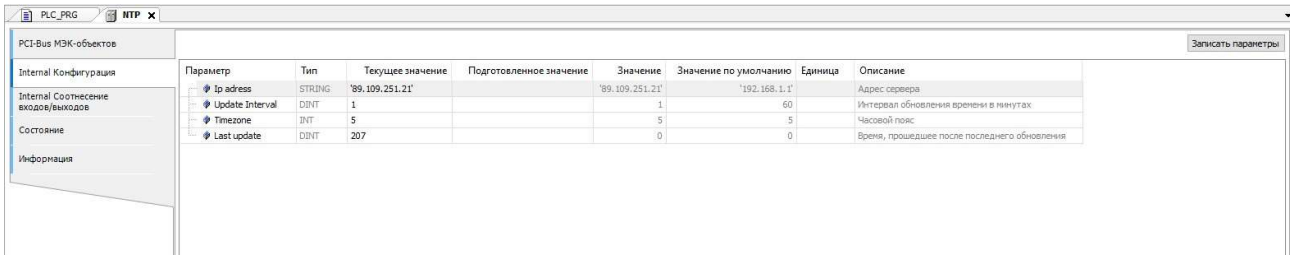


Рис. 3.8.2 Вкладка Конфигурация компонента NTP

Таблица 3.8 – Описание параметров компонента NTP

Канал	Тип	Описание
Вкладка **Конфигурация**		
IP address	STRING	IP адрес сервера NTP в локальной сети
Update Interval	DINT	Интервал обновления времени, в минутах
TimeZone	INT	Часовой пояс
Last update	DINT	Время, прошедшее после последнего обновления, мин

Инструкция по запуску NTP сервера в OS WINDOWS.

Для запуска NTP сервера на компьютере необходимо создать .bat файл со следующим содержимым:

```
:: Создаем копию раздела реестра
REG export
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time
"%~dp0W32Time.reg" /y
:: Остановить службу времени Windows
net stop w32time
:: Включить NTP сервер в реестре
REG ADD
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\W32Time\Tim
eProviders\NtpServer /v Enabled /d 1 /t REG_DWORD /f
:: Сделать сервер NTP всегда доверенным
REG ADD
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Con
fig /v AnnounceFlags /d 5 /t REG_DWORD /f
:: Настроить запуск службы времени при наличие сети
sc triggerinfo w32time start/networkon stop/networkoff
:: Запустить службу времени Windows
net start w32time
:: Разрешить исходящие подключения по udp 123 порт
netsh advfirewall firewall add rule name="udp 123 out" dir=out
protocol=udp localport=123 action=allow
:: Разрешить входящие подключения по udp 123 порт
netsh advfirewall firewall add rule name="udp 123 in" dir=in
protocol=udp localport=123 action=allow
:: Вывести текущий конфиг w32time
w32tm /query /configuration
@echo off
pause
```

Далее запустить этот файл от имени администратора. Файл вносит изменения в реестр и создает правила для порта UDP 123 в брандмауэр Windows.

Если на компьютере установлен антивирус, то необходимо создать правила для порта UDP 123 и в нем.

4 Техническое обслуживание

4.1 Общие указания

В целях обеспечения правильной и безопасной эксплуатации обслуживающий персонал должен пройти производственное обучение на рабочем месте. В процессе обучения персонал должен быть ознакомлен в объеме, необходимом для данной должности, с назначением, техническими данными, работой и устройством модуля, с порядком подготовки и включения модуля в работу и другими требованиями данного руководства.

4.2 Меры безопасности

По способу защиты от поражения электрическим током в соответствии с ГОСТ 12.2.007.0 модуль с номинальным напряжением питания 24 В постоянного тока относятся к классу III.

При эксплуатации, техническом обслуживании и поверке необходимо соблюдать требования ГОСТ 12.3.019-80, «Правил эксплуатации электроустановок потребителей» и «Правил охраны труда при эксплуатации электроустановок потребителей».

Любые подключения к модулю и работы по его техническому обслуживанию производятся только при отключенном питании модулю и подключенных к модулю устройств.

Не допускается работа модуля с открытым корпусом.

Подключение и техническое обслуживание модуля должны производиться только квалифицированными специалистами, изучившими настоящее руководство по эксплуатации.

При обнаружении неисправностей, необходимо отключить модуль от электрической сети и произвести замену прибора.

Запрещается эксплуатирование модуля с имеющимися неисправностями.

4.3 Порядок технического обслуживания изделия

Для обеспечения нормальной работы модуля рекомендуется выполнять в установленные сроки, следующие мероприятия:

В ПЕРИОД НАЛАДКИ

Проверять правильность функционирования модуля в составе средств управления по показаниям контрольно-измерительных приборов, фиксирующих протекание регулируемых технологических процессов, или с помощью SCADA систем.

ЕЖЕМЕСЯЧНО

– очищать корпус и клеммные колодки прибора от пыли, грязи и посторонних предметов;

– проверять качество крепления модуля на DIN-рейке;

– проверять качество подключения внешних связей.

Обнаруженные при осмотре недостатки следует немедленно устранить.

В ПЕРИОД КАПИТАЛЬНОГО РЕМОНТА ОБОРУДОВАНИЯ И ПОСЛЕ РЕМОНТА МОДУЛЯ

Производить проверку технического состояния и измерения параметров модуля в лабораторных условиях.

4.4 Консервация

Перед упаковыванием модуль должен пройти консервацию согласно требованиям ГОСТ 9.014-78.

Консервацию проводить по варианту защиты ВЗ-10. Вариант внутренней упаковки - ВУ-5.

Срок защиты без переконсервации – 2 года.

5 Хранение

Условия хранения модуля приведены в таблице 3.

Срок хранения в упаковке производителя - 2 года.

6 Транспортировка

Условия транспортировки модуля приведены в таблице 2.

Модуль, упакованный в транспортную тару, может транспортироваться железнодорожным транспортом без ограничения скорости и расстояния, автомобильным транспортом на расстоянии не более: 4000 км по шоссе; 1000 км по грунтовым дорогам; 300 км по бездорожью.

При транспортировке воздушным транспортом груз должен быть помещен в герметизированный отсек. Модули, упакованные в транспортную тару, должны храниться в отапливаемом или неотапливаемом помещении.

7 Утилизация

После вывода из эксплуатации и демонтажа, изделие подлежит ликвидации (в том числе утилизации и захоронению) в установленном порядке ГОСТ Р 52108-2003 «Ресурсосбережение. Обращение с отходами. Основные положения».

Образующиеся при ликвидации изделия отходы соответствуют 5 классу опасности. Особых требований к обращению с образовавшимися отходами не предъявляется.

8 Гарантийные обязательства

ООО «АСУ ПРО» (далее по тексту - Производитель) гарантирует работоспособность модуля и его качество (соответствие требованиям ТУ 26.20.30.000-020-73619730-2018) при соблюдении условий транспортирования, хранения, монтажа и эксплуатации, установленных настоящим руководством.

Гарантийный срок эксплуатации – 1 год с момента ввода модуля в эксплуатацию, но не более 2 лет с момента продажи.

Гарантийный срок хранения модуля в упаковке Производителя – 2 года.

В рамках настоящих гарантий Производитель обязуется осуществить ремонт во взаимосогласованные сроки любой и каждой неисправности оборудования, за исключением нижеуказанных случаев.

Производитель не несет гарантийных обязательств, если модуль:

- имеет механические повреждения;
- хранился или транспортировался с нарушением правил, указанных в настоящем руководстве или чётко оговорённых иным образом (в заключенном Договоре, технической документации и т.д.);
- поврежден в процессе установки (монтажа);

- модифицирован, изменен или восстановлен без письменного согласия Производителя;

- установлен или эксплуатируется с нарушением требований настоящего руководства;

- поврежден, изношен или разрушен из-за использования не по назначению или вследствие небрежного обращения во время эксплуатации;

- при эксплуатации модуля использовались некачественные и/или несоответствующие расходные материалы;

- утрачен или поврежден вследствие действий третьих лиц или в результате наступления обстоятельств непреодолимой силы.

Действие гарантийных обязательств Производителя распространяется на неисправности, установленные в течение гарантийного периода, если уведомление об этих неисправностях отправлено Потребителем Производителю в письменном виде в течение тридцати календарных дней с момента обнаружения предполагаемого дефекта. Датой подачи уведомления считается дата почтового отправления.

Для осуществления гарантийного ремонта или замены модуля в течение указанного выше гарантийного срока, Потребитель, после письменного уведомления Производителя, должен отправить модуль с паспортом и кратким описанием неисправности в офис Производителя в г. Оренбург, либо в другое, указанное Производителем место.

Адрес офиса Производителя:

460000, г. Оренбург, ул. Черепановых, д. 7, ООО «АСУ ПРО»

тел/факс: (3532) 68-90-88 доб. 155, +7 (800) 222-38-82, 1 доб. 155

e-mail: support@asupro.ru

По согласованию сторон, возможен гарантийный ремонт модуля на объекте. В этом случае Потребитель направляет письменный запрос Производителю на вызов специалиста. В запросе должен быть кратко описан предполагаемый дефект модуля для выявления причины дефекта и закупки необходимых запасных частей.

ПРИЛОЖЕНИЕ А

(Обязательное)

Общий вид модуля процессорного КАПП2-00-000-1

ПРИЛОЖЕНИЕ Б

(Обязательное)

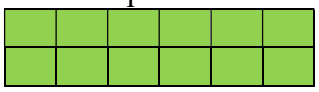
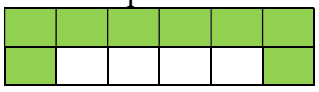
Передняя индикаторная панель КАПП2-00-000-1

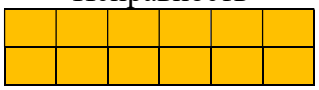
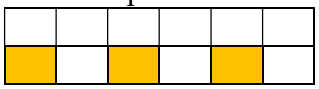
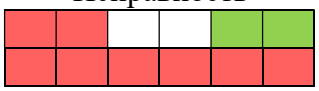
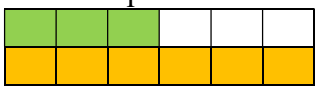
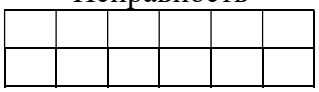
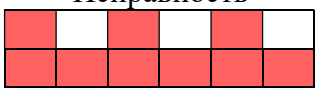
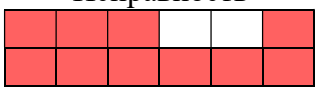
Передняя панель модуля КАПП2-00-000-1 состоит из световой индикации и кнопок управления, а также интерфейсных разъёмов. Кнопок управления – две. Интерфейсный разъёмы: RS232, RS485, Ethernet (10/100 BaseT).

Первая кнопка – «Сброс», предназначена для аппаратного сброса микроконтроллера, с последующей перезагрузкой программы пользователя. Вторая кнопка – «Старт / Стоп», используется для приостановки программы пользователя, а также ее обратного запуска, или, удаления программы пользователя – например, в том случае, если последняя содержит ошибки, которые не позволяют запустить ПЛК и получить к нему доступ. Для остановки программы пользователя необходимо нажать кнопку «Стар / Стоп» не менее половины секунды, после чего индикатор «Пуск» начнет быстро мигать зелёным. Для запуска программы пользователя необходимо выполнить тоже действие, индикатор в случае успешного запуска будет гореть зелёным постоянно.

Сигналы светодиодного индикатора КАПП2-00-000-1 (CODESYS). Индикация выполнена при помощи двух светодиодов, каждый из которых может гореть тремя цветами (красный, оранжевый, зелёный). Первый индикатор отображает исправность периферии, второй состояние программы пользователя (CODESYS).

Расшифровка световых кодов индикации

Сигнал	Описание
Исправность  Пуск	Штатный режим работы, ПЛК прошел тест, и запустил среду исполнения CODESYS, а также работает программа загружена успешно и работает. Индикаторы горят постоянно.
Исправность  Пуск	Штатный режим работы, ПЛК прошел тест, и запустил среду исполнения CODESYS, программа пользователя отсутствует. Индикатор «Пуск» редко мигает зелёным.
Исправность  Пуск	Штатный режим работы, ПЛК прошел тест, и запустил среду исполнения CODESYS, программа пользователя загружена, остановлена, пользователем или средой CODESYS. Индикатор «Пуск» часто мигает зелёным.
Исправность  Пуск	Штатный режим работы, ПЛК прошел тест, и запустил среду исполнения CODESYS, программа пользователя загружена, остановлена отладчиком среды CODESYS – Breakpoint. Индикатор «Пуск» часто мигает зелёным.

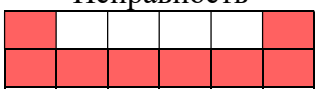
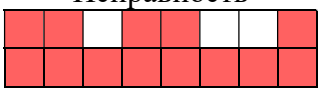
Исправность  Пуск	Загрузка ПЛК, в этот момент происходит базовое самотестирование аппаратных компонентов, загрузка системных модулей.
Исправность  Пуск	Загрузка пользовательской программы и среды исполнения CODESYS, аппаратные компоненты успешно запущены и проверены.
Исправность  Пуск	Работа загрузчика ПЛК, базовая инициализация. Индикатор «Пуск» часто мигает оранжевым, индикатор «Исправность» отключен.
Исправность  Пуск	Прошивка не удалась, аппаратный сбой во время обновления системной микропрограммы. Индикатор «Исправность» попеременно горит красным затем зелёным.
Исправность  Пуск	Идёт процесс обновления прошивки. Индикатор «Исправность» часто мигает зелёным.
Исправность  Пуск	После подачи питания более 5 секунд отсутствует индикация – необходимо проверить питание, если напряжение укладывается в диапазон 18..28В, то, это означает неисправность ПЛК. В этом случае необходимо обратиться в техническую поддержку.
Исправность  Пуск	Аппаратный сбой ПЛК, запуск и работа невозможны. Индикатор «Исправность» очень часто мигает красным.
Исправность  Пуск	Аппаратный сбой ПЛК, неисправность внешней ОЗУ. Индикатор «Исправность» мигает красным, длинный затем короткий сигналы.
	Аппаратный сбой ПЛК, Карта памяти microSD неисправна или

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

Исправность  Пуск	неисправен порт SDIO. Индикатор «Исправность» мигает красным, два редких коротких сигнала.
Исправность  Пуск	Аппаратный сбой ПЛК, Чип FRAM (Reatain RAM в CODESYS) неисправен. Индикатор «Исправность» мигает красным, два длинных и один короткий сигналы.

Важно!!! Для удаления программы пользователя необходимо, нажать кнопку «Сброс», после чего сразу нажать и удерживать кнопку «Старт / Стоп», до тех пор, пока индикатор «Пуск» не начнет мигать оранжевым, при этом индикатор «Исправность» будет постоянно гореть зелёным, после чего отпустить – всё, программа пользователя стёрта из памяти (microSD).

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

ПРИЛОЖЕНИЕ В

(Обязательное)

Функции чтения и записи в режиме ведомого

Работа с флагами(Coils)

FUNCTION **GetCoil** : BOOL – Получить значение флага

Область	Имя	Тип	Комментарий
Return	GetCoil	BOOL	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес флага
Output	result	ModbusError	Результат выполнения операции

FUNCTION **GetCoils** : ModbusError – Получить значения флагов

Область	Имя	Тип	Комментарий
Return	GetCoils	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес первого флага
	count	UINT	Количество флагов
	values	POINTER TO BOOL	Массив флагов, должен быть [0..count-1]

FUNCTION **SetCoil** : ModbusError – Установить значение флага

Область	Имя	Тип	Комментарий
Return	SetCoil	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес флага
	value	BOOL	Значение для установки

FUNCTION **SetCoils** : ModbusError – Установить значения флагов

Область	Имя	Тип	Комментарий
Return	SetCoils	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес флага
	count	UINT	Количество регистров
	values	POINTER TO BOOL	Массив флагов, должен быть [0..count-1]

Работа с дискретными входами (Discrete Inputs)

FUNCTION **GetDiscreteInput** : BOOL – Получить значение дискретного входа

Область	Имя	Тип	Комментарий
Return	GetDiscreteInput	BOOL	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес дискретного входа
Output	result	ModbusError	Результат выполнения операции

FUNCTION **GetDiscreteInputs** : ModbusError – Получить значения дискретных входов

Область	Имя	Тип	Комментарий
Return	GetDiscreteInputs	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес первого дискретного входа
	count	UINT	Количество дискретных входов
	values	POINTER TO BOOL	Массив дискретных входов, должен быть [0..count-1]

FUNCTION **SetDiscreteInput** : ModbusError – Установить значение дискретного входа

Область	Имя	Тип	Комментарий
Return	SetDiscreteInput	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес дискретного входа
	value	BOOL	Значение для установки

FUNCTION **SetDiscreteInputs** : ModbusError – Установить значения дискретных входов

Область	Имя	Тип	Комментарий
Return	SetDiscreteInputs	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес дискретного входа
	count	UINT	Количество регистров
	values	POINTER TO BOOL	Массив дискретных входов, должен быть [0..count-1]

Работа с регистрами хранения (Holding Registers)

FUNCTION **GetHoldingRegister** : UINT – Получить значение регистра хранения

Область	Имя	Тип	Комментарий

Return	GetHoldingRegister	UINT	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес регистра
Output	result	ModbusError	Результат выполнения операции

FUNCTION **GetHoldingRegisters** : ModbusError – Получить значения регистров хранения

Область	Имя	Тип	Комментарий
Return	GetHoldingRegisters	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес первого регистра
	count	UINT	Количество регистров
	values	POINTER TO UINT	Массив регистров хранения, должен быть [0..count-1]

FUNCTION **SetHoldingRegister** : ModbusError – Установить значение регистра хранения

Область	Имя	Тип	Комментарий
Return	SetHoldingRegister	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес регистра
	value	UINT	Значение для установки

FUNCTION **SetHoldingRegisters** : ModbusError – Установить значения регистров хранения

Область	Имя	Тип	Комментарий
Return	SetHoldingRegisters	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес регистра
	count	UINT	Количество регистров
	values	POINTER TO UINT	Массив регистров хранения, должен быть [0..count-1]

Работа с регистрами ввода (Input Registers)

FUNCTION **GetInputRegister** : UINT – Получить значение входного регистра

Область	Имя	Тип	Комментарий
Return	GetInputRegister	UINT	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес регистра
Output	result	ModbusError	Результат выполнения операции

FUNCTION **GetInputRegisters** : ModbusError – Получить значения входных регистров

Область	Имя	Тип	Комментарий
Return	GetInputRegisters	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес первого регистра
	count	UINT	Количество регистров
	values	POINTER TO UINT	Массив входных регистров, должен быть [0..count-1]

FUNCTION **SetInputRegister** : ModbusError – Установить значение входного регистра

Область	Имя	Тип	Комментарий
Return	SetInputRegister	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес регистра
	value	UINT	Значение для установки

FUNCTION **SetInputRegisters** : ModbusError – Установить значения входных регистров

Область	Имя	Тип	Комментарий
Return	SetInputRegisters	ModbusError	
Inout	mapping	ModbusMapping	Таблица регистров Modbus
Input	address	UINT	Адрес регистра
	count	UINT	Количество регистров
	values	POINTER TO UINT	Массив входных регистров, должен быть [0..count-1]

ПРИЛОЖЕНИЕ Г

(Обязательное)

Функции настройки портов в режиме ведущего

FUNCTION **ModbusNewRtu:** ModbusError – Создать контекст Modbus RTU

Область	Имя	Тип	Комментарий
Return	ModbusNewRtu	ModbusError	
Input	port	BYTE	Номер последовательного порта
	baudrate	UDINT	Скорость последовательного порта
	parity	ModbusParity	Чётность
	dataBits	ModbusDataBits	Количество битов данных
	stopBits	ModbusStopBits	Количество стоп-битов
Output	context	ModbusContext	Контекст библиотеки Modbus

FUNCTION **ModbusNewTcp:** ModbusError – Создать контекст Modbus TCP

Область	Имя	Тип	Комментарий
Return	ModbusNewTcp	ModbusError	
Input	ip	STRING	IP-адрес
	port	DINT	Порт
Output	context	ModbusContext	Контекст библиотеки Modbus

FUNCTION **ModbusConnect:** ModbusError – Установить соединение Modbus

Область	Имя	Тип	Комментарий
Return	ModbusConnect	ModbusError	
Inout	context	ModbusContext	Контекст библиотеки Modbus

FUNCTION **ModbusClose:** ModbusError – Закрыть подключение Modbus

Область	Имя	Тип	Комментарий
Return	ModbusConnect	ModbusError	
Inout	context	ModbusContext	Контекст библиотеки Modbus

FUNCTION **ModbusSetSlave:** ModbusError – Установить номер вторичного устройства Modbus (Slave ID)

Область	Имя	Тип	Комментарий
Return	ModbusSetSlave	ModbusError	
Input	slaveId	DINT	Номер вторичного устройства Modbus (Slave ID)
Inout	context	ModbusContext	Контекст библиотеки Modbus

FUNCTION **ModbusSetResponseTimeout:** ModbusError – Установить время таймаута ответа от сервера

Область	Имя	Тип	Комментарий
Return	ModbusSetResponseTimeout	ModbusError	
Input	sec	UDINT	Секунды
	usec	UDINT	Микросекунды
Inout	context	ModbusContext	Контекст библиотеки Modbus

ПРИЛОЖЕНИЕ Д

(Обязательное)

Функции чтения и записи в режиме ведущего

FUNCTION **ReadCoils** : ModbusError – Прочитать значения флагов

Область	Имя	Тип	Комментарий
Return	ReadCoils	ModbusError	
Inout	context	ModbusContext	Контекст библиотеки Modbus
Input	address	UDINT	Адрес первого регистра
	count	UDINT	Количество регистров для чтения
	buffer	UDINT	Буфер для приёма данных ADR (ARRAY OF SINT)

FUNCTION **ReadDiscreteInputs**: ModbusError – Прочитать значения дискретных входов

Область	Имя	Тип	Комментарий
Return	ReadDiscreteInputs	ModbusError	
Inout	context	ModbusContext	Контекст библиотеки Modbus
Input	address	UDINT	Адрес первого регистра
	count	UDINT	Количество регистров для чтения
	buffer	UDINT	Буфер для приёма данных ADR (ARRAY OF SINT)

FUNCTION **ReadHoldingRegisters**: ModbusError – Прочитать значения регистров хранения

Область	Имя	Тип	Комментарий
Return	ReadHoldingRegisters	ModbusError	
Inout	context	ModbusContext	Контекст библиотеки Modbus
Input	address	UDINT	Адрес первого регистра
	count	UDINT	Количество регистров для чтения
	buffer	UDINT	Буфер для приёма данных ADR (ARRAY OF UINT)

FUNCTION **ReadInputRegisters**: ModbusError – Прочитать значения входных регистров

Область	Имя	Тип	Комментарий
Return	ReadInputRegisters	ModbusError	
Inout	context	ModbusContext	Контекст библиотеки Modbus
Input	address	UDINT	Адрес первого регистра
	count	UDINT	Количество регистров для чтения
	buffer	UDINT	Буфер для приёма данных ADR(ARRAY OF UINT)

FUNCTION **WriteCoil:** ModbusError – Записать значение флага

Область	Имя	Тип	Комментарий
Return	WriteCoil	ModbusError	
Inout	context	ModbusContext	Контекст библиотеки Modbus
Input	address	UDINT	Адрес регистра
	value	UINT	Значение для записи

FUNCTION **WriteCoils:** ModbusError – Записать значения флагов

Область	Имя	Тип	Комментарий
Return	WriteCoils	ModbusError	
Inout	context	ModbusContext	Контекст библиотеки Modbus
Input	address	UDINT	Адрес первого регистра
	count	UDINT	Количество регистров для записи
	values	UDINT	Значения для записи ADR(ARRAY OF SINT)

FUNCTION **WriteHoldingRegister:** ModbusError – Записать значение регистра хранения

Область	Имя	Тип	Комментарий
Return	WriteHoldingRegister	ModbusError	
Inout	context	ModbusContext	Контекст библиотеки Modbus
Input	address	UDINT	Адрес регистра
	value	UINT	Значение для записи

FUNCTION **WriteHoldingRegisters:** ModbusError – Записать значения флагов

Область	Имя	Тип	Комментарий
Return	WriteHoldingRegisters	ModbusError	
Inout	context	ModbusContext	Контекст библиотеки Modbus
Input	address	UDINT	Адрес первого регистра
	count	UDINT	Количество регистров для записи
	values	UDINT	Значения для записи ADR(ARRAY OF UINT)

ПРИЛОЖЕНИЕ Е

(Обязательное)

Функции преобразования чисел с плавающей точкой для передачи по протоколу Modbus

FUNCTION **GetFloatABCD** : REAL – Преобразование двух регистров в число с плавающей точкой без перестановки

Область	Имя	Тип
Return	GetFloatABCD	REAL
Inout	values	ARRAY [0..1] OF UINT

FUNCTION **GetFloatBADC** : REAL – Преобразование двух регистров в число с плавающей точкой с перестановкой байтов

Область	Имя	Тип
Return	GetFloatBADC	REAL
Inout	values	ARRAY [0..1] OF UINT

FUNCTION **GetFloatCDAB** : REAL – Преобразование двух регистров в число с плавающей точкой с перестановкой слов

Область	Имя	Тип
Return	GetFloatCDAB	REAL
Inout	values	ARRAY [0..1] OF UINT

FUNCTION **GetFloatDCBA** : REAL – Преобразование двух регистров в число с плавающей точкой с перестановкой байтов и слов

Область	Имя	Тип
Return	GetFloatDCBA	REAL
Inout	values	ARRAY [0..1] OF UINT

FUNCTION **SetFloatABCD** : ModbusError – Преобразование числа с плавающей точкой в два регистра в без перестановки

Область	Имя	Тип
Return	SetFloatABCD	ModbusError
Input	value	REAL
Inout	registers	ARRAY [0..1] OF UINT

FUNCTION **SetFloatBADC** : ModbusError – Преобразование числа с плавающей точкой в два регистра с перестановкой байтов

Область	Имя	Тип
Return	SetFloatBADC	ModbusError

Input	value	REAL
Inout	registers	ARRAY [0..1] OF UINT

FUNCTION **SetFloatCDAB** : ModbusError – Преобразование числа с плавающей точкой в два регистра с перестановкой слов

Область	Имя	Тип
Return	SetFloatCDAB	ModbusError
Input	value	REAL
Inout	registers	ARRAY [0..1] OF UINT

FUNCTION **SetFloatDCBA** : ModbusError – Преобразование числа с плавающей точкой в два регистра с перестановкой байтов и слов

Область	Имя	Тип
Return	SetFloatDCBA	ModbusError
Input	value	REAL
Inout	registers	ARRAY [0..1] OF UINT

ПРИЛОЖЕНИЕ Ж

(Обязательное)

Формуляр согласования реализации протокола МЭК 60870-5-104

Выбранные параметры обозначаются в белых прямоугольниках следующим образом:

	Функция или ASDU не используется.
X	Функция или ASDU используется, как указано в стандарте (по умолчанию).
R	Функция или ASDU используется в обратном режиме.
B	Функция или ASDU используется в стандартном и обратном режимах.

Возможный выбор (пустой, X, R или B) определяется для каждого пункта или параметра. Черный прямоугольник указывает на то, что опция не может быть выбрана в стандарте МЭК 60870-5-104.

Б.1 Система или устройство

(Параметр, характерный для системы; указывает на определение системы или устройства, маркируя один из нижеследующих прямоугольников знаком "X")

	Определение системы.
	Определение контролирующей станции (Ведущий, Мастер).
X	Определение контролируемой станции (Ведомый, Слэйв).

Б.2 Конфигурация сети

(Параметр, характерный для сети; все используемые структуры должны маркироваться знаком "X").

X	Точка-точка	X	Магистральная
	~~Радиальная точка-точка~~		~~Многоточечная радиальная~~

Б.3 Физический уровень

(Параметр, характерный для сети; все используемые интерфейсы и скорости передачи данных маркируются знаком "X").

Скорости передачи (направление управления)

Несимметричные цепи обмена V.24 [3], V.28 [5]; стандартные		Несимметричные цепи обмена V.24 [1], V.28 [5]; Рекомендуются при скорости более 1200 бит/с		Симметричные цепи обмена X.24[6], X.27[7]			
	~~100 бит/с~~		~~2400 бит/с~~		~~2400 бит/с~~		~~56000 бит/~~
	~~200 бит/с~~		~~4800 бит/с~~		~~4900 бит/с~~		~~64000 бит/~~
	~~300 бит/с~~		~~9600 бит/с~~		~~9600 бит /с~~		
	~~600 бит/с~~				~~19200 бит/с~~		
	~~1200 бит/с~~				~~38400 бит/с~~		

Скорости передачи (направление контроля)

Несимметричные цепи обмена V.24 [3], V.28 [5]; стандартные		Несимметричные цепи обмена V.24 [1], V.28 [5]; Рекомендуются при скорости более 1200 бит/с		Симметричные цепи обмена X.24[6], X.27[7]			
	~~100 бит/с~~		~~2400 бит/с~~		~~2400 бит/с~~		~~56000 бит/~~
	~~200 бит/с~~		~~4800 бит/с~~		~~4900 бит/с~~		~~64000 бит/~~
	~~300 бит/с~~	X	9600 бит/с		~~9600 бит /с~~		
	~~600 бит/с~~	X	19200 бит/с		~~19200 бит/с~~		
	~~1200 бит/с~~	X	38400 бит/с		~~38400 бит/с~~		
		X	57600 бит/с				
		X	115200 бит/с				

Б.4 Канальный уровень

(Параметр, характерный для сети; все используемые опции маркируются знаком X.) Указывают максимальную длину кадра. Если применяется нестандартное назначение для сообщений класса 2 при небалансной передаче, то указывают Type ID (или Идентификаторы типа) и COT (Причины передачи) всех сообщений, приписанных классу 2.

~~В настоящем стандарте используются только формат кадра FT 1.2, управляющий символ 1 и фиксированный интервал времени ожидания.~~

Передача по каналу		Адресное поле канального уровня	
X	Балансная передача		~~Отсутствует (только при балансной передаче)~~
X	Небалансная передача		~~Один байт~~
~~Длина кадра~~			~~Два байта~~
	~~Максимальная длина L (число байтов)~~		~~Структурированное~~
		X	Неструктурированное

При использовании небалансного канального уровня следующие типы ASDU возвращаются при сообщениях класса 2 (низкий приоритет) с указанием причин передачи:

~~Стандартное назначение ASDU к сообщениям класса 2 используется следующим образом~~

ИДЕНТИФИКАТОР типа	Причина передачи
9,11,13,21	<1>

~~Специальное назначение ASDU к сообщениям класса 2 используется следующим образом:~~

ИДЕНТИФИКАТОР типа	Причина передачи

~~Примечание - При ответе на опрос данных класса 2 контролируемая станция может посылать в ответ данные класса 1, если нет доступных данных класса 2.~~

Б.5 Прикладной уровень

Режим передачи прикладных данных

В настоящем стандарте используется только режим 1 (первым передается младший байт), как определено в 4.10 ГОСТ Р МЭК 870-5-5.

Общий адрес ASDU

(Параметр, характерный для системы; все используемые варианты маркируются знаком X).

	~~Один байт~~	X	Два байта

Адрес объекта информации

(Параметр, характерный для системы; все используемые варианты маркируются знаком X).

	~~Один байт~~		Структурированный
	~~Два байта~~	X	Неструктурированный
X	Три байта		

Причина передачи

(Параметр, характерный для системы; все используемые варианты маркируются знаком X).

	~~Один байт~~	X	Два байта (с адресом источника). Если адрес источника не используется, то он устанавливается в 0.

Длина APDU

(Параметр, характерный для системы и устанавливающий максимальную длину APDU в системе).

Максимальная длина APDU равна 253 (по умолчанию). Максимальная длина может быть уменьшена для системы.

253	Максимальная длина APDU для системы.

Выбор стандартных ASDU

Информация о процессе в направлении контроля

(Параметр, характерный для станции; каждый Type ID маркируется знаком X, если используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

X	<1>	:= Одноэлементная информация	M_SP_NA_1
	~~<2>~~	~~:= Одноэлементная информация с меткой времени~~	~~M_SP_TA_1~~
	<3>	:= Двухэлементная информация	M_DP_NA_1

	~~<4>~~	~~:= Двухэлементная информация с меткой времени~~	~~M_DP_TA_1~~
	<5>	:= Информация о положении отпаек	M_ST_NA_1
	~~<6>~~	~~:= Информация о положении отпаек с меткой времени~~	~~M_ST_TA_1~~
	<7>	:= Строка из 32 бит	M_BO_NA_1
	~~<8>~~	~~:= Строка из 32 бит с меткой времени~~	~~M_BO_TA_1~~
	<9>	:= Значение измеряемой величины, нормализованное значение	M_ME_NA_1
	~~<10>~~	~~:= Значение измеряемой величины, нормализованное значение с меткой времени~~	~~M_ME_TA_1~~
	<11>	:= Значение измеряемой величины, масштабированное значение	M_ME_NB_1
	~~<12>~~	~~:= Значение измеряемой величины, масштабированное значение с меткой времени~~	~~M_ME_TB_1~~
X	<13>	::=Значение измеряемой величины, короткий формат с плавающей запятой	M_ME_NC_1
	~~<14>~~	~~:= Значение измеряемой величины, короткий формат с плавающей запятой с меткой времени~~	~~M_ME_TC_1~~
	<15>	:= Интегральные суммы	M_IT_NA_1
	~~<16>~~	~~:= Интегральные суммы с меткой времени~~	~~M_IT_TA_1~~
	~~<17>~~	~~:= Действие устройств защиты с меткой времени~~	~~M_EP_TA_1~~
	~~<18>~~	~~:= Упакованная информация о срабатывании пусковых органов защиты с меткой времени~~	~~M_EP_TB_1~~
	~~<19>~~	~~:= Упакованная информация о срабатывании выходных цепей устройства защиты с меткой времени~~	~~M_EP_TC_1~~
	<20>	:= Упакованная одноэлементная информация с определением изменения состояния	M_SP_NA_1
	<21>	:= Значение измеряемой величины, нормализованное значение без описателя качества	M_ME_ND_1
X	<30>	:= Одноэлементная информация с меткой времени CP56Время2a	M_SP TB_1
	<31>	:= Двухэлементная информация с меткой времени CP56Время2a	M_DP_TB_1
	<32>	:= Информация о положении отпаек с меткой времени CP56Время2a	M_ST_TB_1
	<33>	:= Строка из 32 бит с меткой времени CP56Время2a	M_BO_TB_1
	<34>	:= Значение измеряемой величины, нормализованное значение с меткой времениCP56Время2a	M_ME_TD_1
	<35>	:= Значение измеряемой величины, масштабированное значение с меткой времени CP56Время2a	M_ME_TE_1
X	<36>	:= Значение измеряемой величины, короткий формат с плавающей запятой с меткой времени CP56Время2a	M_ME_TF_1
	<37>	:= Интегральные суммы с меткой времени CP56Время2a	M_IT_TB_1
	<38>	:= Действие устройств защиты с меткой времени CP56Время2a	M_EP_TD_1
	<39>	:= Упакованная информация о срабатывании пусковых органов защиты с меткой времени CP56Время2a	M_EP_TE_1
	<40>	:= Упакованная информация о срабатывании выходных цепей устройства защиты с меткой времени CP56Время2a	M_EP_TF_1

Используются ASDU либо из набора <2>, <4>, <6>, <8>, <10>, <12>, <14>, <16>, <17>, <18>, <19>, либо из набора от <30> до <40>.

Информация о процессе в направлении управления

(Параметр, характерный для станции; каждый Type ID маркируется знаком X, если используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

X	<45>	:= Однопозиционная команда	C_SC_NA_1
	<46>	:= Двухпозиционная команда	C_DC_NA_1
	<47>	:= Команда пошагового регулирования	C_RC_NA_1
	<48>	:= Команда уставки, нормализованное значение	C_SE_NA_1
	<49>	:= Команда уставки, масштабированное значение	C_SE_NB_1
X	<50>	:= Команда уставки, короткий формат с плавающей запятой	C_SE_NC_1
	<51>	:= Строка из 32 бит	C_BO_NA_1
X	<58>	:= Однопозиционная команда с меткой времени CP56Время2a	C_SC_TA_1
	<59>	:= Двухпозиционная команда с меткой времени CP56Время2a	C_DC_TA_1
	<60>	:= Команда пошагового регулирования с меткой времени CP56Время2a	C_RC_TA_1
	<61>	:= Команда уставки, нормализованное значение с меткой времени CP56Время2a	C_SE_TA_1
	<62>	:= Команда уставки, масштабированное значение с меткой времени CP56Время2a	C_SE_TB_1
X	<63>	:= Команда уставки, короткое значение с плавающей запятой с меткой времени CP56Время2a	C_SE_TC_1
	<64>	:= Строка из 32 бит с меткой времени CP56Время2a	C_BO_TA_1

Используются ASDU либо из набора от <45> до <51>, либо из набора от<58> до<64>.

Информация о системе в направлении контроля

(Параметр, характерный для станции; для маркировки используется знак X)

	<70>	:= Окончание инициализации	M_EI_NA_1

Информация о системе в направлении управления

(Параметр, характерный для станции; каждый Type ID маркируется знаком X, если используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

X	<100>	:= Команда опроса	C_IC_NA_1
	<101>	:= Команда опроса счетчиков	C_CI_NA_1
	<102>	:= Команда чтения	C_RD_NA_1
X	<103>	:= Команда синхронизации времени (опция, см.7.6)	C_CS_NA_1
	~~<104>~~	~~:= Тестовая команда~~	~~C_TS_NA_1~~
	<105>	:= Команда сброса процесса	C_RP_NA_1
	~~<106>~~	~~:= Команда задержки опроса~~	~~C_CD_NA_1~~
	<107>	:= Тестовая команда с меткой времени CP56Время2a	C_TS_TA_1

Передача параметра в направлении управления

(Параметр, характерный для станции; каждый Type ID маркируется знаком X, если используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

	<110>	:= Параметр измеряемой величины, нормализованное значение	P_ME_NA_1
	<111>	:= Параметр измеряемой величины, масштабированное значение	P_ME_NB_1
	<112>	:= Параметр измеряемой величины, короткий формат с плавающей запятой	P_ME_NC_1
	<113>	:= Активации параметра	P_AC_NA_1

Пересылка файла

(Параметр, характерный для станции; каждый Type ID маркируется знаком X, если используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

	<120>	:= Файл готов	F_FR_NA_1
	<121>	:= Секция готова	F_SR_NA_1
	<122>	:= Вызов директории, выбор файла, вызов файла, вызов секции	F_SC_NA_1
	<123>	:= Последняя секция, последний сегмент	F_LS_NA_1
	<124>	:= Подтверждение приема файла, подтверждение приема секции	F_AF_NA_1
	<125>	:= Сегмент	F_SQ_NA_1
	<126>	:= Директория {пропуск или X; только в направлении контроля (стандартном)}	F_DR_NA_1

Б.6 Основные прикладные функции

Инициализация станции

(Параметр, характерный для станции; если функция используется, то прямоугольник маркируется знаком X)

	Удаленная инициализация

Циклическая передача данных

(Параметр, характерный для станции; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

X	Циклическая передача данных

Процедура чтения

(Параметр, характерный для станции; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

	Процедура чтения

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

Спорадическая передача

(Параметр, характерный для станции; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

X	Спорадическая передача

Дублированная передача объектов информации при спорадической причине передачи

(Параметр, характерный для станции; каждый тип информации маркируется знаком **X**, если оба типа - Type ID без метки времени и соответствующий Type ID с меткой времени - выдаются в ответ на одиночное спорадическое изменение в контролируемом объекте).

Следующие идентификаторы типа, вызванные одиночным изменением состояния объекта информации, могут передаваться последовательно. Индивидуальные адреса объектов информации, для которых возможна дублированная передача, определяются в проектной документации.

	Одноэлементная информация M_SP_NA_1, M_SP_TA_1, M_SP_TB_1 и M_PS_NA_1
	Двухэлементная информация M_DP_NA_1, M_DP_TA_1 и M_DP_TB_1
	Информация о положении отпаек M_ST_NA_1, M_ST_TA_1 и M_ST_TB_1
	Строка из 32 бит M_BO NA_1, M_BO_TA_1 и M_BO_TB_1 (если определено для конкретного проекта)
	Измеряемое значение, нормализованное M_ME_NA_1, M_ME_TA_1, M_ME_ND_1 и M_ME_TD_1
	Измеряемое значение, масштабированное M_ME_NB_1, M_ME_TB_1 и M_ME_TE_1
	Измеряемое значение, короткий формат с плавающей запятой M_ME_NC_1, M_ME_TC_1 и M_ME_TF_1

Опрос станции

(Параметр, характерный для станции; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

X	Общий				
X	Группа 1	X	Группа 8	X	Группа 15
X	Группа 2	X	Группа 9	X	Группа 16
X	Группа 3	X	Группа 10	Адреса объектов информации, принадлежащих каждой группе, должны быть показаны в отдельной таблице	
X	Группа 4	X	Группа 11		
X	Группа 5	X	Группа 12		
X	Группа 6	X	Группа 13		
X	Группа 7	X	Группа 14		

Синхронизация времени

(Параметр, характерный для станции; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

X	Синхронизация времени

Опционально.

Передача команд

(Параметр, характерный для станции; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях)

X	Прямая передача команд
X	Прямая передача команд уставки
	Передача команд с предварительным выбором
	Передача команд уставки с предварительным выбором
	Использование C_SE_ACTTERM
X	Нет дополнительного определения длительности выходного импульса
	Короткий импульс (длительность определяется системным параметром на КП)
	Длинный импульс (длительность определяется системным параметром на КП)
	Постоянный выход
	Контроль максимальной задержки (запаздывания) команд телеуправления и команд уставки в направлении управления
	Максимально допустимая задержка команд телеуправления и команд уставки

Передача интегральных сумм

(Параметр, характерный для станции или объекта; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях).

	Режим A: Местная фиксация со спорадической передачей
	Режим B: Местная фиксация с опросом счетчика
	Режим C: Фиксация и передача при помощи команд опроса счетчика
	Режим D: Фиксация командой опроса счетчика, фиксированные значения сообщаются спорадически
	Считывание счетчика
	Фиксация счетчика без сброса
	Фиксация счетчика со сбросом
	Сброс счетчика
	Общий запрос счетчиков
	Запрос счетчиков группы 1
	Запрос счетчиков группы 2
	Запрос счетчиков группы 3
	Запрос счетчиков группы 4

Загрузка параметра

(Параметр, характерный для объекта; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях).

	Пороговое значение величины
	Коэффициент сглаживания
	Нижний предел для передачи значений измеряемой

	величины
	Верхний предел для передачи значений измеряемой величины

Активация параметра

(Параметр, характерный для объекта; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях).

	Активация/деактивация постоянной циклической или периодической передачи адресованных объектов

Процедура тестирования

(Параметр, характерный для станции; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях).

X	Процедура тестирования

Пересылка файлов

(Параметр, характерный для станции; маркируется знаком X, если функция используется)

Пересылка файлов в направлении контроля

	Прозрачный файл
	Передача данных о нарушениях от аппаратуры защиты
	Передача последовательности событий
	Передача последовательности регистрируемых аналоговых величин
	Пересылка файлов в направлении управления
	Прозрачный файл

Фоновое сканирование

(Параметр, характерный для станции; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях).

	Фоновое сканирование

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

Получение задержки передачи

(Параметр, характерный для станции; маркируется знаком X, если функция используется только в стандартном направлении, знаком R - если используется только в обратном направлении и знаком B - если используется в обоих направлениях).

	~~Получение задержки передачи~~

Определение тайм-аутов

Параметр	Значение по умолчанию	Примечания	Выбранное значение
t0	10 с	Тайм-аут при установлении соединения	
t1	15 с	Тайм-аут при посылке или тестировании APDU	
t2	10 с	Тайм-аут для подтверждения в случае отсутствия сообщения с данными t2<t1	
t3	20 с	Тайм-аут для посылки блоков тестирования в случае долгого простоя	

Максимальный диапазон значений для всех тайм-аутов равен: от 1 до 255 с точностью до 1с.

Максимальное число k неподтвержденных APDU формата I и последних подтверждающих APDU (w)

Параметр	Значение по умолчанию	Примечания	Выбранное значение
k	12 APDU	Максимальная разность между переменной состояния передачи и номером последнего подтвержденного APDU	12 APDU
w	8 APDU	Последнее подтверждение после приема w APDU формата I	8 APDU

Максимальный диапазон значений k: от 1 до 32767 = (215-1) APDU с точностью до 1 APDU. Максимальный диапазон значений w: от 1 до 32767 APDU с точностью до 1 APDU (Рекомендация: - значение w не должно быть более двух третей значения k).

Номер порта

Параметр	Значение	Примечание
Номер порта	2404	Во всех случаях

Согласовано

Взаим. инв. №

Подп. и дата

Инв. № подл.

ПРИЛОЖЕНИЕ З

(Обязательное)

Описание структур, функций и объектов библиотеки МЭК 60870-5-104

TYPE Iec60870MeasuredValueShort : STRUCT - Значение измеряемой величины, короткий формат с плавающей запятой.

Name	Type	Initial	Comment
ObjectAddress	DINT		Адрес объекта информации
TypeId	Iec60870Type	Iec60870Type.M_ME_NC_1	Тип информации
Value	REAL		Значение (короткий формат с плавающей запятой)
Quality	Iec60870Quality		Описатель качества

TYPE Iec60870MeasuredValueShortWithCP56Time2a : STRUCT - Значение измеряемой величины, короткий формат с плавающей запятой с меткой времени CP56Время2a.

Name	Type	Initial	Comment
ObjectAddress	DINT		Адрес объекта информации
TypeId	Iec60870Type	Iec60870Type.M_ME_TF_1	Тип информации
Value	REAL		Значение (короткий формат с плавающей запятой)
Quality	Iec60870Quality		Описатель качества
Timestamp	Iec60870CP56Time2a		Метка времени в формате CP56Время2a

TYPE Iec60870SinglePointInformation : STRUCT - Одноэлементная информация

Name	Type	Initial	Comment
ObjectAddress	DINT		Адрес объекта информации
TypeId	Iec60870Type	Iec60870Type.M_SP_NA_1	Тип информации
Value	BOOL		Значение (одноэлементная информация)
Quality	Iec60870Quality		Описатель качества

TYPE Iec60870SinglePointWithCP56Time2a : STRUCT - Одноэлементная информация с меткой времени СР56Время2а

Name	Type	Initial	Comment
ObjectAddress	DINT		Адрес объекта информации
TypeId	Iec60870Type	Iec60870Type.M_SP_TB_1	Тип информации
Value	BOOL		Значение (одноэлементная информация)
Quality	Iec60870Quality		Описатель качества
Timestamp	Iec60870CP56Time2a		Метка времени в формате СР56Время2а