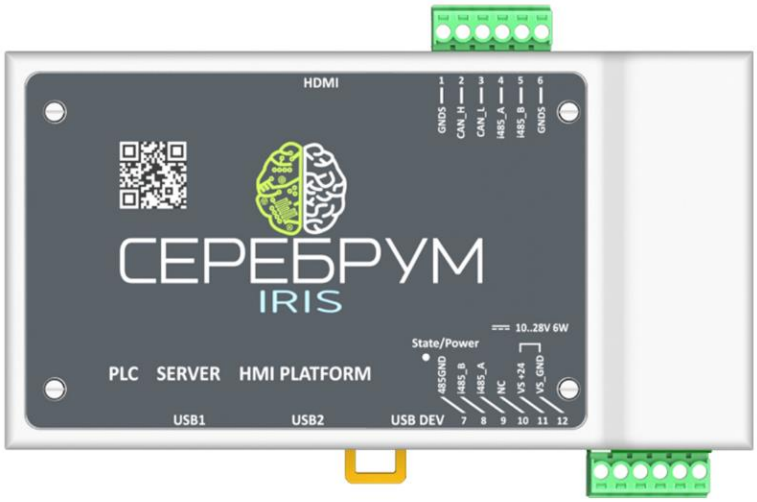
Программируемый логический контроллер

СЕРЕБРУМ IRIS

Техническое руководство



LINUX	Master SCADA *	HDMI	USB HOST 4	32GB < microSD	
Ethernet	USB	RS232	RS485 2	ШИНА модулей I/O	Clock
MODBUS TCP RTU	Telemetry GSM/ Ethernet	UDP YART LINK	USER protocol	OPC UA	REST API
среда YART Studio	языки FBD C	отладка VIRTUAL PLC	отладка VISUAL элементы	AUTO PID	CEREBRUM CLOUD
Audio IN	Audio OUT	CAN	LVDS	Micro phone	

Оглавление

2. Предупреждения

3. Список изменений

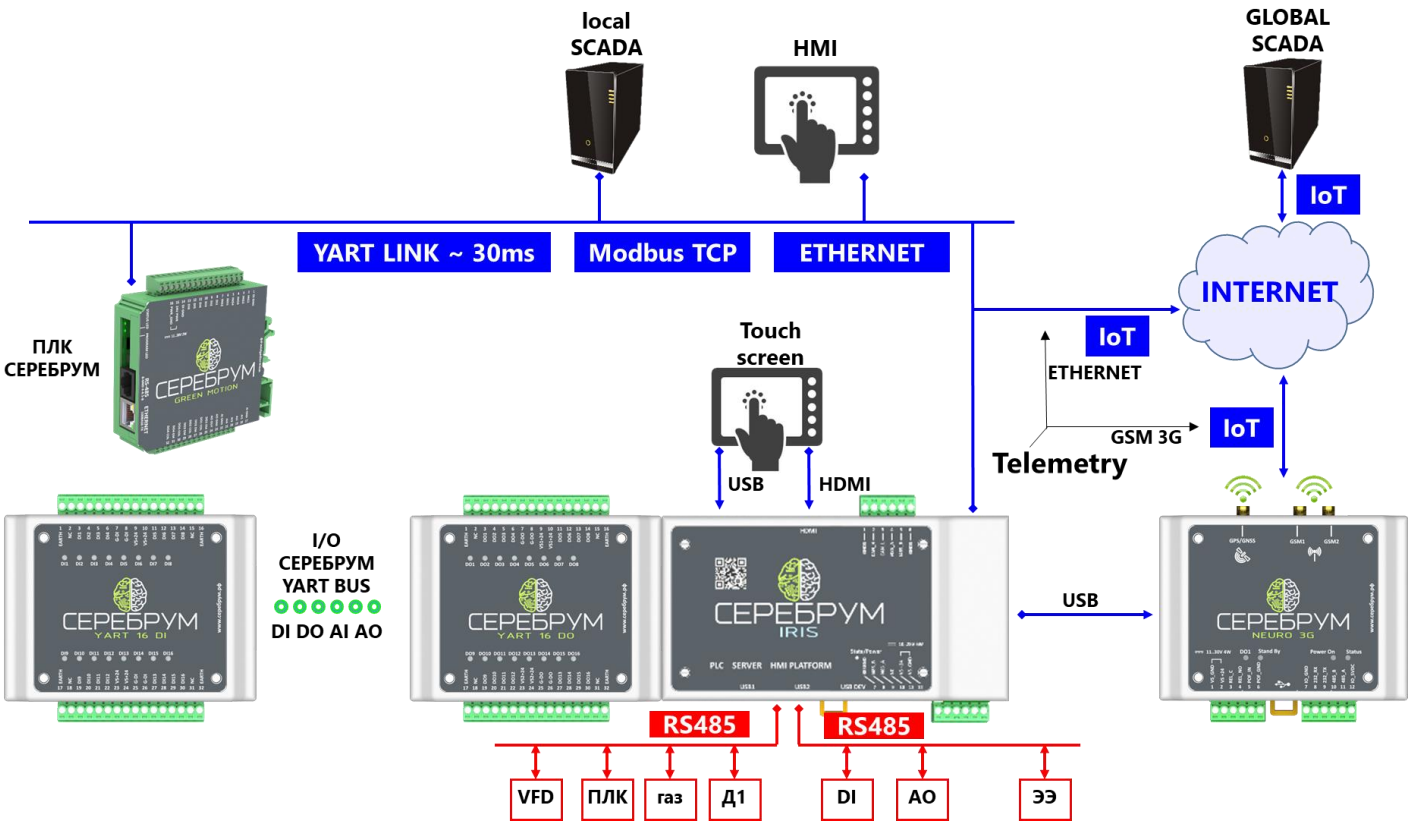
4. Общая информация

Контроллер IRIS — промышленный логический контроллер (ПЛК) с расширенными коммуникационными возможностями.

В зависимости от приложения IRIS способен работать в качестве ПЛК с расширенными коммуникационными возможностями или сервера телеметрии, сервера SCADA, шлюза сети Интернет.

Начальная настройка контроллера производится через встроенный Web интерфейс. Дальнейшая работа и программирование выполняется в среде Yart Studio (если IRIS используется в режиме ПЛК), либо в среде Master SCADA 4D, если установлен пакет программного обеспечения ИнСАТ.

Кроме того, IRIS может использоваться во множестве задач управления в качестве промышленного компьютера.



5. Технические характеристики

5.1 Массогабаритные характеристики

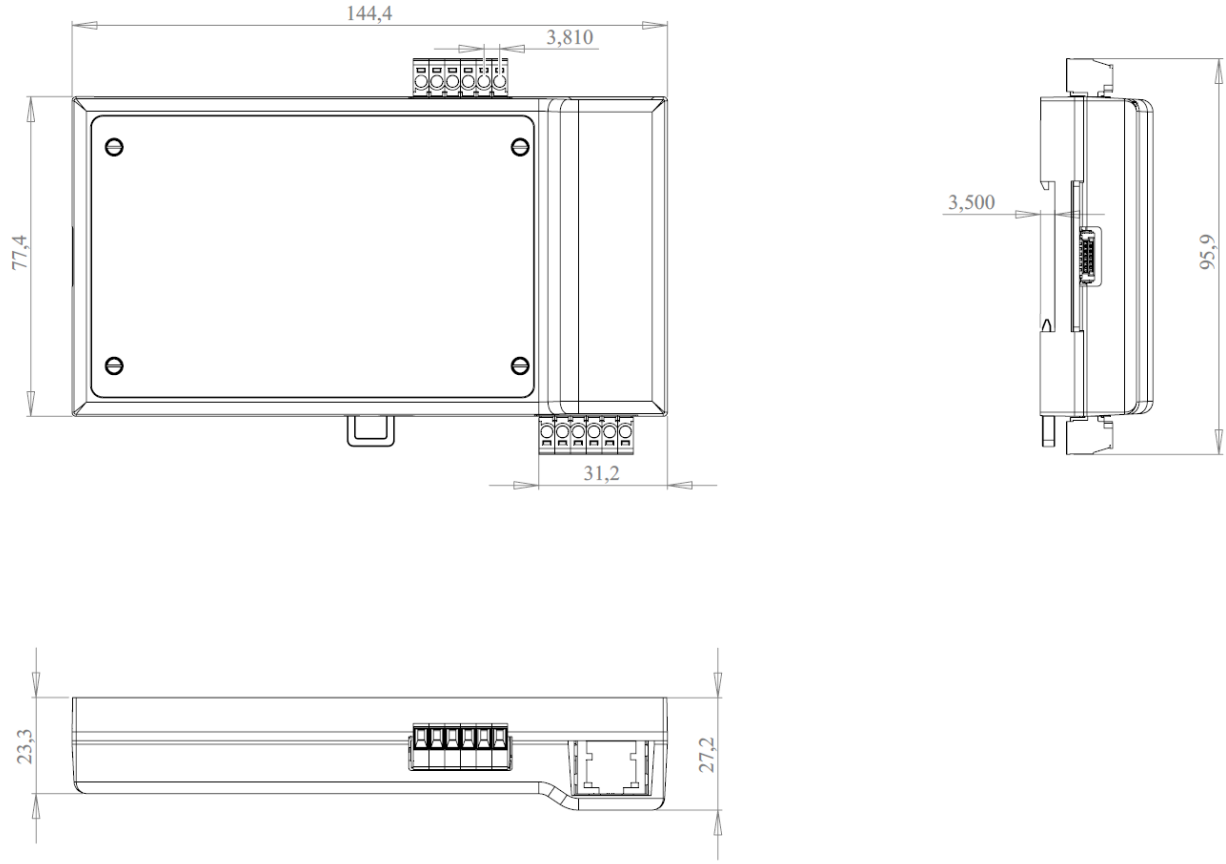
– габаритные размеры, мм, не более:

а) корпуса контроллера 145 x 78 x 28;

б) с учетом клеммных соединений 145 x 98 x 28.

– масса, кг, не более 0,2;

– монтаж на DIN-рейку по стандарту DIN EN 50 022.



5.2 Требования к электропитанию

Электропитание контроллера осуществляется от сети постоянного тока напряжением от 10 до 28 В (номинальное – 24 В).

Потребляемая мощность – не более 4 Вт.

5.3 Условия эксплуатации

Контроллер предназначен для эксплуатации в следующих условиях:

– закрытые взрывобезопасные помещения или шкафы электрооборудования без агрессивных паров и газов;

– рабочая среда воздух;

– диапазон рабочих температур окружающей среды от 5 до 55 ºС;

- верхний предел относительной влажности воздуха при низких температурах без конденсации влаги — 80% при 35 ºС и более;
- атмосферное давление — от 84 до 106,7 кПа;
- высота над уровнем моря — не более 2000 м;
- степень защиты корпуса по ГОСТ 14254-96 — IP20;
- вибрация — амплитуда не более 0,1 мм с частотой не более 25 Гц;

допустимая степень загрязнения 1 по ГОСТ Р 51841-2001 (несущественные загрязнения или наличие только сухих непроводящих загрязнений).

5.4 Конструкция и особенности устройства и работы контроллера

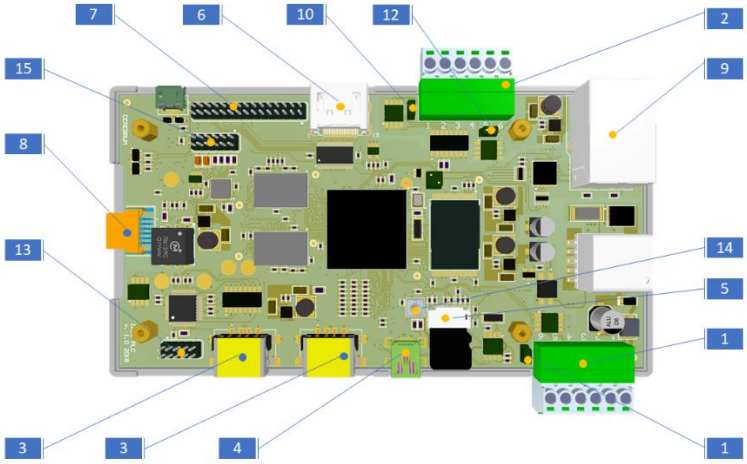


Рисунок 1. Плата контроллера IRIS

1. Клеммник 1
2. Клеммник 2
3. Разъемы USB-A
4. Разъем miniUSB (виртуальный последовательный порт) для подключения к Yart Studio
5. Слот uSD карты.
6. Разъем HDMI
7. Разъем подключения LVDS дисплея
8. Разъем шины YART-BUS
9. Разъем RJ-45, Ethernet
10. Переключатель «терминатора» линии RS-485 (изолированного)
11. Переключатель «терминатора» линии RS-485
12. Переключатель «терминатора» линии CAN-Bus
13. Разъем для подключения внешних USB портов
14. Светодиодный индикатор состояния
15. Разъем подключения аудио

Таблица 1. Технические характеристики IRIS

Параметр		Характеристики
Напряжение питания		от 12 до 28 VDC, гальванически развязанный источник питания
Тип процессора		ARM-Cortex-A9, тактовая частота 1000МГц, 3D/2D ускоритель
Тип разъемов клеммника		Разъемные, под винт, максимальное сечение провода 1.5 мм2
Тип ОС		Linux 4.1, Buildroot
Программирование		ФБД, C-YART, среда программирования YART Studio или MasterSCADA 4D ИнСАТ
Память	Программ	1024 KB память программ (более 500 типов блоков, 5000 вызовов блоков)
	RAM	40 KB память для переменных программы пользователя
	Постоянная	До 4 ГБ eMMC, в зависимости от комплектации
	FRAM	16 KB энергонезависимая память для переменных пользователя, включая область FRAM BIT
	Общая	1024 МБ DDR3-800, 32 bit
	RAM BIT	2 Kb память для переменных пользователя
	FRAM BIT	2 Kb энергонезависимая память для переменных пользователя
uSD карта памяти		до 32 GB для хранения архива приложения
Выполнения цикла программы		от 1 ms
Подключение дисплея	HDMI	Стандартный HDMI 1.4 порт монитора, разрешение до 1920x1080
	LVDS	Подключение LVDS панели, 1366x786 макс., I2C для сенсорного экрана (**)

Аудио	Входы	Линейный вход (стерео) + вход микрофона
	Выходы	Линейный выход (стерео), выход подключения наушников, 16 Ом мин.
Коммуникационные порты	ETHERNET	Разъем RJ-45, 100/10Base-T автоопределение типа кабеля. YART-LINK, MODBUS TCP, IoT, статус WEB SERVER, Сервер телеметрии Серебрум
	USB Host	4 порта: 2 на плате (**), 2 USB-A порта
	USB Device	Виртуальный последовательный порт, Modbus RTU для Yart Studio
	RS485	MODBUS RTU, протоколы библиотеки, пользовательский
	RS485I	MODBUS RTU, протоколы библиотеки, пользовательский, гальванически изолирован от цепей процессора
	CAN	CANOpen с использованием CAN Festival(*)
	YART-BUS	Шина модулей расширения, период опроса всех модулей менее 30 ms, максимальное число подключаемых модулей – 7
Часы реального времени (RTC)		С питанием от батарейки (точность хода при 25 ºС – не более ±2 с в сутки)

(*) – подсистема находится в разработке

(**) – для OEM решений

6. Подключение

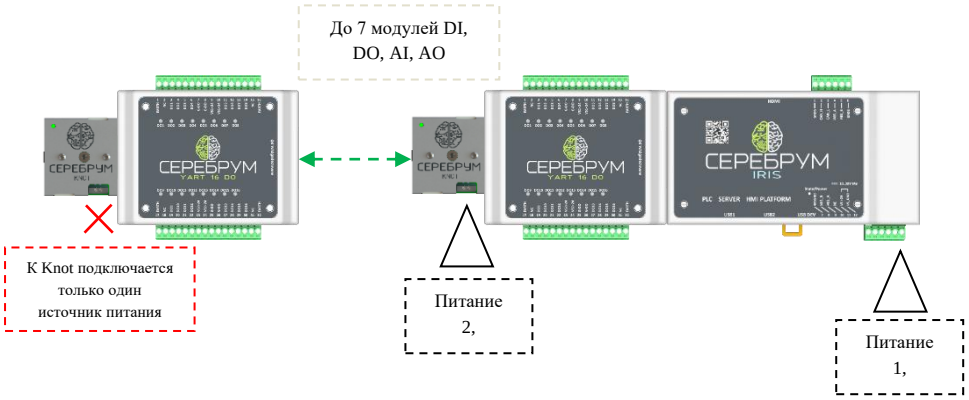


Рисунок 2. Подключение источников питания

Для работы контроллера требуется наличие внешнего питания (см. **Таблицу №1**). Линия питания выведена на клеммы 11-12 клеммника 1, а также на выводы разъема подключения модулей расширения – YART-BUS.

На **Рисунке 2** показана схема питания контроллера.

Основной источник питания подключается к клеммнику 1. Модули расширения, подключенные к IRIS будут работать от основного источника питания через шину расширения YART-BUS.

При необходимости организации резервного питания или при отсутствии возможности подключения к клеммнику №1 питание может осуществляться через модуль Knot, подключаемый к шине расширения ПЛК.

Место подключения не регламентируется.

! Внимание

При одновременном использовании нескольких модулей Knot внешнее питание должно подключаться только к одному из них.

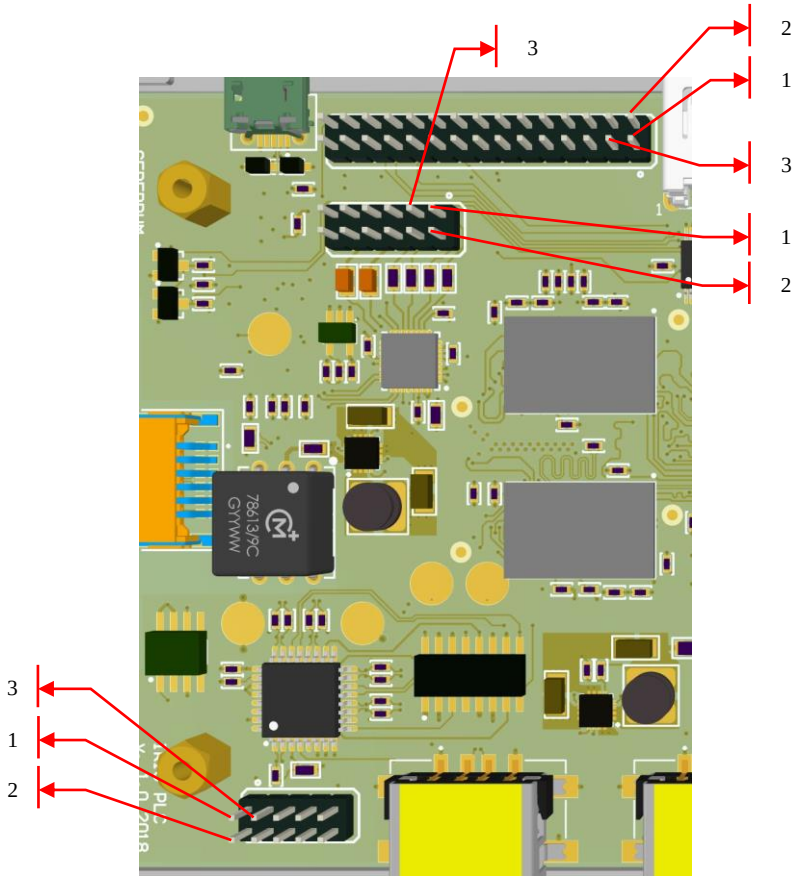


Рисунок 3. Нумерация контактов разъемов расширения

Таблица 2. Разъем аудио порта (Рис. 2)

Номер	Назначение	Комментарии
1	Линейный выход, правый канал	Минимальное Rнагр 10 кОм
2	Линейный выход, левый канал	
3	Общий	
4	-	
5	Линейный вход, правый канал	Входное сопротивление 29 кОм
6	Линейный вход, левый канал	
7	Вход микрофона	Монофонический микрофон, питание от кодека
8	Выход наушников, правый канал	17 мВт на нагрузке 16 Ом.
9	-	
10	Выход наушников, левый канал	
11	-	
12	Общий	

Таблица 3. Разъем дополнительных портов USB (Рис. 2)

Номер	Назначение	Комментарии
1	Vbus, выход A	500 мА макс. на канал
2	Vbus, выход B	500 мА макс. на канал
3	D-, выход A	
4	D-, выход B	
5	D+, выход A	
6	D+, выход B	
7	GND	
8	GND	
9	-	
10	-	

Таблица 4. Разъем LVDS дисплея (Рис. 2)

Номер	Назначение	Комментарии
1	LVDS0_N	Данные дисплея Lane0
2	LVDS0_P	
3	GND	
4	LVDS2_P	Данные дисплея Lane2
5	LVDS2_N	
6	GND	
7	LVDS3_N	Данные дисплея Lane3
8	LVDS3_P	
9	GND	
10	LVDS1_P	Данные дисплея Lane1
11	LVDS1_N	
12	SCL1	I2C1, для подключения сенсорного экрана
13	LVDS_CLK_P	Данные дисплея CLK
14	LVDS_CLK_N	
15	+5V	Питание дисплея, 500мА макс.
16	+5V	
17	+3.3V	Логический уровень дисплея, 100 мА макс.
18	+3.3V	
19	SDA1	I2C1, для подключения сенсорного экрана
20	PWM	Сигнал ШИМ для регулировки яркости подсветки
21	+12V	Питание подсветки 12В, 400мА макс.
22	+12V	
23	TOUCH_INT	Сигнал прерывания сенсорного экрана
24	TOUCH_RST	Сброс контроллера сенсорного экрана
25	SDA2	Дополнительный интерфейс I2C
26	SCL2	Дополнительный интерфейс I2C
27	GPIO1	Дополнительная линия GPIO
28	GPIO2	Дополнительная линия GPIO
29	GND	
30	GND	

Таблица 5. Клеммник 1

Номер	Назначение	Комментарии
7	485-GND	Общий для порта RS-485-1 (порт гальванически развязан от линии питания (VS +24; VS GND)
8	485-B	Сигнал B порта RS-485-1
9	485-A	Сигнал A порта RS-485-1
10	-	
11	VS +24	Терминал подключения питания 10..28В
12	VS GND	Терминал подключения питания 10..28В

Таблица 6. Клеммник 2

Номер	Назначение	Комментарии
1	GNDS	Общий для порта RS-485 -2 (порты RS-485-2 и CAN полностью гальванически развязаны)
2	CAN-H	
3	CAN-L	
4	i485-A	Сигнал A порта RS-485-2
5	I485-B	Сигнал B порта RS-485-2
6	GNDS	

7. Начало работы

Обмен данными с IRIS в основном осуществляется через интерфейс Ethernet. Для комфортной работы с контроллером уделите внимание корректным настройкам сетевого оборудования и сетевой инфраструктуры в целом.

Подключение к Yart Studio также может быть выполнено через порт USB Device, однако некоторые настройки могут быть изменены только через Ethernet.

7.1 Настройки сети

Настройка сетевого интерфейса осуществляется через встроенный Web интерфейс.

Внешний вид интерфейса может меняться в зависимости от версии установленного программного обеспечения.

При производстве, IRIS получает статический IP адрес равный ***192.168.1.254***

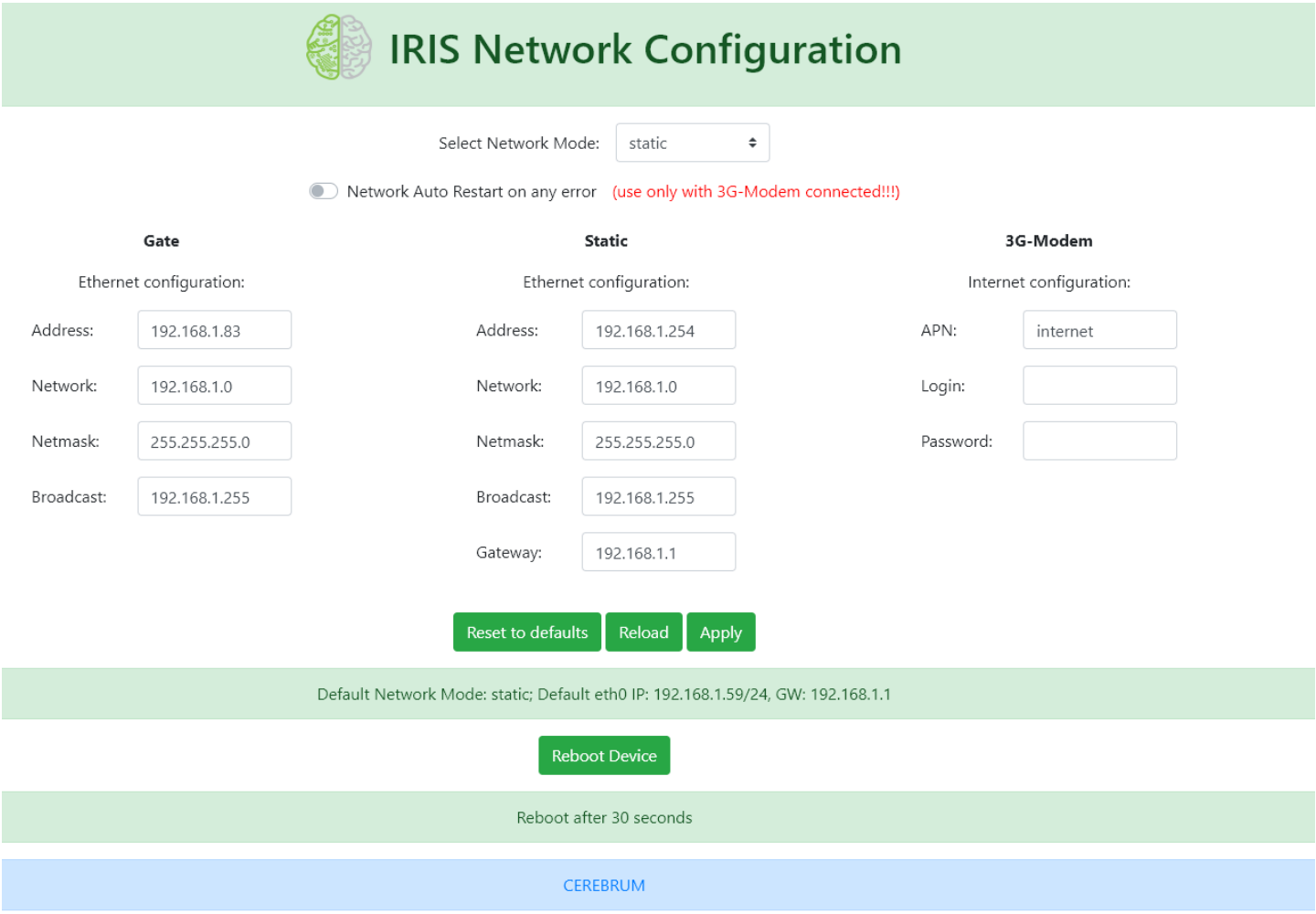


Рисунок 4. Встроенный интерфейс настройки

В центральной колонке (**Рис. 4**) расположены настройки основного сетевого подключения.

В верхней части экрана расположено окно выбора режимов работы сетевого интерфейса IRIS

Доступны три основных режима:

- ***Static*** – статический IP адрес
- ***DHCP*** – автоматическое определение сетевого адреса по протоколу DHCP
- ***Gate*** – работа в качестве роутера – другие абоненты сети могут пользоваться Интернет через подключенный к IRIS модем Neuro

Колонка ***Gate*** определяет параметры IRIS в режиме роутера. IP адрес назначается статически.

Колонка ***Static*** содержит настройка сетевого подключения в обычном режиме работы. Если включен режим DHCP, то данные в колонке Static будут содержать информацию, полученную от DHCP сервера в процессе инициализации сетевого интерфейса.

В колонке 3G Modem перечислены настройки Neuro для осуществления регистрации в мобильной сети передачи данных.

7.2 Работа с 3G модемом Neuro

В отличие от Yart 1.8 контроллер IRIS подключается к 3G модему Neuro только через USB интерфейс.

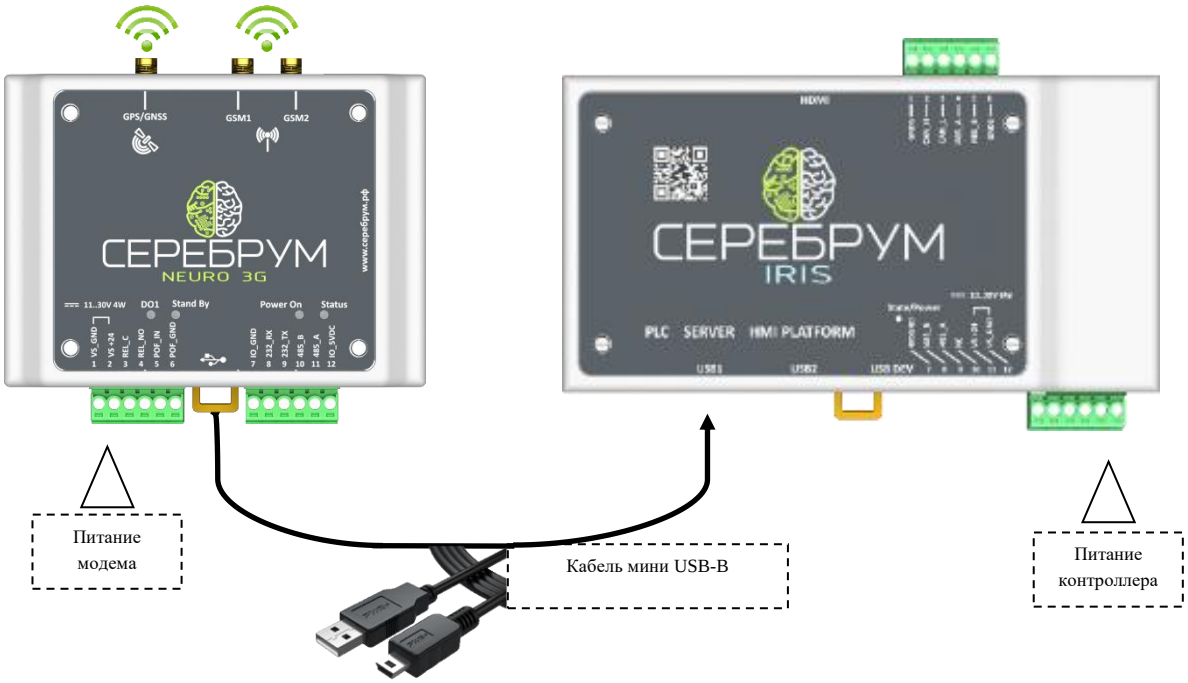


Рисунок 5. Подключение модема Neuro

Для подключения доступны порты USB Host (**поз. 3, Рис. 1**). К одному контроллеру подключается не более одного модема.

При работе с Neuro соединение с Интернет будет поддерживаться автоматически.

8. Использование модулей расширения

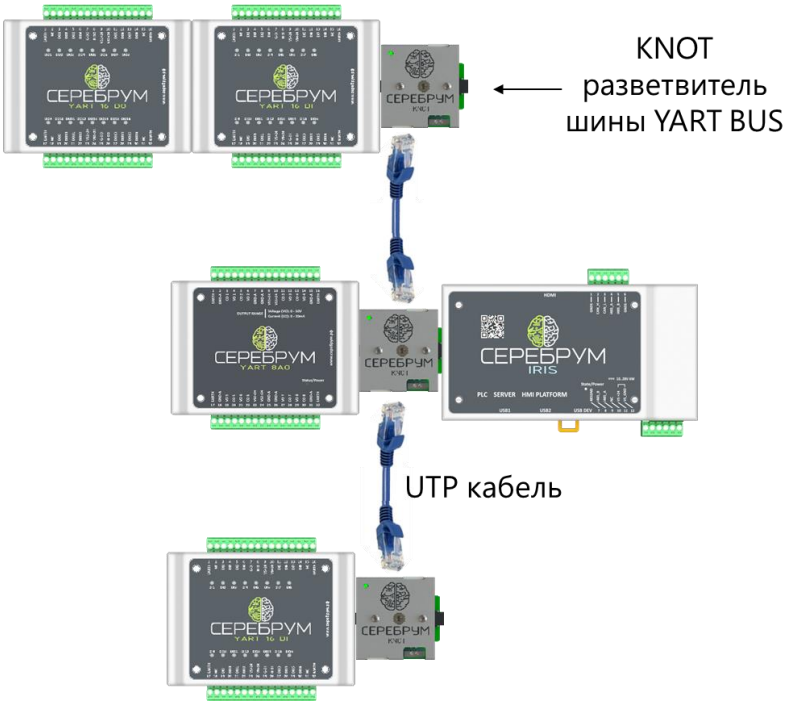


Рисунок 6. Возможная топология шины расширения

Пример возможной конфигурации шины YART-BUS представлен на **Рисунке 6**.

При проектировании конфигурации шины необходимо учитывать следующие правила:

- Контроллер IRIS поддерживает до семи различных модулей расширения на одной шине
- Суммарная длина всех соединений не должна превышать 5 метров
- На оконечном модуле должен быть включен согласующий «терминатор»
- Учитывайте падение напряжения между модулями. В среднем на каждом модуле теряется 0.1В
- При низком значении напряжения питания удаленный модуль может отключиться
- Диапазон разрешенных адресов для конфигурации модулей расширения 1 – 7
- Порядок присоединения модулей может быть любым

Программная конфигурация модулей осуществляется при программе YartStudio.

При каждом запуске контроллер осуществляет сканирование и идентификацию всех возможных модулей. В процессе работы найденные модули опрашиваются, а результат работы сохраняется в специализированных регистрах процессора.

На рисунке ниже показан пример автоматически определенных модулей.

IRIS может работать в двух режимах опроса модулей расширения:

- В ***безопасном режиме работы*** любая ошибка опроса модулей расширения вызовет аварийную остановку прикладной программы. При этом все выходы модулей будут переведены в безопасное состояние (нуль).
- В ***обычном режиме работы*** ошибка обмена с модулем расширения не вызывает остановку выполнения прикладной программы, но состояние ошибки фиксируется в соответствующем регистре процессора для анализа и диагностики.

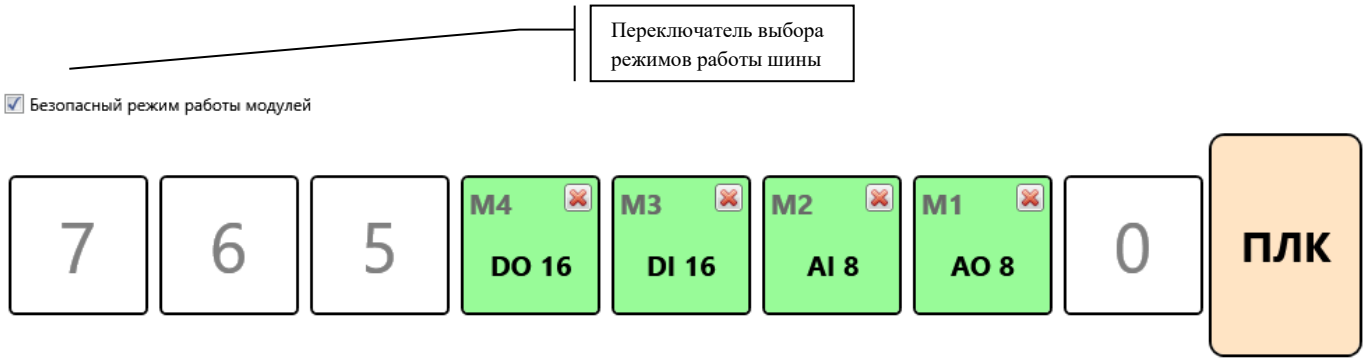


Рисунок 7. Конфигурация модулей расширения

Выбор режима работы шины расширения осуществляется в окне настроек проекта – модули расширения.

Каждый модуль расширения должен иметь уникальный адрес, который назначается вращающимся переключателем, расположенным на плате модуля.

Для IRIS доступны адреса модулей в диапазоне 1 – 7.

Пример модуля расширения и расположение переключателей показан на **Рисунке 8**.

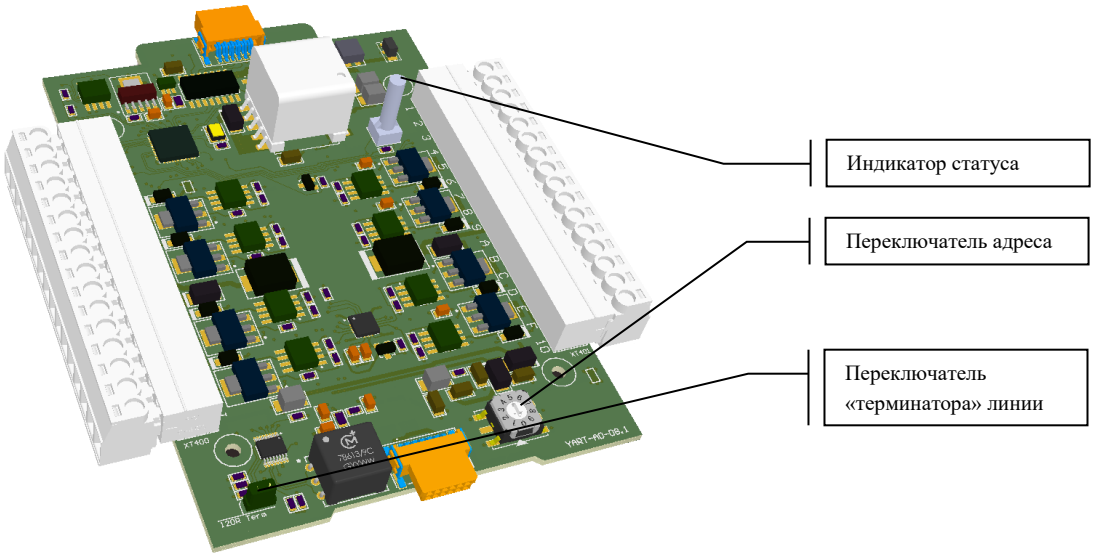


Рисунок 8. Переключатели модулей расширения

Перед началом работы убедитесь, что переключатели адресов модулей расширения установлены в правильные позиции, совпадений между ними нет.

В процессе работы в окне статуса модулей расширения Yart Studio будет отображаться текущее состояние шины расширения.

Для успешной работы отображаемое состояние модулей должно в точности соответствовать проектному.

Каждый модуль расширения оснащается статусным индикатором. Для исправного модуля расширения данный индикатор должен всегда быть включен.

9. Индикация состояний

Отображение состояния ПЛК IRIS осуществляется при помощи светодиодного индикатора (**поз. 14 Рис. 2**)

Различные режимы отображения представлены в **Таблице 7**.

Таблица 7. Состояния встроенного индикатора

№	Режим индикатора	Состояние ПЛК
1	Горит оранжевым	Самодиагностика ПЛК, режим начальной загрузки
2	Горит зеленым	Выполнение прикладной программы (режим Run)
3	Мигает зеленым	Выполнение прикладной программы остановлено
4	Мигает красным	Аварийная остановка прикладной программы

10. ПЛК, особенности архитектуры и память

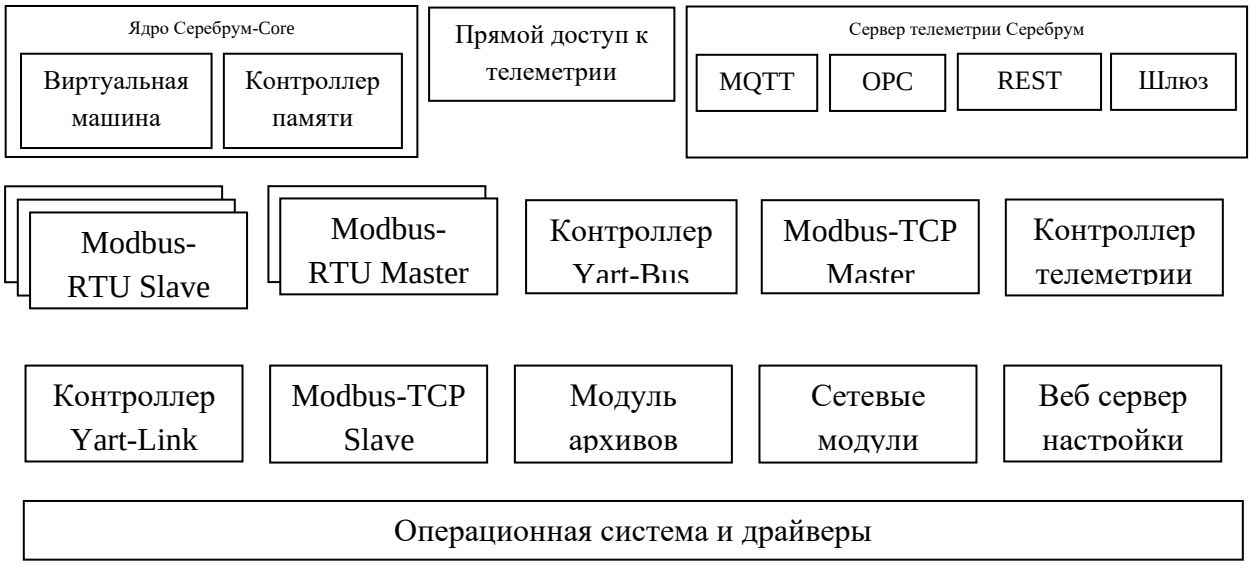


Рисунок 9. Архитектура контроллера IRIS

В основе вычислительного ядра контроллера лежит система ***Серебрум-Core***, состоящая из виртуальной машины и контроллера виртуальной памяти.

Все взаимодействие между модулями осуществляется через общее пространство памяти, в которой определены участки периферии, данных пользователя и настроек. Более подробно структура памяти рассматривается ниже.

Независимо от виртуальной машины ПЛК выполняет задачи обработки внешних интерфейсов, приёма и передачи данных, а также поддерживает функционирование встроенного сервера телеметрии.

При изменении настроек загруженного проекта все сопутствующие программы автоматически перенастраиваются.

Часть программных модулей настраиваются и обрабатываются непосредственно из программы пользователя. Например, работа программ обмена по Modbus конфигурируется через специальный API, что ускоряет и упрощает процесс настройки.

IRIS может одновременно работать в качестве сервера телеметрии и программируемого контроллера. В ряде случаев программе контроллера требуется управлять несколькими удалёнными ПЛК, подключёнными к встроенному серверу телеметрии. Специальный модуль прямого обмена данными между пространством виртуальной машины и сервером телеметрии обеспечивает быстрый и удобный доступ к пространству глобальных данных удалённых контроллеров.

10.1 Память программируемого контроллера

Вся память контроллера (**Таблица 8**) условно разделена на две основных части:

- Память программы
- Память данных

В памяти программы хранится исполняемый код виртуальной машины, настройки проекта, список переменных, а также настройки обмена и телеметрии.

При каждом старте контроллера содержимое памяти программы проверяется и, если все поля верны - даётся разрешение на исполнение кода виртуальной машины.

Память данных содержит четыре основных сегмента

10.1.1 Системная память

В системной памяти хранятся все настройки периферии контроллера, а также текущее значение данных, полученных от модулей ввода.

Чтение и запись в системную память осуществляют специальные блоки программы пользователя, обеспечивающие корректную работу с аппаратными ресурсами ПЛК

10.1.2 Оперативная память

Основной блок памяти, предназначенный для хранения данных прикладной программы, настроек программ окружения и данных последовательных портов.

Пространство оперативной памяти автоматически сбрасывается при рестарте контроллера. Исключение составляют лишь переменные, для которых определены начальные значения.

Нижнее пространство оперативной памяти выделено для хранения значений булевых переменных.

Помимо обычного режима доступа (байт, слово, двойное слово) булевые переменные имеют битовый режим доступа.

10.1.3 Энергонезависимая память (FRAM)

FRAM дает даёт возможность сохранения данных программы виртуальной машины при отсутствии основного питания контроллера.

Доступ к FRAM осуществляется аналогично оперативной памяти, однако скорость работы контроллера при этом будет снижена.

При разработке прикладных программ следует избегать постоянной перезаписи значений FRAM, поскольку контроллеру требуется дополнительное время на индексацию данных энергонезависимой памяти.

Все пространство энергонезависимой памяти разделено на страницы по 128 байт. После завершения каждого цикла исполнения виртуальной машины встроенная программа проверяет каждую страницу и осуществляет перезапись содержимого физической FRAM, если данные были изменены.

Специальный алгоритм контролирует запись страниц, предотвращая потерю данных при отключении питания контроллера.

Верхние 256 байт FRAM выделены под хранение энергонезависимых булевых переменных аналогично булевым переменным оперативной памяти.

10.1.4 Эмулятор батарейной памяти

Данный участок памяти функционально аналогичен оперативной памяти и организован для обеспечения совместимости с программами, разработанными для других моделей ПЛК Серебрум.

Таблица 8. Карта памяти контроллера

0000h	System, 1k	Настройки Данные в/в Данные системы	0000h	Память программ
03FFh		Управление телеметрией		Настройки проекта
				Список переменных
1000h	RAM, 32k	Встроенные данные		Значения констант
				Данные телеметрии
		Данные пользователя		Настройки
				Настройки ВМ
				Бинарный код ВМ
8E00h				
8FFFh		256 байт Bool		
B000h	FRAM, 15k	256 байт Bool		
	120 x 128 байт	Страница FRAM – 128B		
EBFFh				Контрольная сумма
F000h	BRAM, 4k	Эмуляция памяти Yart 1.8		
FFFFh			1FFFFFh	

10.2 Типы данных, используемых в системе

- ***Bool*** – Битовая переменная.
  Данные переменные хранятся в Bool сегментах (Таблица 8).
  В C-Yart доступны предопределенные слова «***true***», «***false***» соответствующие значениям «1» и «0» соответственно.
- ***Byte*** – Один байт данных. Беззнаковый тип в диапазоне чисел 0 – 255.
- ***Short*** – Знаковый 16 – разрядный тип данных. Может принимать значения в диапазоне [−32768, +32767]
- ***Int*** - Знаковый 32 – разрядный тип данных. Может принимать значения в диапазоне [−2 147 483 648, +2 147 483 647]
- ***Float*** – вещественные 32-разрядные числа в формате с плавающей точкой (IEEE 754-1985)
- ***Символ*** – ASCII символ в кодировке IBM CP866. Поддерживаются строки длиной до 254 символов
- ***Дата*** – внутренний тип данных в BCD формате даты: DD.MM.YY
  Нумерация лет в рамках диапазона [0, 99]
- ***Время*** – внутренний формат времени в BCD формате HH:MM:SS

11. Телеметрия

Все программируемые контроллеры Серебрум изготавливаются с поддержкой системы удаленного мониторинга и управления. В зависимости от типа используемого оборудования возможно подключение через Ethernet, 2G/3G модем или оба варианта.

Все подключенные контроллеры имеют встроенную возможность передавать через защищенное соединение данные на удаленный сервер, принимать команды и отображать в пространстве программы пользователя данные о состоянии сетевого подключения.

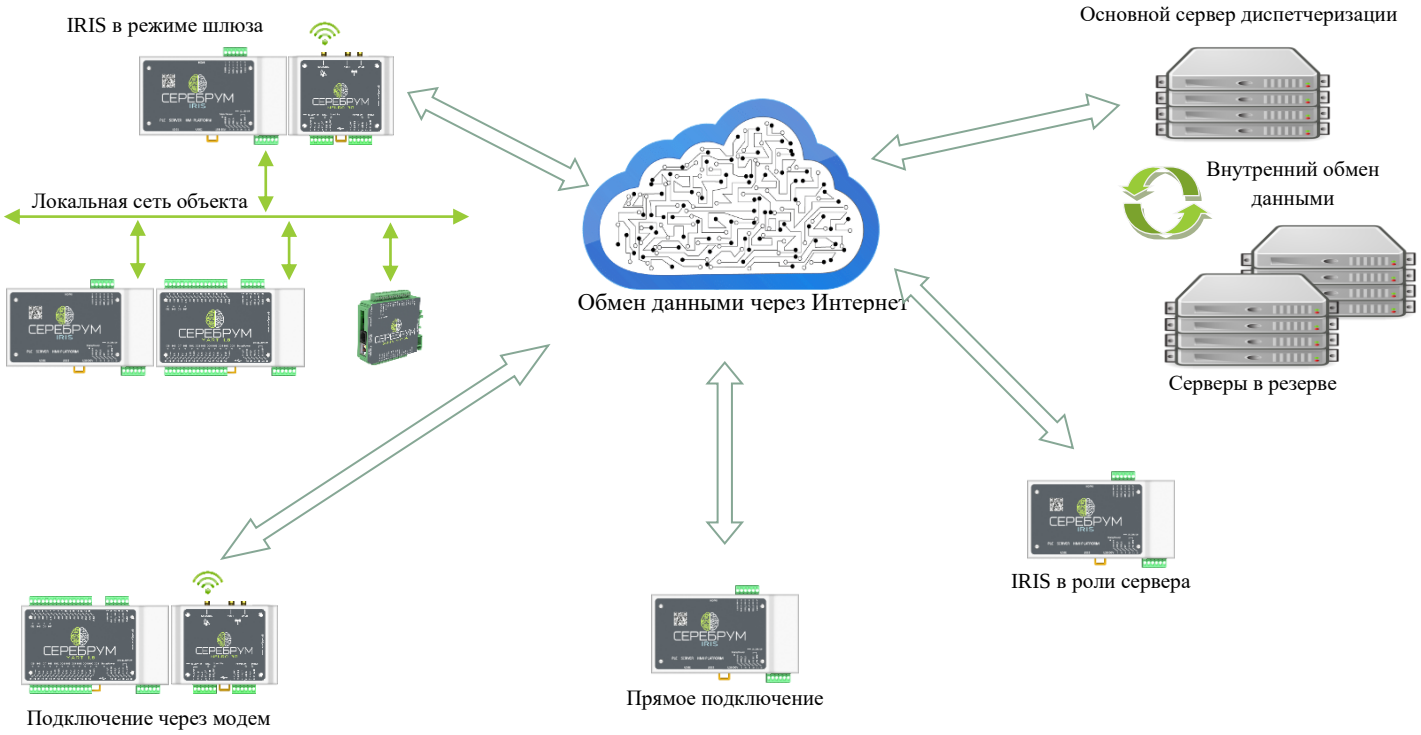


Рисунок 10. Удаленная диспетчеризация и управление Серебрум

Инфраструктура системы телеметрии Серебрум (**Рис. 10**) состоит из выделенного сервера и множества удаленных контроллеров. При необходимости вся система может быть развернута в локальной сети или сети предприятия.

Головная часть состоит из одного или нескольких серверов, которые могут быть связаны друг с другом для обеспечения горячего резервирования данных.

11.1 Использование IRIS в качестве сервера телеметрии

Для небольших систем диспетчеризации и управления (до 20 удаленных ПЛК) IRIS может быть использован в качестве сервера телеметрии.

Прикладная программа или сервер ***MasterSCADA*** дают возможность организации централизованного управления объектами и их мониторинга.

Для доступа к общим данным из прикладной программы ПЛК предусмотрен асинхронный прямой интерфейс к данным телеметрии. Данный интерфейс практически не оказывает влияния на быстродействие ПЛК – по мере готовности новых данных в пространстве RAM выставляются флаги готовности для активации тех или действий программы.

Доступ SCADA систем к общим данным осуществляется через OPC UA интерфейс или посредством специализированного REST API.

В случае доступа через ***OPC UA*** рекомендуем использовать бесплатную программу ***UA Expert***.

Внешний вид программы показан на **Рисунке 11**.

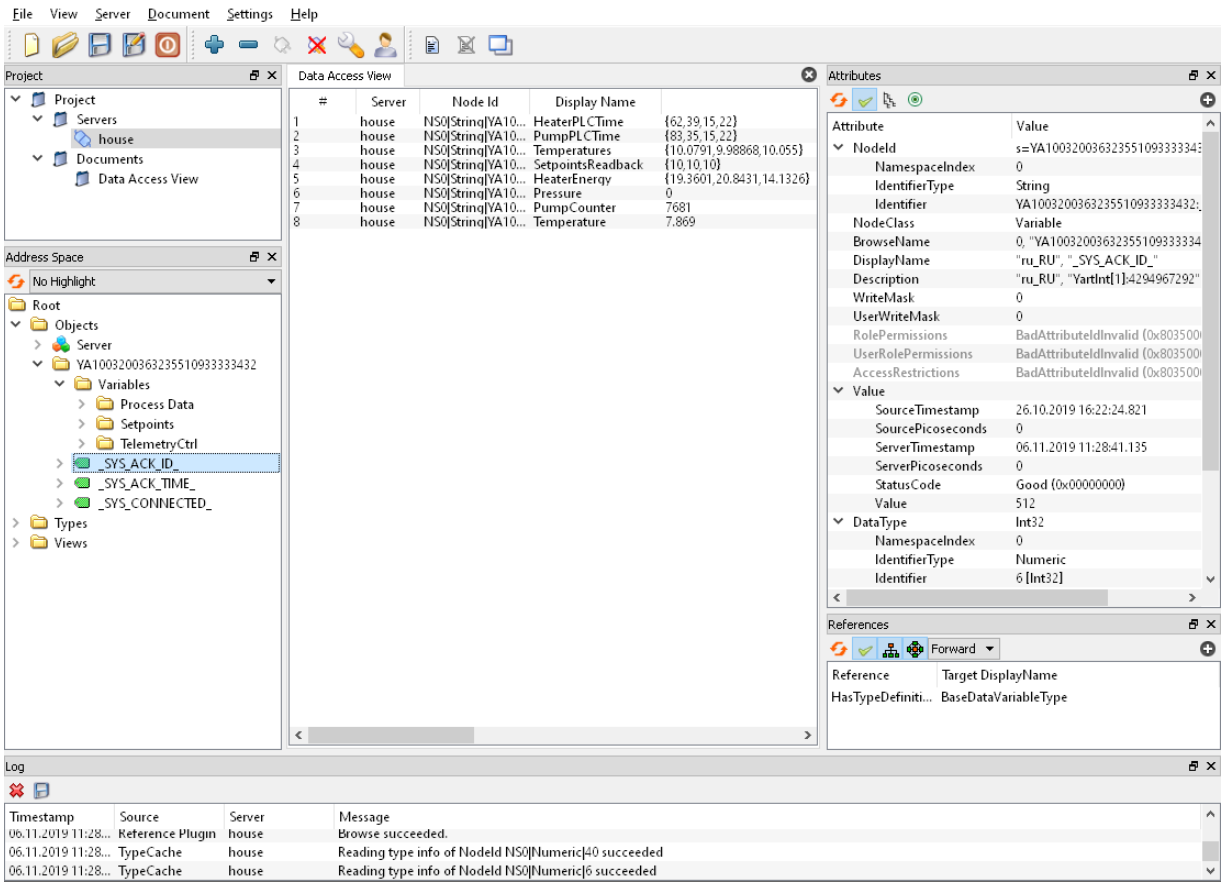


Рисунок 11. UA Expert

Программное обеспечение сервера телеметрии настроено на автоматический старт при загрузке контроллера. После запуска сервер сконфигурирован для работы следующим образом:

Порт ***MQTT: 1883*** (TCP)

Порт ***OPC UA: 16664*** (TCP)

Порт ***REST API: 5556*** (TCP)

Специальное расширение для Yart Studio позволяет осуществить настройку работы сервера телеметрии и подключаемых контроллеров.

Каждый контроллер, подключенный к IRIS, создает свой каталог тегов, доступных для работы. Данные теги конфигурируются в проектах Yart Studio. Более подробно данный процесс рассматривается в главе 7.2.

Корневым узлом дерева тегов (**Рис. 11**) является идентификатор контроллера. Он состоит из кода типа ПЛК и уникального серийного номера устройства.

В данном примере к серверу подключен контроллер ***YA1003200363235510933333432***.

Таблица 9. Типы префиксов контроллеров Серебрум

Префикс	Тип контроллера
YA1	Контроллер Yart 1.8
CO1	Контроллер Cobalt
GM1	Контроллер Green Motion
YV1	Контроллер IRIS или виртуальный ПЛК

В **Таблице 9** указаны возможные коды контроллеров, таким образом YA1 в примере соответствует Yart 1.8.

Наименования переменных, объявленных в исходном проекте YartStudio сохранятся во всем пространстве телеметрии. К использованию также допускаются кириллические имена переменных и массивы.

При использовании REST API наименования переменных не меняются.

Например: «***YA1003200363235510933333432:Variables:Process Data:Pressure»***

Для удобства работы с данными идентификатор контроллера может быть изменен на более удобное наименование – alias. Новый идентификатор должен быть уникальным в адресуемом пространстве сервера.

Подключение удаленных контроллеров предполагает обязательную процедуру аутентификации, кроме того, все данные, передаваемые через сеть, шифруются.

Для правильной работы системы рекомендуется сменить ключ шифрования, установленный по умолчанию. Данная настройка также доступна через интерфейс YartStudio.

11.2 Подключение к удаленному серверу телеметрии

Контроллер IRIS поддерживает работу в режиме удаленного модуля телеметрии. При этом подключение к Интернет может быть осуществлено через встроенный интерфейс Ethernet, а также через внешний модем Neuro.

Выбор режима работы осуществляется автоматически в зависимости от настроек сетевого интерфейса. Физическое соединение с модемом осуществляется через канал USB.

Для настройки телеметрии необходимо выполнить ряд действий.

Шаг 1. В окне настройки проекта выберите интерфейс передачи данных Ethernet, а также укажите настройки соединения с сервером телеметрии

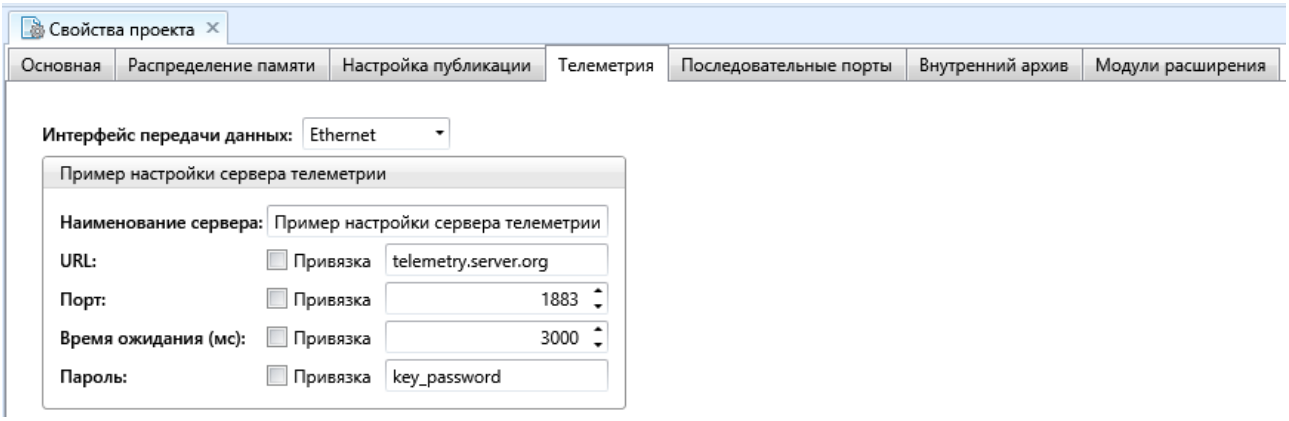


Рисунок 12. Простая настройка телеметрии

В том случае если контроллер еще на был зарегистрирован в системе – оставьте значение поля «Пароль» равным «***hitchhiker***».

В поле URL укажите адрес основного сервера телеметрии.
Порт – 1883, время ожидания – рекомендуется использовать значение 3000

В данном примере показана базовая настройка, не предполагающая смену рабочих серверов при работе контроллера.
Для динамического управления подключениями следует использовать режим привязок.

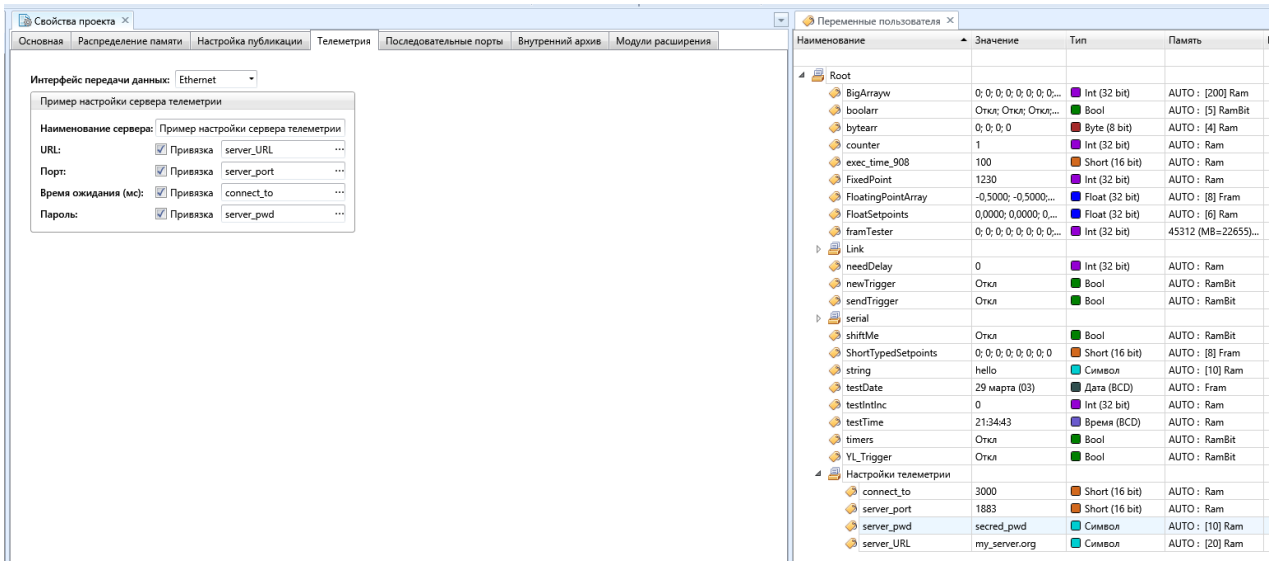


Рисунок 13. Привязка параметров подключения к серверу к переменным

На **Рисунке 13** показан пример использования переменных для настройки подключения к серверу. Значения переменных объявлены заранее, однако в процессе работы их можно изменить и дать команду повторного подключения.

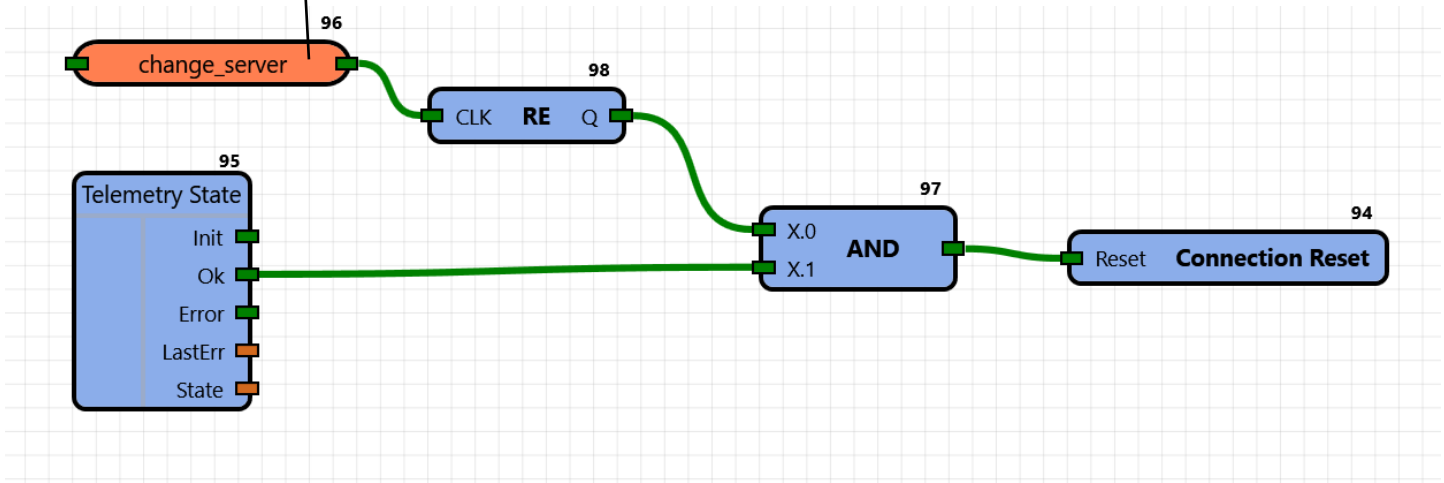


Рисунок 14. Пример команды сброса соединения с сервером

Перед каждой попыткой соединения с сервером программа телеметрии будет считывать значения переменных, указанных в виде ссылок (**Рис. 13**). Если соединение установить не удалось, программа снова считает эти данные и повторит попытку.

Если требуется установить соединение с другим сервером, то в приложении необходимо предусмотреть место, где переменные **server_URL**, **server_pwd**, **server_port** и **connect_to** будут изменены.

Для сброса и обновления данных подключения работающего соединения следует использовать блок “Connection Reset”, **Рисунок 14**.

На этом примере соединение сбросится и новые данные будут скопированы из привязанных переменных при наличии успешного соединения (блок Telemetry State) и нарастающего фронта “change_server”.

Параметры подключения можно изменять удаленно, поместив данные основных и резервных серверов в список публикуемых переменных.

Шаг 2. Определение списков удаленных переменных

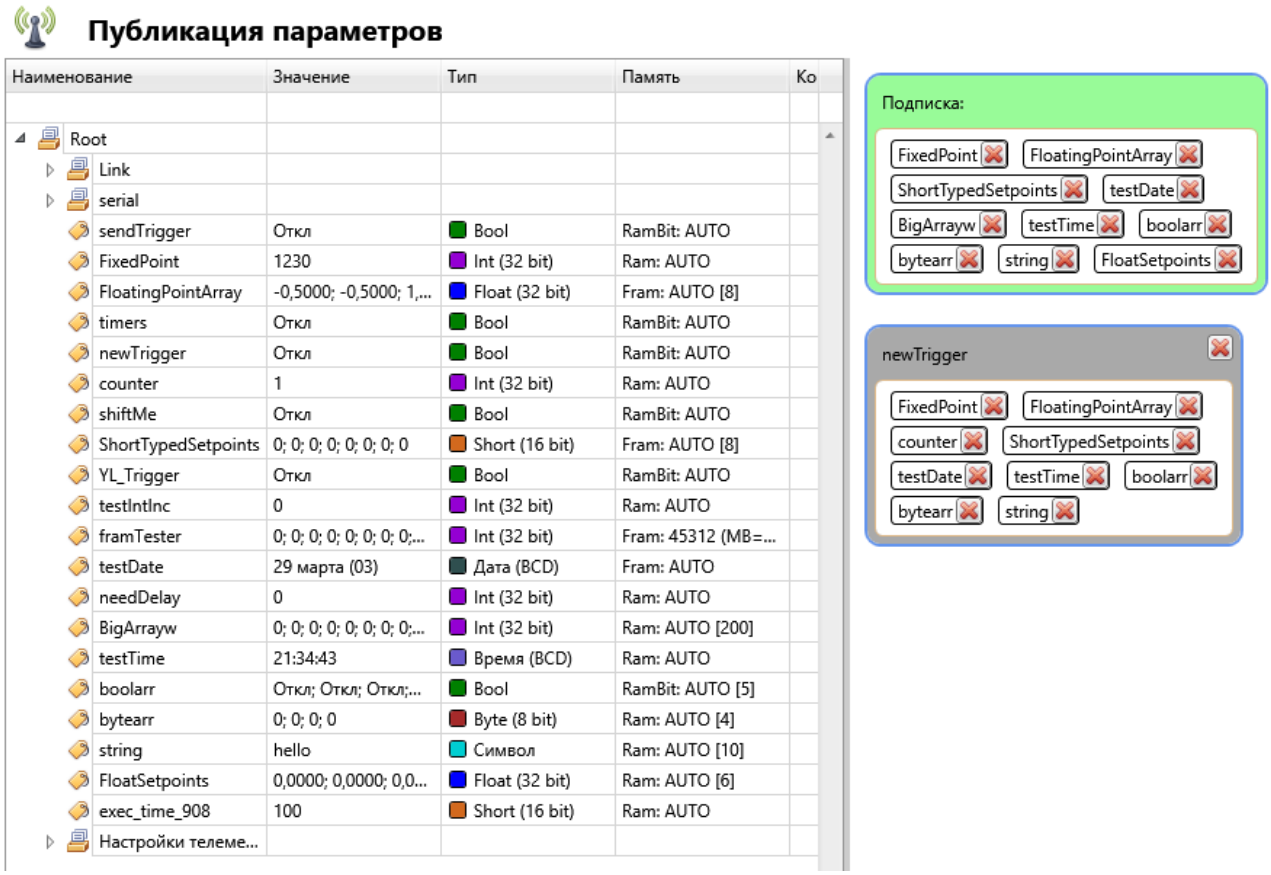


Рисунок 15. Параметры публикации

Все публикуемые переменные условно разделяются на два множества: переменные в подписке и переменные для публикации.

В списке «Подписка» (**Рис. 15**) находятся переменные, которые допускается изменять со стороны сервера – настройки, параметры работы и прочие данные.

Шаг 3. Публикация данных

Отправка данных на сервер осуществляется по флагам передачи. Флагом является любая переменная типа bool, которая объявлена в таком качестве. Для этого требуется перенести нужную переменную на правое поле – появится серый контейнер, куда аналогично можно помещать переменные, значения которых будут отправлены при наличии «1» в бите соответствующего флага.

В примере выше в качестве флага используется переменная “newTrigger”. После того, как программа присвоит данной переменной значение «1» на текущий сервер будут отправлены значения переменных:

FixedPoint, FloatingPointArray, counter, ShortTypedSetpoints, testDate, testTime, boolarr, bytearr, string.

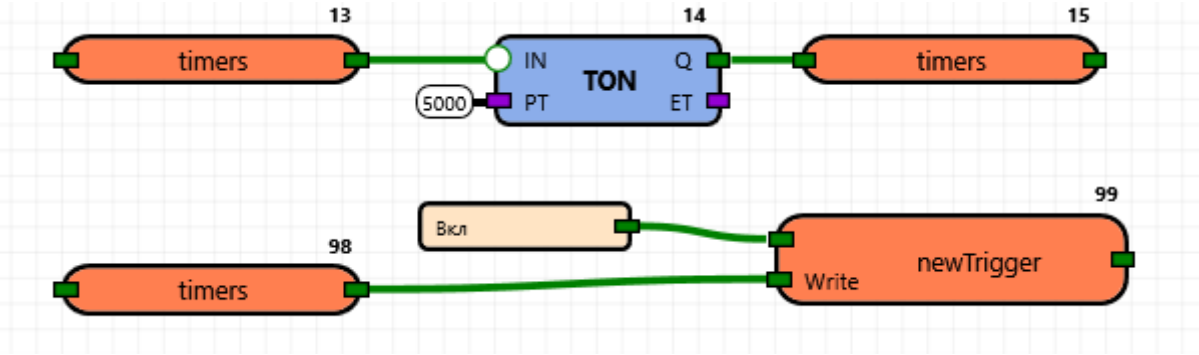


Рисунок 16. Отправка данных на сервер каждые 5 секунд

В качестве примера приложение (**Рис. 16**) отправляет данные на сервер через каждые 5000мс.

Обратите внимание, что переменной newTrigger присваивается только значение «1» - после успешной отправки данных значение переменной-флага автоматически сбросится в «0».

Переменные, перечисленные в списке «Подписка», меняются автоматически, если от сервера пришла команда на их изменение.

В случае, когда контроллеру необходимо «знать» о изменении тех или иных переменных в прикладной программе, необходимо предусмотреть дополнительные функции контроля за счет специального системного блока SubscribeEvent – данный блок выдает информацию о идентификаторе записанной переменной и флаг активации.

Одни и те же переменные могут быть перечислены в обоих множествах (подписка и передача) – контроллер будет отправлять на сервер значение тех переменных, которые также могут быть изменены удаленно.

12. Модули обмена данными

Как видно из **Рис. 9** все интерфейсные блоки реализованы отдельно от виртуальной машины что дает возможность независимой (асинхронной) работы коммуникационных модулей. Конфигурация режимов работы и получение статуса связи выполняется посредством внутреннего API, а данные для передачи передаются в/из пространство памяти контроллера независимо от работы основной программы.

В зависимости от модификации контроллера состав интерфейсных модулей может изменяться, однако основные принципы конфигурации и обмена остаются неизменными.

12.1 Modbus TCP

Modbus TCP является основным протоколом обмена для связи ПЛК с системой YartStudio. Кроме того, данный протокол поддерживается большинством производителей промышленного оборудования и используется повсеместно.

Контроллер IRIS поддерживает режимы Master и Slave одновременно.

Modbus TCP Slave активен всегда и доступен через стандартный порт 502 (TCP). Количество внешних подключений к контроллеру не ограничено.

Адресация Modbus привязана к физическим адресам контроллера, однако ряд системных адресов защищен от записи для обеспечения безопасной работы контроллера.

Наиболее удобный способ получения информации об адресации – это использование YartStudio.

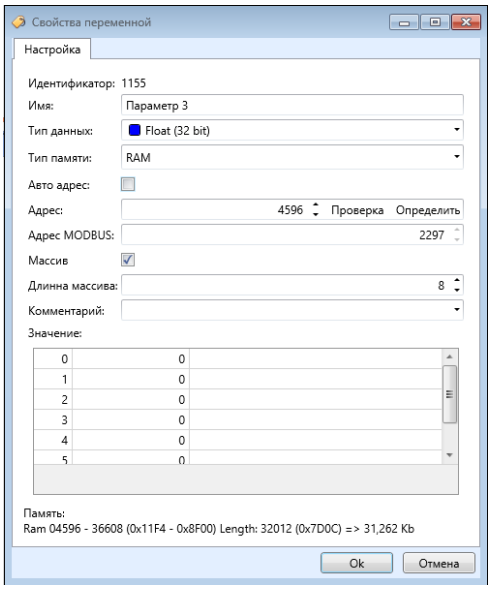


Рисунок 17. Редактирование переменных

Для точного определения любой переменной в Modbus пространстве необходимо использовать окно редактора переменных (**Рис. 17**).

YartStudio допускает два режима физического размещения переменных: автоматический и ручной (переключатель «Авто адрес»).

В автоматическом режиме размещения компилятор виртуальной машины последовательно размещает переменные в памяти данных. В этом режиме нет возможности однозначной идентификации физического адреса Modbus.

В ручном режиме пользователь сам выбирает тот адрес, где будет размещена переменная. При этом система предлагает пользователю возможность автоматизированного назначения адресов и проверки введенного адреса на пересечение с уже объявленными адресами переменных.

В поле «Адрес» указывается текущий начальный адрес размещения переменной. Кнопки «Проверить» и «Определить» позволяют осуществить проверку использованного адреса на предмет пересечений, и выбрать новый свободный в случае необходимости.

Ниже, в поле «Modbus» указан начальный адрес, по которому данная переменная будет доступна через Modbus протокол. В примере (**Рис. 17**) этот адрес равен 4596 (Holding Register).

Необходимо отметить, что стандартная спецификация Modbus не предполагает чтение переменных типа Float, более того разные платформы могут иметь различающийся формат записи чисел с плавающей точкой. Чтение Float можно реализовать как чтение двух последовательных Holding регистров и дальнейшее их объединение.

В случае линейки контроллеров Серебрум данные преобразования реализованы автоматически – достаточно прочитать/записать требуемое количество слов.

Все данные в памяти контроллера размещаются последовательно, без промежутков, поэтому в приведенном примере массив «Параметр 3» доступен по Modbus адресам 2297 – 2312.

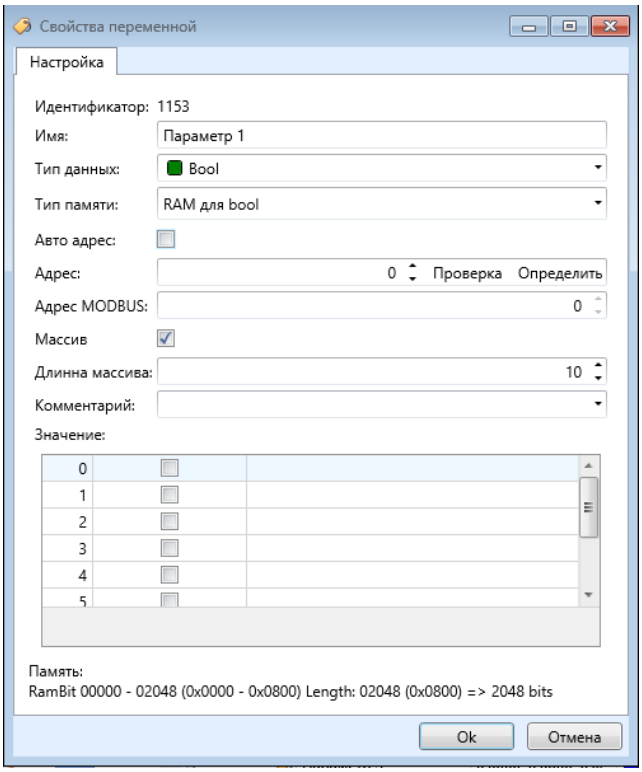


Рисунок 18. Пример с Bool

На **Рис. 18** определен массив переменных типа bool. Обратите внимание, что адрес указан как 0, поскольку доступ к bool памяти организован при помощи команд 01, 05, 15 – Coils Read/Write.

Изменение режима адресации переменной никак не сказывается на использование данной переменной в других программных модулях. Эта переменная может быть использована в модуле Modbus-RTU, телеметрии, архиве и т. д.

YartStudio позволяет осуществить экспорт списка переменных для последующей интеграции в продукты других компаний.

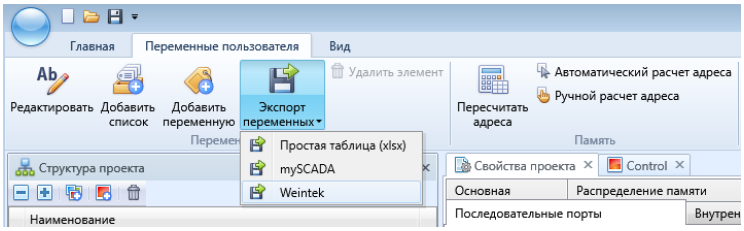


Рисунок 19. Меню экспорта списка переменных

В пункте основного меню «Экспорт переменных» можно выбрать формат выходных данных и осуществить экспорт. В зависимости от выбранного формата определенные в ручном режиме переменные будут сохранены в файл на диск.

12.1.1 Modbus TCP Master

Данный режим позволяет ПЛК IRIS работать с внешними устройствами, подключенными через интерфейс Ethernet.

Работа данного модуля выполняется в автоматическом режиме по заранее заданной конфигурации из прикладной программы.

На примере (**Рис. 20**) показана программа инициализации и контроля обмена по протоколу Modbus TCP Master.

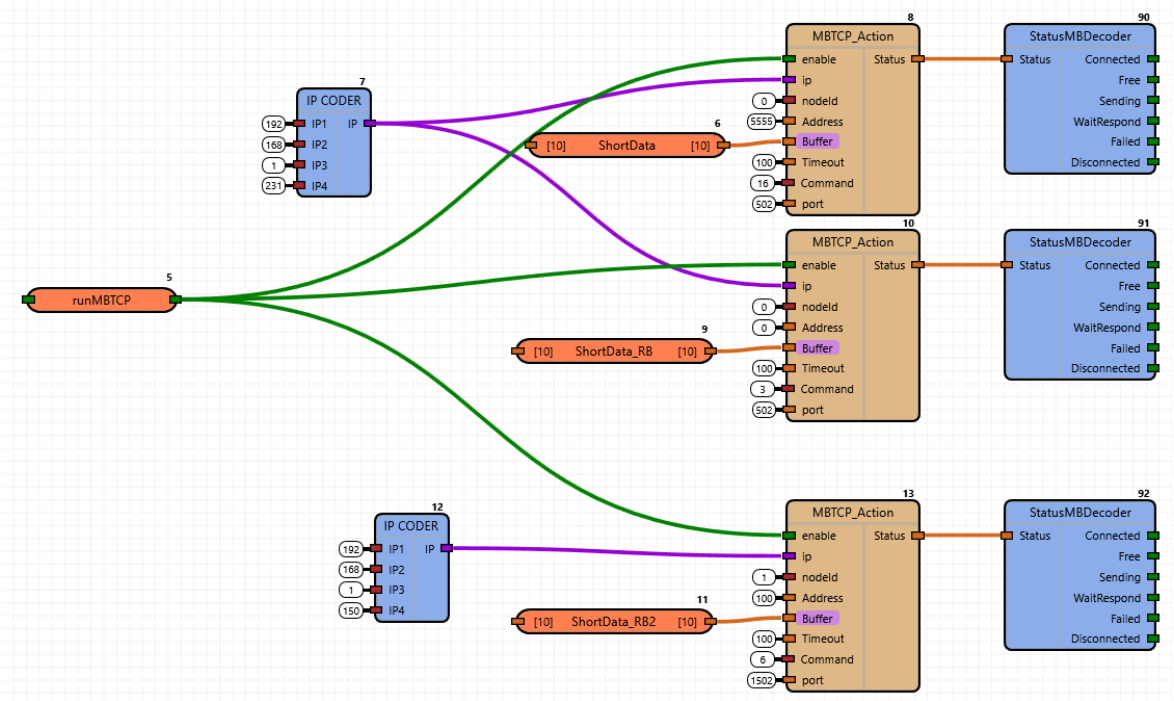


Рисунок 20. Работа с Modbus TCP Master

В данном примере IRIS работает с двумя сетевыми устройствами, расположенными по адресам 192.168.1.231 и 192.168.1.150.

Для устройства с IP адресом 192.168.1.231 определены две команды: 16 – Запись нескольких Holding регистров и 03 - чтение Holding регистров.

Команды ***MBTCP_Action*** сконфигурированы таким образом, что при записи данных во внешнее устройство данные берутся из переменной «ShortData», а чтение выполняется в переменную «ShortData_RB».

Для другого устройства, с IP 192.168.1.150 определена только одна команда 06 – запись одного регистра.

Обратите внимание, что порт второго устройства указан как 1502 (не стандартный для Modbus TCP).

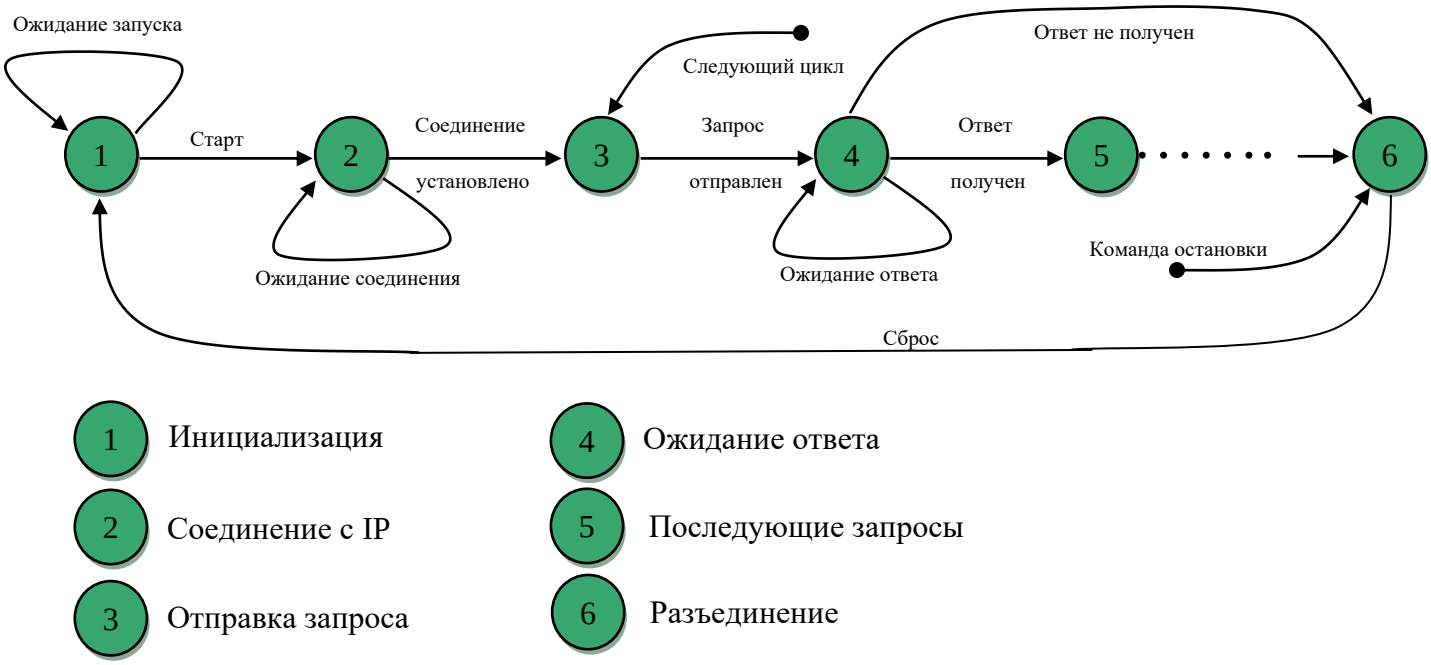


Рисунок 21. Диаграмма работы модуля Modbus TCP Master

Для каждого удаленного устройства (уникальный IP адрес) IRIS создает отдельный поток опроса данных. На **Рис. 21** показана обработка одного потока Modbus TCP Master.

После старта потока контроллер пытается установить TCP соединение (2) с удаленным устройством. Затем начинается отправка команд (3) чтения/записи данных. Для каждой из команд можно установить свое время ожидания отклика (4). Как только отклик получен для отработанной команды выставляет значение статусного регистра (выход блоков 8, 10, 13, Рис. 20) и программа переходит к отправке следующего запроса (5).

В случае, если в заданное время ответ не получен программа осуществляет отключение от внешнего хоста (6) и переходит к шагу инициализации (1).

Также состояние (6) активируется в случае остановки прикладной программы контроллера для перезагрузки.

При необходимости исполнение той или иной команды может быть остановлено, см. входы «Enable» блоков 8, 10, 13, Рис. 20.

Для расшифровки текущего состояния команды Modbus TCP используется функциональный блок StatusMBDecoder.

12.2 Modbus RTU

Работа Modbus RTU осуществляется через последовательные порты контроллера.

В **Таблице 10** указано обозначение портов.

Для контроллера IRIS COM1 всегда настроен в режиме Modbus RTU SLAVE. Оставшиеся два порта настраиваются через вкладку настройки проекта «Последовательные порты»

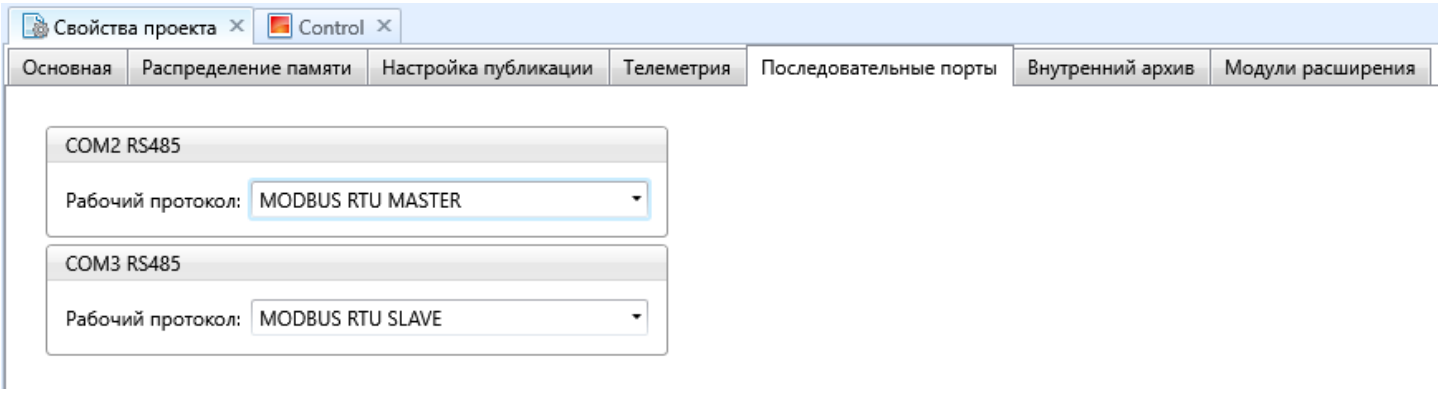


Рисунок 22. Базовая настройка последовательных портов

Для каждого порта допускается три рабочих режима, каждый из которых будет рассмотрен далее:

- Modbus RTU Master
- Modbus RTU Slave
- Пользовательский протокол

Таблица 10. Последовательные порты

Именование	Расположение	Обозначение в программе
COM1	Последовательный порт через USB, поз. 4, Рис. 1	1
COM2	Изолированный порт, поз. 2, Рис. 1	2
COM3	Основной порт, поз. 1, Рис. 1	3

В режиме Modbus RTU Slave последовательный порт предоставляет доступ сторонним устройствам к записи/чтению данных памяти контроллера.

Система адресации и порядок распределения памяти полностью аналогичен Modbus TCP Slave (раздел 8.1).

Однако команды доступа к Bool разделам памяти недоступны.

Для правильного использования режима Modbus RTU Slave в прикладной программе необходимо указать локальный адрес устройства и режим работы последовательного порта (**Рис. 22**).

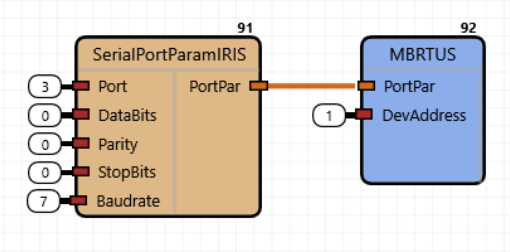


Рисунок 23. Настройка параметров Modbus RTU Slave

В блоке ***SerialPortParamIRIS*** указывается условный номер порта (**Таблица 10**), и параметры обмена:

DataBits – Код числа бит в слове обмена

Parity – Режим четности

StopBits – Число Стоп бит

Baudrate – Скорость обмена

Подробная информация по командам содержится в описании программирования ПЛК Серебрум

В примере (**Рис. 23**) ***DataBits*** = 0 (8 бит), ***Parity*** = 0 (нет бита четности), ***StopBits*** = 0 (один стоп бит), ***Baudrate*** = 7 (скорость обмена 57600 бод).

При получении корректных Modbus RTU запросов на указанный порт IRIS автоматически сгенерирует и отправит ответ. Участия прикладной программы не требуется.

Участок кода, показанный на примере, **Рис. 23**, будет выполнен лишь в самом начале работы программы, соответственно параметры работы последовательного порта устанавливаются на всем протяжении времени жизни программы.

При необходимости изменить сетевой адрес или режим соединения потребуется перезагрузка прикладной программы контроллера.

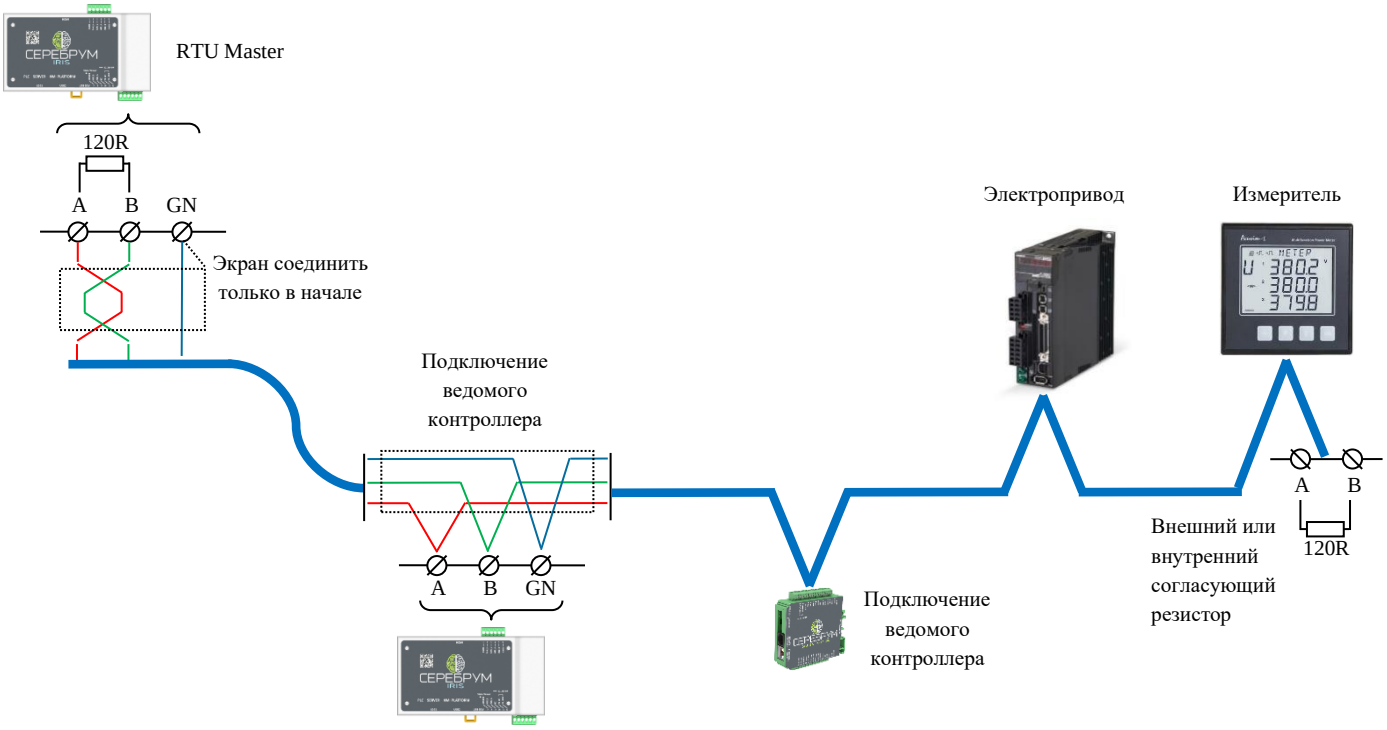


Рисунок 24. Подключение к шине Modbus RTU

На **Рисунке 24** показана приблизительная схема подключения устройств в шине RS-485 (Modbus RTU). В качестве ведущего устройства выступает контроллер IRIS – COM3.

Поскольку в данной конфигурации ведущий физически расположен в самом начале шины, то для порта COM3 включается согласующий резистор 120 Ом. Для этого устанавливается джампер (**поз. 11, Рис. 1**).

Другой контроллер IRIS подключается к шине через порт COM2.

На оконечном устройстве (в данном примере это Измеритель) также должен быть подключен согласующий резистор.

12.2.1 Modbus RTU Master

Если COM порт сконфигурирован в проекте в качестве ведущего устройства, то автоматически инициализируется программа обмена по сети Modbus RTU.

Работа и конфигурация данной программы во много схожа с Modbus TCP Master.

На **Рисунке 25** приведен пример программы настройки обмена через порт Modbus RTU Master.

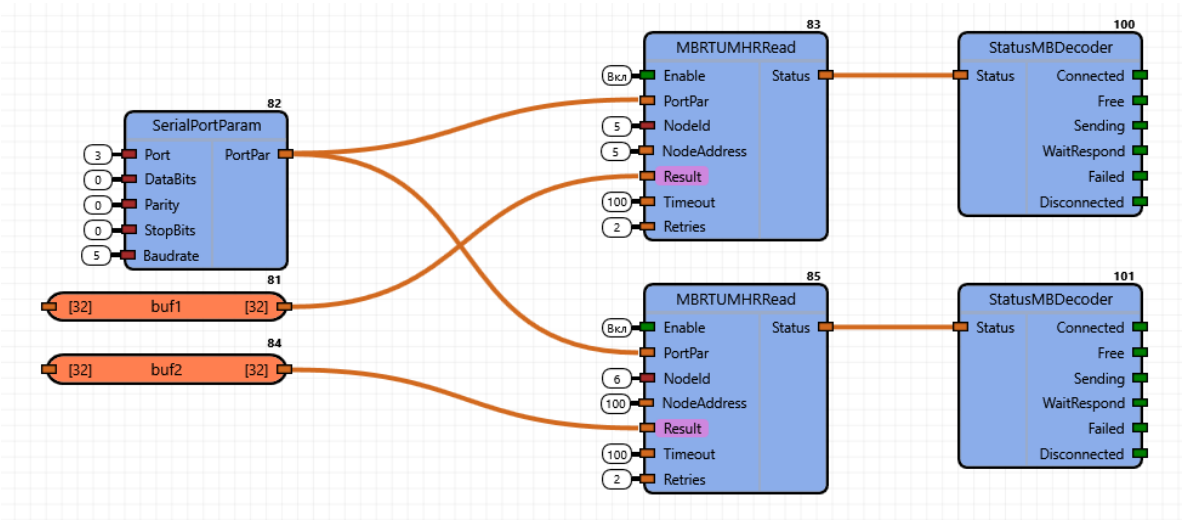


Рисунок 25. Конфигурация обмена Modbus RTU Master

Для каждого коммуникационного слота (команда для обмена) может быть определена своя уникальная настройка последовательного порта. В ряде случаев такой подход позволяет организовать несколько RTU сетей, работающих параллельно.

В примере настройки порта не изменяются и используется порт COM3, 19200 бод в формате 8N1 (блок 82 – ***SerialPortParam***).

Команда 83 – ***MBRTUMHRRead*** конфигурирует коммуникационный слот для чтения группы Holding регистров из устройства по сетевому адресу 5 (***NodeId***). Адрес первого регистра на удаленном устройстве тоже 5 (***NodeAddress***). Данные будут прочитаны в пользовательскую переменную buf1.

Обратите внимание, что длина запроса берется равной длине пользовательской переменной buf1, т. е. 32 слова.

Для данной команды определен таймаут отклика ведомого устройства (Timeout) 100 мс. Если отклик не будет получен, то программа Modbus RTU Master повторит запрос 2 (Retries) раза.

Статус выполнения команды расшифровывается в блоке ***StatusMBDecoder***.

Аналогично настроена и команда 85, за исключением того, что обращение производится к устройству по адресу 6, начальный адрес регистра 100.

В отличие от программы Modbus TCP Master в системе RTU запросы делятся по используемым ими портам. Каждый порт обрабатывается индивидуально. Запросы, запрошенные в программе для порта, следуют поочередно (**Рис. 26**).

При необходимости запрос может быть пропущен, если вход Enable при начале обработки запроса равен «0».

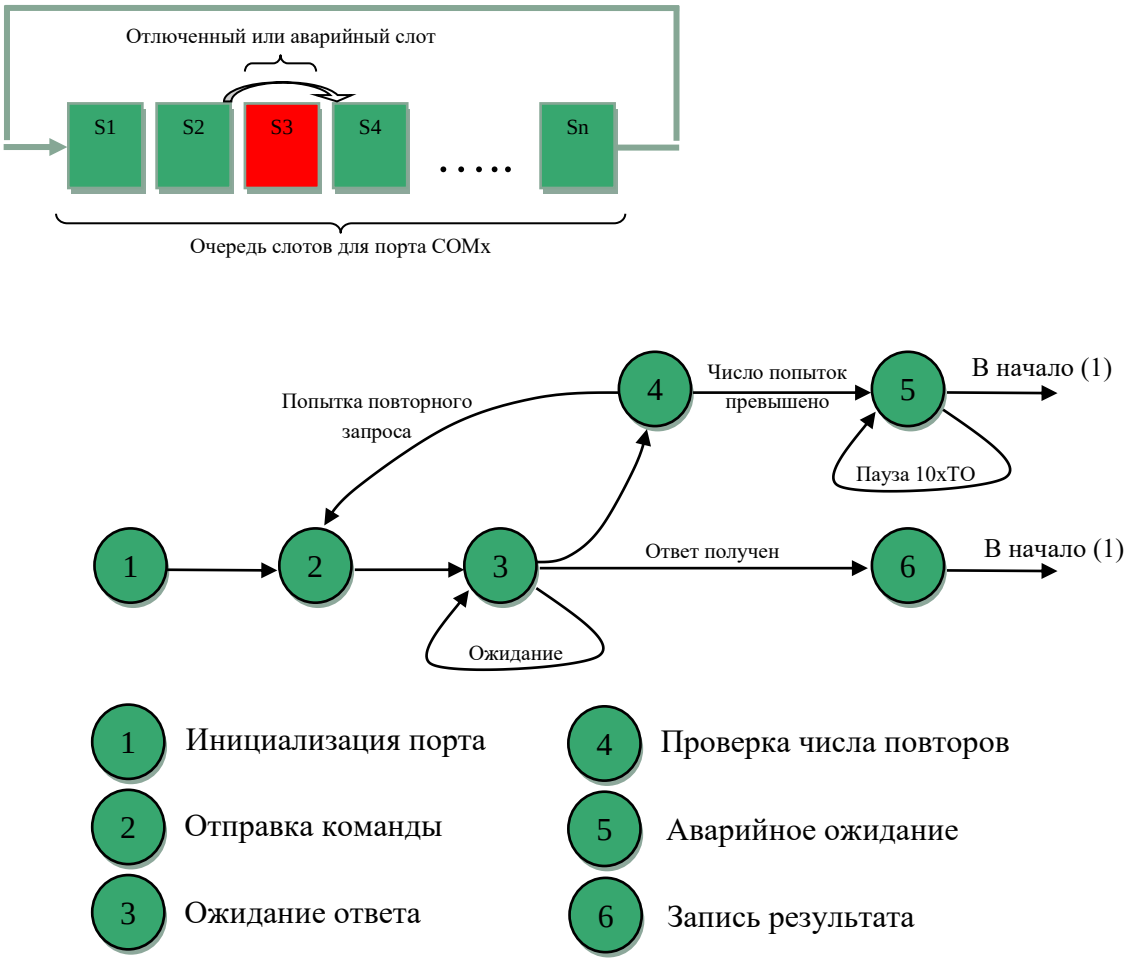


Рисунок 26. Работа слотов и диаграмма состояний коммуникационного слота.

Также слот пропускается в том случае, если все попытки получить отклик от удаленного устройства были неудачными (состояние 5). Пауза, в течение которой слот не будет задействован как правило равна десяти таймаутам для данной команды.

При успешном завершении процедуры обмена (состояние 6) программа перейдет к обслуживанию следующего по очереди слота.

Обработка очереди запросов прекращается если контроллер переходит в состояние Стоп. Кроме того, очередь сбрасывается при каждом новом запуске контроллера.

12.3 Пользовательские протоколы

В контроллере IRIS предусмотрен API для самостоятельного программирования протоколов обмена через последовательные порты.

При необходимости использования собственного протокола (или какого-либо из поставляемых в составе библиотеки функциональных блоков) необходимо в окне основного режима COM портов (**Рис. 22**) выбрать пункт «Пользовательский протокол».

API рассчитан на использование языка C-Yart и содержит следующие функции:

Попытка открытия порта с заданными аргументами.

port_open (port, baud, flags)

В процессе выполнения данной команды операционная система проверят правильность конфигурации порта (режим «Пользовательский протокол»), доступность порта (порт не уже не открыт) и конфигурирует порт в соответствии с заданными в команде параметрами.

Значение параметра port должно быть равно коду порта (**Табл. 10**).

baud – значение скорости обмена: 9600, 19200, и т.д.

Поддерживаются следующие значения:

- 1200 бод
- 2400 бод
- 4800 бод
- 9600 бод
- 19200 бод
- 38400 бод
- 57600 бод
- 115200 бод

flags - битовое поле конфигурации

биты 31-3	**бит 2**	**бит 1**	**бит 0**
зарезервировано	STOPB	Parity	

STOPB: 1 – два стоповых бита; 0 – 1 стоповый бит

Parity: 0 – нет четности; 01 – Нечетный режим; 10 – Четный режим

В случае успешного открытия порта функция вернет значение 1.

В случае ошибки значения будут:

- -1 – порт уже открыт

- -2 – некорректный режим порта

Для закрытия порта используется функция ***port_close(port)***

port – аналогично ***port_open***

Все соединения сделанные ранее разрываются, параметры сбрасываются. Если порт находился в режиме приема или передачи данных, то работа будет остановлена.

port_check (port) – проверяет текущий режим порта. Функция возвращает «1», если порт открыт и «0», если порт закрыт.

port_send (port, address, len) – функция отправки сообщения через последовательный порт

port – индекс порта для отправки сообщения. Перед отправкой порт должен быть открыт

address – адрес в пространстве оперативной памяти (**Табл. 8**) где находится сообщение

len – длина сообщения в байтах

Функция возвращает:

- 0 - отправка началась
- -1 – ошибочный порт
- -5 – Оправка не удалась, передатчик занят
- -6 – Ошибка работы порта

port_receive (port, address, len)

port – индекс порта для получения сообщения. Перед приемом порт должен быть открыт

address – адрес в пространстве оперативной памяти (Табл. 8) где будет находиться сообщение

len – длина сообщения в байтах

Для информации о текущем статусе приема или передачи предусмотрены функции:

port_transmit_status (port) – проверка очереди передачи

port_receive_status (port) – проверка очереди приема

Функции возвращают число байт, оставшихся к приему или передаче.

0 – означает, что операция завершена

Отрицательное число обозначает ошибку.

stop_port_receive(port) – останавливает прием данных. ***port_receive_status()*** будет возвращать 0.

12.4 Yart Link

YartLink — это протокол межконтроллерного обмена данными Серебрум. Он основан на UDP датаграммах и позволяет разворачивать децентрализованные решения на основе ПЛК Серебрум.

При этом один источник может передавать данные множеству получателей без потери производительности и загрузки сетевого интерфейса.

Получатели данных, в свою очередь, "подписываются" на публичную рассылку от нужного источника. Данные приходят асинхронно от выполнения основной программы. Факт получения регистрируется временной меткой, по которой прикладная программа может сделать вывод о достоверности текущего значения.

Настройка работы YartLink осуществляется отдельно на источнике и приемниках данных.

В источнике данных формируется датаграмма для передачи в сеть. Размер датаграммы ограничен и равен 1024 байт. В процессе работы программы пользователя датаграмма заполняется через блоки YL xx, где xx - тип данных переменной на входе блока, **Рис. 27**.

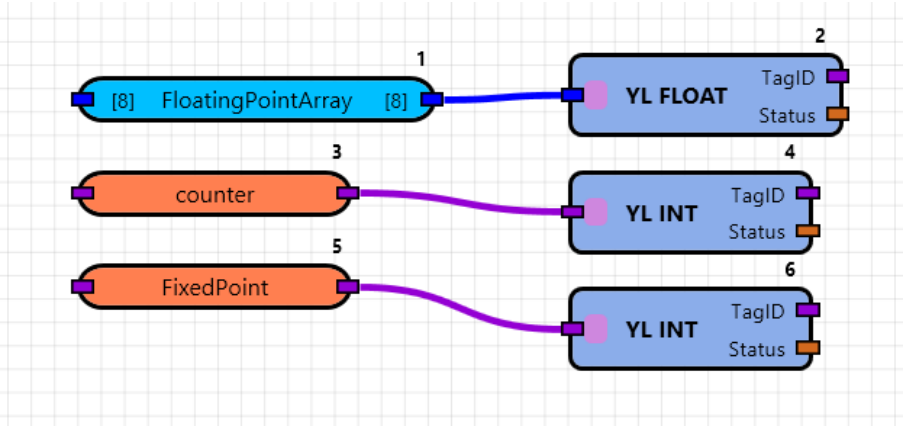


Рисунок 27. Пример программы передатчика YartLink

В процессе работы программы выход блока формирования пакета выдает идентификатор переменной для подписки и статус выполнения операции.

Статус операции подготовки может принимать 3 значения:

- 0 – завершено без ошибок
- 1 – не корректный идентификатор переменной
- 2 – тип переменной не поддерживается
- 3 – пакет отправки переполнен

Отправка подготовленного пакета осуществляется или периодически, блоком "***YL SEND PERIODIC***", либо блоком "***YL SEND***", который начинает отправку блока при наличии "1" на входе.

Фактически датаграмма будет отправлена синхронно, после окончания выполнения такта программы пользователя, в котором был активирован флаг передачи сообщения.

Датаграмма отправляется в широковещательном режиме - дальнейшая обработка будет осуществлена на каждом подключенном устройстве.

Для получения опубликованных данных на приемном контроллере должна быть сконфигурирована подписка.

Для подписки (пример на **Рис. 28**) необходимо указать IP адрес передающего контроллера, идентификатор переменной, которая передается (выход Status блоков ***YL XX***), а также ссылка на переменную куда будет записано значение.

Важной особенностью является то, что типы данных и длина должно полностью совпадать.

Выход Duration показывает время с момента получения последних данных от источника. При реализации ответственных объектов данный параметр необходимо контролировать и переводить систему в безопасное состояние в случае, если данное время превысило предопределенный порог.

Каждая подписка должна иметь свой уникальный идентификатор (RecID), программируемый в диапазоне 0-63. В примере используется специализированный функциональный блок "ID CONTROL", формирующий код RecID на основе предыдущего.

Как видно из параметра RecID всего в контроллере допускается до 64 подписок на глобальные переменные одновременно.

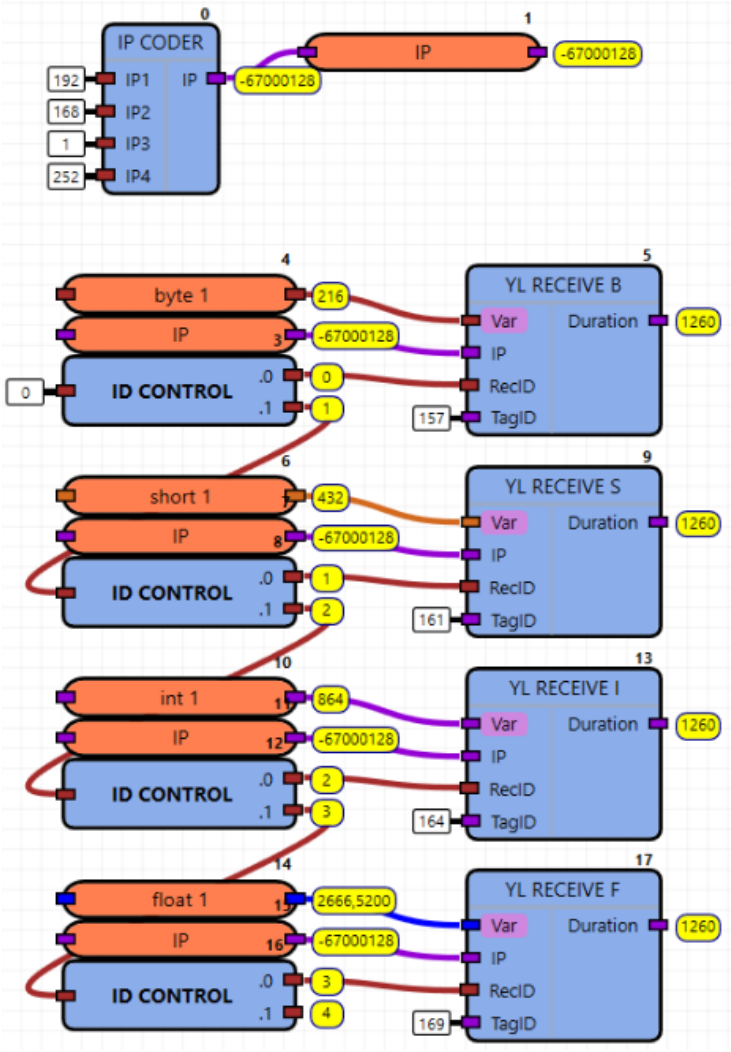


Рисунок 28. Подписка на глобальные публикуемые данные

12.5 USB порт Yart Studio

В ряде случаев сетевое соединение с ПЛК невозможно.

Для прямого подключения к IRIS допускается использование USB подключения через встроенный порт (поз. 4 Рис. 1). При подключении к ПК в системе появится дополнительный USB-COM порт.

ОС должна определить новое устройство (***Gadget Serial***, **Рис. 29**).

Официальный inf файл для работы устройства можно скачать по ссылке

https://www.kernel.org/doc/Documentation/usb/linux-cdc-acm.inf

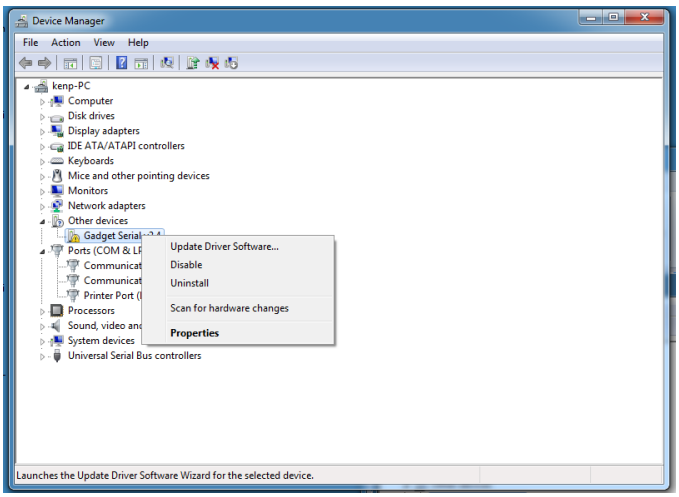


Рисунок 29. Подключение IRIS к ПК через USB порт

Доступ к данным IRIS будет осуществляться через протокол Modbus-RTU по адресу 1.

12.6 Обмен со встроенным сервером телеметрии

Важной особенностью IRIS является возможность осуществлять обмен со встроенным сервером телеметрии. Это дает возможность обработки удаленных данных в прикладной программе контроллера и формирования управляющих команд для разнесенных систем управления.

Непосредственная работа с данными телеметрии осуществляется через набор функциональных блоков чтения и записи информации в/из сервер телеметрии (**Рис. 30**).

Рисунок 30. Блоки обена с сервером телеметрии.

Процесс обмена данными с сервером длительный, поэтому программа контроллера динамически составляет список переменных для передачи на сервер.

Для настройки команды требуется указать следующие параметры:

URI - имя глобального тега (см. **Раздел 11.1**). Например: ***my_PLC:Variables:Process Data:Pressure***

VarID - идентификатор внутренней переменной, куда будут сохранены или откуда будут считаны данные

Write/Read - битовый вход, нарастающий фронт которого поместит команду в очередь исполнения.

В выходном поле Readback содержится текущий статус обработки команды:

- 0 - инициализация
- 1 - ожидание ответа
- 2 - ошибка обмена данными
- 3 - команда выполнена
- 4 - данные отсутствуют (команда была выполнена, но сервер не вернул данных)
- 5 - ошибка конфигурации
- 6 - ошибка записи

Как видно, в описания команд тип данных не указывается. При формировании запроса к серверу система автоматически сверяет тип переменной, переданной по идентификатору. В случае, если тип данных в ПЛК отличается от типа данных на сервере - произойдет автоматическое преобразование типов.

В ряде случае данное преобразование может повредить данные (например Float в Int) - при проектировании программы необходимо убедиться, что данные для записи и чтения сопоставимы с существующими тегами, хранящимися на сервере телеметрии.

13. Работа с периферийными модулями

Настройка работы периферийных модулей производится в разделе «Состав оборудование». В данном разделе отображаются модули, добавленные в разделе настройки проекта – «Модули расширения» (**Рис. 7**).

Для модулей аналогового ввода необходим также выбор основного режима работы.

В процессе работы контроллера производится опрос внешних модулей расширения: модули ввода опрашиваются до старта очередного цикла программы, в модули вывода данные заносятся после выполнения программы.

Поскольку скорость выполнения прикладной программы может превышать физические характеристики периферийных модулей (например AI8) – фактические данные измерений будут обновляться не на каждом такте исполнения прикладной программы.

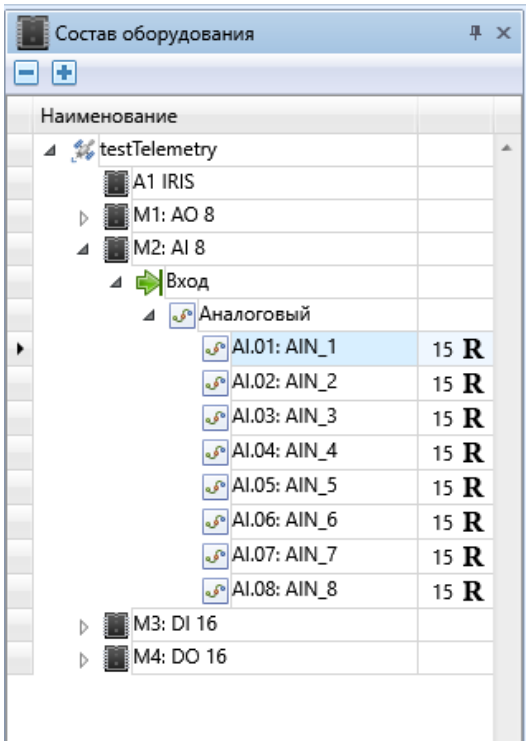


Рисунок 31. Панель настройки периферийных модулей

13.1 Аналоговые входы

Модуль AI8 (**Рис. 32, 33**) основан на 12-разрядном АЦП со встроенным программируемым усилителем.

Подключение модуля рекомендуется выполнять в соответствии со схемой (**Рис. 32**). При этом клеммы «Earth», 1, 16, 17, 32 должны подключаться к контуру заземления шкафа непосредственно в месте установки модуля.

Необходимо подключение всех клемм заземления.

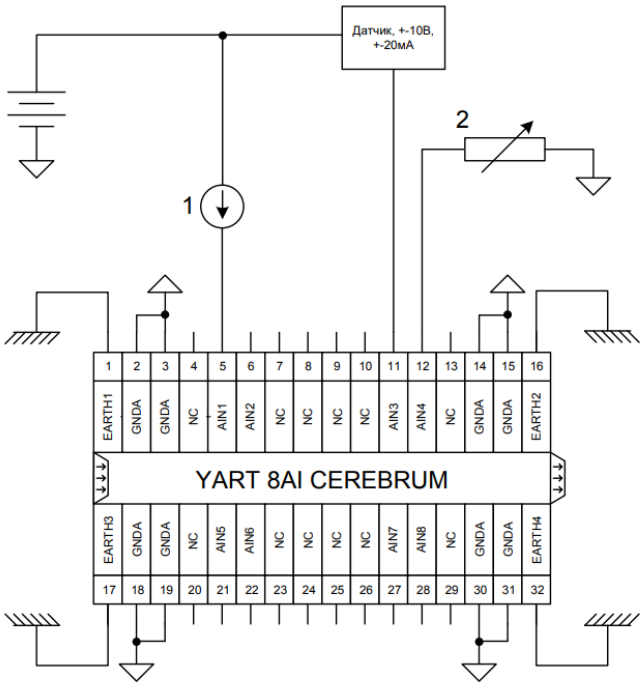


Рисунок 32. Модуль 8AI

На схеме (**Рис. 32**) цифрами обозначены:

1 – Токовый датчик, подключенный по 2-проводной схеме
2 – Измеряемое внешнее сопротивление (датчик температуры, дискретный вход)

Все каналы модуля имеют групповую гальваническую развязку от напряжения питания шины расширения и контроллера.

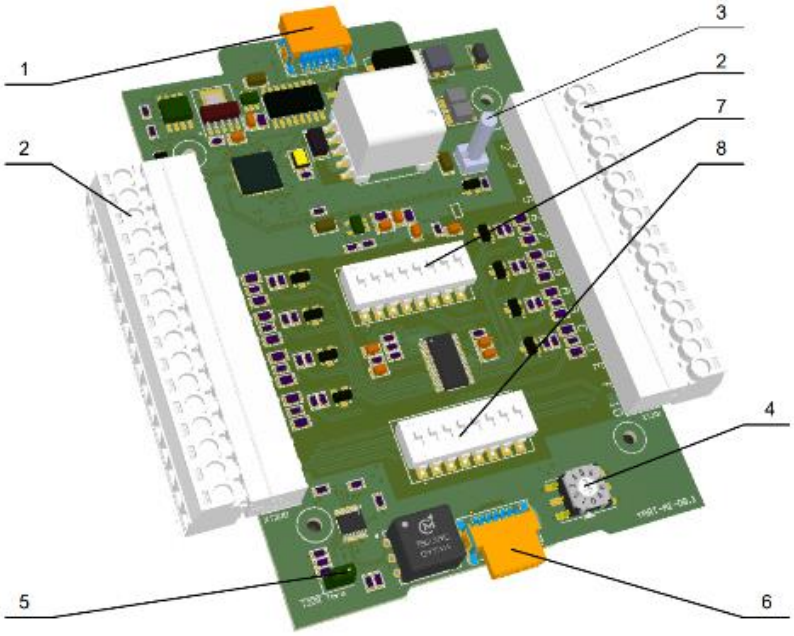


Рисунок 33. Модуль 8AI

1 – разъем для подключения к контроллеру или «ведущему» модулю расширения
2 – группа клемм для подключения внешних соединителей
3 – индикатор состояния работы модуля
4 – переключатель адреса в шине модулей расширения
5 – перемычка включения терминатора линии передачи данных
6 – разъем для подключения последующих модулей расширения
7 – переключатели 1 активации режима измерения сопротивления
8 – переключатели 2 активации режима измерения тока

Кроме того, входной каскад АЦП имеет внешние переключатели для выбора режима работы каналов. Доступно три основных режима (**Таблица 11**):

- измерение напряжения
- измерение тока
- измерение сопротивления

Таблица 11. Режимы работы модуля AI8

Режим работы	Переключатель 1	Переключатель 2
Тестовый режим	Включен	Включен
Измерение сопротивления	Включен	Выключен
Измерение тока	Выключен	Включен
Измерение напряжения	Выключен	Выключен

Каждый канал настраивается индивидуально. В зависимости от настроек программируемого усилителя изменяется входной диапазон измерений, в рамках которого работает 12-разрядный АЦП. Специальные

блоки программного обеспечения ПЛК выполняют дальнейший пересчет полученных кодов АЦП, предоставляя пользователю уже обработанные данные (в физических величинах: В, мА, Ом).

Обратите внимание, что при измерении биполярного напряжения (или тока) 12 разрядов распределяются на весь диапазон измерения, т. е. 11 разрядов АЦП на положительное напряжение и 11 разрядов на отрицательное напряжение.

Также предусмотрены режимы, когда выходом блока AI8 являются необработанные данные АЦП.

Дальнейшие настройки выполняются в YartStudio в соответствии с Таблицей 11

Измерение напряжения

Доступны восемь режимов:

- 0-1.25В / ±0-1.25В
- 0-2.5В / ±0-2.5В
- 0-5.0В / ±0-5.0В

Изменение тока

Все указанные диапазоны приведены с учетом подключаемого нагрузочного резистора 250 Ом.

- 0-20.0mA / ±0-20.0mA
- 0-10.0mA / ±0-10.0mA
- 0-5.0mA / ±0-5.0mA
- 0-2.5mA / ±0-2.5mA

Измерение входного сопротивления / дискретный вход

Измерение сопротивления осуществляется за счет измерения падения напряжения на входе канала при подаче нормированного тока возбуждения.

Схема измерения не является дифференциальной, поэтому подвержена влиянию синфазной помехи, возникающей на длинных кабелях.

При необходимости измерения сопротивления на расстояниях более 10 метров, а также при работе каналов измерения рядом с мощными силовыми установками рекомендуется использовать преобразователи сопротивление/ток.

Доступно три диапазона измерения:

- 0-4k
- 0-18k
- 0-1M

Канал измерения сопротивления способен имитировать дискретный вход. Программа автоматически определяет состояние низкого входного сопротивления, формируя «0» на выходе.

Для использования данных модуля AI8 в прикладной программе (Рисунок 35) необходимо использовать индивидуальный функциональный блок Aix или весь модуль сразу.

Добавление входа в программу осуществляется путем перетаскивания из выпадающего списка каналов (**Рис. 34**).

Пункт списка «Аналоговый» добавит все каналы одновременно.

Возможно добавления каждого канала по отдельности, выбирая элементы AI.xx:AIN_x.

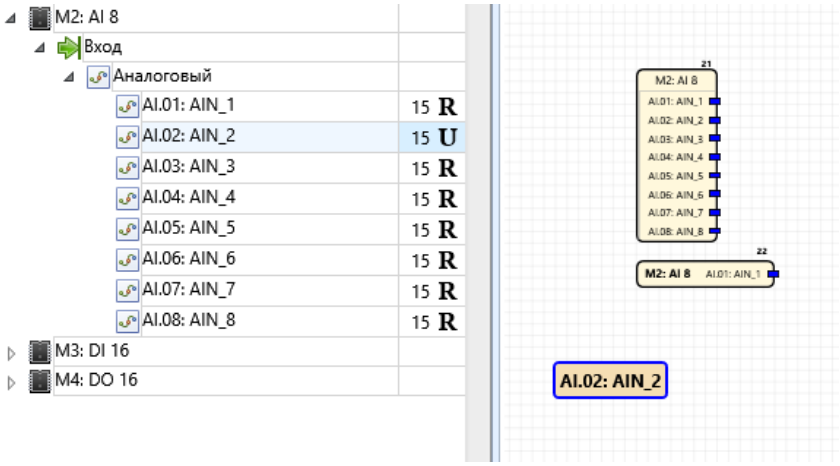


Рисунок 34. Функциональные блоки аналоговых входов

Выходы программных блоков модуля AI формируют выходное значение в формате с плавающей точкой.

Значения выходов соответствуют измеренным физическим величинам в зависимости от выбранного режима работы (R, U, I, DI).

В режиме DI выход имеет тип Bool.

В состав встроенной библиотеки включены уже готовые функциональные блоки для осуществления дальнейших преобразований: датчики температуры, давления и т. д.

13.2 Аналоговые выходы

Модуль 8АО (**Рис. 35**) формирует восемь независимых каналов напряжения и тока. Формирователи тока и напряжения независимые и могут работать одновременно (каждый выходной канал имеет две клеммы: ток и напряжение).

Для работы токовых каналов модулю необходимо внешнее питание (поз. 1 и 2, **Рис 35**). Для устойчивой работы модуля достаточно запитать один вход – второй может оставаться в резерве.

Все каналы модуля имеют групповую гальваническую развязку от напряжения питания шины расширения и контроллера.

Для фильтрации помех предусмотрены клеммы 1, 17, 32, 16 (Earth), которые должны присоединяться к заземляющему контуру системы управления.

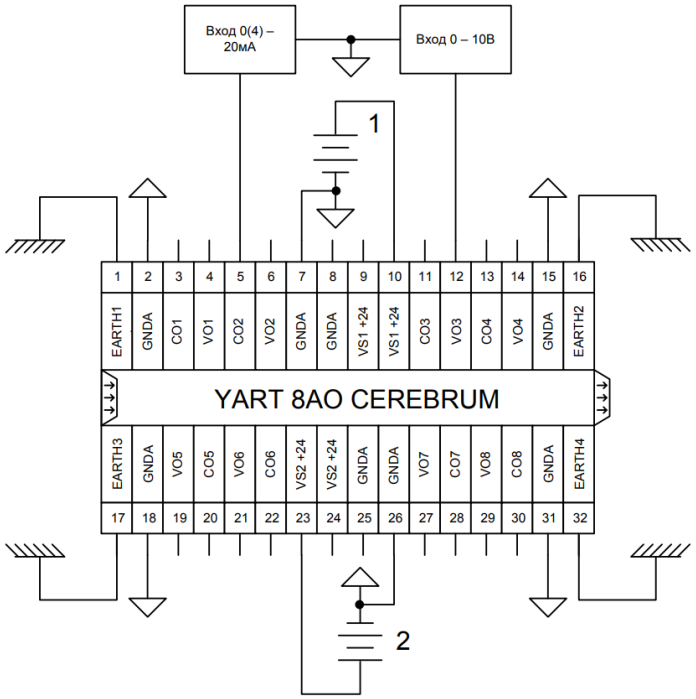


Рисунок 35. Модуль 8АО

Клеммы COx – токовые выходы 0 -20мА

Клеммы Vox – выходы напряжения 0 -10В

Для использования аналогового выхода в программе соответствующий блок необходимо разместить на программном поле, аналогично п. 9.1, Рис. 31.

На вход блока (Рис. 36) подается число в формате с плавающей точкой в диапазоне 0 -100.0

0.0 соответствует выходному току/напряжению 0мА/0.0В

100 соответствует выходному току/напряжению 20мА/10.0В

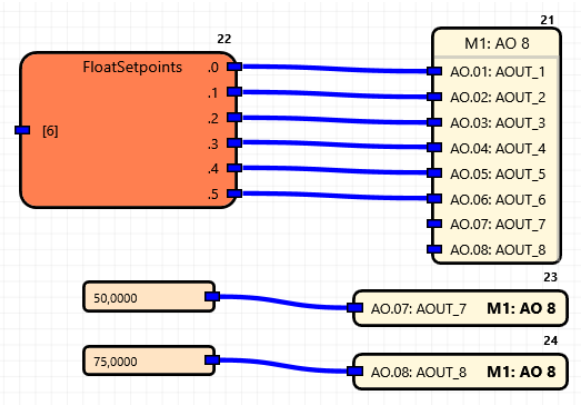


Рисунок 36. Использование модулей аналоговых выходов.

13.3 Дискретные входы

Модуль 16DI содержит 16 входных каналов для дискретных сигналов. Работа входных каскадов модуля осуществляется от внешнего источника питания. При этом все каналы модуля гальванически развязаны от напряжения питания шины расширения.

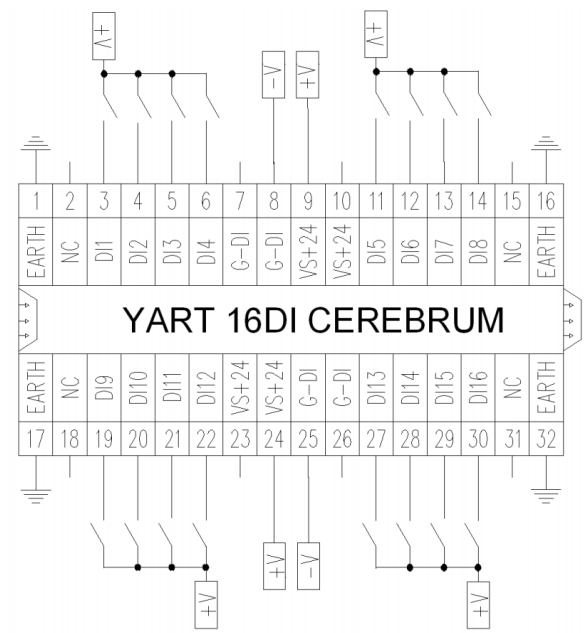


Рисунок 37. Модуль 16DI

Для устойчивой работы модуля достаточно присоединить питание к одной паре клемм.

Клеммы 1, 16, 17, 32 (**Рис. 37**) должны быть присоединены к внутреннему контуру заземления.

Аналогично модулям 8AI 8AO модуль 16DI представлен в виде функционального блока, доступного для размещения на поле программы (**Рис. 38**).

Блоки допускается размещать в любом месте программы.

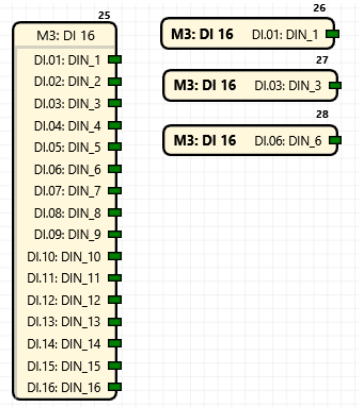


Рисунок 38. Блоки дискретного ввода

13.4 Дискретные выходы

Модуль 16DO содержит 16 выходных каналов с максимальной токовой нагрузкой 1А. Для каждого выходного блока из восьми линий (**Рис. 39**) предусмотрен отдельный вход питания.

При этом все каналы модуля гальванически развязаны от напряжения питания шины расширения.

Выходы модуля имеют встроенную защиту от перегрузки.

Клеммы 1, 16, 17, 32 (**Рис. 39**) должны быть присоединены к внутреннему контуру заземления.

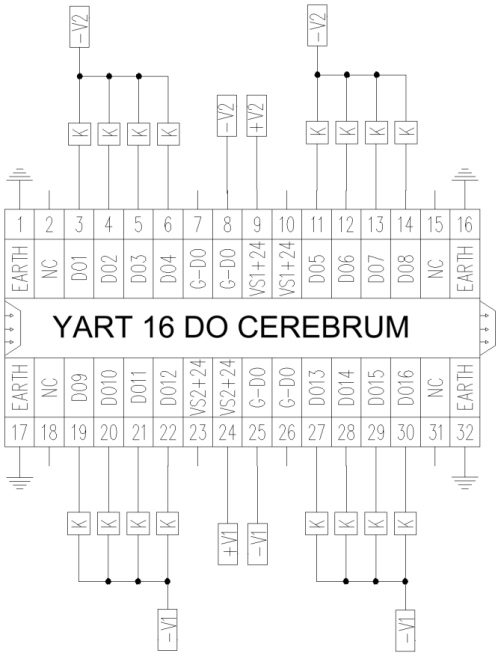


Рисунок 39. Модуль 16DO

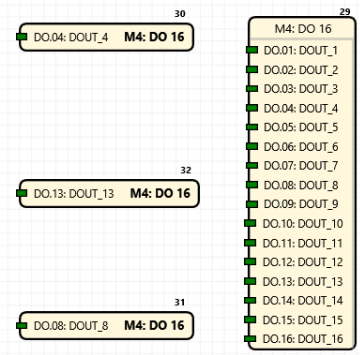


Рисунок 40. Функциональные блоки DO

Значения дискретных выходов задаются при помощи функциональных блоков DO (**Рис. 40**). Данные блоки могут быть размещены в любом месте программы. Компилятор YartStudio автоматически проверяет однократное использование каждого выхода.

При многократной попытке записи в один и тот же канал будет сгенерирована ошибка компиляции.

14. Архив данных

Встроенный слот для uSD карты (**поз. 5 Рис. 1**) позволяет сформировать локальный архив исторических данных контроллера.

В процессе выполнения прикладной программы данные, указанные в окне "Внутренний архив" (**Рис. 41**) будут автоматически сохранены на карту.

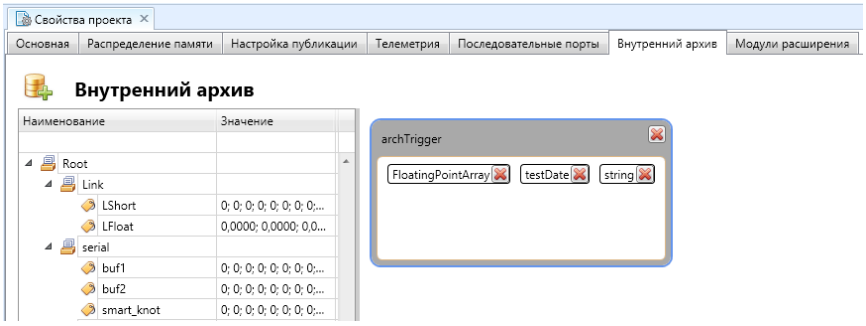


Рисунок 41. Настройка встроенного архива данных

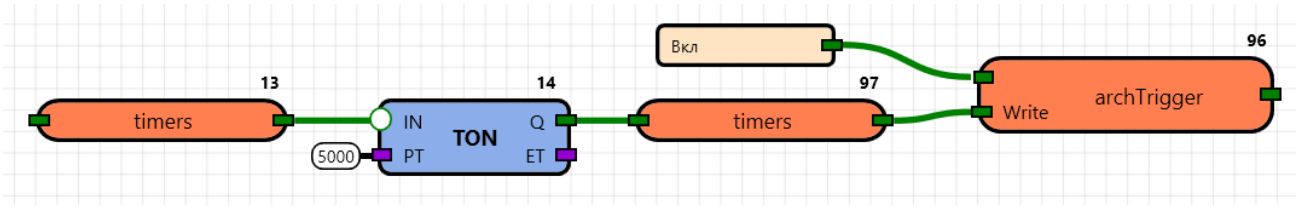


Рисунок 42. Сохранение данных в архив

Программа архива постоянно проверяет флаги записи в архив и переносит на uSD карту данные в случае "1" в флаге управления. На **Рисунке 42** показан пример программы для сохранения данные в архив каждые 5 секунд.

После того, как данные сохранены флаг записи сбрасывается в "0".

Для работы с контроллером uSD карта должна иметь один раздел данных, отформатированный в формате EXT4.

Перенос архивной информации с карты на ПК осуществляется через YartStudio.

В разделе "Локальный архив" необходимо выбрать интервал времени архива и параметры считывания.

Архивные данные считываются через интерфейс Ethernet.

При необходимости архивные данные доступны непосредственно на ПЛК по адресу:

PLC_IP_ADDR/archive.html

Где ***PLC_IP_ADDR*** текущий IP адрес контроллера в локальной сети.

Архив представляет собой список файлов, по одному файлу на каждый день в году. Скачанные с контроллера файлы могут быть расшифрованы при помощи YartStudio.

15. ИнСАТ MasterSCADA

Программируемый логический контроллер IRIS может поставляться в комплекте с системой MasterSCADA 4D, разработки компании Инсат - https://insat.ru

MasterSCADA 4D – это продукт нового поколения SCADA-систем. В нем реализованы инструменты по созданию крупных распределенных систем с возможностью использования технологий Интернета контроллеров СЕРЕБРУМ, повышено удобство и гибкость, увеличено число поддерживаемых уровней систем управления и реализована миграция функционала между уровнями.

В MasterSCADA 4D легко разрабатывать проекты любого масштаба и сложности. Для этого предлагаются различные подходы, обеспечивающие наиболее комфортные условия разработки под каждый тип проекта.

Техническую поддержку MasterSCADA 4D выполняет компания Инсат - support.ms4d@insat.ru . Техническую поддержку контроллера IRIS выполняет компания СЕРЕБРУМ support@serebrum.ru .

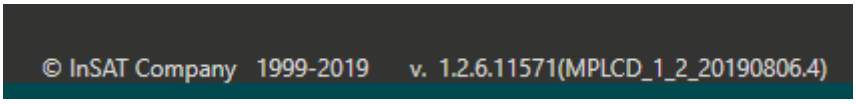
Базовая версия MasterSCADA 4D, интегрированная в IRIS, поддерживает работу с 2500 тегами и 5 пользователей.

Загрузить программное обеспечение MasterSCADA 4D можно с сайта

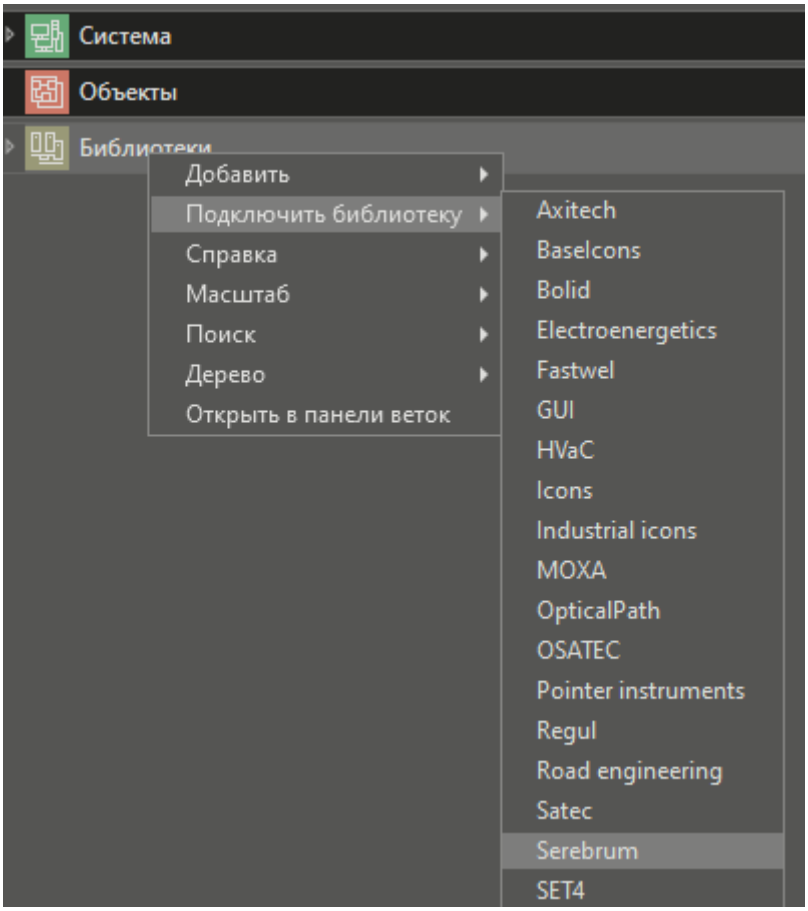
https://insat.ru/services/support/demos/

15.1 Инструкция по использованию контроллера IRIS в MasterSCADA 4D

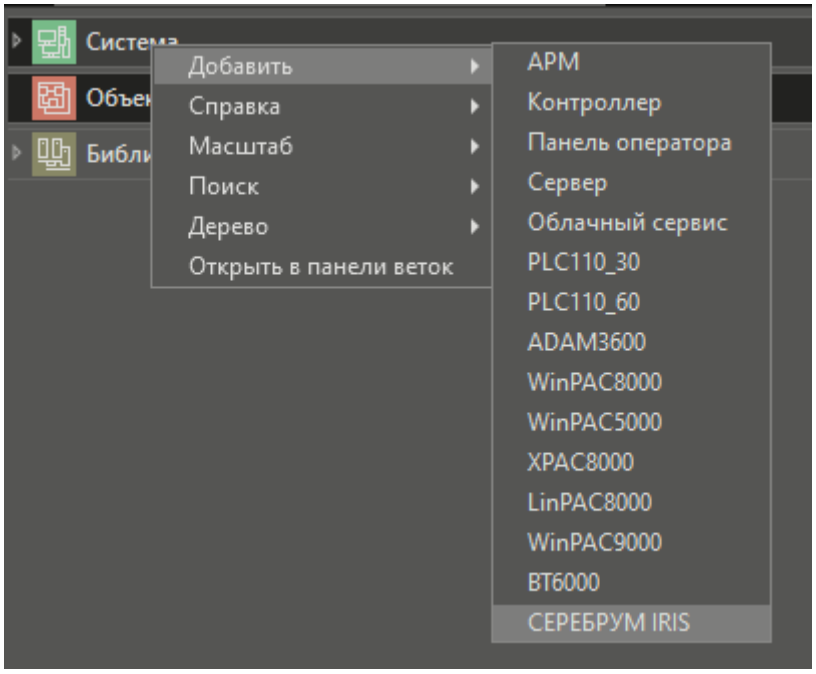
1. Запустите MasterSCADA 4D;
2. Убедитесь, что версия MasterSCADA 4D не ниже, чем 1.2.6.11571 (указана в нижней части экрана);



3. Создайте новый проект;
4. ПКМ (правой кнопкой мыши) по Библиотеки – Подключить библиотеку – Serebrum;



5. ПКМ по Система – Добавить – СЕРЕБРУМ IRIS;



6. В свойствах добавленного СЕРЕБРУМ IRIS необходимо изменить следующие параметры:
   - IP адрес – по умолчанию у IRIS адрес: 192.168.1.83
   - Максимальный размер UDP пакета: 1400
7. При необходимости добавьте модули расширения (Система – СЕРЕБРУМ IRIS – Встроенные модули) и протокол подключения к встроенному в контроллер OPC-серверу (Система – СЕРЕБРУМ IRIS – Протоколы – Добавить – OPC UA)
   - Имя пользователя для OPC UA – по умолчанию: ua_client
   - Пароль для OPC UA – по умолчанию: ua_password
   - URI: opc.tcp://192.168.1.83:16664
8. Далее работа ведется согласно инструкциям от MasterSCADA 4D.

16.Транспортирование и хранение

Контроллеры транспортируются в заводской упаковке в транспортной таре любым видом транспорта с защитой от дождя и снега. Крепление тары в транспортных средствах должно производиться согласно правилам, действующим на соответствующих видах транспорта.

Условия транспортирования должны соответствовать условиям 5 по ГОСТ 15150-69 при температуре окружающего воздуха от минус 40 до 50 °С с соблюдением мер защиты от ударов и вибраций.

Пребывание в условиях транспортирования – не более 3 месяцев.

Условия хранения в заводской упаковке на складе изготовителя и потребителя должны соответствовать условиям 1 по ГОСТ 15150-69. Наличие в воздухе агрессивных примесей не допускается.

После транспортирования при отрицательных температурах контроллеры перед включением необходимо выдержать в нормальных условиях не менее 24 ч

17. Гарантийные обязательства

Изготовитель гарантирует соответствие контроллера требованиям ТУ при соблюдении условий эксплуатации, транспортирования, хранения и монтажа.

Гарантийный срок– 36 месяцев со дня продажи.

В случае выхода контроллера из строя в течение гарантийного срока при соблюдении пользователем условий эксплуатации, транспортирования, хранения и монтажа предприятие-изготовитель обязуется осуществить его бесплатный ремонт или замену.