ООО «ЭМТИАЙ»
ИНН: 7733418955; ОГРН: 1237700542419; КПП: 773301001

Программный продукт
«Интегрированная среда разработки UpakNet»

Документация, содержащая информацию, необходимую для эксплуатации экземпляра программного обеспечения, предоставленного для проведения экспертной оценки в Экспертном совете при Минцифры России

Москва
2025 г.

СРЕДА РАЗРАБОТКИ
UpakNet CSyntax_V2.1

Руководство пользователя

2025

Добро пожаловать в документацию по Интегрированной среде разработки UpakNet (IDE) для программируемого микроконтроллера ET-XX, разработанного компанией ЭМТИАЙ. Данная среда предоставляет полный набор инструментов для эффективной разработки, отладки и прошивки встроенного программного обеспечения, ориентированного на архитектуру ET-XX.

IDE от ЭМТИАЙ разработана с учетом потребностей как начинающих, так и опытных разработчиков встроенных систем. Она обеспечивает интуитивно понятный интерфейс, мощные средства анализа и отладки, а также гибкие механизмы взаимодействия с аппаратной частью.

Микроконтроллер ET-XX представляет собой высокопроизводительное решение для встраиваемых приложений с поддержкой периферийных интерфейсов, энергоэффективных режимов работы и масштабируемой архитектуры. В сочетании с данной IDE он открывает широкие возможности для создания надежных и функциональных систем — от простых сенсорных узлов до сложных устройств управления.

В этой документации вы найдете все необходимое для начала работы с IDE: инструкции по установке, описание пользовательского интерфейса, примеры кода, а также руководство по отладке и прошивке устройств на базе ET-XX.

Готовы приступить? Начнем с установки среды разработки.

Обзор возможностей программы

Интегрированная среда разработки UpakNet (далее также UpakNet) - это среда разработки пользовательской управляющей программы для ПЛК (программируемого логического контроллера) серии ЕТ-ХХ.

UpakNet предназначен для:

- Взаимодействие с ПЛК ЕТ-ХХ (загрузка/выгрузка программы, диагностика состояния)
- Создание и редактирование программы для ПЛК ЕТ-ХХ в текстовом виде.
- Компиляции написанной программы

Типовые задачи, решаемые с помощью UpakNet:

- Конфигурация контроллера ЕТ-ХХ в рамках текущего проекта
- Настройка соединений, связей и параметров ПЛК
- Создание текста пользовательской программы на одном из встроенных языков программирования (булевые операции, язык Си).
- Отладка написанного кода, поиск ошибок, в том числе в режиме реального времени.
- Загрузка пользовательской программы в ПЛК.
- Диагностика состояния ПЛК.
- Сохранение/загрузка пользовательских программ

Системные требования

Для стабильной и эффективной работы UpakNet рекомендуется использовать следующую конфигурацию:

Частота процессора (CPU): 1 GHz

Количество ядер процессора (CPU): 1

Объем оперативной памяти (RAM): 2 Gb

Объем свободного места на диске (HDD): 20 Mb

Операционная система (OS): Windows XP и старше

Начало работы

Данный раздел поможет вам быстро установить, настроить и начать работать с UpakNet.

Перед началом работы, пожалуйста, ознакомьтесь с системными требованиями и лицензионным соглашением.

Основные понятия и термины

Перед началом работы в UpakNet рекомендуем ознакомиться с основными понятиями и терминами:

Флаг

1 бит в битовой области, который управляется контроллером.

Регистр

1 бит в битовой области, который управляется внешним устройством.

Настройка

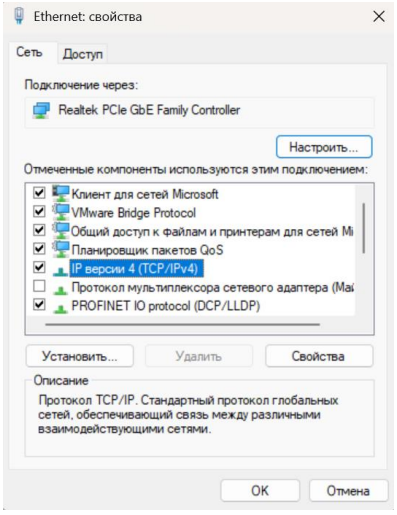
Для начала работы в UpakNet рекомендуем предварительно выполнить следующие настройки окружения:

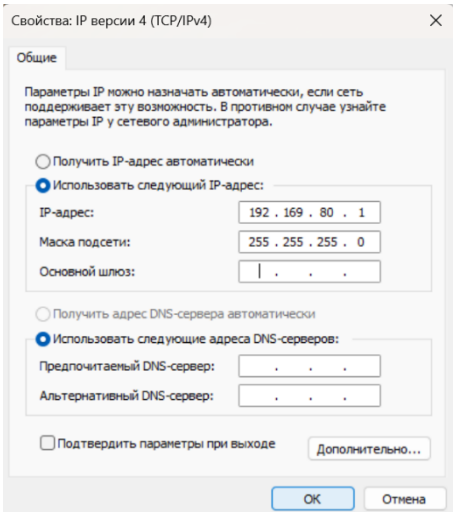
Настройка сетевого соединения с ПЛК

Перед началом работы с контроллером **ET-XX** необходимо правильно настроить сетевую карту компьютера. Это требуется для корректного взаимодействия с ПЛК и исключения конфликтов с другими устройствами в сети.

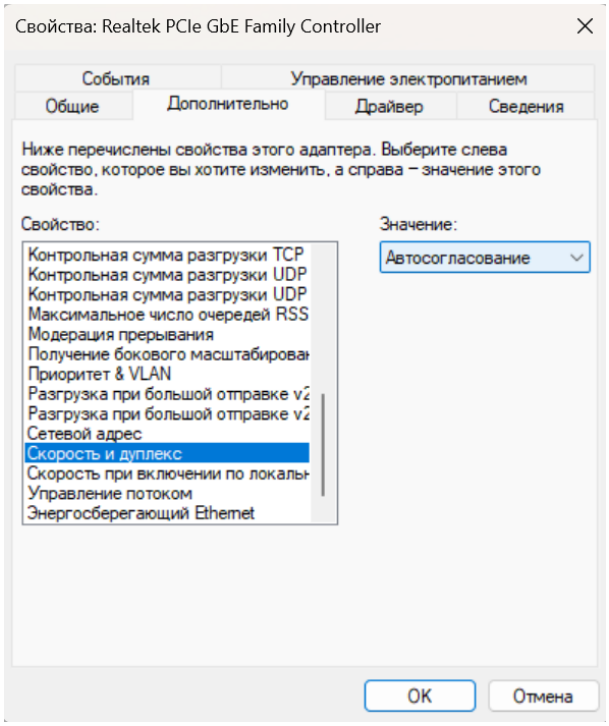
Для настройки сетевой карты выполните следующие действия:

1. Откройте меню управления сетевыми подключениями:
    - Нажмите **Win + R**, введите **ncpa.cpl** и нажмите Enter,
    - или через «Панель управления» → «Сеть и Интернет» → «Центр управления сетями и общим доступом» → «Изменение параметров адаптера».
2. Определите, к какой сетевой карте вашего компьютера подключён контроллер **ET-XX.**
3. Убедитесь, что эта сетевая карта не подключена к интернету и не использует DHCP.
4. Установите статический IP-адрес для сетевой карты:
    - IP-адрес сетевой карты компьютера: 192.168.81.x (например, 192.168.81.1)
    - Маска подсети: 255.255.255.0
    - Стандартный IP контроллера: 192.168.80.80





5. Для обеспечения стабильного соединения рекомендуется установить скорость соединения сетевой карты в режим автосогласования.



Эти действия гарантируют корректную связь между компьютером и контроллером и позволяют использовать все функции среды разработки UpakNet.

Проверка соединения с ПЛК

После настройки сетевой карты рекомендуется убедиться в корректности соединения с контроллером **ET-XX**. Это позволит избежать проблем при загрузке программы и диагностике.

1. Подключите контроллер к настроенной сетевой карте компьютера.
2. Откройте командную строку Windows (Win + R → введите cmd → Enter).
3. Выполните команду проверки связи:
4. ping 192.168.80.80 -t
5. Убедитесь, что получаете ответы от контроллера.

- Если ответы приходят — соединение установлено корректно.
- Если соединение не установлено — проверьте настройки IP сетевой карты и маску подсети.

После успешной проверки соединения можно переходить к запуску среды **UpakNet** и работе с ПЛК.

Подготовка антивирусного ПО

1. Добавьте папку с установленной программой **UpakNet** в исключения стандартного антивируса Windows Defender:

    - «Параметры» → «Обновление и безопасность» → «Безопасность Windows» → «Защита от вирусов и угроз» → «Параметры защиты от вирусов и других угроз» → «Исключения» → «Добавить или удалить исключения» → «Папка» → выберите каталог установки **UpakNet**.

2. Если на компьютере установлены сторонние антивирусы или средства защиты, временно **отключите их** на время работы с **UpakNet**, чтобы исключить блокировку соединения с ПЛК.

Эти действия гарантируют корректную работу среды разработки и стабильную связь с контроллером **ET-XX**.

Запуск

Запуск программы UpakNet

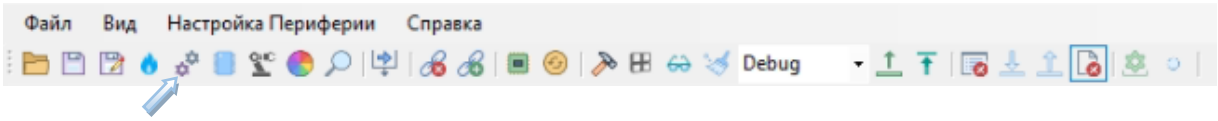
Для запуска среды разработки **UpakNet** выполните одно из следующих действий:

1. Через ярлык на рабочем столе – дважды щёлкните по ярлыку **UpakNet**.

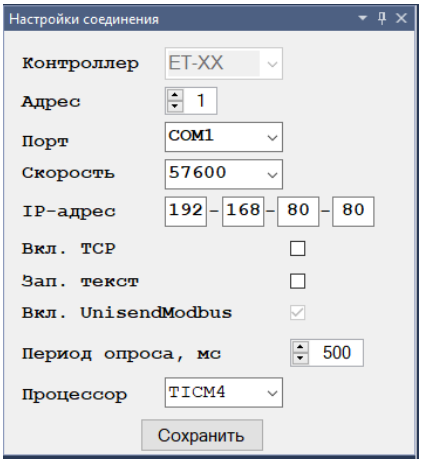
Первоначальная проверка связи с контроллером

При первом запуске рекомендуется выполнить проверку соединения с контроллером **ET-XX**:

1. Нажмите на кнопку **Настройки** в виде шестерёнки в верхней части окна программы.



2. В открывшемся окне проверьте следующие параметры:



- IP-адрес контроллера — должен соответствовать стандартному значению, указанному в документации на ПЛК.
- Тип процессора контроллера — сверьте с указанным в документации.

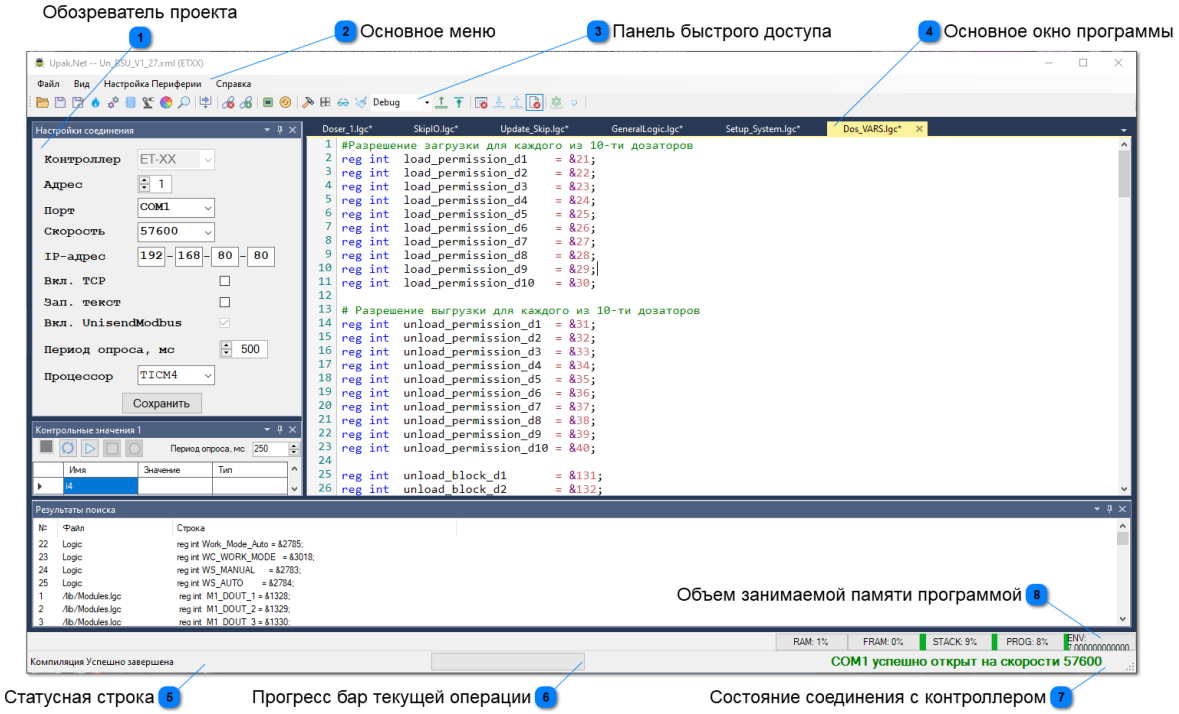
- Включите обмен по TCP, чтобы обеспечить корректную работу UpakNet**.**

3. Отдельно предусмотрена возможность работы по протоколу Modbus, для этого пункт «Вкл. TCP» должен быть отключен и должны быть настроены COM-порт и скорость (по умолчанию для связи используется скорость 57600 – для более подробной информации смотрите отдельные пункты руководства или отдельное руководство на ПЛК ET-XX).

Пользовательский интерфейс

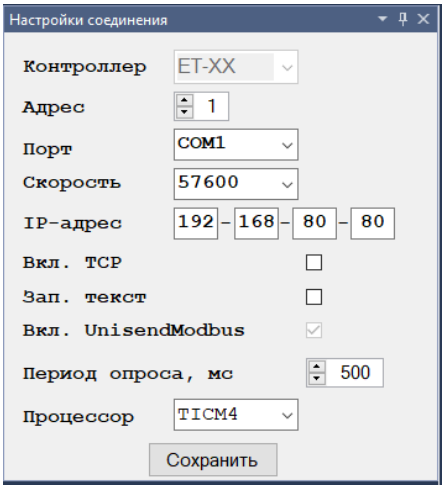
Этот раздел описывает основные элементы пользовательского интерфейса UpakNet CSyntax_V2.1: основных режимов работы, предназначение окон и экранов, доступные операции.

Главное окно программы

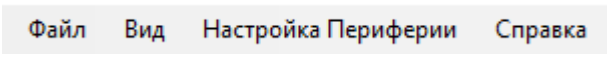


1 Обозреватель проекта

Предназначен для создания файлов проекта и настроек среды.



2 Основное меню



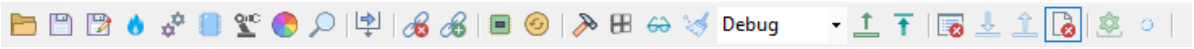
Файл — загрузка и сохранение программы.

Вид — настройка окон UpakNet.

Диагностика — диагностика устройств Modbus.

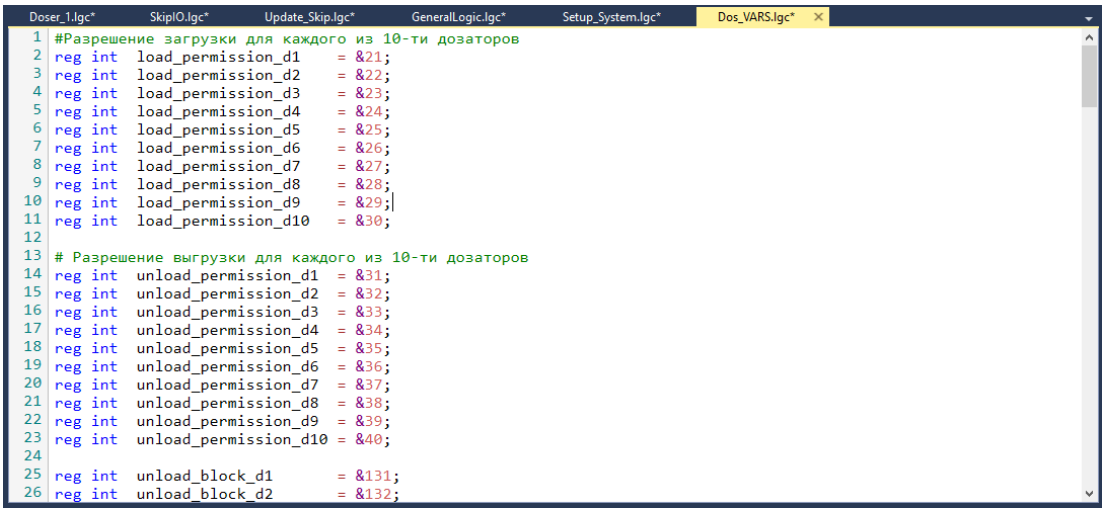
Справка — справка по работе с программой.

3 **Панель быстрого доступа**

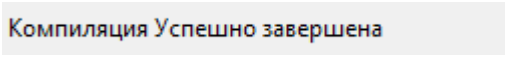


Обеспечивает быстрый доступ к основным функциям Upak_Net.

4 **Основное окно программы**



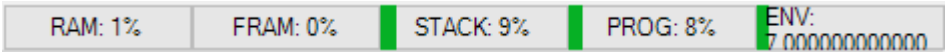
5 **Статусная строка**



6 **Прогресс бар текущей операции**

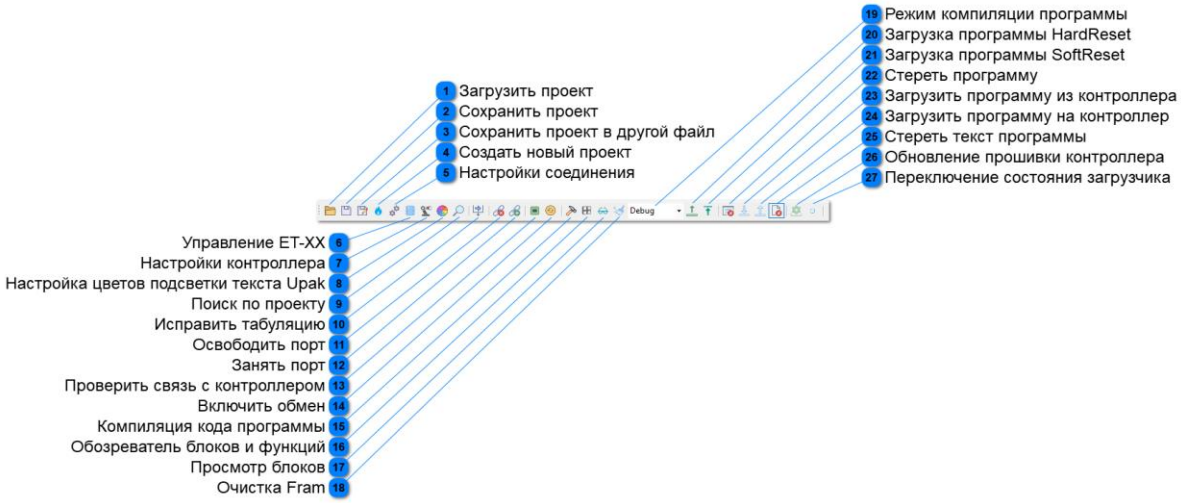
7 **Состояние соединения с контроллером**

8 **Объем занимаемой памяти программой**



Панель быстрого доступа

Обеспечивает быстрый доступ к основным функциям Upak_Net.



1 Загрузить проект

2 Сохранить проект

3 Сохранить проект в другой файл

4 Создать новый проект

5 Настройки соединения

Описание.

6 Управление ET-XX

Описание.

7 Настройки контроллера

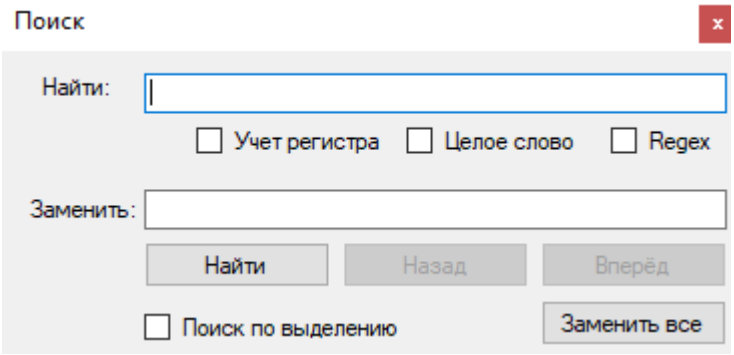
Описание.

8 Настройка цветов подсветки текста Upak

Описание.

9 Поиск по проекту

Поиск и замена по проекту.



10 **Исправить табуляцию**

Исправляет табуляцию в коде.

Пример

До исправления:

```
void Check_Power()
{
 #Проверка состояния питания
 if (PWR_SENS=0)
 {
  Power_Off_IN=1;
  if (Power_Off_OUT)
  {
      #Запись в Журнал, пропадание питания
      if (PWR_STATE=1)
      {
        Logic->Save_To_Journal(0,0,0.0);
      }
      PWR_STATE = 0;
      i0=0;
      #Переходим в Sleep режим
      Power_Off_IN=0;
      Power_Off_OUT=0;
      mstd->CPU__PutToSleep();
  }
 }
 else
 {
  if (PWR_STATE=0)
  {
    #Поднимаем флаг что питание востановленно
```

```
        PWR_STATE=1;
        Power_Off_IN=0;
        #Делаем запись в Журнал что питание востановленно
        Logic->Save_To_Journal(20,0,0.0);
      }
     }
    }
```

После исправления:

```
void Check_Power()
{
    #Проверка состояния питания
    if (PWR_SENS=0)
    {
        Power_Off_IN=1;
        if (Power_Off_OUT)
        {
            #Запись в Журнал, пропадание питания
            if (PWR_STATE=1)
            {
                Logic->Save_To_Journal(0,0,0.0);
            }
            PWR_STATE = 0;
            i0=0;
            #Переходим в Sleep режим
            Power_Off_IN=0;
            Power_Off_OUT=0;
            mstd->CPU__PutToSleep();
        }
    }
    else
    {
        if (PWR_STATE=0)
        {
            #Поднимаем флаг что питание востановленно
            PWR_STATE=1;
            Power_Off_IN=0;
            #Делаем запись в Журнал что питание востановленно
            Logic->Save_To_Journal(20,0,0.0);
        }
    }
}
```

11 **Освободить порт**

 Освобождает порт связи с контроллером для использования другими приложениями.

12 **Занять порт**

 Занимает порт связи с контроллером.

13 **Проверить связь с контроллером**

 Отправляет одиночное сообщение на контроллер, в статусной строке отображает результат:

"Успешно" - контроллер ответил.

"Ошибка проверки!" - после 10 попыток ответ от контроллера не был получен.

14 **Включить обмен**

 Включает периодический опрос битовой области для подсветки регистров во вкладке Upak.

15 **Компиляция кода программы**

 Компилирует код программы для последующей загрузки в контроллер. В случае ошибок компиляции автоматически откроется окно "Список ошибок".

16 **Обозреватель блоков и функций**

 Открывает окно для просмотра блоков и функций. Описание.

17 **Просмотр блоков**

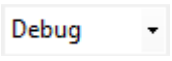
 Открывает окно просмотр блоков. Описание.

ВНИМАНИЕ!!! Режим просмотра блоков доступен в "Debug" режиме компиляции. Для "Release" режима необходимо включить отладку на Панели быстрого доступа  .

18 **Очистка Fram**

 Позволяет очистить внутреннюю энергонезависимую память контроллера.

19 **Режим компиляции программы**



Выбор режима компиляции программы Debug / Release. В Debug режиме контроллер позволяет просматривать память блоков и всю оперативную память контроллера, но при этом снижается производительность контроллера на 20-30%.

20 **Загрузка программы HardReset**

 Загрузка программы с полным перезапуском контроллера.

21 **Загрузка программы SoftReset**

 Загрузка программы без остановки текущей выполняемой программы, с последующим "горячим" переключением на новую программу.

ВНИМАНИЕ!!! Режим загрузки SoftReset не возможен если были внесены изменения в частях программы:

1. Параметры Таймеров, Счетчиков, Дозаторов.
2. Добавлены новые функции или блоки.
3. Объявлены новые переменные.

22 **Стереть программу**

 Стереть программу в контроллере.

23 **Загрузить программу из контроллера**

 Загрузить текст программы из контроллера.

24 **Загрузить программу на контроллер**

 Загрузить текст программы на контроллер.

25 **Стереть текст программы**

 Стереть сохраненный текст проекта в контроллере.

26 **Обновление прошивки контроллера**

 Обновление прошивки базовой прошивки контроллера.

ВНИМАНИЕ!!! Для контроллеров с процессором TICM4:

1. Обновление возможно только по TCP.
2. Необходимо установить IP компьютера 192.168.80.1.
3. Брандмауэр Windows должен быть отключен.

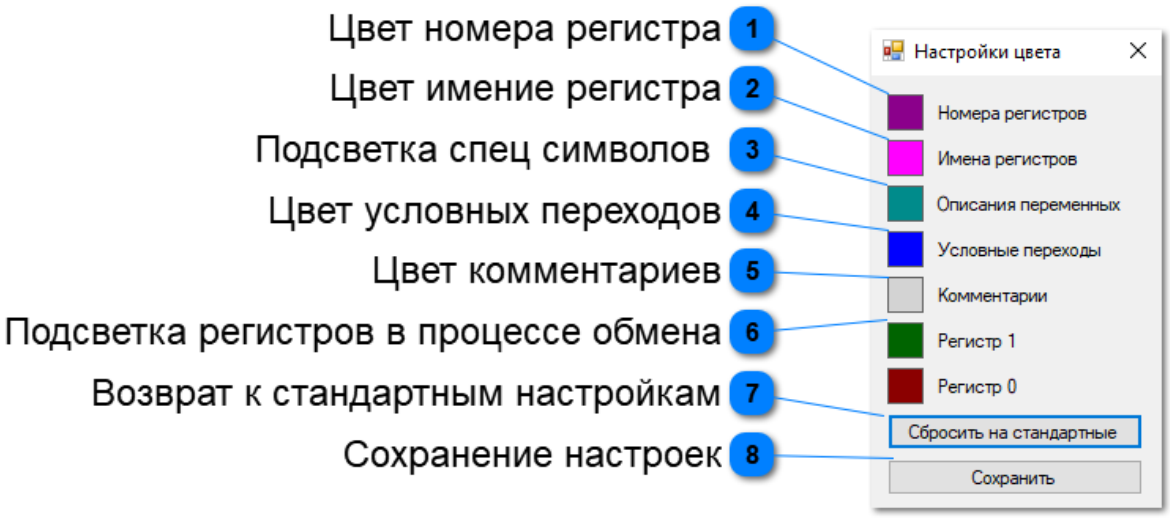
27 **Переключение состояния загрузчика**

Окна настройки программы

Окна **Настройки программы** позволяют редактировать параметры и глобальные настройки программы UpakNet.

Окно "Настройки цвета"

Позволяет настроить подсветку в программе UPAK.



1 **Цвет номера регистра**

Пример:

```
#?90=Doz9Allow
```

2 **Цвет имение регистра**

Пример:

```
#?90=Doz9Allow
```

3 **Подсветка спец символов**

Подсветка спец символов используемых для объявления переменных

4 **Цвет условных переходов**

Пример:

```
#* START 1050=1
        #* STOP
```

5 **Цвет комментариев**

6 **Подсветка регистров в процессе обмена**

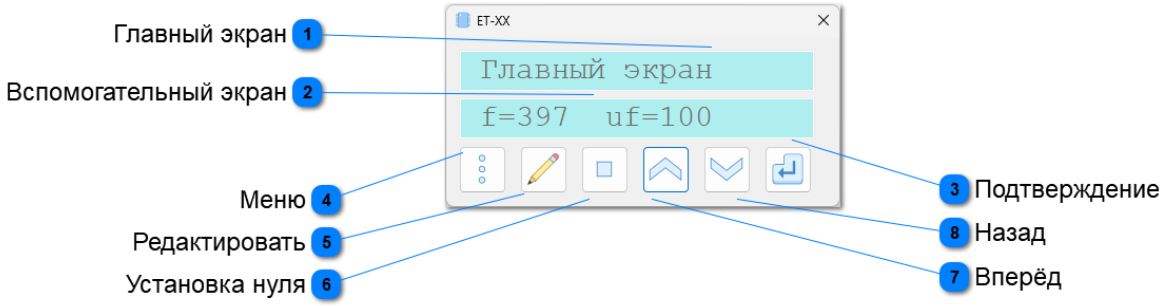
Цвет подсветки регистров при включенном обмене для состояний 0/1.

7 **Возврат к стандартным настройкам**

8 **Сохранение настроек**

ВНИМАНИЕ!!! Настройки будут применены только после перезапуска UpakNet.

Управление ET-XX



1 Главный экран

Главный экран

2 Вспомогательный экран

f=397 uf=100

3 Подтверждение

4 Меню

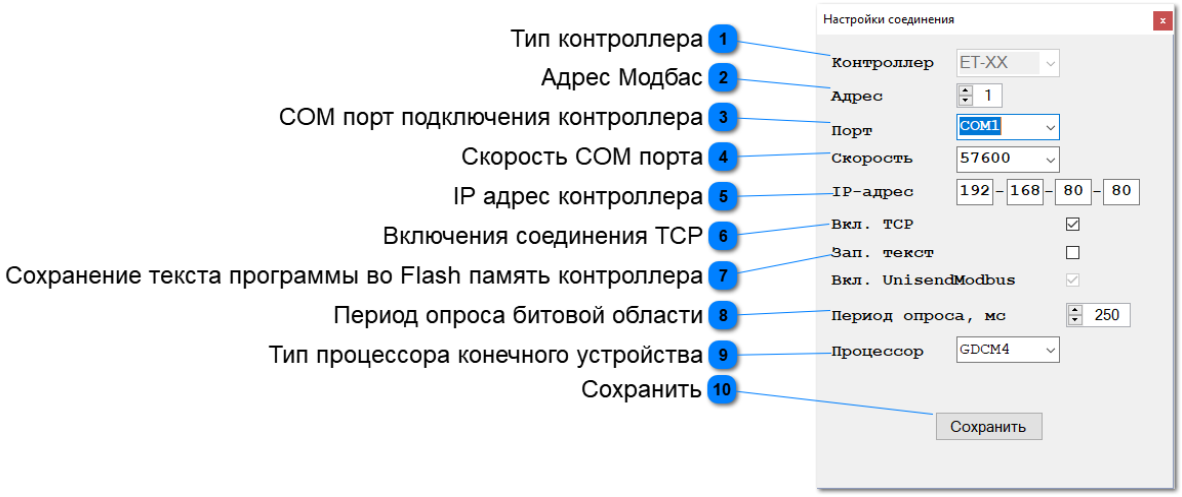
5 Редактировать

6 Установка нуля

7 Вперёд

8 **Назад**

Окно "Настройки соединения"



1 **Тип контроллера**

Текущая версия программы поддерживает только ET-XX.

2 **Адрес Модбас**

Модбас адрес подключенного контроллера.

3 **COM порт подключения контроллера**

COM порт компьютера к которому подключен контроллер.

4 **Скорость COM порта**

Скорость COM порта на которой идет обмен с контроллером. Заводская настройка 57 600.

5 **IP адрес контроллера**

Для корректной работы Ethernet адаптер компьютера должен быть настроен на туже подсеть, что и контроллер.

Для процессоров TICM4 при смене прошивки IP адрес компьютера должен быть 192.168.80.1.

6 **Включения соединения TCP**

При выборе соединения TCP, COM порт компьютера будет освобожден и будет установлено соединение TCP с контроллером.

7 **Сохранение текста программы во Flash память контроллера**

При включении, после записи программы в контроллер, также происходит запись файла программы на внешнюю Flash память контроллера.

В последствии программа может быть загружена с контроллера на ПК.

8 **Период опроса битовой области**

Период опроса битовой области, для отображения состояние регистров во вкладке Upak, в режиме обмена.

9 **Тип процессора конечного устройства**

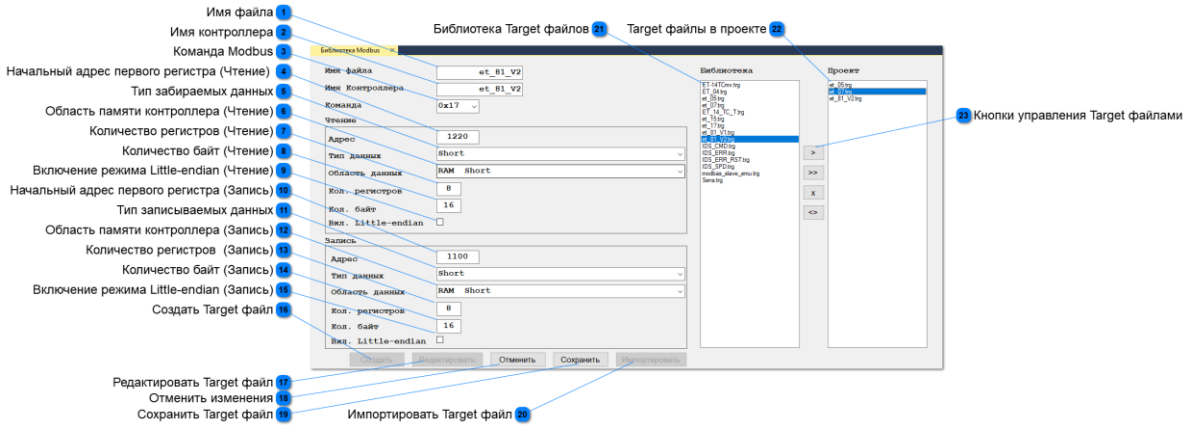
Возможные значения TICM4/GDCM4. Для определения типа процессора используйте окно "Настройки контроллера".

10 **Сохранить**

Применение настроек происходит сразу после нажатия кнопки сохранить. При изменении IP адреса автоматически разрывается соединение с контроллером, потребуется новое подключение.

Окно "Библиотека Modbus"

"Библиотека Modbus" позволяет создавать Target файлы для связи со Slave устройствами по протоколу Modbus. Target файл - файл, который описывает порядок взаимодействия контроллера со Slave устройствами. После добавления Target файла в проект, он может быть использован в проекте. см. Окно "Модули UniSend".



1 Имя файла

2 Имя контроллера

Имя контроллера для отображения в таблице опроса.

3 Команда Modbus

Поддерживаемые команды:

```
0x01 - Чтение DO
0x02 - Чтение DI
0x03 - Чтение AO
0x04 - Чтение AI
0x05 - Запись одного DO
0x06 - Запись одного AO
0x0F - Запись нескольких DO
0x10 - Запись нескольких AO
0x17 - Чтение / Запись нескольких регистров
```

4 Начальный адрес первого регистра (Чтение)

Значение адреса будет использовано в сообщении без преобразований.

5 Тип забираемых данных

Формат данных забираемых с устройства.

Описание типов данных.

6 Область памяти контроллера (Чтение)

Область памяти контроллера куда надо положить полученные данные с устройства.

ВНИМАНИЕ!!! Размер забираемых данных должен совпадать с размером в области памяти.

При не совпадении размеров могут быть повреждены данные.

7 **Количество регистров (Чтение)**

Количество регистров забираемых с устройства.

8 **Количество байт (Чтение)**

Количество значимых байт в ответе устройства.

9 **Включение режима Little-endian (Чтение)**

Переключение режима распаковки данных ответа устройства (Big endian / Little endian). При включенном режиме порядок байт - LSB.

10 **Начальный адрес первого регистра (Запись)**

Значение адреса будет использовано в сообщении без преобразований.

11 **Тип записываемых данных**

Формат данных записываемых в устройство.

Описание типов данных.

12 **Область памяти контроллера (Запись)**

Область памяти контроллера откуда забирать данные для записи в устройство.

ВНИМАНИЕ!!! Размер записываемых данных должен совпадать с размером в области памяти.

При не совпадении размеров могут быть повреждены данные.

13 **Количество регистров (Запись)**

Количество записываемых регистров.

14 **Количество байт (Запись)**

Количество значимых байт в посылке.

15 **Включение режима Little-endian (Запись)**

Переключение режима упаковки данных для отправки (Big endian / Little endian). При включенном режиме порядок байт - LSB.

16 **Создать Target файл**

Файл создается в директории "\TargetFiles" в папке компилятора. Директория создается автоматически.

17 **Редактировать Target файл**

Изменение выбранного Target файла.

18 **Отменить изменения**

19

Сохранить 20 **Target файл**

Импортировать Target файл

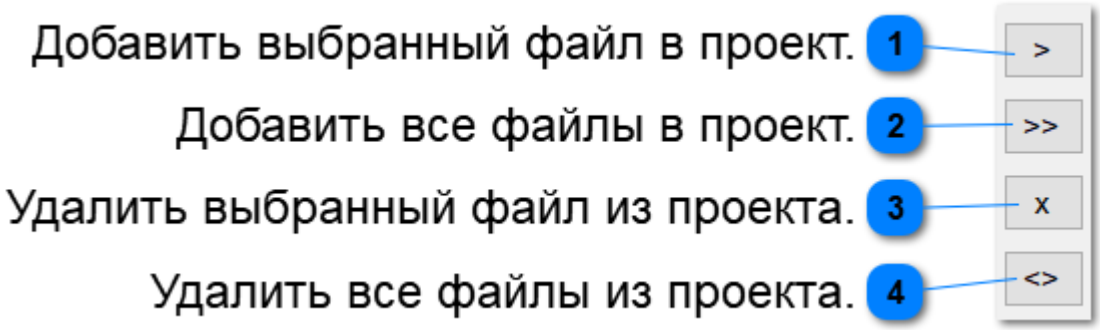
21 **Библиотека Target файлов**

Список всех доступных Target файлов в библиотеке.

22 **Target файлы в проекте**

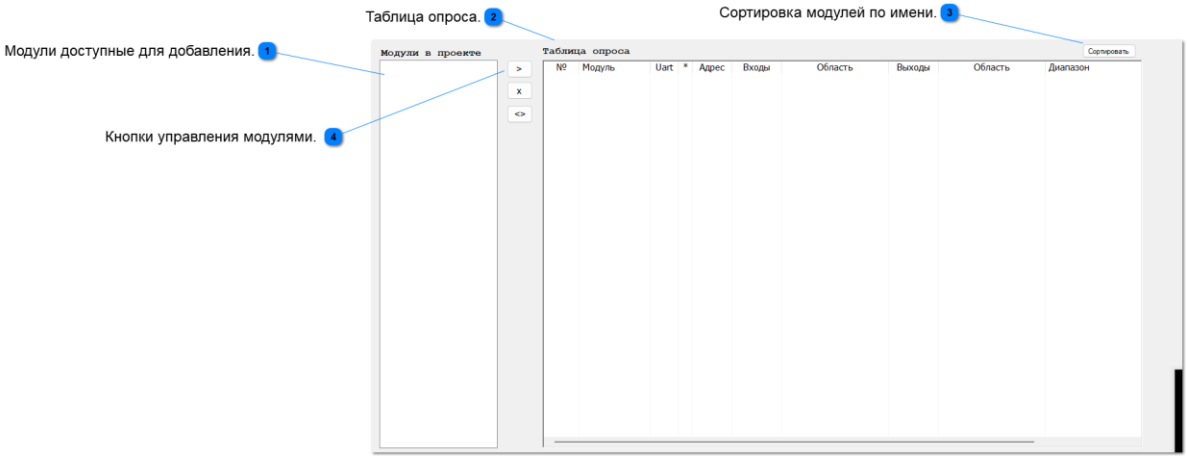
Target файлы доступные для использования в проекте.

23 **Кнопки управления Target файлами**



Окно "Модули UniSend"

Окно "Модули UniSend" используется для настройки периферийных устройств подключенных к контроллеру по RS-485. Настройка параметров обмена происходит в окне "Настройки контроллера".



1 **Модули доступные для добавления.**

Список модулей доступных в проекте. см. "Библиотека Modbus".

2 **Таблица опроса.**

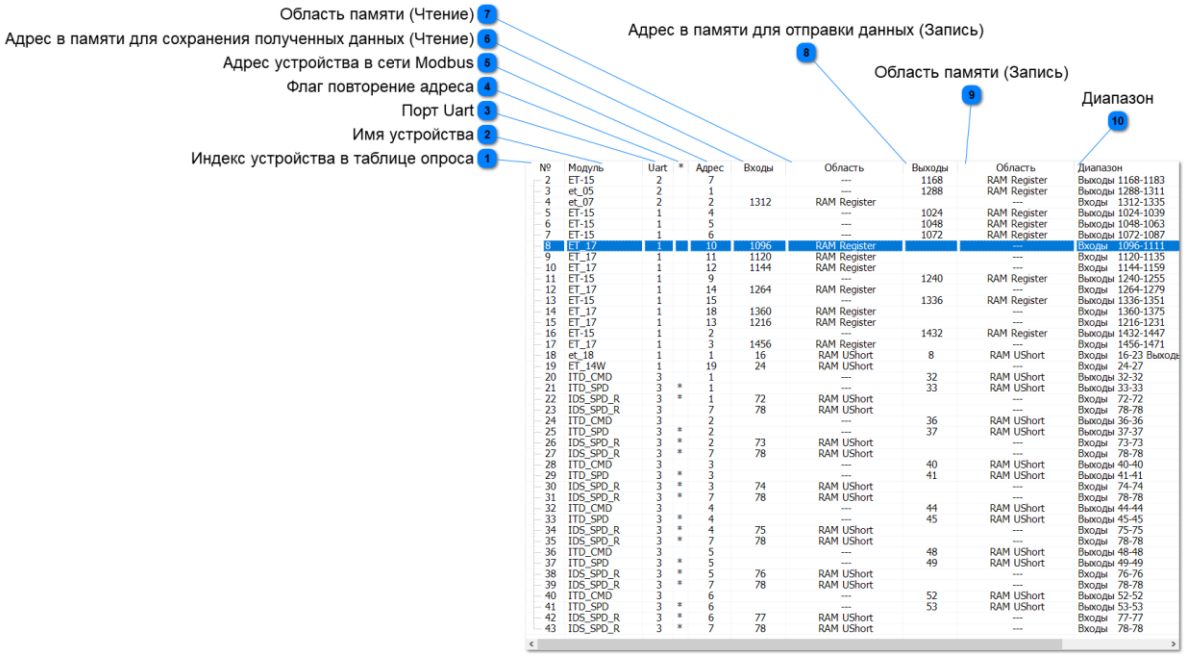
Список Slave устройств включенных в проект. см. "Таблица опроса".

3 **Сортировка модулей по имени.**

4 **Кнопки управления модулями.**

Таблица опроса

Опрашиваемые устройства по интерфейсам Modbus RTU на шине RS-485 контроллера ET-XX настраиваются с помощью таблицы опроса на основе target-файлов с указанием Имени опрашиваемого модуля, используемого порта Uart, адреса в сети Modbus, адресов регистров чтения/записи и области памяти, в которой они находятся.



№	Модуль	Uart	*	Адрес	Входы	Область	Выходы	Область	Диапазон
2	ET-15	2		7		---	1168	RAM Register	Выходы 1168-1183
3	et_05	2		1		---	1288	RAM Register	Выходы 1288-1311
4	et_07	2		2	1312	RAM Register		---	Входы 1312-1335
5	ET-15	1		4		---	1024	RAM Register	Выходы 1024-1039
6	ET-15	1		5		---	1048	RAM Register	Выходы 1048-1063
7	ET-15	1		6		---	1072	RAM Register	Выходы 1072-1087
8	ET_17	1		10	1096	RAM Register		---	Входы 1096-1111
9	ET_17	1		11	1120	RAM Register		---	Входы 1120-1135
10	ET_17	1		12	1144	RAM Register		---	Входы 1144-1159
11	ET-15	1		9		---	1240	RAM Register	Выходы 1240-1255
12	ET_17	1		14	1264	RAM Register		---	Входы 1264-1279
13	ET-15	1		15		---	1336	RAM Register	Выходы 1336-1351
14	ET_17	1		18	1360	RAM Register		---	Входы 1360-1375
15	ET_17	1		13	1216	RAM Register		---	Входы 1216-1231
16	ET-15	1		2		---	1432	RAM Register	Выходы 1432-1447
17	ET_17	1		3	1456	RAM Register		---	Входы 1456-1471
18	et_18	1		1	16	RAM UShort	8	RAM UShort	Входы 16-23 Выходы
19	ET_14W	1		19	24	RAM UShort		---	Входы 24-27
20	ITD_CMD	3		1		---	32	RAM UShort	Выходы 32-32
21	ITD_SPD	3	*	1		---	33	RAM UShort	Выходы 33-33
22	IDS_SPD_R	3	*	1	72	RAM UShort		---	Входы 72-72
23	IDS_SPD_R	3		7	78	RAM UShort		---	Входы 78-78
24	ITD_CMD	3		2		---	36	RAM UShort	Выходы 36-36
25	ITD_SPD	3	*	2		---	37	RAM UShort	Выходы 37-37
26	IDS_SPD_R	3	*	2	73	RAM UShort		---	Входы 73-73
27	IDS_SPD_R	3	*	7	78	RAM UShort		---	Входы 78-78
28	ITD_CMD	3		3		---	40	RAM UShort	Выходы 40-40
29	ITD_SPD	3	*	3		---	41	RAM UShort	Выходы 41-41
30	IDS_SPD_R	3	*	3	74	RAM UShort		---	Входы 74-74
31	IDS_SPD_R	3	*	7	78	RAM UShort		---	Входы 78-78
32	ITD_CMD	3		4		---	44	RAM UShort	Выходы 44-44
33	ITD_SPD	3	*	4		---	45	RAM UShort	Выходы 45-45
34	IDS_SPD_R	3	*	4	75	RAM UShort		---	Входы 75-75
35	IDS_SPD_R	3	*	7	78	RAM UShort		---	Входы 78-78
36	ITD_CMD	3		5		---	48	RAM UShort	Выходы 48-48
37	ITD_SPD	3	*	5		---	49	RAM UShort	Выходы 49-49
38	IDS_SPD_R	3	*	5	76	RAM UShort		---	Входы 76-76
39	IDS_SPD_R	3	*	7	78	RAM UShort		---	Входы 78-78
40	ITD_CMD	3		6		---	52	RAM UShort	Выходы 52-52
41	ITD_SPD	3	*	6		---	53	RAM UShort	Выходы 53-53
42	IDS_SPD_R	3	*	6	77	RAM UShort		---	Входы 77-77
43	IDS_SPD_R	3	*	7	78	RAM UShort		---	Входы 78-78

1 Индекс устройства в таблице опроса

Индекс устройства в таблице опроса контроллера, начинается с нуля.

> **ВНИМАНИЕ!!!** Устройства определенные в окне дозатора индексируются первыми.

2 Имя устройства

3 Порт Uart

Номер порта RS-485 (Uart 0- 4) контроллера к которому подключено устройство.

ВНИМАНИЕ!!!! Не должен совпадать с портом PC см. "Настройки контроллера".

4 Флаг повторение адреса

Устройства с одинаковым адресом в таблице обмена помечаются "*". Необходимо в тех случаях когда с одного и того же устройства забираются разные данные.

5 **Адрес устройства в сети Modbus**

6 **Адрес в памяти для сохранения полученных данных (Чтение)**

Адрес в выбранной области памяти начиная с которого будут записывать данные полученные со Slave устройства.

7 **Область памяти (Чтение)**

Область памяти контроллера в которую будут записываться данные для полученные со Slave устройства.

8 **Адрес в памяти для отправки данных (Запись)**

Адрес в выбранной области памяти начиная с которого будут отправляться данные в Slave устройство.

9 **Область памяти (Запись)**

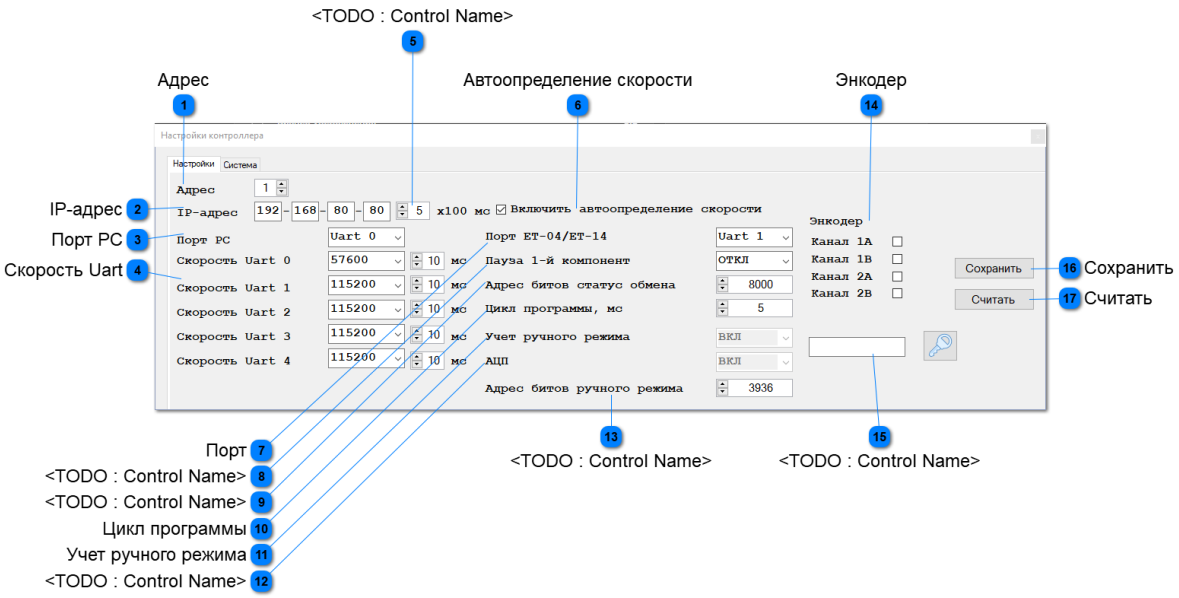
Область памяти контроллера из которой будут забираться данные для отправки в Slave устройство.

10 **Диапазон**

Диапазон памяти которую использует контроллер для Slave устройства.

Окна настройки контроллера

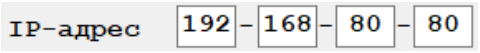
Окно "Настройки контроллера"



1 Адрес

Сетевой slave-адрес ПЛК ET-XX в сети Modbus-TCP.

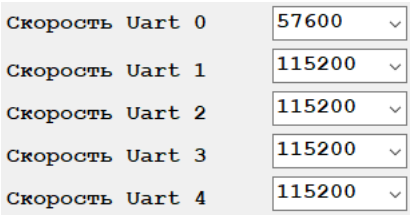
2 IP-адрес



3 Порт PC

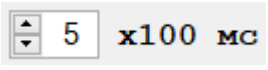
Выбранный порт контроллера будет использован как slave-устройство Modbus.

4 Скорость Uart

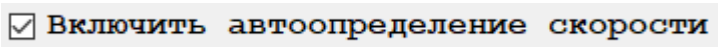


Каждому порту Uart может быть назначена своя скорость обмена по Modbus-RTU

5 Таймаут обмена по Modbus-TCP



6 Автоопределение скорости

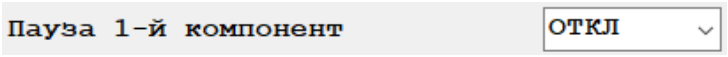


Автоопределение скорости соединения по Ethernet

7 **Порт**

Номер интерфейса встроенных весовых дозаторов.

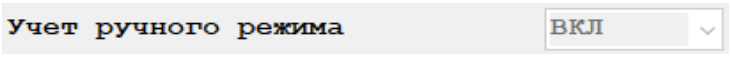
8 **Включение паузы перед загрузкой первого компонента в дозатор**



9 **Адреса регистров ошибок обмена в соответствии с таблице обмена**

10 **Цикл программы**

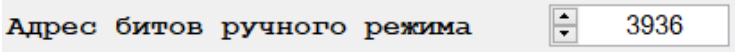
11 **Учет ручного режима**



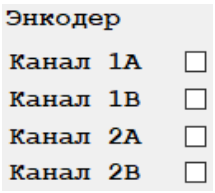
Включение встроенного алгоритма учета набранного веса в дозаторы в ручном режиме работы

12 **Включение/отключение встроенного АЦП в ПЛК ET-XX**

13 **Адреса регистров учета ручного режима (рекомендуется оставить это значение)**



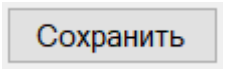
14 **Энкодер**



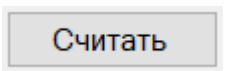
Включение встроенного в ПЛК энкодера

15 **Пароль для входа в режим суперпользователя**

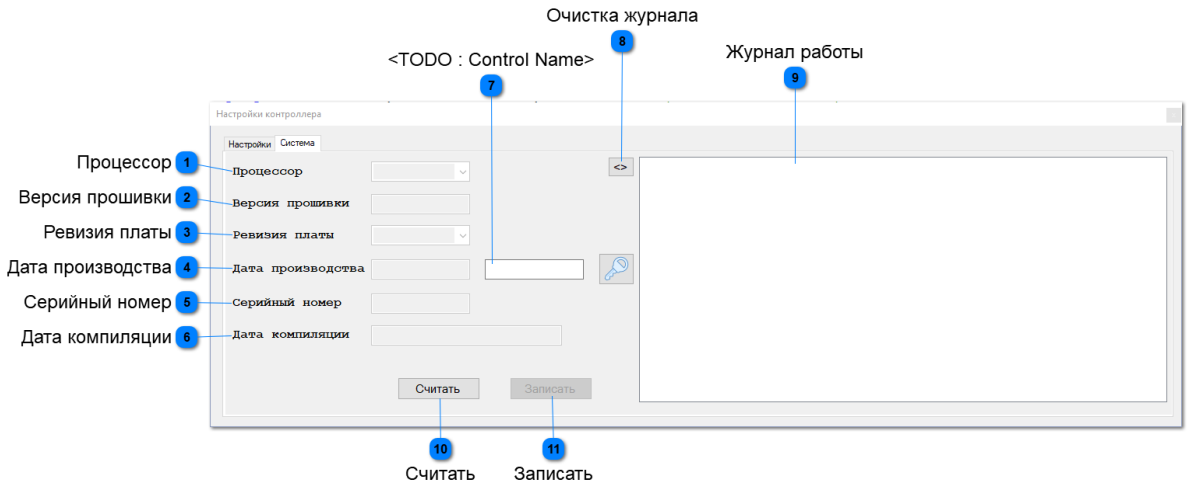
16 **Сохранить**



17 **Считать**



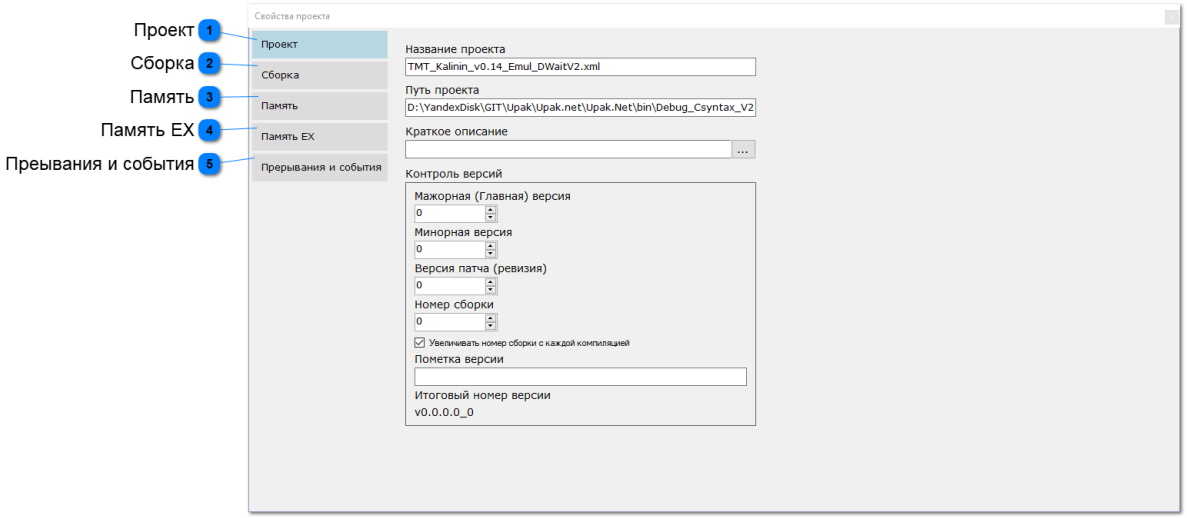
Вкладка "Система"



Эта вкладка позволяет считать сервисные параметры и версию встроенного ПО в ПЛК

Окно "Свойства проекта"

Окно свойство проекта позволяет настроить распределение памяти контроллера, память обмена Modbus, прерывания и события.



1 Проект

Описание проекта и системы контроля версий.

2 Сборка

Настройка параметров сборки проектов.

3 Память

Настройка распределения памяти RAM и FRAM.

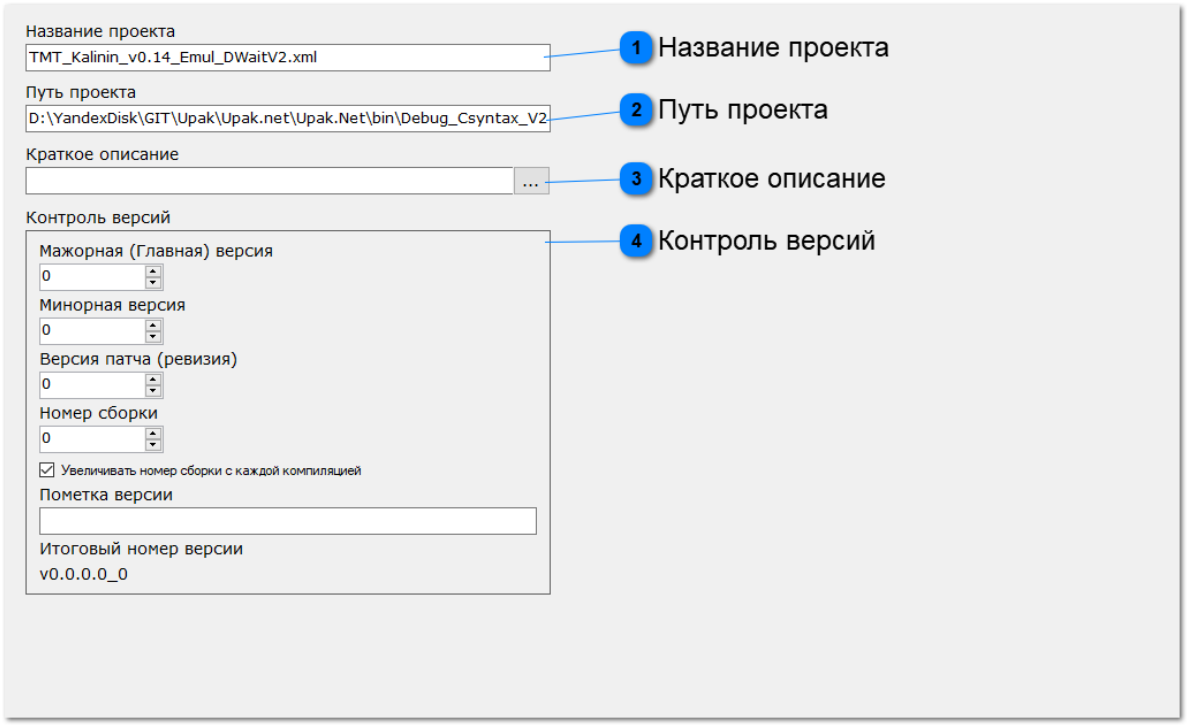
4 Память EX

Настройка распределения памяти обмена.

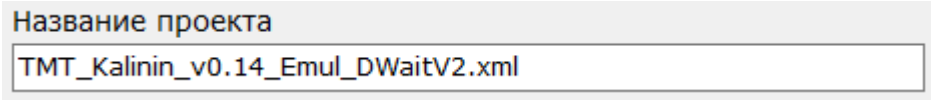
5 Преывания и события

Настройка прерываний.

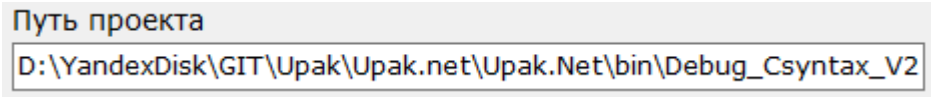
Вкладка "Проект"



1 Название проекта

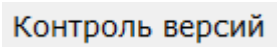


2 Путь проекта

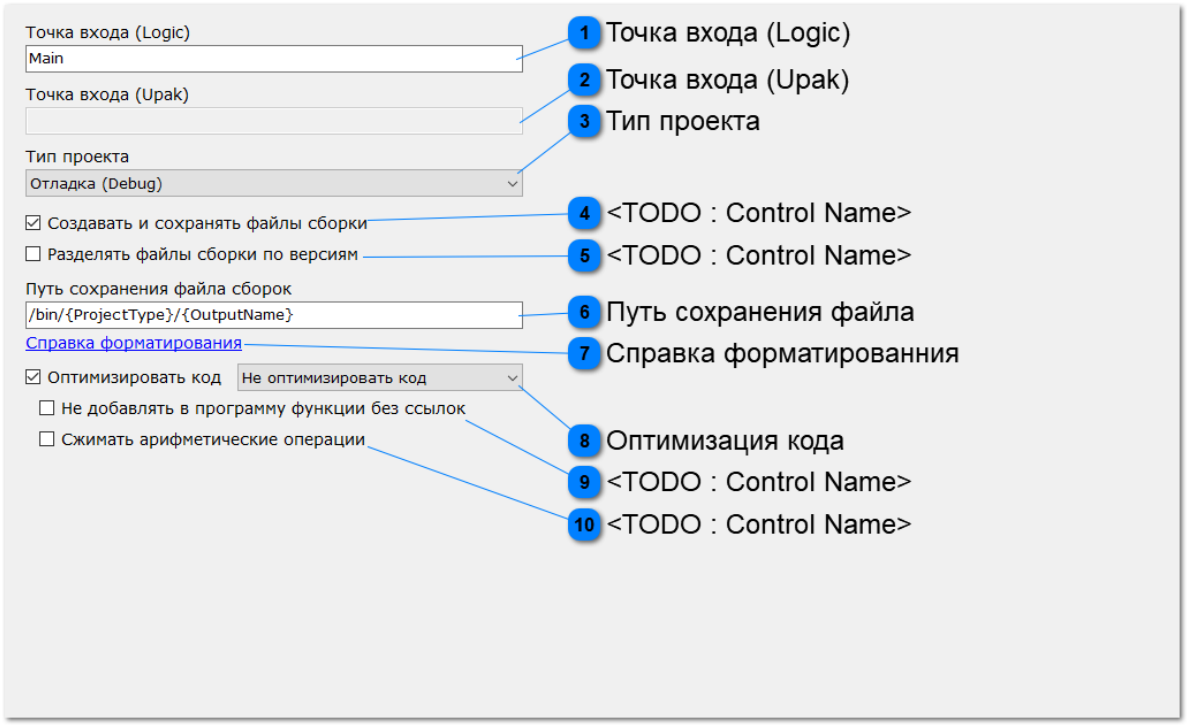


3 Краткое описание

4 Контроль версий



Вкладка "Сборка"



1 Точка входа (Logic)

Точка входа (Logic)

Main

2 Точка входа (Upak)

Точка входа (Upak)

3 Тип проекта

Тип проекта

Отладка (Debug)

4 <TODO : Control Name>

☑ Создавать и сохранять файлы сборки

5 <TODO : Control Name>

☐ Разделять файлы сборки по версиям

6 Путь сохранения файла

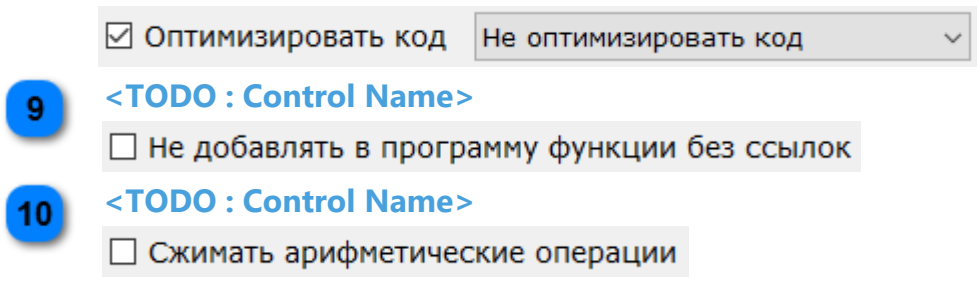
Путь сохранения файла сборок

/bin/{ProjectType}/{OutputName}

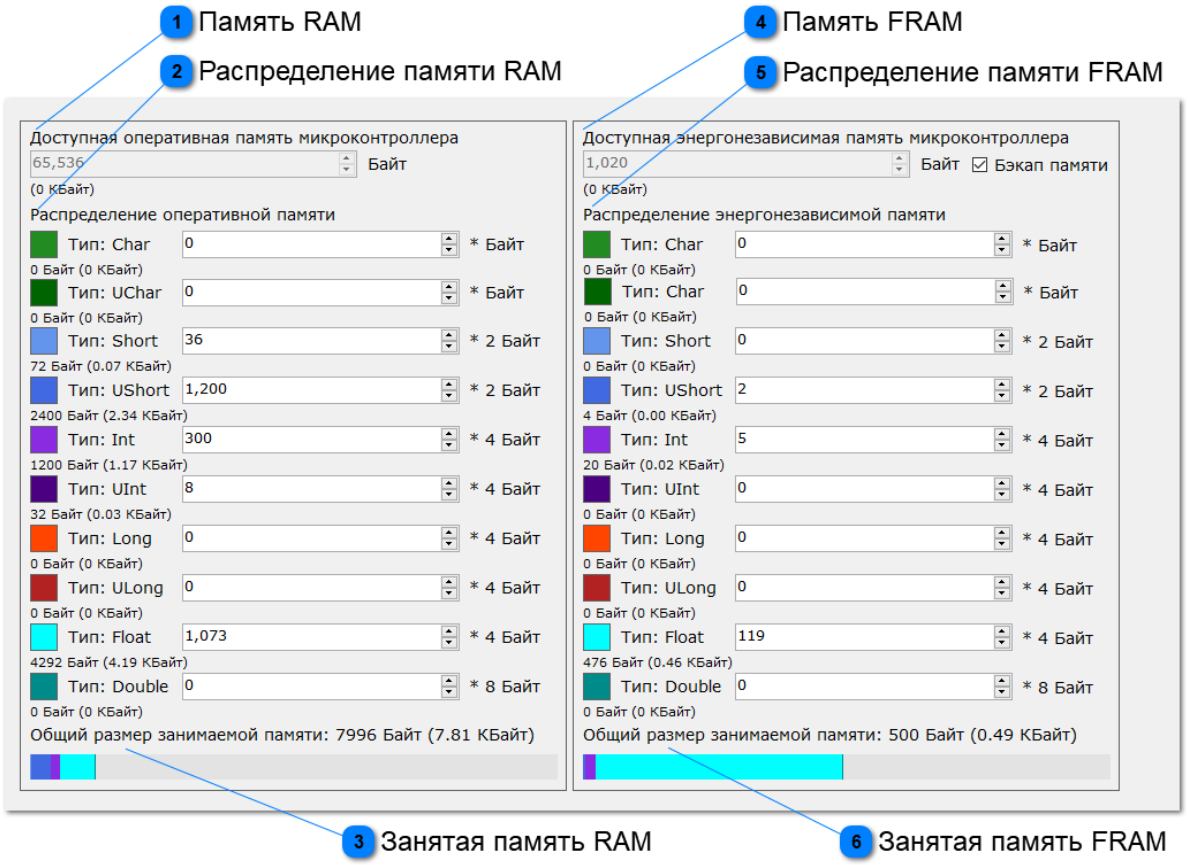
7 Справка форматированния

Справка форматирования

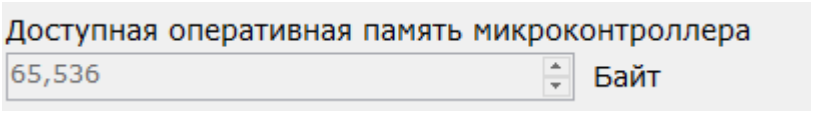
8 Оптимизация кода


Оптимизировать код
Не оптимизировать код
9
<TODO : Control Name>
Не добавлять в программу функции без ссылок
10
<TODO : Control Name>
Сжимать арифметические операции

Вкладка "Память"



1 Память RAM

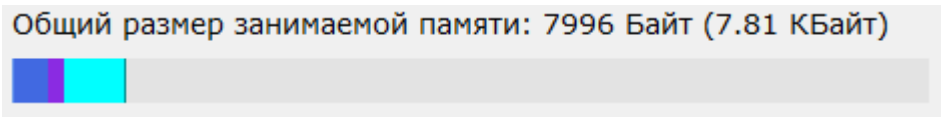


Объем доступной оперативной памяти (RAM) контоллера 65 536 байт.

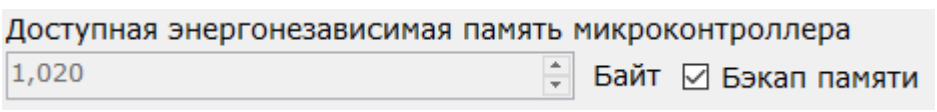
2 Распределение памяти RAM

Количество переменных и их тип, которые использованны в проекте.

3 Занятая память RAM



4 Память FRAM



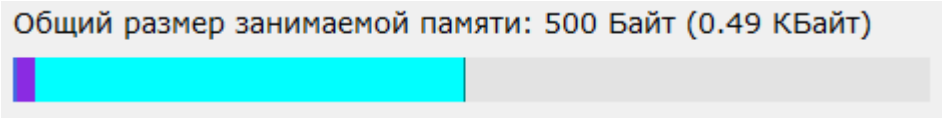
Общий размер энергонезависимой памяти (FRAM) 2048 байт, из них 8 байт заняты под служебную информацию. Сохранения данных происходит каждые 1000 мс.

В режиме "Бэкап" память разбивается на две страницы по 1020 байт, сохранение данных в этом режиме происходит со сменой страницы после каждого цикла записи.

5 **Распределение памяти FRAM**

Количество переменных и их тип, которые использованны в проекте.

6 **Занятая память FRAM**



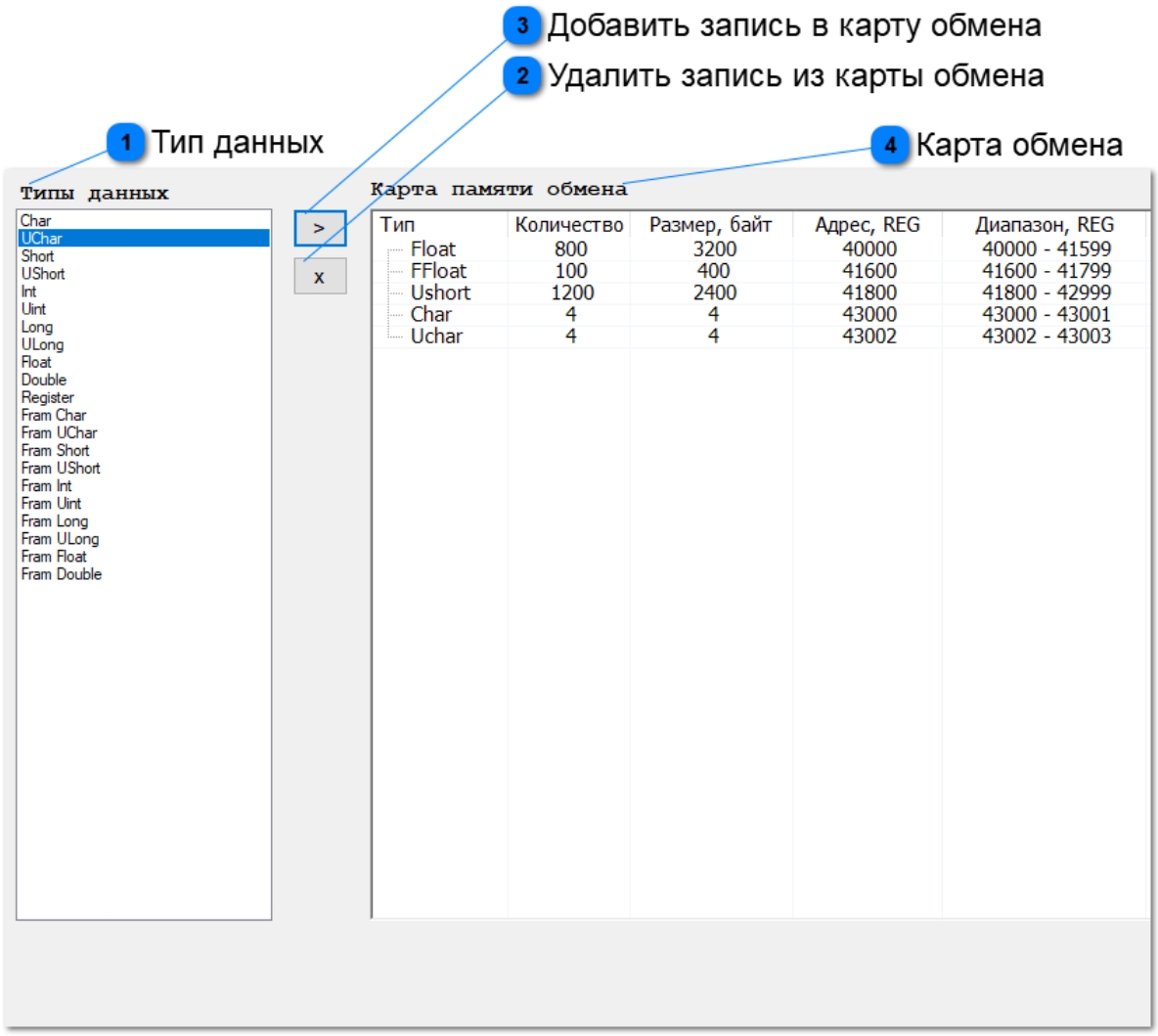
Вкладка "Память EX"

Вкладка "Память EX" позволяет настроить карту обмена контроллера в режиме работы "Release".

Доступ к переменным из карты обмена начинаетс с адреса 40 000 (0x9C40). Размер области 15 120 регистров.

> Доступ к переменнымтипа `reg int` осуществляется по адресу 6 000 (0x1770). Размер области 512 регистров.

Пример заполнения карты обмена:



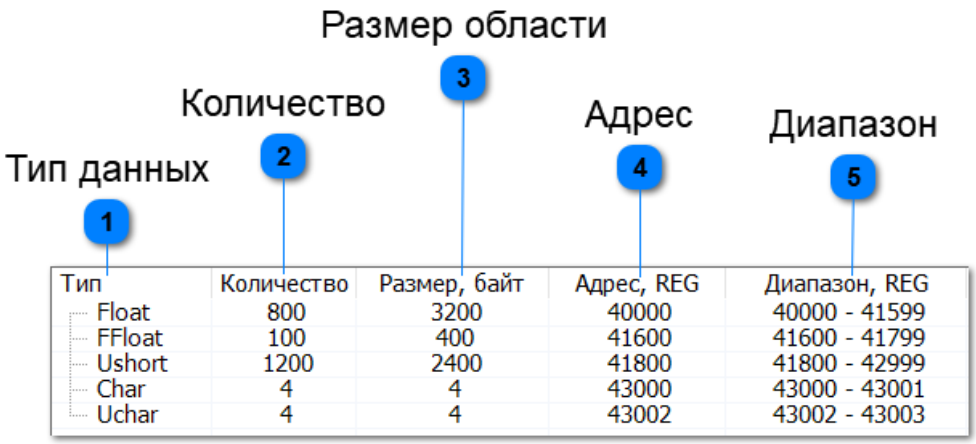
1 **Тип данных**

Тип данных для добавление в карту обмена.

2 **Удалить запись из карты обмена**

3 **Добавить запись в карту обмена**

4 Карта обмена



Тип	Количество	Размер, байт	Адрес, REG	Диапазон, REG
Float	800	3200	40000	40000 - 41599
FFloat	100	400	41600	41600 - 41799
Ushort	1200	2400	41800	41800 - 42999
Char	4	4	43000	43000 - 43001
Uchar	4	4	43002	43002 - 43003

1 Тип данных

2 Количество

Количество элементов для выбранного типа данных.

3 Размер области

Размер области данных в байтах.

4 Адрес

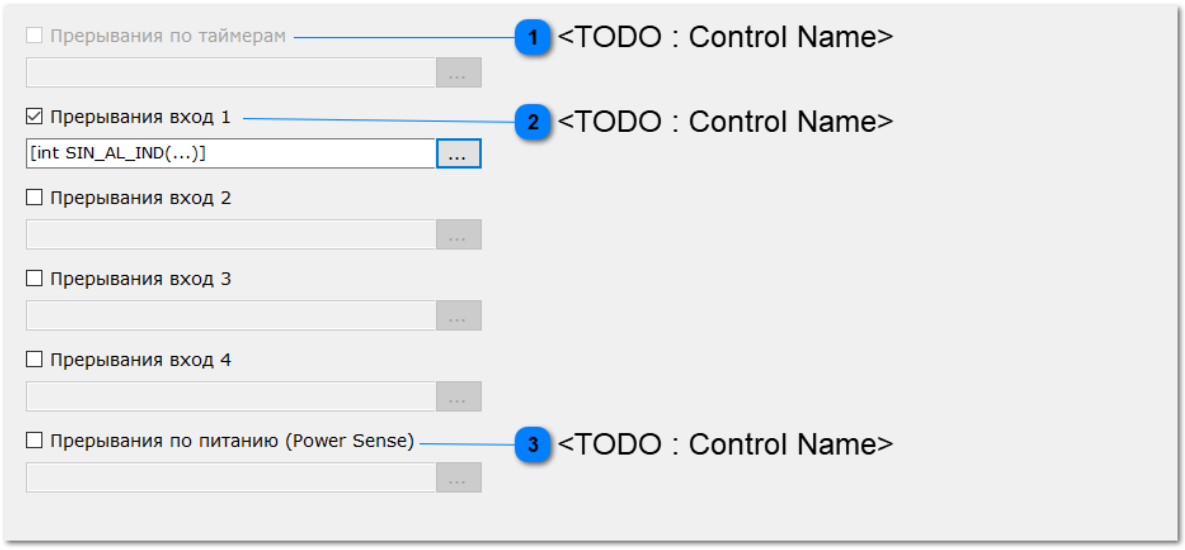
Адрес Modbus начала области для выбранного типа данных.

5 Диапазон

Диапазон адресов Modbus для выбранной области.

Вкладка "Прерывания и события"

Вкладка "Прерывания и события" позволяет настроить функций по прерыванием от таймера, дискретных входов и прерывания по пропаданию питания.



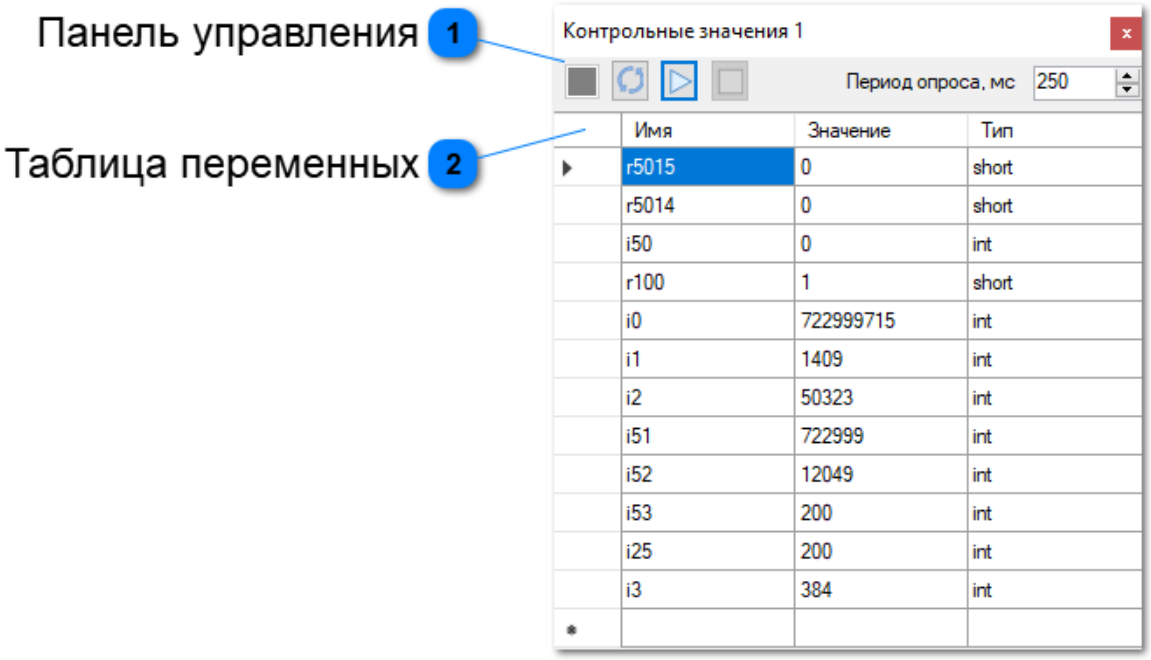
Окна Отладки

Окна отладки, предназначены для откладки кода в процессе выполнения программы.

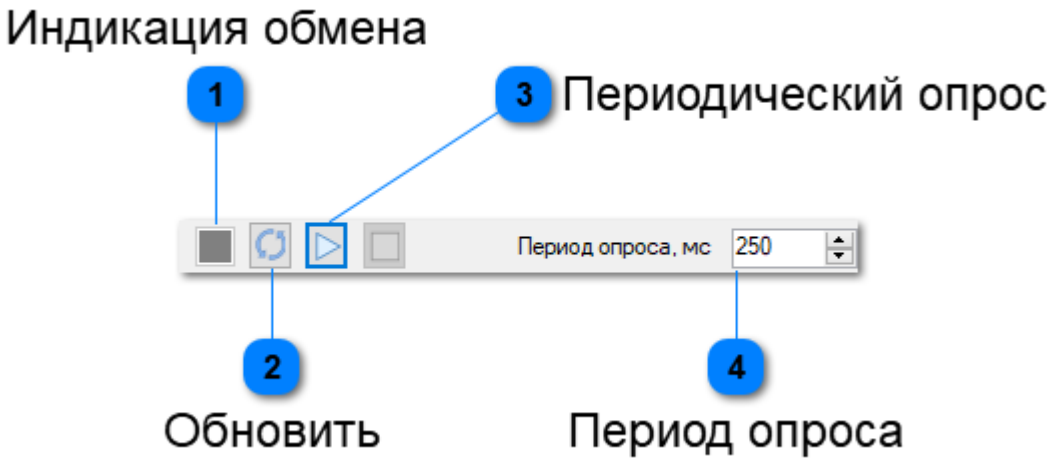
Окно "Контрольные значения"

Окно контрольных значений предназначено для отладки программного кода в процессе его выполнения. Одновременно можно открыть три окна контрольных значений, с различным набором переменных.

ВНИМАНИЕ!!! Для просмотра доступны только глобальные переменные. Для просмотра переменных внутри функции / блока используйте окно "Просмотр блоков".



1 Панель управления



1 Индикация обмена

Если обмен есть, происходит переключение зеленый / серый.

2 **Обновить**

При нажатии на кнопку обновить происходит однократное чтение данных из контроллера.

3 **Периодический опрос**

Обновление данных таблицы переменных с заданным периодом.

4 **Период опроса**

Период опроса контролера / обновления данных таблицы переменных в миллисекундах.

Минимальное значение 250 мс.

2 **Таблица переменных**

1 **Столбец выбора записи**

Для удаления одной или нескольких записей, после выделения записей, надо нажать клавишу "Delete".

2 **Имя переменной**

ВНИМАНИЕ!!! Для корректного отображения требуется компиляция программы.

3 **Значение переменной**

В столбце значение переменной отображается актуальное значение в памяти контроллера. Для изменения переменной в памяти контроллера, необходимо ее отредактировать и

нажать клавишу "Enter".

ВНИМАНИЕ!!! Для полного доступа к переменным, компиляцию проекта следует производить в режиме "Debug". В режиму "Release" доступ к переменным ограничен настройками

"Памяти EX".

Для полного доступа в режиме "Release" необходимо включить отладку на Панели быстрого доступа  .

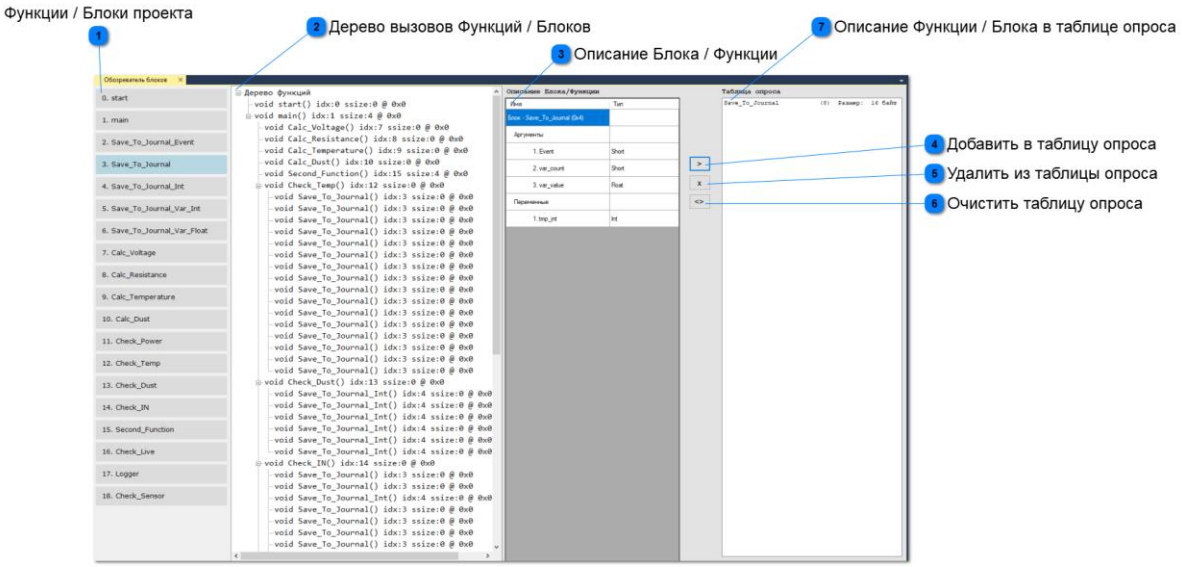
4 **Формат данных**

Формат данных переменной (см. Описание).

Окно "Обозреватель блоков"

Обозреватель блоков показывает структуру откомпилированной программы, с деревом вызовов всех функций в проекте. И позволяет добавить в окно "Просмотр блоков" любую функцию или блок для просмотра в реальном времени следующие параметры:

1. Аргументы Функции / Блока.
2. Статические переменные, определенные внутри Функции / Блока.
3. Динамические переменные определенные внутри Функции / Блока.



1 **Функции / Блоки проекта**

Список всех Функций / Блоков объявленных в проекте.

2 **Дерево вызовов Функций / Блоков**

Дерево вызовов, отображает порядок вызовов Функций / Блоков в проекте.

Горячие клавиши:

F12 - переход к месту вызова функции.

3 **Описание Блока / Функции**

Описание Аргументов и Переменных Функции / Блока.

4 **Добавить в таблицу опроса**

Добавляет Функцию / Блок в таблицу опроса для просмотра в Окне "Просмотр блоков".

5 **Удалить из таблицы опроса**

Удаляет Функцию / Блок из таблицы опроса.

6 **Очистить таблицу опроса**

Полностью очищает таблицу опроса.

7 Описание Функции / Блока в таблице опроса

1 Имя Функции / Блока

2 Индекс вызова

Индекс вызова Функции / Блока однозначно определяет точку вызова в программе.

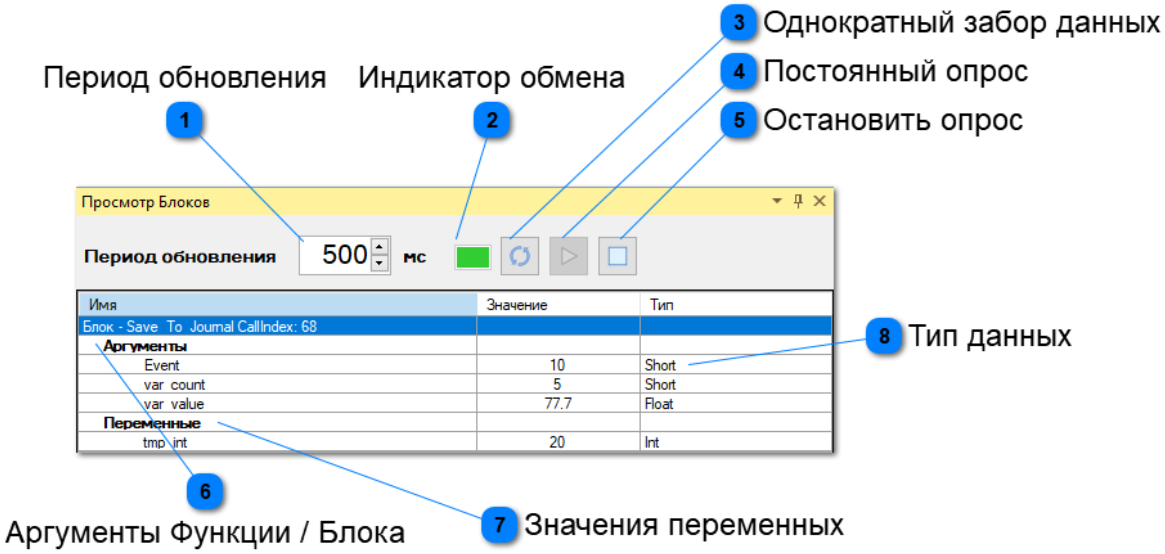
3 Используемая память

Используемая память в Статической Куче.

Окно "Просмотр блоков"

Окно "Просмотр блоков" позволяет просмотреть в реальном времени Аргументы и Переменные внутри работающей Функции / Блока.

ВНИМАНИЕ!!! Доступ возможен только в "Debug" режиме.



1 **Период обновления**

Период обновления данных в миллисекундах.

ВНИМАНИЕ!!! Для большого объема данных предпочтительней использовать соединение TCP.

2 **Индикатор обмена**

```
Переключение Зеленый / Голубой - обмен идет.
Оранжевый                      - попытка опроса контроллера
Серый                          - обмена нет.
```

3 **Однократный забор данных**

Отправка данных в контроллер и однократный запрос данных функций / Блоков.

4 **Постоянный опрос**

Постоянный опрос контроллера с заданным периодом.

5 **Остановить опрос**

ВНИМАНИЕ!!! При записи новой программы в контроллер опрос будет остановлен автоматически.

6 **Аргументы Функции / Блока**

Значения аргументов вызываемой Функции / Блока.

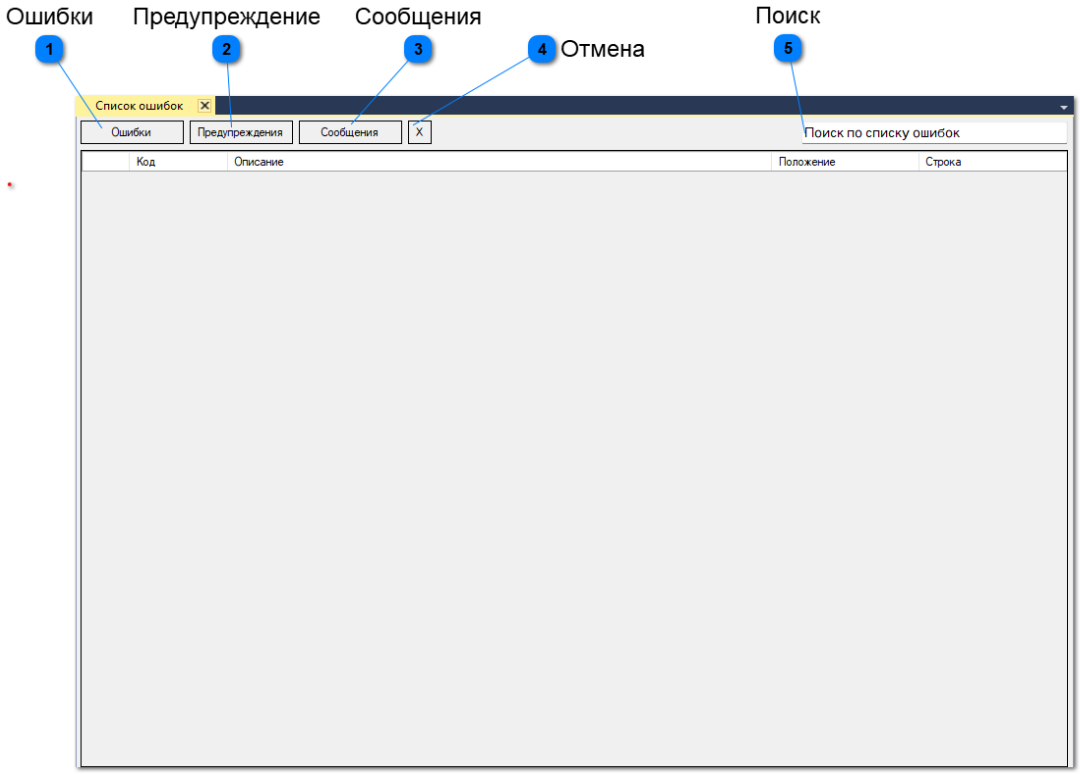
7 **Значения переменных**

Значения переменных используемых внутри Функции / Блока.

Тип данных

Тип данных аргументов и переменных. См. Типы переменных.

Окно "Список ошибок"



Возникающие в процессе компиляции события можно фильтровать следующими кнопками:

1 **Ошибки**

Ошибки

2 **Предупреждение**

Предупреждения

3 **Сообщения**

Сообщения

4 **Отмена**

X

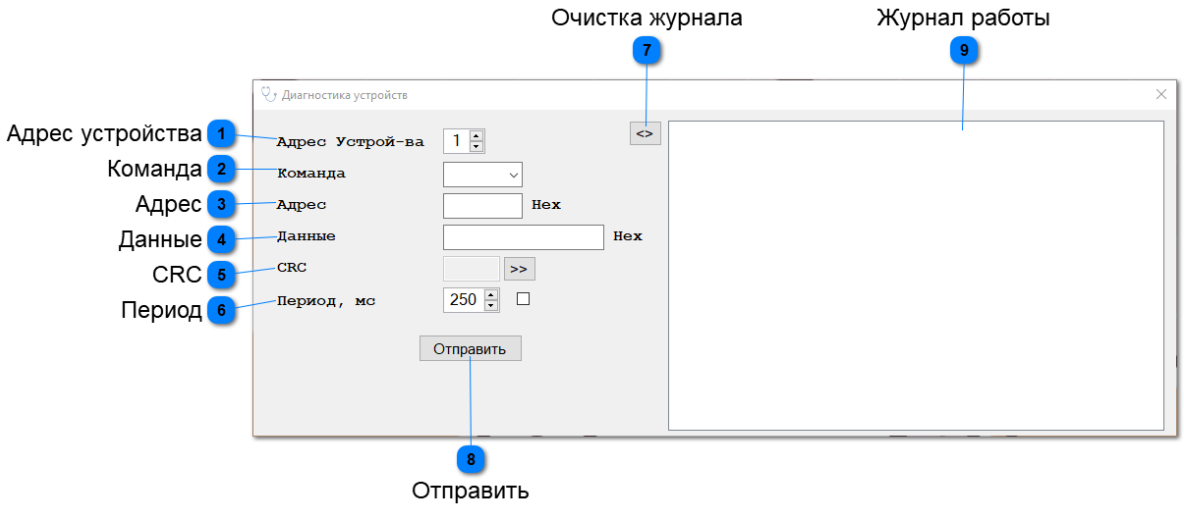
Снятие фильтра типа сообщений

5 **Поиск**

Поиск по списку ошибок

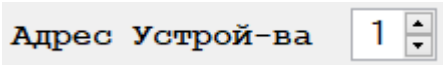
Окна "Настройки периферии"

Окно "Диагностика устройств"

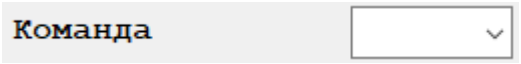


Окно позволяет формировать произвольные запросы в порты Modbus-RTU для slave-устройств

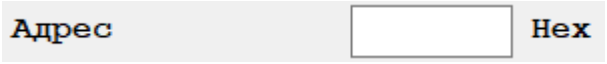
1 Адрес устройства



2 Команда

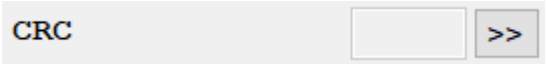


3 Адрес

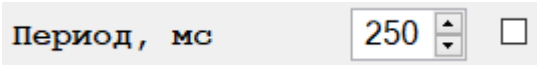


4 Данные

5 CRC

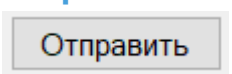


6 Период

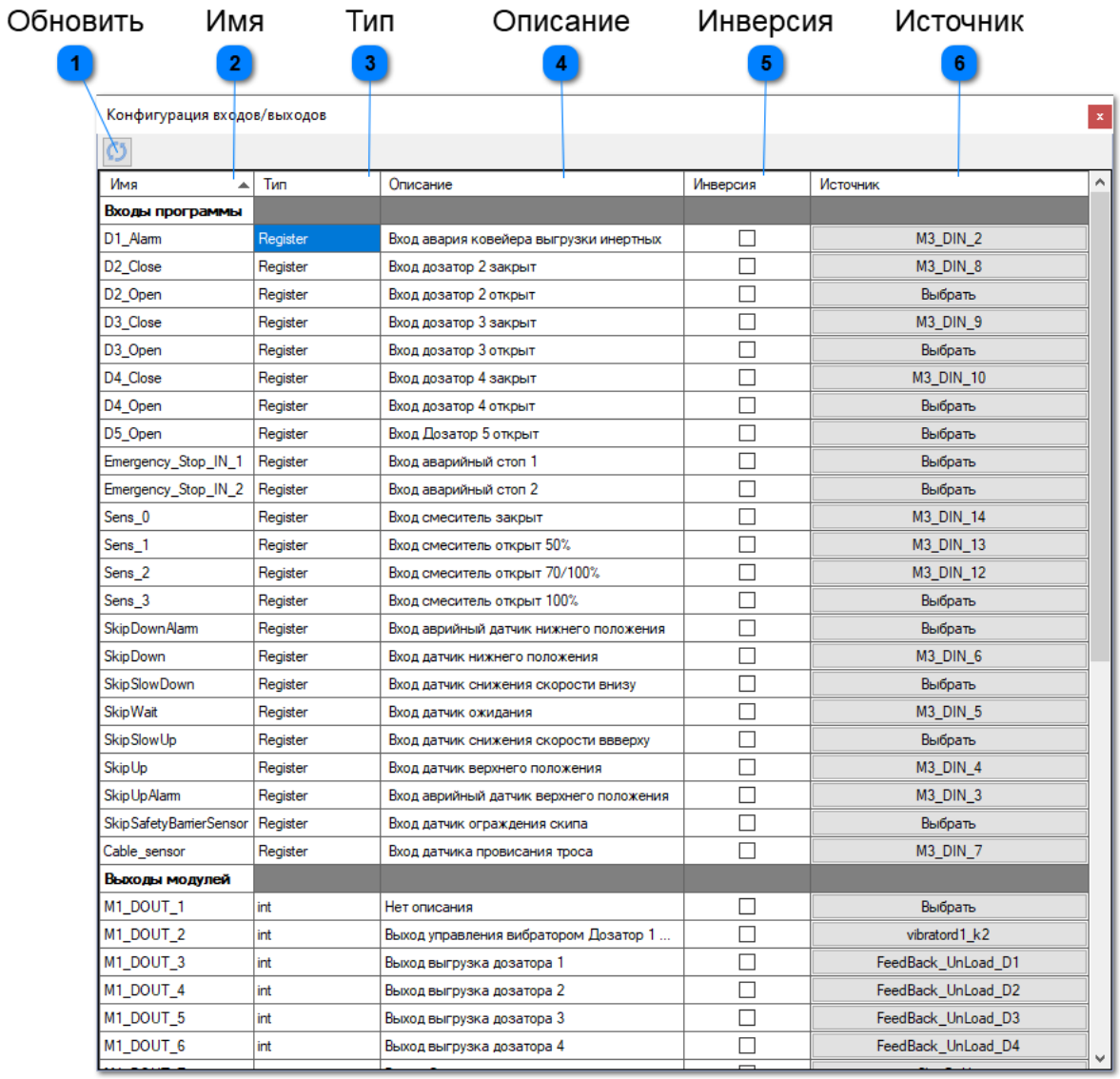


7 Очистка журнала

8 Отправить



Окно настройки входов/выходов

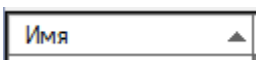


Имя	Тип	Описание	Инверсия	Источник
Входы программы				
D1_Alarm	Register	Вход авария ковейера выгрузки инертных	☐	M3_DIN_2
D2_Close	Register	Вход дозатор 2 закрыт	☐	M3_DIN_8
D2_Open	Register	Вход дозатор 2 открыт	☐	Выбрать
D3_Close	Register	Вход дозатор 3 закрыт	☐	M3_DIN_9
D3_Open	Register	Вход дозатор 3 открыт	☐	Выбрать
D4_Close	Register	Вход дозатор 4 закрыт	☐	M3_DIN_10
D4_Open	Register	Вход дозатор 4 открыт	☐	Выбрать
D5_Open	Register	Вход Дозатор 5 открыт	☐	Выбрать
Emergency_Stop_IN_1	Register	Вход аварийный стоп 1	☐	Выбрать
Emergency_Stop_IN_2	Register	Вход аварийный стоп 2	☐	Выбрать
Sens_0	Register	Вход смеситель закрыт	☐	M3_DIN_14
Sens_1	Register	Вход смеситель открыт 50%	☐	M3_DIN_13
Sens_2	Register	Вход смеситель открыт 70/100%	☐	M3_DIN_12
Sens_3	Register	Вход смеситель открыт 100%	☐	Выбрать
SkipDownAlarm	Register	Вход аврийный датчик нижнего положения	☐	Выбрать
SkipDown	Register	Вход датчик нижнего положения	☐	M3_DIN_6
SkipSlowDown	Register	Вход датчик снижения скорости внизу	☐	Выбрать
SkipWait	Register	Вход датчик ожидания	☐	M3_DIN_5
SkipSlowUp	Register	Вход датчик снижения скорости ввверху	☐	Выбрать
SkipUp	Register	Вход датчик верхнего положения	☐	M3_DIN_4
SkipUpAlarm	Register	Вход аврийный датчик верхнего положения	☐	M3_DIN_3
SkipSafetyBarrierSensor	Register	Вход датчик ограждения скипа	☐	Выбрать
Cable_sensor	Register	Вход датчика провисания троса	☐	M3_DIN_7
Выходы модулей				
M1_DOUT_1	int	Нет описания	☐	Выбрать
M1_DOUT_2	int	Выход управления вибратором Дозатор 1 ...	☐	vibratord1_k2
M1_DOUT_3	int	Выход выгрузка дозатора 1	☐	FeedBack_UnLoad_D1
M1_DOUT_4	int	Выход выгрузка дозатора 2	☐	FeedBack_UnLoad_D2
M1_DOUT_5	int	Выход выгрузка дозатора 3	☐	FeedBack_UnLoad_D3
M1_DOUT_6	int	Выход выгрузка дозатора 4	☐	FeedBack_UnLoad_D4

1 **Обновить**

Обновляет данные списка.

2 **Имя**



Входы программы это переменные, помеченные модификатором "input".

Выходы модулей это переменные, определенные в файле "/lib/Modules.lgc".

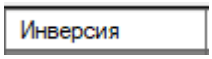
3 **Тип**

Тип переменной или модуля.

4 **Описание**

Описание переменных в файле как "#d Описание". Редактируемое.

5 **Инверсия**



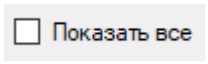
Будет ли значение источника инвертировано при записи.

6 **Источник**

Источник данных для переменной или модуля.

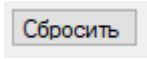
Окно выбора переменной

1 Показать все



Отображает все переменные независимо от того, использованы они или нет.

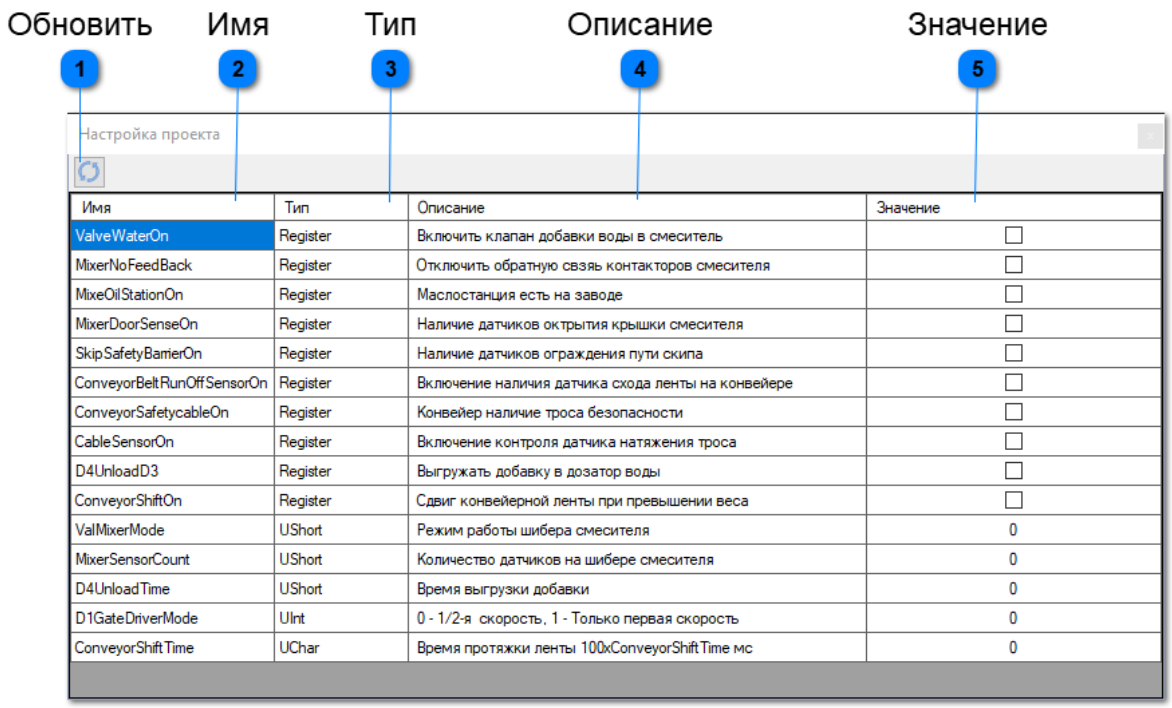
2 Сброс



Сбрасывает значение переменной.

Окно настройки проекта

В этом окне находятся все переменные, помеченные модификатором "config".



Имя	Тип	Описание	Значение
ValveWaterOn	Register	Включить клапан добавки воды в смеситель	☐
MixerNoFeedBack	Register	Отключить обратную связь контакторов смесителя	☐
MixeOilStationOn	Register	Маслостанция есть на заводе	☐
MixerDoorSenseOn	Register	Наличие датчиков октрытия крышки смесителя	☐
SkipSafetyBarrierOn	Register	Наличие датчиков ограждения пути скипа	☐
ConveyorBeltRunOffSensorOn	Register	Включение наличия датчика схода ленты на конвейере	☐
ConveyorSafetycableOn	Register	Конвейер наличие троса безопасности	☐
CableSensorOn	Register	Включение контроля датчика натяжения троса	☐
D4UnloadD3	Register	Выгружать добавку в дозатор воды	☐
ConveyorShiftOn	Register	Сдвиг конвейерной ленты при превышении веса	☐
ValMixerMode	UShort	Режим работы шибера смесителя	0
MixerSensorCount	UShort	Количество датчиков на шибере смесителя	0
D4UnloadTime	UShort	Время выгрузки добавки	0
D1GateDriverMode	UInt	0 - 1/2-я скорость, 1 - Только первая скорость	0
ConveyorShiftTime	UChar	Время протяжки ленты 100xConveyorShiftTime мс	0

1 **Обновить**

Обновляет список переменных.

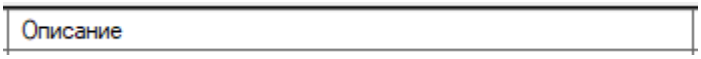
2 **Имя**

Имя переменной.

3 **Тип**

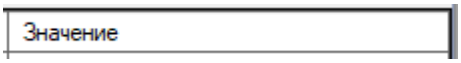
Тип переменной.

4 **Описание**



Описание переменной. Редактируемое.

5 **Значение**



Значение переменной.

Горячие клавиши

Следующий раздел содержит все сочетания клавиш и способы управления при помощи мыши, поддерживаемые UpakNet.

Общие

F1 — показать контекстную справку.
F12 — перейти к объявлению функции, переменной.
Ctrl+N — создать новый проект.
Ctrl+O — открыть проект.
Ctrl+S — сохранить открытый проект.

Поиск

Ctrl+F — найти в текущем файле.
Ctrl+Shift+F — поиск по проекту.

Редактирование

Alt+BackSpace — отменить.
Shift+Delete — вырезать.
Shift+Insert — вставить.
Ctrl+C — копировать.
Ctrl+Insert — копировать.
Ctrl+X — вырезать.
Ctrl+V — вставить.
Ctrl+Y — повторить.
Ctrl+A — выбрать все.
Ctrl+Z — отменить.
Ctrl+Shift+Z — повторить.

Описание языков программирования

Программа состоит из двух частей – «Upak» и «Logic». Выполнение этих двух частей выполняется последовательно друг за другом в цикле с заданным периодом. Период выполнения программы настраивается индивидуально для каждого проекта в окне "Настройки контроллера".

ВНИМАНИЕ!!! В контроллере предусмотрена функция автоматической коррекции периода выполнения программы при загрузке процессора выше 100%.

Шаг коррекции периода - 1 мс.

Описание языка Upak

Upak - язык релейной логики (ELL).

Роль переменными в программе исполняют номера логических регистров. Программа состоит из строк, каждая из которых (в случае если она не пустая или не является комментарием) представляет собой одну логическую формулу. Формула представляет собой логическое выражение, состоящее из двух частей, разделённых знаком «=». Левая часть содержит номера регистров, операторы и скобки, правая – номер регистра, которому необходимо присвоить результат вычисления левой части.
Операторов реализовано четыре: «+» - логическое ИЛИ, «*» - логическое И, «^» - исключающее ИЛИ, «~» - логическое НЕ.

Операторы «+», «^» и «*» применимы к двум операндам, которыми могут быть либо переменные, либо выражения в скобках. Оператор «~» применим к переменной, либо
к выражению в скобках.

Операции осуществляются со значениями находящимися в логических регистрах. Номера допустимых регистров от 0 до 8191. Часть регистров зарезервирована для специальных целей в приборе (см. системные переменные типа r) или используется для ввода
и вывода через модули Modbus, для использования таймерами и т.п.

В программе возможна замена номеров логических регистров на буквенные обозначения. Для этого необходимо в тексте программы добавить строку-замену следующего вида:

```
#?100=FLAG_START
#?101=TEST_BIT
```

Запись программы в номерах регистров:

```
100=101
```

Запись программы в именованных переменных:

```
#?100=FLAG_START
#?101=TEST_BIT

FLAG_START=TEST_BIT
```

```
# Программа может содержать комментарии. Комментарием является всё, что
находится
# справа от символа комментария «#».
```

Далее рассмотрим простейший пример реализации релейной схемы запуска двигателя в программе "Upak"

При написании программы в выражениях на языке ELL необходимо учитывать приоритет выполнения операций.
Самый высокий приоритет имеет выражение, заключенное в скобки.
Далее приоритет операций уменьшается в следующем порядке: «~», «*», «^», «+».

Для операции отрицания вместо символа «~» допустимо использовать символ «!». Данные символы операции отрицания полностью взаимозаменяемы.

Для ускорения выполнения программы возможен пропуск обработки части выражений
в текущем цикле обработки. Решение о том надо или не надо обрабатывать блок формул принимается прибором в каждом цикле обработки на основе результатов проверки условного регистра. Номер этого регистра указывается в программе.

Любой условный блок начинается с сочетания символов #* , далее следует признак начала условного перехода - START и само выражение «REG_NAME=1». Данная запись обозначает, что если регистр REG_NAME на данном этапе выполнения программы равен логической «1», то будут вычислены все выражения, следующие за данной строкой и вплоть до первой строки с признаком конца условного блока «STOP».
Условие перехода можно записать и так: «#* START REG_NAME=0», в таком случае выражения в условном блоке будут выполнены, если регистр равен логическому «0».

Применение условных операторов START / STOP

Для прерывания обработки текущего цикла логической программы предусмотрена команда досрочного завершения. Аналогично условным блокам, при совпадении условия в команде
прерывания - текущий цикл обработки логической программы прерывается.

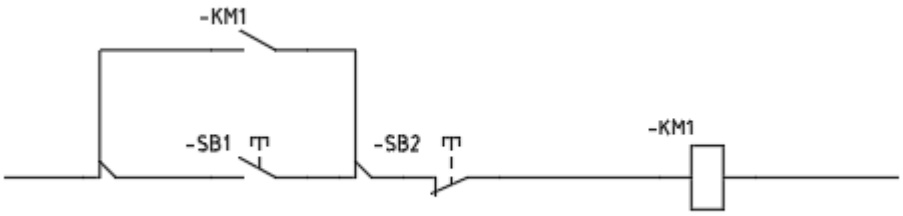
Пример:

```
#* BREAK TEST_BIT=1
```

Примеры использования

Запуск Двигателя

Релейная схема примера:



Реализация в программе Upak:

```
#?1000=KM1
#?1001=SB1
#?1002=SB2

(SB1+KM1)*~SB2=KM1
```

Условные операторы START / STOP

Пример использования:

```
#?100=FLAG_START
#?101=TEST_BIT

#* START FLAG_START=1
 #Код программы, который выполняется когда FLAG_START=1
 FLAG_START=TEST_BIT
#* STOP
```

Пример использования вложенных условных переходов:

```
#* START FLAG_START=1

 #Код программы, который выполняется когда FLAG_START=1
 FLAG_START=TEST_BIT
 #* START FLAG_START_2=1
    #Код программы, который выполняется когда FLAG_START_2=1
    FLAG_START_2=TEST_BIT_2
 #* STOP

#* STOP
```

Описание языка Logic

Logic (LC) – это текстовый язык высокого уровня общего назначения, по синтаксису схожий с языком C. Удобен для программ, включающих числовой анализ или сложные алгоритмы. Может использоваться в программах, в теле функции или функционального блока.

Выражения в LC выглядят точно также, как и в языке C:

[variable] = [value];

Порядок их выполнения – справа налево. Выражения состоят из операндов и операторов. Операндом является литерал, переменная, структурированная переменная, компонент структурированной переменной, обращение к функции или прямой адрес.

Программа состоит из двух обязательных функций:

```
# Функция start, выполняющаяся один раз на старте контроллера
void start()
{

}

# Функция main, выполняющаяся в цикле с заданным периодом
void main()
{

}
```

Области памяти / Типы переменных

В контроллере доступно 3 типа памяти:

1. RAM - Оперативная память размером 65 кБайт.
2. FRAM - Энергонезависимая память контроллера размером 1020 / 2040 Байт. см. Свойства проекта / Память.
3. Битовые переменные - Размер области 8 192 бита.

Типы глобальных переменных в RAM:

```
char    - целочисленная переменная размером 1 байт
(от -128 до +127)
short   - целочисленная переменная размером 2 байт
(от -32,768 до +32,767)
int     - целочисленная переменная размером 4 байт
(от -2,147,483,648 до 2,147,483,647)
long    - целочисленная переменная размером 4 байт
(от -2,147,483,648 до 2,147,483,647)
uchar   - целочисленная без знаковая переменная размером 1
байт     (от 0 до 255)
ushort  - целочисленная без знаковая переменная размером 2
байт     (от 0 до 65,535)
uint    - целочисленная без знаковая переменная размером 4
байт     (от 0 до 4,294,967,295)
ulong   - целочисленная без знаковая переменная размером 4
байт     (от 0 до 4,294,967,295)

float   - дробная переменная с плавающей точкой размером 4
байт
double  - дробная переменная с плавающей точкой размером 8
байт
```

Типы глобальных переменных в FRAM:

```
nvm char    - целочисленная переменная размером 1 байт
(от -128 до +127)
nvm short   - целочисленная переменная размером 2 байт
(от -32,768 до +32,767)
nvm int     - целочисленная переменная размером 4 байт
(от -2,147,483,648 до 2,147,483,647)
nvm long    - целочисленная переменная размером 4 байт
(от -2,147,483,648 до 2,147,483,647)
nvm uchar   - целочисленная без знаковая переменная размером 1
байт (от 0 до 255)
```

```
nvm ushort  - целочисленная без знаковая переменная размером 2
байт (от 0 до 65,535)
nvm uint    - целочисленная без знаковая переменная размером 4
байт (от 0 до 4,294,967,295)
nvm ulong   - целочисленная без знаковая переменная размером 4
байт (от 0 до 4,294,967,295)

nvm float   - дробная переменная с плавающей точкой размером 4
байт
nvm double  - дробная переменная с плавающей точкой размером 8
байт
```

Битовые глобальные переменные RAM Register:

```
reg int    - битовая переменная (0/1)
```

Примеры объявления:

```
Определение переменной типа float
float var_float;
Определение переменной типа Float по абсолютному адресу
float var_float = &0;
```

Конструкции языка

Формульные выражения записываются в обычном математическом виде, с учётом приоритетов операций. В левой части выражения указывается операнд, в который записывается результат вычисления правой части выражения. Конец строки всегда должен завершаться точкой с запятой «;».

Арифметические операции:

"+" – сложение
"–" – вычитание
"*" – умножение
"/" – деление
"^" – возведение в степень
"%" – деление по модулю

ВНИМАНИЕ!!!! Переменные в выражении должны быть одного типа, если тип данных отличается то требуется преобразование.
Пример:

```
int tmp;
float tmp_f;
tmp_f=1.0+(float)tmp;
```

Логические операции:

"&" – логическое (побитовое) И
"|" – логическое (побитовое) ИЛИ
"!" – логическое (побитовое) отрицание

Приоритеты операций в порядке уменьшения:

1) ! отрицание
2) ^ возведение в степень
3) /, %, * деление, деление по модулю, умножение
4) +, - сложение, вычитание
5) & логическое И
6) | логическое ИЛИ

Операции сравнения:

"=" - равно
"~" - не равно
">" - больше

"<" - меньше

Пример записи операции сравнения:

```
if (T1_FAIL=1)
{
  Logic->Save_To_Journal(1,1,T1);
}
else
{
  Logic->Save_To_Journal(22,1,T1);
}
```

Цикл While.

Служит для определения цикла с предусловием. Цикл будет исполняться до тех пор, пока выражение в предложении WHILE возвращает TRUE. Формат конструкции следующий:

```
while <Boolean-Expression>
{
   <Statement List>
}
```

Значение <Boolean-Expression> проверяется на каждой итерации. Завершение цикла произойдет, если выражение <Boolean-Expression> вернет FALSE.

Например:

```
int a;
a = 10;

float b;
b = 1.0;

while(a > 0)
# Сначала происходит проверка выражения
{
   # Потом выполняется соответствующий ему код
   a = a - 1;
   b = b * 2.0;
}
```

Функции

Функции - отдельные блоки кода, которые можно вызвать к выполнению в любой момент программы, и получить результирующие данные.
Можно считать функции как отдельные малые программы, имеющие доступ к памяти контроллера, а всю программу в общем и целом - сборник этих функций.

Функции состоят из двух основных частей - объявление и блок кода.
Объявление функции - её описание, описывает входные данные - аргументы, тип выходных данных и название.
Общий формат выглядит следующим образом:
<type> <name>(<type> <name>, <type> <name>, ...)

Объявим простейшую функцию, она не принимает данных и ничего не возвращает:
тип `void` - обозначает что функция возвращает пустой тип данных размером 0
то есть буквально ничего, в таком случае, функция не обязана возвращать данные.

Пример объявления и вызова функции:

```
void FunctionsTest()
{
   i50=i50+1;
}

void start()
{

}

void main()
{
 Logic->FunctionsTest();
}
```

В результате выполнения программы, переменная i50 будет инкреминитироваться каждый цикл программы.

Объявим функцию которая и принимает и возвращает значение.
Переменные которые объявлены в круглых скобках называются - "Аргументы" функции.

⚠ Внутри функции аргументы доступны только для чтения.

```
float DoubleValue(float value)
{
    float tmp_f;
    tmp_f = value * 2.0;

    return tmp_f;
}
```

Использование статичеких перменных внутри функций.

Модификатор `static` перед типом переменной указывает компилятору, что для данной переменной необходимо выделить память
в статической куче, таким образом, после выполнения функции переменная не будет очищена и при следующем вызове функции значение
переменной будет загружено из памяти.

Пример использования:

```
void test_static()
{
  static int tmp_i;

  tmp_i=i50+tmp_i+1;

  i50=tmp_i;

}

void start()
{

}


void main()
{
 if (r100=1)
 {
```

```
        r100=0;
        Logic->test_static();
    }
}
```

Функциональные блоки

Функциональные блоки - функции с определёнными дополнительными возможностями обращения и хранения статической памяти. Для каждого вызова или объявления функционального блока по телу программы создается отдельный экземпляр статических перменных, которые загружаются при вызове блока.

Пример объявления Функционального блока:

```
block void SomeBlock(int initIndex)
{
    static float blockFloat1;
    static float blockFloat2;

    if(initIndex = 0)
    {
        blockFloat1 = 123;
        blockFloat2 = 321;
    }

    if(initIndex = 1)
    {
        blockFloat1 = 456;
        blockFloat2 = 654;
    }
}
```

При вызове функционального блока ему можно присвоить имя объекта, и после этого обращаться к статическим переменным внутри этого блока.

Пример:

```
void BlocksTest()
{
    Logic->SomeBlock(0) as Block1;

    Logic->SomeBlock(1) as Block2;

    float _r1;
    float _r2;
```

```
    _r1 = Block1.blockFloat1;
    _r2 = Block2.blockFloat1;

    # Несмотря на то, что обращение идёт к одной и той же переменной,
внутри _r1 будет "123", а внутри _r2 будет "456"

    blockResult1 = Block1.blockFloat1;
    blockResult2 = Block1.blockFloat2;
    blockResult3 = Block2.blockFloat1;
    blockResult4 = Block2.blockFloat2;
}
```

Для отладки работы функциональных блоков воспользуйтесь окнами "Обозреватель блоков" и "Просмотр блоков".

Указатели

В отличие от *статических* переменных, помеченных ключевым словом **static**, вы так же можете использовать ключевое слово **pointer**, что сделает переменную *указателем*.
В отличие от *статической* переменной, которая хранит какое-то полезное значение типа "0" или "42" (и это значение хранится в отдельной области памяти, размеры которой неизменны и определены заранее), *указатель* хранит адрес памяти, где это полезное значение лежит.
Таким образом, *указатель* не хранит в себе какое-либо значение, он указывает на область оперативной памяти контроллера, в которой лежит какое-либо значение.

Создадим указатель:

```
pointer int somePointer;
```

После своего создания, в отличие от переменных типа **static**, *указатель* всё ещё нельзя использовать, т.к. при создании все такие переменные указывают на адрес 0, поэтому данную переменную необходимо обязательно инициализировать начальным значением, иначе возможны критические ошибки при выполнении работы пользовательской программы.

Для проведения инициализации используется ключевое слово `new`. Это ключевое слово выделет необходимый размер области памяти, зависящий от типа данных (в примере выше это `int`) и выдаст указателю адрес.

Теперь наш пример выглядит так:

```
pointer int somePointer;        # Создание указателя int
somePointer = new; # Инициализация указателя
```

Так как регисты в памяти контроллера лежат вне выделенной памяти, для них существует отдельный указатель при инициализации - `regs`.
Суть процедуры инициализации при этом не меняется:

```
pointer int pRegs; # Создание указателя reg int
pRegs = regs; # Инициализация указателя
```

Какие задачи позволяют решать указатели?

В основном, главной задачей указателей является работа с памятью и массивами данных внутри контроллера.

Системные переменные

Используются контроллером для настройки и отображения режимов работы и переменных окружения.

Переменные типа i

Переменные типа i (Целочисленные переменные)

```
i0  (R/W) Счетчик времени в миллисекундах;
i1  (R/W) Текущее время в формате ЧЧММСС;
i2  (R/W) Текущая дата в формате ДДММГГ;
i3  (R)   Частота обмена с приборами;
i4  (R)   Загрузка процессора в процентах;
i5  (R/W) Адрес битовой области для внутренних входов/выходов # 4
бита входы 8 бит выходы;
i6  (R)   Период выполнения программы мили секунды;
i7  (R)   Частота выполнения программы;
i8  (R)   Частота обмена с компьютером;
i9  (R)   Загруженная страница памяти программы;
i10 (R/W) Энкодер CH1;
i11 (R/W) Энкодер CH2;
i12 (R/W) Не используется;
i13 (R/W) Не используется;
i13 (R/W) Не используется;
i15 (R)   Размер программы;
i16 (R)   Размер переменных окружения программы;
i17 (R)   Внутренний аналоговый вход  0;
i18 (R)   Внутренний аналоговый вход  1;
i19 (R)   Внутренний аналоговый вход  2;
i20 (R)   Внутренний аналоговый вход  3;
i21 (W)   Внутренний аналоговый выход 0;
i22 (W)   Внутренний аналоговый выход 1;
i23 (W)   Внутренний аналоговый выход 2;
i24 (W)   Внутренний аналоговый выход 3;
i25 (R)   Количество ошибок обмена;
i26 (R)   Частота обмена Uart 0;
i27 (R)   Частота обмена Uart 1;
i28 (R)   Частота обмена Uart 2;
i29 (R)   Частота обмена Uart 3;
i30 (R)   Частота обмена Uart 4;
i31 (R)   Состояние программы контроллера:
                    0 - программа не выполняется
```

1 - программа выполняется
2 - ошибка программы
3 - ошибка распределения памяти;

i32 (R/W) Счетчик IN1;
i33 (R/W) Счетчик IN2;
i34 (R/W) Счетчик IN3;
i35 (R/W) Счетчик IN4;
i36 (R/W) Состояние загрузки функций программы:
0 - функции загружены
1 - ошибка загрузки функций

Переменные типа r

Переменные типа r (Логические регистры)

r3936-4015 (R) Бит старта ручной загрузки по компонентам 1-8 для Дозаторов 1-10 (см. Описание);
r4016-4095 (R) Биты блокировки ручного режима для компонентов 1-8 каждого из 10 Дозаторов (см. Описание);
r8001–r8010 (R) Индикация процесса загрузки материала по Дозаторам 1-10 (1 – идет загрузка 0 – загрузки нет);
r8011 (R) Энкодер CH1 направление вращения 1 - вперед 0 - назад;
r8012 (R) Энкодер CH2 направление вращения 1 - вперед 0 - назад;
r8014 (R) Системный таймер, меандр 100 мс;
r8015 (R) Системный таймер, меандр 1000 мс;
rYYXX (R) Используются для индикации ошибок связи с модулями(0 – есть обмен 1 – нет обмена),
где XX номер модуля начиная с 00, первыми идут модули дозатора, затем модули из таблицы обмена.
Настройка области производится в настройках контроллера.

Стандартные функции

В компиляторе предусмотрены стандартные функции, доступ к которым осуществляется с ключевым словом `mstd->X`, где X - имя функции.

Пример вызова стандартной функции:

```
int regIndex;
int regValue;

regIndex = 123;
regValue = 0;

mstd->SetBit(regIndex, regValue);
```

Управление CPU

Функции управления CPU:

```
CPU__PutToSleep(); -> войти в Sleep режим (выход возможен из прерываний)
CPU__WakeUp();     -> выйти из Sleep режима
Reset();           -> перезагрузка контроллера
```

Пример использования:

```
mstd->CPU__PutToSleep();
mstd->CPU__WakeUp();
mstd->Reset();
```

Работа с Flash

В контроллере доступна пользовательская Flash память размером 7 Мбайт .

Функции работы с Flash:

WriteToFlash (int Address, *int Var_Pointer, int Size) -> Запись данных
ReadFromFlash (int Address, *int Var_Pointer, int Size) -> Чтение данных

Описание аргументов функций:

Address	- адрес во Flash памяти диапазон 0 - 7 340 032
Var_Pointer	- адрес чтения/записи берется из переменной в аргументе
Size	- количество байт для записи

Пример использования:

```
float f1 = &0;
float f2 = &1;
int   Address;
int   Size;

# Запись во флэш память переменных f1,f2
Address=0;
Size=8;
mstd->WriteToFlash(Address, f1, Size);

# Чтение из флэш памяти переменных f1,f2
Address=0;
Size=8;
mstd->ReadFromFlash(Address, f1, Size);
```

Математические функции

Математические функции:

```
  float ln    (float arg)                  -> натуральный логарифм
  float pow   (float base, float exponent) -> возведение в степень
  float sin   (float arg)                  -> синус
  float cos   (float arg)                  -> косинус
  float tan   (float arg)                  -> тангенс
  float abs   (float arg)                  -> модуль
  float sqrt  (float arg)                  -> квадратный корень
```

Пример использования:

```
float tmp_f;
float tmp_f1;
tmp_f=1.7;
tmp_f1=mstd->ln(tmp_f);
```

Функции модуля Дозатора

Функции работы с модулем Дозатора:

```
 float GetDoserMassNet(n);          -> Вес Нетто
 float GetDoserMassBrut(n);         -> Вес Брутто
 float SetDoserActualMass(n,ves)    -> Задает текущую массу для виртуального
```
дозатора

Описание аргументов функций:

n - номер дозатор 0-9

ves - вес в дозаторе

Пример использования:

```
float Ves_Netto  = &0;
float Ves_Brutto = &1;

# Получения веса нетто дозатора 0
Ves_Netto = mstd->GetDoserMassNet(0);

# Получения веса брутто дозатора 0
Ves_Brutto = mstd->GetDoserMassBrut(0);

# Задание вес дозатора 1, значение Ves_Brutto
mstd->SetDoserActualMass(1,Ves_Brutto);
```

Состояние соединения TCP

В контроллере доступна пользовательская Flash память размером 7 Мбайт.

Функции работы с соединением TCP:

```
mstd->GetConnectionState();
```

Пример использования:

```
int   State = &0;

# Чтение состояния соединения TCP
State=mstd->GetConnectionState();

Если State = 1 - соединение установленно.
```

Стандартные блоки
Блоки битовых операций

Список блоков:

Bit_Clear

Назначение:

Очищает бит с номером n в числе Number.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Number	Изменяемое число
reg int	n	Бит для изменения

Пример использования:

```
blocks->Bit_Clear(Number,n);
```

Bit_Set

Назначение:

Устанавливает бит с номером n в числе Number.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Number	Изменяемое число
reg int	n	Бит для изменения

Пример использования:

```
blocks->Bit_Set(
    Number,
```

```
        n
);
```

Bit_Toggle

Назначение:

Инвертирует бит с номером n в числе Number.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Number	Изменяемое число
reg int	n	Бит для изменения

Пример использования:

```
blocks->Bit_Toggle(
        Number,
        n
);
```

Bit_Get

Назначение:

Возвращает в переменную Out бит с номером n из числа Number.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Number	Изменяемое число
reg int	n	Бит для изменения
reg int	Out	Переменная для сохранения выходного значения

Пример использования:

```
blocks->Bit_Get(
        Number,
        n,
        Out
);
```

Блоки обмена

Список блоков:

Uart_RW

Назначение:

Блок управляет работой с протоколом Modbus RTU.

Описание работы блока:

Если у контроллера с индексом Uart_Index есть поток, State присвоит себе значение "1" и блок завершит работу. Иначе, блок создаст новый поток:
в зависимости от кода операции Modbus RTU (MB_Code) будут выполнены следующие команды:

"0x04" - чтение регистров.
"0x06" - запись данных в регистр.
"0x10" - запись данных в несколько регистров.

Переменные adr, Uart_Index, NOFregs, Start_Reg используются для указания адреса устройства-получателя, номера используемого интерфейса контроллера-отправителя, количество отправляемых регистров и номер начального регистра.

Работа блока завершается в случаях если:
1) Данный поток уже используется;
2) Uart_Index имеет значение, которое больше, чем "4".
3) Поток контроллера с индексом Uart_Index уже используется.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Uart_Index	Индекс интерфейса контроллера (0-4)
reg int	adr	Адрес устройства

reg int	MB_Code	Код операции Modbus RTU
reg int	NOFregs	Кол-во регистров
reg int	Start_Reg	Начальный регистр
short	Data	Отправляемая информация
short	State	Состояние потока: 1 - поток работает, 2 - ошибка создания потока.

Пример использования:

```
blocks->Uart_RW(
      Uart_Index,
      adr,
      MB_Code,
      NOFregs,
      Start_Reg,
      Data,
      State
);
```

Uart_RW_REGS

Назначение:

Блок управляет работой с протоколом Modbus RTU.

Описание работы блока:

Если у контроллера с индексом Uart_Index есть поток, State присвоит себе значение "1" и блок завершит работу. Иначе, блок создаст новый поток:
в зависимости от кода операции Modbus RTU (MB_Code) будут выполнены следующие команды:

"0x04" - чтение регистров.
"0x06" - запись данных в регистр.
"0x10" - запись данных в несколько регистров.

Переменные adr, Uart_Index, NOFregs, Start_Reg используются для указания адреса устройства-получателя, номера используемого интерфейса контроллера-отправителя, количество отправляемых регистров и номер начального регистра.

Работа блока завершается в случаях если:

1) Данный поток уже используется;

2) Uart_Index имеет значение, которое больше, чем "4".

3) Поток контроллера с индексом Uart_Index уже используется.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Uart_Index	Индекс интерфейса контроллера (0-4)
reg int	adr	Адрес устройства
reg int	MB_Code	Код операции Modbus RTU
reg int	NOFregs	Кол-во регистров
reg int	Start_Reg	Начальный регистр
short	Data	Отправляемая информация
short	State	Состояние потока: 1 - поток работает, 2 - ошибка создания потока.

Пример использования:

```
blocks->Uart_RW_REGS(
     Uart_Index,
     adr,
     MB_Code,
     NOFregs,
     Start_Reg,
     Data,
     State
);
```

Uart_RW_Merc

Назначение:

Блок управляет работой с протоколом Modbus RTU.

Описание работы блока:

Если у контроллера с индексом Uart_Index есть поток, State присвоит себе значение "1" и блок завершит работу. Иначе, блок создаст новый поток:
в зависимости от кода операции Modbus RTU (MB_Code) будут выполнены следующие команды:

"0x04" - чтение регистров.
"0x06" - запись данных в регистр.
"0x10" - запись данных в несколько регистров.

Переменные adr, Uart_Index, NOFregs, Start_Reg используются для указания адреса устройства-получателя, номера используемого интерфейса контроллера-отправителя, количество отправляемых регистров и номер начального регистра.

Работа блока завершается в случаях если:
1) Данный поток уже используется;
2) Uart_Index имеет значение, которое больше, чем "4".
3) Поток контроллера с индексом Uart_Index уже используется.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Uart_Index	Индекс интерфейса контроллера (0-4)
reg int	adr	Адрес устройства
reg int	MB_Code	Код операции Modbus RTU
reg int	NOFregs	Кол-во регистров
reg int	Start_Reg	Начальный регистр
short	Data	Отправляемая информация

short	State	Состояние потока: 1 - поток работает, 2 - ошибка создания потока.

Пример использования:

```
blocks->Uart_RW_Merc(
     Uart_Index,
     adr,
     MB_Code,
     NOFregs,
     Start_Reg,
     Data,
     State
);
```

Блоки управления дозированием

Список блоков:

1. DosingBlock;
2. SetExtraTimeUnload;
3. LatchWithTimeOut;
4. ToggleWithTimeOut;
5. SetUPAEmpty;
6. SaveAndRestoreBits;
7. StoneGateDriver;
8. DozingValveEuromix.
9. DozingValveAmmann

DosingBlock

Назначение:

Блок DosingBlock предназначен для управления системой дозаторов от 1-го до 10-ти, количество дозаторов определятся настройками в окне "Дозаторы". Блок формирует все сигналы для работы Scada - "Weigher", формирует сигналы для учета материалов в ручном режиме работы, и обрабатывает сигналы Пауза и Ресет для каждого из дозаторов.

Описание работы блока

В ручном режиме:

Когда WorkMode имеет значение "0", запускается ручной режим. Когда LoadPermission имеет значение "1", режим загрузки дозатора блокируется, если LoadPermission имеет значение "0", режим загрузки дозатора включается. Когда ManualUnload_In имеет значение "1" и UnloadBlock имеет значение "0", дозатор включает режим необходимости выгрузки. Когда Emul_ON имеет значение "1" блок начинает обрабатывать режим эмуляции. Дальше идет обработка аварийного стопа для всех дозаторов: если происходит аварийная остановка у какого-либо дозатора, его режим загрузки и выгрузки блокируется.

В автоматическом режиме:

Дозатор в ручном режиме переключается в автоматический режим путём присваивания его внутренней переменной work_mode значение "1". Когда WorkMode имеет значение "1", запускается автоматический режим. Когда LoadPermission имеет

значение "1", режим загрузки дозатора блокируется, если LoadPermissiong имеет значение "0", режим загрузки дозатора включается. По необходимости дозатор становится на паузу и сбрасывается путем присвоения дозаторам состояния DozReset. При LoadPermission равном "1", блок инициализирует старт работы блока в автоматическом режиме, устанавливая стандартные значения переменным в этом режиме. Если дозатор является весовым и его работа еще не закончена, блок меняет скорость его работы. Когда UnloadPermission имеет значение "1" и UnloadBlock имеет значение "0", режим выгрузки дозатора включается. Затем весовщик обрабатывает конец дозировки. Когда последний компонент загружен, блок выключает его. Дальше, если есть необходимость, начинается выгрузка каждого дозатора по его номеру замеса. Когда разрешение работы дозатора принимает ложное значение и последний компонент загружен, блок завершает работу дозатора. Когда Emul_ON имеет значение "1" блок начинает обрабатывать режим эмуляции. Затем идет обработка аварийного стопа для всех дозаторов: если происходит аварийная остановка у какого-либо дозатора, его режим загрузки и выгрузки блокируется.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	LoadPermission	Бит разрешения загрузки для дозатора Д1
reg int	WorkMode	Бит режима работы для дозатора Д1, 0 - ручной, 1 -автоматический
reg int	UnloadPermission	Бит разрешения выгрузки для дозатора Д1
reg int	UnloadBlock	Бит блокировки выгрузки для дозатора Д1
reg int	ManualUnload_In	Бит выгрузки в ручном режиме дозатора Д
short	MixerNumber	Номер смесителя с которым работает дозатор Д1
reg int	Emul_ON	Бит включения режима эмуляции

Пример использования:

```
  blocks->DosingBlock(
      LoadPermission,
      WorkMode,
      UnloadPermission,
```

```
        UnloadBlock,
        ManualUnload_In,
        MixerNumber,
        Emul_ON
    );
```

Определение переменных:

> ⚠ **ВНИМАНИЕ!!! Распределение в памяти аргументов блока должно быть последовательным как в примере.**

```
reg int  load_permission_d1    = &21;
reg int  load_permission_d2    = &22;
reg int  load_permission_d3    = &23;
reg int  load_permission_d4    = &24;
reg int  load_permission_d5    = &25;
reg int  load_permission_d6    = &26;
reg int  load_permission_d7    = &27;
reg int  load_permission_d8    = &28;
reg int  load_permission_d9    = &29;
reg int  load_permission_d10   = &30;

reg int  work_mode_d1          = &41;
reg int  work_mode_d2          = &42;
reg int  work_mode_d3          = &43;
reg int  work_mode_d4          = &44;
reg int  work_mode_d5          = &45;
reg int  work_mode_d6          = &46;
reg int  work_mode_d7          = &47;
reg int  work_mode_d8          = &48;
reg int  work_mode_d9          = &49;
reg int  work_mode_d10         = &50;

reg int  unload_permission_d1  = &31;
reg int  unload_permission_d2  = &32;
reg int  unload_permission_d3  = &33;
reg int  unload_permission_d4  = &34;
reg int  unload_permission_d5  = &35;
reg int  unload_permission_d6  = &36;
reg int  unload_permission_d7  = &37;
reg int  unload_permission_d8  = &38;
```

```
reg int  unload_permission_d9  = &39;
reg int  unload_permission_d10 = &40;

reg int  unload_block_d1       = &131;
reg int  unload_block_d2       = &132;
reg int  unload_block_d3       = &133;
reg int  unload_block_d4       = &134;
reg int  unload_block_d5       = &135;
reg int  unload_block_d6       = &136;
reg int  unload_block_d7       = &137;
reg int  unload_block_d8       = &138;
reg int  unload_block_d9       = &139;
reg int  unload_block_d10      = &140;

reg int  unload_manual_d1      = &71;
reg int  unload_manual_d2      = &72;
reg int  unload_manual_d3      = &73;
reg int  unload_manual_d4      = &74;
reg int  unload_manual_d5      = &75;
reg int  unload_manual_d6      = &76;
reg int  unload_manual_d7      = &77;
reg int  unload_manual_d8      = &78;
reg int  unload_manual_d9      = &79;
reg int  unload_manual_d10     = &80;

ushort   MixerIndex_d1         = &0;
ushort   MixerIndex_d2         = &1;
ushort   MixerIndex_d3         = &2;
ushort   MixerIndex_d4         = &3;
ushort   MixerIndex_d5         = &4;
ushort   MixerIndex_d6         = &5;
ushort   MixerIndex_d7         = &6;
ushort   MixerIndex_d8         = &7;
ushort   MixerIndex_d9         = &8;
ushort   MixerIndex_d10        = &9;
```

Вызов блока:

SetExtraTimeUnload

Назначение:

Установка дополнительной паузы выгрузки для дозатора.

Описание аргументов блока:

Тип	Переменная	Описание
char	Doz_Number	Номер дозатора
char	Extra_Time	Установка дополнительного времени выгрузки для дозатора

Пример использования:

```
blocks->SetExtraTimeUnload(
    Doz_Number,
    Extra_Time
);
```

LatchWithTimeOut

Назначение:

Защелка с таймаутом.

Описание работы блока:

Когда Signal_IN переходит в состояние "1", Signal_Out переходит в состояние "1" через время Time_Out, после этого момента выходной сигнал блоком не изменяется. Timer_Index привязывает защелку к внутреннему таймеру контроллера.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Signal_IN	Вход
reg int	Signal_Out	Выход
char	Timer_Index	Номер таймера
ushort	Time_Out	Таймаут если таймер не задан

Пример использования:

```
blocks->LatchWithTimeOut(
    Signal_IN,
```

```
        Signal_Out,
        Timer_Index,
        Time_Out
    );
```

ToggleWithTimeOut

Назначение:

Переключатель с таймаутом.

Описание работы блока:

Блок работает, когда Signal_IN имеет значение "1". При Signal_State равном "-1", Signal_Out меняет значение на противоположное. Иначе Signal_Out присваивается значение Signal_State.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Signal_IN	Вход
reg int	Signal_Out	Выход
char	Signal_State	Состояние в которое надо переключить если -1 инверсия состояния
char	Timer_Index	Номер таймера
char	Time_Out	Таймаут если таймер не задан

Пример использования:

```
    blocks->ToggleWithTimeOut(
        Signal_IN,
        Signal_Out,
        Signal_State,
        Timer_Index,
        Time_Out
    );
```

SetUPAEmpty

Назначение:

Разрешение выгрузки для дозатора после сигнала "дозатор пустой".

Описание работы блока:

Когда Mixer_State равен MixerPermission_State, данный блок формирует разрешение (UnloadPermission_Out) выгрузки первого дозатора с номером (Doz1Number) во второй с номером (Doz2Number) после того, как второй загрузиться.

Описание аргументов блока:

Тип	Переменная	Описание
char	Doz1Number	Номер дозатора для формирования разрешения выгрузки
char	Doz2Number	Номер дозатора по которому формирует разрешения выгрузки
short	Mixer_State	Состояние смесителя
char	MixerPermission_State	Состояние смесителя для формирования разрешения
reg int	UnloadPermission_Out	Выход разрешения загрузки смесителя

Пример использования:

```
blocks->SetUPAEmpty(
     Doz1Number,
     Doz2Number,
     Mixer_State,
     MixerPermission_State,
     UnloadPermission_Out
);
```

SaveAndRestoreBits

Назначение:

Сохранение и восстановение бит дозаторов при смене режима.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	WorkMode	Режим работы: 0 - ручной, 1 - автомат
reg int	UnloadPermission	Номер первого бита разрешения выгрузки

Пример использования:

```
blocks->SaveAndRestoreBits(
    WorkMode,
    UnloadPermission
);
```

StoneGateDriver

Назначение:

Управляет "челюстями" хранилища сыпучих материалов.

Описание работы блока

В режиме одной челюсти:

Режим работы одной челюсти включается, когда Work_Mode имеет значение "1". При SP_1 равном "1", переменным OUT_SP1 и OUT_SP2 присваивается значение "1". Когда SP_1 имеет значение "0", то переменным OUT_SP1 и OUT_SP2 присваивается значение "0".

В режиме двух челюстей:

Режим работы двух челюстей включается, когда Work_Mode имеет значение "0". При SP_1 равному "1", OUT_SP1, OUT_SP2 меняются на "1". При SP_2 равному "1", OUT_SP1 меняется на "0", OUT_SP2 меняется на "1". При отсутствии сигналов на SP_1 или SP_2 OUT_SP1, OUT_SP2 меняются на "0".

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Work_Mode	Режим работы 0 - один распределитель 2-е челюсти; 1 - одна челюсть
reg int	SP_1	Вход первая скорость
reg int	SP_2	Вход вторая скорость
reg int	OUT_SP1	Выход первая скорость
reg int	OUT_SP2	Выход вторая скорость
reg int	Middle_Sensor	Датчик среднего положения

Пример использования:

```
blocks->StoneGateDriver(
    Work_Mode,
    SP_1,
    SP_2,
    OUT_SP1,
    OUT_SP2,
    Middle_Sensor
);
```

DozingValveEuromix

Назначение:

Управляет работой дозирующего клапана.

Описание работы блока:

Работа данного блока осуществляется в зависимости от состояния дозирующего клапана.

Описание состояний дозирующего клапана:

Состояние	Значение
CLOSE	Open_Out и Stop_Out изменяют своё значение на "0".
OPENING_SP2	Дозирующий клапан открывается с 2-ой скоростью, устанавливая Open_Out, Stop_Out значение "1". Когда время открытия клапана со 2-ой скоростью превышает время SP2_TIME_OPEN, клапану устанавливают состояние OPEN_SP2, а также таймер блока обновляется.
SWITCHING_SP2	Stop_Out, Open_Out устанавливаются в значение "1". Когда время полного закрытия затвора превышает SP1_TIME_CLOSE, таймер блока обновляется, а состояние дозирующего клапана становится OPENING_SP2. Когда SENSOR_SP2_IN имеет значение "1", таймер блока обновляется и состояние клапана меняется на OPEN_SP2
OPEN_SP1	Open_Out, Stop_Out переходят в значение "1"
OPEN_SP2	Open_Out, Stop_Out переходят в значение "0", "1" соответственно.
CLOSING	Дозирующий клапан закрывается,

устанавливая Open_Out, Stop_Out значение "0". Когда время полного закрытия затвора превышает SP1_TIME_CLOSE, таймер блока обновляется, а состояние дозирующего клапана становится CLOSE.

Данный блок возвращает состояние дозирующего клапана.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Load_Sp1	Загрузка скорость 1
reg int	Load_Sp2	Загрузка скорость 2
reg int	SENSOR_SP2_IN	Вход датчик второй скорости
short	SP1_TIME_CLOSE	Время полного закрытия затвора x100 мс
short	SP2_TIME_OPEN	Время полного открытия затвора на 2-й x100 мс
reg int	Open_Out	Выход на управление открытием
reg int	Stop_Out	Выход на управление остановкой в задней точке

Пример использования:

```
blocks->DozingValveEuromix(
     Load_Sp1,
     Load_Sp2,
     SENSOR_SP2_IN,
     SP1_TIME_CLOSE,
     SP2_TIME_OPEN,
     Open_Out,
     Stop_Out
);
```

DozingValveAmmann

Назначение:

Управляет работой дозирующего клапана.

Описание работы блока:

Работа данного блока осуществляется в зависимости от состояния дозирующего клапана.

Описание состояний дозирующего клапана:

Состояние	Значение
CLOSE	Open_Out и Stop_Out изменяют своё значение на "0".
OPENING_SP2	Дозирующий клапан открывается с 2-ой скоростью, устанавливая Open_Out значение "1". Когда время открытия клапана со 2-ой скоростью превышает время SP2_TIME_OPEN, клапану устанавливают состояние OPEN_SP2, а также таймер блока обновляется.
SWITCHING_SP2	Open_Out устанавливается в значение "1". Когда время полного закрытия затвора превышает SP1_TIME_CLOSE, таймер блока обновляется, а состояние дозирующего клапана становится OPENING_SP2.
OPEN_SP1	Open_Out переходит в значение "1"
OPEN_SP2	Open_Out, Stop_Out переходят в значение "0", "1" соответственно.
CLOSING	Дозирующий клапан закрывается, устанавливая Open_Out, Stop_Out значение "0". Когда время полного закрытия затвора превышает SP1_TIME_CLOSE,

таймер блока обновляется, а состояние дозирующего клапана становится CLOSE.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Load_Sp1	Загрузка скорость 1
reg int	Load_Sp2	Загрузка скорость 2
reg int	SENSOR_SP2_IN	Вход датчик второй скорости
short	SP1_TIME_CLOSE	Время полного закрытия затвора x100 мс
short	SP2_TIME_OPEN	Время полного открытия затвора на 2-й x100 мс
reg int	Open_Out	Выход на управление открытием
reg int	Stop_Out	Выход на управление остановкой в задней точке

Пример использования:

```
blocks->DozingValveAmmann(
     Load_Sp1,
     Load_Sp2,
     SENSOR_SP2_IN,
     SP1_TIME_CLOSE,
     SP2_TIME_OPEN,
     Open_Out,
     Stop_Out
);
```

Блоки управления смесителем

Список блоков:

1. MixingBlock;
2. MixingBlock V2;
3. GateDriver;
4. GateSensEmul.

MixingBlock

Назначение:

Описание аргументов блока:

Тип	Переменная	Описание
reg int	MixerMode	Вход режим работы 0/1 Прямое включение/Звезда треугольник
short	MixerIndex	Индекс смесителя
reg int	AutoMode	Выбор режима работы смесителя
reg int	MixerOpenIN	
reg int	MixerOpenOut	
reg int	MixerCloseIN	
reg int	MixerCloseOut	
reg int	Star_Mixer_Out	Выход смеситель вкл звезда
reg int	Star_Mixer_IN	Вход смеситель вкл звезда
reg int	Triangle_Mixer_Out	Выход смеситель вкл треугольник
reg int	Triangle_Mixer_IN	Вход смеситель вкл треугольник
short	MixerTime	Время переключения в секундах
short	SensorCount	Количество датчиков положения
short	SensorPosition	Номер датчика на котором находится шибер
reg int	SensorPointer	Указатель на биты датчиков положения
reg int	Enable_Filter	
short	DryFilter	Маска признака сухого перемешивания
short	WetFilter	Маска признака влажного перемешивания
short	MixerFail	Ошибка включения смесителя
short	TimersPointer	Массив с заданием времени нахождения в

		точках остановок
short	MixerState	Состояние смесителя
short	Emul_ON	

Пример использования:

```
blocks->MixingBlock(
      MixerMode,
      MixerIndex,
      AutoMode,
      MixerOpenIN,
      MixerOpenOut,
      MixerCloseIN,
      MixerCloseOut,
      Star_Mixer_Out,
      Star_Mixer_IN,
      Triangle_Mixer_Out,
      Triangle_Mixer_IN,
      MixerTime,
      SensorCount,
      SensorPosition,
      SensorPointer,
      Enable_Filter,
      DryFilter,
      WetFilter,
      MixerFail,
      TimersPointer,
      MixerState,
      Emul_ON
);
```

MixingBlock_V2

Назначение:

Осуществляет работу смесителя

Описание работы блока

В режиме прямого включения:

Когда MixerMode имеет значение "0", Star_Mixer_OUT, Triangle_Mixer_OUT присваивается значение "1", мотору смесителя присваивается значение WAIT_TRIANGLE.

В режиме "Звезда-треугольник":

Когда MixerMode имеет значение "1", мотору смесителя присваивается значение STAR. В этом режиме время на переключение между схемами включения составляет 20% от времени работы MixerTime.

В ручном режиме:

Когда AutoMode имеет значение "0", блок отключает таймеры перемешивания. Когда MixerCloseIN имеет значение "1", а Sensor_Position равен "0" или SensorPointer равен "1", Sensor_Position и MixerCloseIN присваивается значение "0".

В автоматическом режиме:

Когда AutoMode имеет значение "1" и мотор смесителя имеет состояние "MIXERON", блок обрабатывает состояния смесителя:

При поступлении сигнала запуска смесителя в автоматическом режиме, смеситель переходит в режим MIXERON. Если есть разрешение работы смесителя и он находится в состоянии необходимости выгрузки, смеситель переходит в режим LOADING.

В режиме LOADING при соответствии номеров выгруженных дозаторов маске DryFilter, включается таймер сухого перемешивания, а смеситель переходит в режим DRYMIX.

В режиме DRYMIX по истечению таймера сухого перемешивания, смеситель переходит в режим DRYMIX_END.

В режиме DRYMIX_END при соответствии номеров выгруженных дозаторов маске WetFilter, включается таймер влажного перемешивания, а смеситель переходит в режим WETMIX.

В режиме WETMIX по истечению таймера влажного перемешивания начинается выгрузка смесителя, а смеситель переходит в режим ST_UNLOADING.

Режим ST_UNLOADING открывает смеситель. Смеситель переходит в режим UNLOADING.

Режим UNLOADING открывает смеситель

Режим OPEN запускает таймер выгрузки. По истечению этого таймера смеситель закрывается и переходит в режим CLOSE.

В режиме CLOSE при SensorPointer равном "1", блок считает, что смеситель закрылся и переходит в режим MIXERON.

Описание состояний смесителя (при обработке в автоматическом режиме)

Состояние смесителя	Значение	Следующее состояние
MIXERON	Смеситель в режиме перемешивания	LOADING
LOADING	Состояние после режима перемешивания	DRYMIX
DRYMIX	Смеситель в режиме сухого перемешивания	DRYMIX_END
DRYMIX_END	Состояние после режима сухого перемешивания	WETMIX
WETMIX	Смеситель в режиме влажного перемешивания	ST_UNLOADING
ST_UNLOADING	Состояние после режима влажного перемешивания	UNLOADING
UNLOADING	Состояние после обработки режима	OPEN

	ST_UNLOADING	
OPEN	Смеситель открывается	CLOSE
CLOSE	Смеситель закрывается	MIXERON

Описание аргументов блока:

Тип	Переменная	Описание
reg int	MixerMode	Вход режим работы 0/1 Прямое включение/Звезда треугольник
short	MixerIndex	Индекс смесителя
reg int	AutoMode	Выбор режима работы смесителя
reg int	MixerOpenIN	Вход бит открытия смесителя в ручном режиме
reg int	MixerOpenOUT	Выход бит открытия смесителя
reg int	MixerCloseIN	Вход бит закрытия смесителя в ручном режиме
reg int	MixerCloseOut	Выход бит закрытия смесителя
reg int	Star_Mixer_OUT	Выход смеситель вкл звезда
reg int	Star_Mixer_IN	Вход смеситель вкл звезда
reg int	Triangle_Mixer_OUT	Выход смеситель вкл треугольник
reg int	Triangle_Mixer_IN	Вход смеситель вкл треугольник
short	MixerTime	Время переключения в секундах
short	SensorCount	Количество датчиков положения
short	SensorPosition	Номер датчика на котором находится шибер
reg int	SensorPointer	Указатель на биты датчиков положения
reg int	Enable_Filter	Первый бит датчика положения
short	DryFilter	Маска признака сухого перемешивания
short	WetFilter	Маска признака влажного перемешивания
short	MixerFail	Ошибка включения смесителя

short	TimersPointer	Массив с заданием времени нахождения в точках остановок
short	MixerState	Состояние смесителя
short	Emul_ON	Режим эмуляции

Пример использования:

```
blocks->MixingBlock_V2(
      MixerMode,
      MixerIndex,
      AutoMode,
      MixerOpenIN,
      MixerOpenOUT,
      MixerCloseIN,
      MixerCloseOut,
      Star_Mixer_OUT,
      Star_Mixer_IN,
      Triangle_Mixer_OUT,
      Triangle_Mixer_IN,
      MixerTime,
      SensorCount,
      SensorPosition,
      SensorPointer,
      Enable_Filter,
      DryFilter,
      WetFilter,
      MixerFail,
      TimersPointer,
      MixerState,
      Emul_ON
);
```

GateDriver

Назначение:

Блок управляет открытием смесителя

Описание работы блока

В режиме управления открытием смесителя с помощью кнопок без фиксации:

Данный режим активируется, когда GateMode имеет значение "1". При MixerOpenIN равному 1, MixerOpen_Out присваивается значение "1". При MixerCloseIN равному "1", MixerOpen_Out присваивается значение "0". MixerClose_Out переходит в значение "0".

В режиме управления открытием смесителя с помощью кнопок с фиксацией:

Данный режим активируется, когда GateMode имеет значение "2". MixerOpen_Out присваивается значение MixerOpenIN, а MixerClose_Out присваивается значение MixerCloseIN.

Описание аргументов блока:

Тип	Переменная	Описание
char	GateMode	Режим работы: 0 - не выбран, 1 - управления открытием смесителя с помощью кнопок без фиксации, 2 - управления открытием смесителя с помощью кнопок с фиксацией
reg int	MixerOpenIN	Вход Бит открытия смесителя
reg int	MixerCloseIN	Вход Бит закрытия смесителя
reg int	MixerOpen_Out	Выход Бит открытия смесителя
reg int	MixerClose_Out	Выход Бит закрытия смесителя

Пример использования:

```
blocks->GateDriver(
    GateMode,
```

```
        MixerOpenIN,
        MixerCloseIN,
        MixerOpen_Out,
        MixerClose_Out
    );
```

GateSensEmul

Назначение:

Эмулятор датчика положения шибера.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	MixerOpenIN	Вход Бит открытия смесителя
reg int	MixerCloseIN	Вход Бит закрытия смесителя
reg int	SensorOpen	Вход смеситель открыт
reg int	SensorCloce	Выход датчик закрытого положения смесителя
short	TimerClose	Время закрытия смесителя и выставление сигнала смеситель закрыт

Пример использования:

```
    blocks->GateSensEmul(
        MixerOpenIN,
        MixerCloseIN,
        SensorOpen,
        SensorCloce,
        TimerClose
    );
```

Блоки управления скипом

Список блоков:

1. Skip.

Skip

Назначение:

Осуществляет работу скипа.

Описание работы блока

В автоматическом режиме:

Когда Auto_Mode имеет значение "1", запускается автоматический режим. После того, как скип заполнился материалом (SkipFull_IN = 1), SkipFull_OUT присваивается значение "1" и начинает движение вверх (Skip_UP = 1). При достижении скипа верхнего положения, начинается выгрузка скипа. Если материала в скипе нет (SkipFull_IN = 0), то блок изменяет состояние на "Авария". После выгрузки скип начинает движение вниз, устанавливая SkipFull_OUT, SkipGoUpAllow значение "0", а Skip_DOWN значение "1". Когда срабатывает нижний датчик, движение скипа вниз останавливается, Skip_DOWN переходит в значение "0".

В ручном режиме:

Когда Auto_Mode имеет значение "0", запускается ручной режим. Skip_UP устанавливается значение In_Skip_UP, Skip_DOWN значение In_Skip_Down. Когда In_Stop = 1, Skip_UP, Skip_DOWN, In_Skip_Up, In_Skip_Down устанавливаются в значение "0", останавливая скип.

Эмуляция работы скипа:

Включается, когда Emul_ON имеет значение "1". Когда позиция скипа (Sensor_Position) равна 255, она переходит в значение "1", включается нижний датчик скипа.

Когда скип двигается вверх, (Skip_UP = 1), блок начинает проверять позицию скипа (Sensor_Position). Когда скип занимает нижнюю позицию, он начинает движение вверх (Sensor_Position = Middle), а нижний датчик выключается. Когда скип находится в движении и разница времени внутреннего таймера контроллера и таймера скипа больше 16с, таймер скипа обновляется, а датчик ожидания включается. Если скип находится в режиме ожидания и разница времени внутреннего таймера контроллера

и таймера скипа больше 4с, таймер скипа обновляется, датчик ожидания выключается, но верхний датчик включается.

Когда скип двигается вниз, (Skip_DOWN = 1), блок начинает проверять позицию скипа (Sensor_Position). Если скип находится в верхнем положении и разница времени внутреннего таймера контроллера и таймера скипа больше 4с, скип переходит в режим ожидания, его таймер обновляется, а датчик ожидания включается. Верхний датчик выключается. Когда скип находится в режиме ожидания и разница времени внутреннего таймера контроллера и таймера скипа больше 0.6с, таймер скипа обновляется, датчик ожидания включается, а скип переходит в среднее положение. Когда скип занимает среднее положение и разница времени внутреннего таймера контроллера и таймера скипа больше 16с, таймер скипа обновляется, а нижний датчик включается.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	AutoMode	Выбор режима работы скипа 0 - ручной 1 автомат
reg int	SkipFull_IN	Бит установки скип полный вход
reg int	SkipFull_OUT	Бит Сигнализации скип полный
reg int	SkipGoUpAllow	Бит Сигнализации скип полный
reg int	Sensor_Pointer	Указатель на датчики положения скипа 0 - аварийный датчик внизу 1 - нижний датчик 2 - нижний датчик второй скорости 3 - датчик ожидания 4 - верхний датчик второй скорости 5 - верхний датчик 6 - аварийный датчик вверху
short	Sensor_Position	Позиция скипа
reg int	Skip_UP	Скип движение вверх
reg int	Skip_DOWN	Скип движение вниз
reg int	Skip_Slow	Бит включения второй скорости
short	Timer_N	Номер таймера выгрузки скипа
reg int	In_Skip_Up	вход движение вверх в ручном режиме
reg int	In_Skip_Down	вход движение вниз в ручном режиме
reg int	In_Stop	Аварийная остановка

short	SkipState	Состояние скипа
short	MixerState	Состояние смесителя
short	Emul_ON	Режим эмуляции

Пример использования:

```
blocks->Skip(
     AutoMode,
     SkipFull_IN,
     SkipFull_OUT,
     SkipGoUpAllow,
     Sensor_Pointer,
     Sensor_Position,
     Skip_UP,
     Skip_DOWN,
     Skip_Slow,
     Timer_N,
     In_Skip_Up,
     In_Skip_Down,
     In_Stop,
     SkipState,
     MixerState,
     Emul_ON
);
```

Прочие блоки

Список блоков:

1. GetTimerValue;
2. Timer;
3. Timer Off;
4. Vibrator;
5. Motor;
6. MotorLite;
7. MotorLiteDirect;
8. Valve;
9. StartMaker;
10. StartMakerAdd
11. Преобразователь сигнала ET14M (ET14M Scale);
12. Преобразователь сигнала ADC Scale Izm (Scale Izm);
13. Формирователь переменного меандра (Variable Meandr);
14. Драйвер ПЧ (frc driver);
15. Burner

GetTimerValue

Назначение:

Возвращает значение таймера.

Описание аргументов блока:

Тип	Переменная	Описание
short	Timer_Index	Индекс внутреннего таймера контроллера

Пример использования:

```
blocks->GetTimerValue(
    Timer_Index
);
```

Timer

Назначение:

Осуществляет работу таймера.

Описание работы блока:

Когда In имеет значение "1", по истечению времени (Set * Step) блок изменит значение Out на 1. Когда In имеет значение "0", Out присваивается значение "0".

Описание аргументов блока:

Тип	Переменная	Описание
reg int	In	Входное значение активации таймера
reg int	Step	Шаг таймера (мс)
reg int	Set	Время срабатывания таймера
reg int	Out	Выходное значение сигнала таймера

Пример использования:

```
blocks->Timer(
    In,
```

```
      Step,
      Set,
      Out
);
```

Timer_Off

Назначение:

Осуществляет работу таймера

Описание работы блока:

Когда In имеет значение "1", по истечению времени (Set * Step) блок поменяет значение Out на 0.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	In	Входное значение активации таймера
reg int	Step	Шаг таймера (мс)
reg int	Set	Время срабатывания таймера
reg int	Out	Выходное значение сигнала таймера

Пример использования:

```
blocks->Timer_Off(
      In,
      Step,
      Set,
      Out
);
```

Vibrator

Назначение:

Блок управляет работой вибраторов.

Описание работы блока:

Vib_Out используется блоком для включения (Vib_Out=1) и выключения (Vib_Out=0) мотора. Vib_Os предназначен для получения обратной связи от мотора: если её нет

(Vib_Os = 0), ставится таймер на время Vib_Os_TimeOut. Если по окончанию времени Vib_Os_TimeOut обратная связь не появилась (Vib_Os не стала "1"), мотор выключается (Vib_Out меняет значение на "0"), а блок переходит в аварийное состояние (Vib_Alarm меняет значение на "1"). Если блок запущен в постоянном режиме (Mode = "1"), мотор вибратора (Vib_Out) включается на время Vib_Time_On, затем выключается на время Vib_Time_Off.

В автоматическом режиме:

Если Vib_Auto имеет значение "1", то вибратор переходит в автоматический режим. Если блок находится в состоянии аварии (Vib_Alarm = 1), мотор вибратора выключается (Vib_Out меняет значение на "0"). Работа блока продолжается, когда блок запущен в ручном или автоматическом режиме, иначе мотор вибратора отключается (Vib_Out переходит в "0") и блок заканчивает работу. Когда блок запущен в импульсном режиме (Mode = "0"), мотор вибратора включается (Vib_Out меняет значение на "1").

В ручном режиме:

Если Vib_Auto имеет значение "0", то вибратор переходит в ручной режим. Если блок находится в состоянии аварии (Vib_Alarm = 1), мотор вибратора выключается (Vib_Out меняет значение на "0"). Работа блока продолжается, когда блок запущен в ручном или автоматическом режиме, иначе мотор вибратора отключается (Vib_Out переходит в "0") и блок заканчивает работу. Когда блок запущен в импульсном режиме (Mode = "0"), мотор вибратора включается (Vib_Out меняет значение на "1").

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Mode	Входное значение режима работы: 0 - импульсный, 1 - постоянный
reg int	Vib_On	Входное значение для работы блока в автоматическом режиме
reg int	Vib_On_Manual	Входное значение для работы блока в ручном режиме
reg int	Vib_Auto	Входное значение для работы блока в автоматическом режиме
reg int	Vib_Out	Выходное значение мотор включен
reg int	Vib_Os	Входное значение обратной связи

short	Vib_Os_TimeOut	Время на получение обратной связи, умноженное на 100мс
reg int	Vib_Alarm	Выходное значение состояния аварии
reg int	Vib_Reset	Входное значение сброса аварийного состояния
short	Vib_Time_On	Время включенного состояния
short	Vib_Time_Off	Время выключенного состояния

Пример использования:

```
blocks->Vibrator(
      Mode,
      Vib_On,
      Vib_On_Manual,
      Vib_Auto,
      Vib_Out,
      Vib_Os,
      Vib_Os_TimeOut,
      Vib_Alarm,
      Vib_Reset,
      Vib_Time_On,
      Vib_Time_Off
);
```

Motor

Назначение:

Управляет работой моторов.

Описание работы блока

Работа мотора осуществляется только при Work_Permission равном "1". При включении блока сигналами StartBT или Auto_Start в течении времени Os_Timer блок ждет значения "1" у Input_OS. Если в течении заданного времени Input_OS не переходит в состояние "1", блок переходит в состояние аварии, изменяя свой статус на ALARM и OutPut_Alarm переходит в значение "1". Для того чтобы блок вышел из состояния аварии, нужно присвоить Reset_IN значение "1". Compare_Mode обеспечивает совместимость с некоторыми системами благодаря выставлению OutPut_Alarm в значение "1" на период запуска мотора.

Когда Ignore_Os = "1", отключается проверка наличия обратной связи сигнала Input_OS.

Данный блок возвращает свое состояние.

В ручном режиме:

Когда StartBT и Work_Permission имеют значение "1" и значение Work_Mode - "0", то запускается ручной режим, тогда Start_BT будет отвечать за пуск/стоп двигателя. При Start_BT равному "1", Out_Put присваивается значение "1".

В автоматическом режиме:

Когда Work_Mode имеет значение "1" и мотор остановлен, запускается автоматический режим, в котором запуск и остановка мотора осуществляется с задержкой Timer_Start и Timer_Stop соответственно. В автоматическом режиме Auto_Start будет отвечать за запуск мотора. При Auto_Start равному "1", Out_Put присваивается значение "1".

Описание возвращаемых состояний блока:

Возвращаемое значение блока	Числовое возвращаемое значение блока	Описание возвращаемого значения	Причина возвращаемого значения блока
STOP	0	Мотор остановлен	StartBT имеет значение "0"
DELAY_START	1	Подготовка запуска мотора	Work_Mode, Auto_Start имеют значение "1", ожидание в течении Timer_Start, как только время достигнет Timer_Start, блок перейдет в состояние WAIT_START
WAIT_START	2	Ожидание запуска мотора	StartBT имеет значение "1", Input_Os имеет значение "0", время ожидания еще не превысило Os_Timer

WORK	3	Мотор работает	StartBT, Input_Os имеют значение "1"
DELAY_STOP	4	Подготовка остановки мотора	Work_Mode имеет значение "1", Auto_Start изменяет значение с "1" на "0" , через время Timer_Stop блок перейдет в состояние STOP
ALARM	5	Авария мотора	StartBT имеет значение "1", Input_Os имеет значение "0", время ожидания превысило Os_Timer

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Work_Permission	Входное значение разрешения выполнения блока
reg int	Work_Mode	Входное значение режима работы
reg int	StartBT	Пуск стоп в ручном режиме
reg int	Auto_Start	Запуск в автоматическом режиме
short	Timer_Start	Входное значение задержки запуска мотора в автоматическом режиме
short	Timer_Stop	Входное значение задержки остановки мотора в автоматическом режиме
reg int	Reset_IN	Входное значение для выхода из состояния аварии
reg int	Out_Put	Выходное значение включения мотора
short	Os_Timer	Таймер до появления обратной связи
reg int	Input_OS	Обратная связь контактора
reg int	OutPut_Alarm	Выходное значение аварийного состояния
reg	Ignore_Os	Выходное значение игнорирования обратной

int		связи
reg int	Compare_Mode	Режим совместимости, на время включения выставляется бит аварии

Пример использования:

```
MotorState = blocks->Motor(
       Work_Permission,
       Work_Mode,
       StartBT,
       Auto_Start,
       Timer_Start,
       Timer_Stop,
       Reset_IN,
       Out_Put,
       Os_Timer,
       Input_OS,
       OutPut_Alarm,
       Ignore_Os,
       Compare_Mode
  );
```

MotorLite

Назначение:

Управляет работой мотора.

Описание работы блока:

При вызове данного блока возвращается состояние мотора. При Ignore_Os равном "1", отключается проверка наличия обратной связи сигнала Input_OS. Работа блока продолжается только при Work_Permission равном "1".
При Work_Permission равном "0", блок выключит мотор (Out_Put поменяется на "0"), а также изменит состояние мотора на "STOP".

Когда мотор находится в режиме "STOP", при поступлении "1" на Start, мотор переходит в режим "WAIT_START", устанавливает таймер до включения мотора и Out_Put присваивается значение "1", включая мотор.

При Compare_Mode равном "1", OutPut_Alarm переходит в "1" на время запуска мотора.

Когда мотор находится в режиме "WAIT_START", по истечении времени Os_Timer, при Input_OS равном "1", мотор переключается в режим "WORK", но если Input_OS равен "0", состояние мотора переключится на "ALARM" и он выключится (Out_Put изменится на "0").

Когда мотор находится в режиме "WORK", при Input_OS равном "0", по истечению 1.5 секунд мотор входит в режим "ALARM" и выключается (Out_Put переходит в "0").

Когда мотор находится в режиме "ALARM", мотор выключается (Out_Put переходит в "0"), а OutPut_Alarm присваивается "1".
Аварийное состояние сбрасывается при Reset_IN равном "1", а мотор переходит в режим "STOP".

Описание состояний мотора в блоке:

Режим мотора в блоке	Числовое значение	Описание режима
STOP	0	Мотор остановлен
WAIT_START	2	Ожидание запуска мотора
WORK	3	Мотор работает
ALARM	5	Авария мотора

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Start	Входное значение запуска блока
reg int	Out_Put	Выходное значение включения мотора
short	Os_Timer	Таймер до появления обратной связи
reg int	Input_OS	Обратная связь контактора
reg int	OutPut_Alarm	Выходное значение аварийного состояния
reg int	Reset_IN	Входное значение для выхода из состояния аварии
reg	Ignore_Os	Выходное значение игнорирования обратной

int		связи
reg int	Compare_Mode	Режим совместимости, на время включения выставляется бит аварии
reg int	Work_Permission	Входное значение разрешения выполнения блока

Пример использования:

```
MotorState = blocks->MotorLite(
     Work_Permission,
     Start,
     Out_Put,
     Os_Timer,
     Input_OS,
     OutPut_Alarm,
     Reset_IN,
     Ignore_Os,
     Compare_Mode
);
```

MotorLiteDirect

Назначение:

Управляет работой мотора.

Описание работы блока:

Данный блок возвращает состояние мотора. При Ignore_Os равном "1", отключается проверка наличия обратной связи сигнала Input_OS.

Когда мотор находится в режиме "STOP", он переключает мотор в режим "WAIT_START", устанавливает таймер до включения мотора и присваивает Out_Put значение "1", включая мотор.

При Compare_Mode равном "1", OutPut_Alarm переходит в "1" на время запуска мотора.

Когда мотор находится в режиме "WAIT_START", по истечении времени Os_Timer, при Input_OS равном "1", мотор переключается в режим "WORK", но если Input_OS равен "0", состояние мотора переключится на "ALARM" и он выключится (Out_Put изменится на "0").

Когда мотор находится в режиме "WORK", при Input_OS равном "0", по истечению 1.5 секунд мотор входит в режим "ALARM" и выключается (Out_Put переходит в "0")

Когда мотор находится в режиме "ALARM", мотор выключается (Out_Put переходит в "0"), а OutPut_Alarm присваивается "1". При Reset_IN равном "1", аварийное состояние сбрасывается, а мотор переходит в режим "STOP".

Описание возвращаемых значений блока:

Возвращаемое значение блока	Числовое возвращаемое значение блока	Описание возвращаемого значения
STOP	0	Мотор остановлен
WAIT_START	2	Ожидание запуска мотора
WORK	3	Мотор работает

```
    ALARM           5                   Авария мотора
```

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Start	Входное значение запуска блока
reg int	Out_Put	Выходное значение включения мотора
short	Os_Timer	Таймер до появления обратной связи
reg int	Input_OS	Обратная связь контактора
reg int	OutPut_Alarm	Выходное значение аварийного состояния
reg int	Reset_IN	Входное значение для выхода из состояния аварии
reg int	Ignore_Os	Выходное значение игнорирования обратной связи
reg int	Compare_Mode	Режим совместимости, на время включения выставляется бит аварии

Пример использования:

```
MotorState = blocks->MotorLiteDirect(
       Start,
       Out_Put,
       Os_Timer,
       Input_OS,
       OutPut_Alarm,
       Reset_IN,
       Ignore_Os,
       Compare_Mode
  );
```

Valve

Назначение:
Управляет работой клапанов.

Описание работы блока:
Когда Open равен "1", блок подает сигнал на открытие клапана , устанавливая Open_Out в состояние "1". Далее блок ожидает, пока клапан откроется и Open_IN перейдет в значение "1". Если по истечению времени Move_Time на Open_IN не поступает сигнал, статус блока переходит в состояние аварии и присваивает Alarm_Out значение "1", а также присваивает клапану статус "ERROR_OP" , не изменяя все остальные сигналы. Если при закрытии клапана не придет сигнал о том, что он закрылся вовремя, статус блока переходит в состояние аварии , клапану будет присвоен статус "ERROR_CL". Если на блок одновременно будут поданы сигналы Open_IN и Close_IN со значением "1", статус блока перейдет в состояние аварии и клапану будет присвоен статус "ERROR_SENSORS".

Данный блок возвращает состояние клапана.

Описание возвращаемых значений блока:

Возвращаемое значение блока	Числовое возвращаемое значение блока	Описание возвращаемого значения	Причина возвращаемого значения блока
UNDEFINED_POSITION	0	Неопределенная позиция клапана	Open_IN и Close_IN имеют значение "0" в момент включения завода
CLOSE	1	Закрытое состояние клапана	Open_IN и Open имеют значение "0", Close_IN имеет значение "1"
OPEN	2	Открытое состояние клапана	Open_IN и Open имеют значение "1", Close_IN имеет значение "0"
OPENING	3	Состояние открывания	Open_IN и Close_IN имеют значение "0",

		клапана	Open изменяет значение с "0" на "1",ещё не прошло время Move_Time
CLOSING	4	Состояние закрывания клапана	Open_IN и Close_IN имеют значение "0", Open изменяет значение с "1" на "0",ещё не прошло время Move_Time
ERROR_OP	5	Аварийное состояние открывания клапана	
ERROR_CL	6	Аварийное состояние закрывания клапана	
ERROR_SENSORS	7	Аварийное состояние работы датчиков	

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Open	Входной сигнал открытия клапана
reg int	Open_IN	Выходной сигнал открытого состояния клапана
reg int	Close_IN	Выходной сигнал закрытого состояния клапана
reg int	Open_Out	Выходной сигнал начала открытия клапана
reg int	Alarm_Out	Выходной сигнал аварии клапана
short	Move_Time	Время открытия клапана

Пример использования:

```
ValveState = blocks->Valve(
     Open,
     Open_IN,
     Close_IN,
     Open_Out,
     Alarm_Out,
     Move_Time
 );
```

StartMaker

Назначение:

Управляет запуском ячеек.

Описание работы блока:

Данный блок реализует перемещение модуля по массиву ячеек в пределах Start_Position и End_Position. Движение по массиву ячеек модулем осуществляется вверх (если Start_Up равен "1") или вниз (есои Start_Down равен "1"). При перемещении модуля по массиву ячеек, он инициализирует их запуск.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Start_Position	Входное значение начала массива
reg int	End_Position	Входное значение конца массива
reg int	Start_Up	Старт вверх
reg int	Start_Down	Старт вниз

Пример использования:

```
blocks->StartMaker(
    Start_Position,
    End_Position,
    Start_Up,
    Start_Down
);
```

StartMakerAdd

Назначение:

Описание аргументов блока:

Тип	Переменная	Описание
reg int	AutoStart_Pointer	Указатель на бит запуска ячейки
reg int	CellStarted_Pointer	Указатель на бит успешного запуска ячейки
reg int	Cell_Number	Номер ячейки

Пример использования:

```
blocks->StartMakerAdd(
    AutoStartPointer,
    CellStarted_Pointer,
    Cell_Number
);
```

ET14M_Scale

Назначение:

Измерения сигнала датчиков

Описание работы блока:

На входной сигнал Input подается измеренное модулем значение и преобразуется температуру в цельсиях.

Описание аргументов блока:

Тип	Переменная	Описание
short	Input	Входное значения
float	Uref	Входное значение опорного напряжения
float	R_Ref	Входное значение опорного сопротивление сигнала
float	R0	Входное значение номинального сопротивления датчика
short	Accuracy	Входное значение точности возвращаемого значения

Пример использования:

```
blocks->ET14M_Scale(
    Input,
    Uref,
    R_Ref,
    R0,
    Accuracy
);
```

Scale_Izm

Назначение:

Приведение сигнала из одного типа в другой.

Описание работы блока:

Блок приводит диапазоны сигналов с начальными и конечными точками друг к другу. Фильтрацией контролируется скорость изменения значения Input.

Описание аргументов блока:

Тип	Переменная	Описание
short	Input	Входное значение
short	Filter_Number	Коэффициент фильтрации сигнала (0-100)
short	ADC_Start	Начальное значение входных данных
short	ADC_End	Конечное значение входных данных
float	Signal_Start	Начальное значение выходных данных
float	Signal_End	Конечное значение выходных данных
short	Accuracy	кол-во цифр после запятой выходных данных

Пример использования:

```
blocks->Scale_Izm(
      Input,
      Filter_Number,
      ADC_Start,
      ADC_End,
      Signal_Start,
      Signal_End,
      Accuracy
  );
```

Variable_Meandr

Назначение:

Формирует последовательность импульсов определенной длины.

Описание работы блока:

Когда Reg_In равен "1" происходит переключение сигнала Reg_Out в состояние "1" на время TimeOn. После сигнал Reg_Out изменяет свое состояние на "0" на время TimeOff.

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Reg_In	Вход
reg int	Reg_Out	Выход
short	Time_On	Время вкл.
short	Time_Off	Время выкл.

Пример использования:

```
blocks->Variable_Meandr(
    Reg_In,
    Reg_Out,
    TimeOn,
    TimeOff
);
```

frc_driver

Назначение:

Управление частотным преобразователем.

Описание работы блока:

При появлении одного из сигналов Forward_In или Back_In происходит включение блока изменяя сигнал Forward_Out или Back_Out на "1". При поступлении "1" на противоположный сигнал направления вращения и Slow_In равному "0" блок остановит свою работу, присваивая Stop_Out значение "1" на период Time_Out. По истечении времени блок продолжит свою работу, устанавливая Stop_Out значение "0" и выставит соответствующий сигнал направления вращения.

При Slow_In равному "1" переключение направления вращения происходит без задержки.

При поступлении "1" на сигнал Stop_In сигналы Forward_Out и Back_Out меняют значения на "0", Stop_Out становится равен "1".

Если на вход Forward_In и Back_In были одновременно поданы значения "1", блок переходит в аварийное состояние, присваивая Alarm_FD значение "1".

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Forward_In	Вход бит включения
reg int	Back_In	Вход бит включения реверса
reg int	Stop_In	Вход бит остановки
reg int	Slow_In	Вход бит медленного режима
reg int	Forward_Out	Выход бит включения
reg int	Back_Out	Выход бит включения реверса

reg int	Stop_Out	Выход бит остановки
reg int	Slow_Out	Выход бит медленного режима
reg int	Alarm_FD	Выход бит аварии
ushort	Time_Out	Таймаут переключения (x100 мс)

Пример использования:

```
blocks->frc_driver(
    First_Bit_In,
    First_Bit_Out,
    TimeOut
);
```

Burner

Назначение:

Управление горелкой.

Описание значений переменной Burner_State:

Значение переменной Burner_State Описание

1

2

3

4

5

6

7

0

40

41

69

Описание аргументов блока:

Тип	Переменная	Описание
reg int	Outputs_Pointer	Указатель дискретные выходы
reg int	Inputs_Pointer	Указатель дискретные входы
short	AI_Pointer	Указатель аналоговые входы
short	AO_Pointer	Указатель аналоговые выходы
char	Save_Data	Прочитать данные из флэш памяти
char	Load_Data	Записать данные во флэш память

short	usSet	Точки настройки
short	BURN_Pow_Set	
short	Burner_State	

Пример использования:

```
blocks->Burner(
     Outputs_Pointer,
     Inputs_Pointer,
     AI_Pointer,
     AO_Pointer,
     Save_Data,
     Load_Data,
     usSet,
     BURN_Pow_Set,
     Burner_State
);
```

Карта регистров "Weigher"

Для обмена с системой управления верхнего уровня "Weigher" в контроллере предусмотрено две области памяти регистров обмена (Записи / Чтения).

Карта регистров области Записи

Наименование регистра	Адрес начала	Кол-во бит	Описание
StartAutomatic	3328	1	Бит запуска в дозирования в автоматическом режиме, формируется при запуске дозирования со стороны ПК
NU_3329	3329	1	Не используется
AutomaticMode	3330	1	Регистр разрешения автоматического режима с ПК 1 - режим автомат разрешен 0 - режим автомат запрещен
BatchEnded1FlagReset	3331	1	Регистр сброса флага замес окончен смеситель №1, формируется программой после получение от контроллера флага замес окончен (см. описание)
UnloadComplete1FlagReset	3332	1	Регистр сброса флага смеситель №1 выгрузился, формируется программой после получения от контроллера флага смеситель №1 выгрузился (см. описание)
NU_3333	3333	1	Не используется
ClearOrder	3334	1	Регистр остановки выполнения текущей программы с установкой программы в начальное положение
SetAlgorithmPause	3335	1	Регистр установки паузы выполнения программы
DoserWaitingFlagReset	3336	10	Регистр сброса флага ожидания Дозатора 1-10, формируется после прочтения результатов загрузки по компоненту (см. Дозаторы)
ErrorFlagReset	3346	10	Регистр сброса ошибки дозирования для Дозаторов 1-10
NU_3356_3369	3356	14	Не используется
TurnOnMixer_1	3370	1	Регистр включения смесителя №1, формируется программой автоматически при старте Наряд заказа
TurnOnMixer_2	3371	1	Регистр включения смесителя №2, формируется программой автоматически при старте Наряд заказа
Ring1	3372	1	Регистр управления звонком смесителя №1, формируется программой автоматически при окончании Наряд заказа
Ring2	3373	1	Регистр управления звонком смесителя №2, формируется программой автоматически при окончании Наряд заказа
NU_3374	3374	1	Не используется
ManualAllow	3375	1	Регистр разрешения ручного режима 1 - ручной режим разрешен 0 - ручной режим запрещен

NU_3376	3376	1	Не используется
NU_3376	3377	1	Не используется
Mixer1UnloadBan	3378	1	Регистр запрета выгрузки смеситель №1
Mixer2UnloadBan	3379	1	Регистр запрета выгрузки смеситель №2
NU_3380_3390	3380	1	Не используется
StartDosing	3390	10	Регистр запуска Дозаторов 1-10, выставляется при старте замеса, до момента появления флагов включенных скоростей Дозатора (см. описание)
SecondSpeedAllow	3400	10	Регистр разрешение работы на второй скорости для Дозаторов 1-10
NU_3410_3426	3410	26	Не используется
DoserGlobalPause	3426	10	Регистр включения паузы Дозаторов 1-10 (см. Дозаторы)
ResetDoserGlobal	3436	10	Регистр инициализации Дозаторов 1-10 (см. Дозаторы)
Mixer1LoadBun	3446	1	Регистр запрета загрузки материалов в смеситель
OrderFromQueueStart	3447	1	Регистр старта нового наряд заказа из очереди
OrderDoserNewQueueStart	3448	10	Регистр старта нового наряд заказа из очереди для каждого из 10 Дозаторов

Карта регистров область Чтения

Флаги выставляемые контроллером в процессе работы. Диапазон 3072 - 3328.

Наименование регистра	Адрес начала	Кол-во бит	Описание
NU_3072_3079	3072	8	Не используется
DoserWaiting	3080	10	Флаг выставляемый контроллером после загрузки компонента для Дозаторов 1-10 (см. Дозаторы)
AutomatedStartSucceded	3090	1	Флаг подтверждения запуска в автоматическом режиме
BatchEnded1	3091	1	Флаг смеситель №1 закончил перемешивание
Mixing1	3092	1	Флаг смеситель №1 идет перемешивание
MixerUnloading1	3093	1	Флаг смеситель №1 идет выгрузка
UnloadComplete1	3094	1	Флаг смеситель №1 выгрузка окончена
NU_3095	3095	1	не используется
MixerLoading1	3096	1	Флаг смеситель №1 идет выгрузка
NU_3097_3101	3097	4	Не используется
DosersSpeeds	3102	20	Флаги индикации включения 1-й и 2-й скорости для Дозаторов 1-10
NU_3122_3139	3122	17	Не используется
AutomaticMode	3130	1	Флаг включен автоматический режим
ManualMode	3131	1	Флаг включен ручной режим
SettingUpMode	3132	1	Флаг включен наладочный режим
NU_3133_3149	3133	16	Не используется
DosEnd	3150	10	Флаг дозирование по Дозатору 1-10 закончено
DosWorking	3160	10	Флаг Дозатор 1-10 в работе
DosUnload	3170	10	Флаг Дозатор 1-10 выгружается
DosEmpty	3180	10	Флаг Дозатор 1-10 пустой
ErrorFlags	3190	10	Флаг Дозатор 1-10 ошибка
ErrorCodes	3200	30	Флаг Дозатор 1-10 код ошибки Бит 0 - ошибка нуля, Бит 1 - ошибка дозирования, Бит 2 - резерв
NU_3230_3269	3230	39	Не используется
GlobalPauseSetSuccessfully	3260	10	Флаг Дозатор 1-10 подтверждение паузы
GlobalPauseResetSuccessfully	3270	10	Флаг успешной инициализации Дозатор 1-10 (см. Дозаторы)
OrderFromQueueStartSet	3280	1	Флаг подтверждения чтения регистра старт наряд заказа из очереди

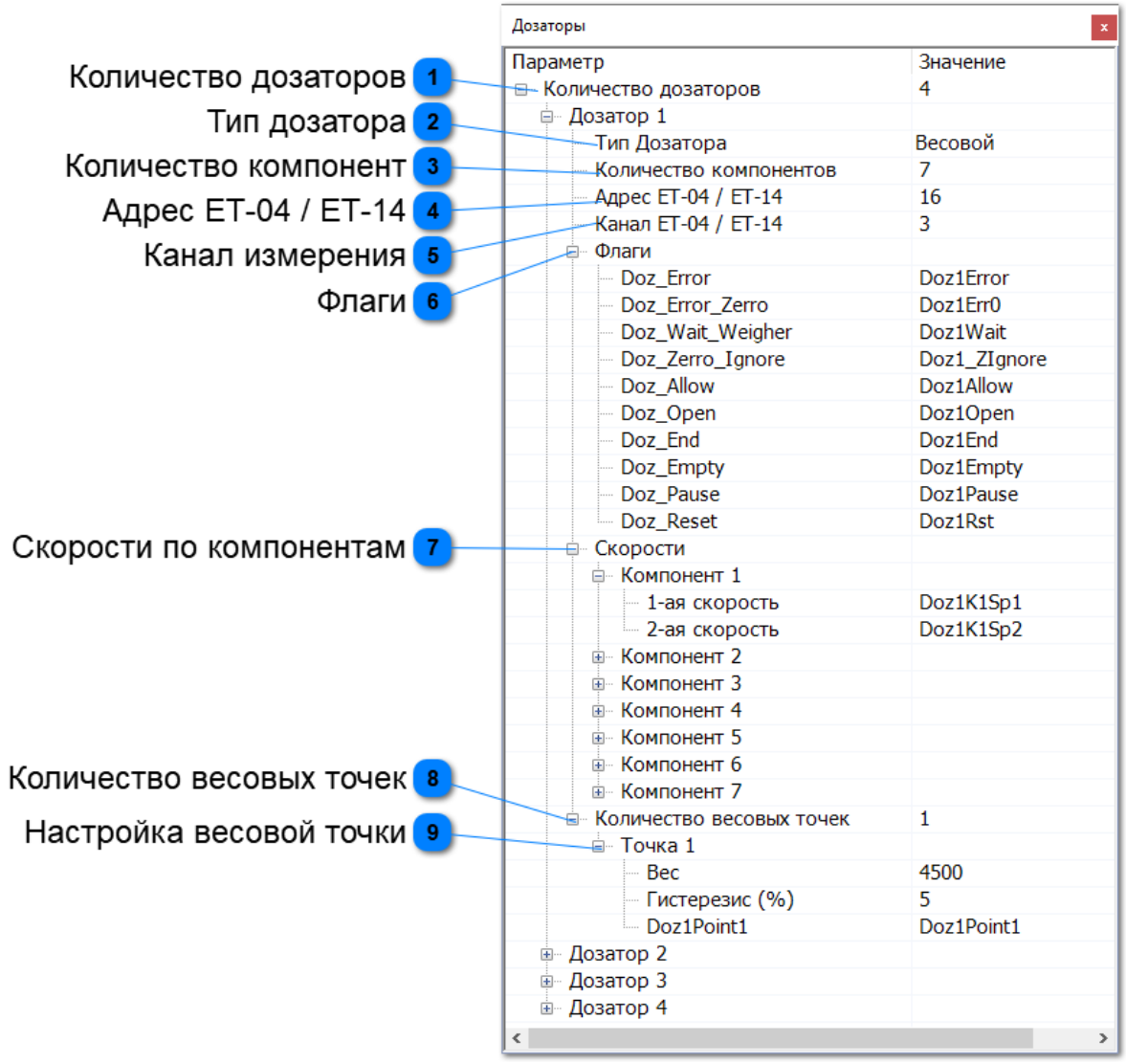
Интегрированные в контроллер функциональные блоки

Дозаторы

Функциональный блок "Дозаторы", предназначен для упрощения написания программ автоматического дозирования. Контроллер позволяет одновременно работать с 10-ю дозаторами. Каждый блок дозатора может дозировать 16 компонент с функций грубого и точного дозирования. Для улучшения качества дозирования, контроллер после каждого цикла дозирования производит автоматическую коррекцию момента отключения исполнительного механизма. Предусмотрено 3 режима автоматической корректировки момента отключения : Статистика, Производная, Время (см. Режимы дозирования).

Для настройки Дозаторов в программе UpakNet используется окно "Дозаторы".

Окно "Дозаторы"



1 Количество дозаторов

Количество дозаторов используемых в программе.

2 Тип дозатора

Тип Дозатора, определяет его алгоритм работы. Контроллер поддерживает следующие типы Дозаторов:

1. Весовой - традиционное весовое дозирование
2. Весовой ПВ - весовое дозирование с выгрузкой после каждого компонента
3. Вычитающий - весовое дозирование из накопительного бункера с материалом
4. Импульсный - дозирование с помощью импульсных счетчиков

5. Виртуальный - дозирование с программно задаваемым весом, предназначен для подачи материалов по времени

3 **Количество компонент**

Количество компонент используемых Дозатором, от 1-го до 16-ти.

4 **Адрес ET-04 / ET-14**

Modbus адрес весового контроллера.

ВНИМАНИЕ!!! При инициализации контроллера происходит проверка всех подключенных весовых модулей и если один из них не пройдет проверку, контроллер не даст разрешения на выполнение программы. Для контроля за состоянием выполнения программы используется системная переменная i31(см. Описание).

Uart для подключения весовых блоков настраивается в окне "Настройки контроллера".

5 **Канал измерения**

Канал измерения весового модуля к которому подключены тензодатчики. Значения 0-2 для ET-04 и 0-5 для ET-14.

6 **Флаги**

Регистры для взаимодействия с модулем дозирования.

1. Doz_Error - Ошибка дозирования.
   После окончания процесса дозирования по каждому компоненту, контроллер сравнивает значение ошибки дозирования с допустимой ошибкой, в случае превышения заданного порогового значения устанавливает регистр Doz_Error в 1 и останавливает процесс дозировки до очистки регистра Doz_Error.
2. Doz_Error_Zero - Ошибка нуля.
   Перед началом дозирования 1-го компонента, контроллер сравнивает текущий вес дозатора с сохраненным весом при калибровке и если отклонение значения веса превышает заданный порог, контроллер выставит регистр Doz_Error_Zero в 1 и остановит процесс дозировки.

ВНИМАНИЕ!!! Регистр Doz_Error_Zero не может быть очищен программно, так как это критическая ошибка, которая может повлиять

на качество выпускаемого продукта.
Возможные причины:
1. Налипание материала на дозаторе
2. Повреждение тензодатчика
3. Повреждение кабеля тензодатчика

3. Doz_Wait_Weigher.

Регистр Doz_Wait_Weigher выставляется контроллером после окончания дозирования каждого компонента, для обеспечения синхронизации с ПО верхнего уровня.

Для автономной работы, в контроллере надо отключить режим связи с ПК.

3. Doz_Zerro_Ignore - Игнорировать ошибку нуля.

Регистр Doz_Zerro_Ignore позволяет программно отключать проверку массы дозатора перед началом дозирования 1-го компонента. (для отключения установить в 1)

4. Doz_Allow - Разрешение работы.

Регистр Doz_Allow предназначен для управления программным блоком дозирования контроллера. Для корректной работы модуля дозирования должен быть установлен в 1

на весь цикл дозирования.

ВНИМАНИЕ!!! При установки в 0 регистра Doz_Allow все флаги модуля дозирования будут сброшены и он перейдет в начальное состояние.

5. Doz_Open - Выгрузка.

Регистр Doz_Open устанавливается контроллером и сигнализирует о необходимости выгрузки дозатора.

6. Doz_End - Дозирование окончено.

Регистр Doz_End формируется контроллером после окончания дозирования по всем компонентам.

7. Doz_Empty - Дозатор пустой.

В процессе выгрузки контроллер проверяет оставшийся вес в дозаторе и сравнивает с пороговым значением "Дозатор пустой", после снижения веса ниже порогового значения запускается таймер задержки выгрузки, после окончания отсчета таймера регистр Doz_Empty устанавливается в 1.

8. Doz_Pause - Установить паузу по Дозатору.

После установки регистра Doz_Pause в 1, модуль дозирования переходит в режим остановки, регистры управляемые контроллером остаются в том же состоянии, в котором были до перехода в режим остановки.

> **ВНИМАНИЕ!!!** Для корректной работы и избегания аварийных ситуаций, необходимо заблокировать все выходные регистры модуля дозирования в момент установки паузы.

9. Doz_Reset - Инициализация дозатора.

Если регистры Doz_Allow и Doz_Pause установлены в 1, то после установки регистра Doz_Reset в 1, модуль дозирования перейдет к 1-му компоненту и произведет обнуление веса по 0-му компоненту без обнуления веса в дозаторе. Если регистры Doz_Allow и Doz_Pause установлены в 0, то после установки регистра Doz_Reset в 1, происходит обнуление дозатора.

> **ВНИМАНИЕ!!!** После установки в 1 регистр Doz_Reset автоматически сбрасывается контроллером, поэтому в программе необходимо обеспечить установку регистра только на один такт программы (см. Описание).

7 Скорости по компонентам

Регистры управления скоростями (по два регистра на каждый компонент) для каждого из 16 компонентов Дозатора. Управляются модулем Дозирования контроллера.
Предназначены для управления механизмами подачи материала в дозатор.
После старта дозирования компонента устанавливается режим грубого дозирования
оба регистра в 1. После достижения настраиваемого порога включения второй скорости регистр 1-я Скорость переходит в 0, регистр 2-я Скорость остается включенным до окончания дозирования по компоненту.

8 Количество весовых точек

Количество весовых точек для модуля Дозатора (0-4).

9 Настройка весовой точки

Вес - задание веса при котором произойдет срабатывание регистра Doz1Point1
Гистерезис - задание гистерезиса

Doz1Point1 - адрес регистра весовой точки

Логика работы:

Регистр Doz1Point1 будет установлен если вес нетто в дозаторе будет больше задания Вес, и перейдет в 0, когда вес нетто будет меньше задания (Вес - (Вес*Гистерезис/100)).

Режимы дозирования

Контроллер поддерживает 4 режима работы корректировки момента отключения:

1. Фиксированный.

Момент корректировки задается вручную и не меняется в процессе работы.

2. Статистика.

В режиме "Статистика" контроллер после каждой дозировки рассчитывает ошибку дозирования и корректирует момент упреждения отключения исполнительного механизма на величину ошибки по следующей формуле:

Mhm=Mhm+(dM/Ккр)

Где,

Mhm - Вес упреждения отключения исполнительного механизма.
Moff - Вес отключения.
dM - полученная ошибка дозирования.
Ккр - количество точек по которым необходимо настраивать коррекцию.
Мз - заданный Вес.

Вес отключения рассчитывается по следующей формуле:

Moff=Мз-Mhm

3. Производная.

В режиме "Производная" в процессе дозирования контроллер рассчитывает скорость загрузки материала и на каждом цикле работы модуля дозатора пересчитывает момент отключения исполнительного механизма по формуле:

Mhm=dM*dMdV

Moff=Мз-Mhm

Где,

Mhm - Вес упреждения отключения исполнительного механизма.
Moff - Вес отключения.
Мз - Заданный Вес.
dM - Скорость загрузки материала кг/сек.
dMDv - Коэффициент учитывающий количество просыпающегося материала после отключения исполнительного механизма.
Ккр - количество точек по которым необходимо настраивать коррекцию.

Коэффициент dMdV пересчитывается после каждого дозирования на величину dMdVт/Ккр.

4. Время.

Режим дозирования "Время" используется для дозирования малых весов, для которых не возможно использовать режим "Статистика" или "Производная". В этом режиме регулируется время включенного состояния исполнительного механизма (шаг регулирования 10 мс). По факту окончания дозирования рассчитывается время загрузки и коррекция по времени на следующую загрузку.

1. Корректировка времени может быть ограничена допустимым максимальным шагом коррекции по времени. Настройка производится из Меню контроллера, либо в программе "Крутой Замес".
2. В случае если за одну загрузку засыпалось меньше чем % <Ошибки дозирования> * 10 будет совершена повторная загрузка.

Учет материалов в ручном режиме

Каждый из модулей Дозаторов 1-10 позволят вести учет дозирования в ручном режиме по первым 8-ми компонентам каждого дозатора.
Регистры управления ручном режиме зарезервированы в программе описаны в разделе "Карта регистров ручного режима".

Карта регистров ручного режима

Дозатор 1

```
# ------------------------------
# Биты ручного режима Дозатор 1
# ------------------------------
#?3936=MAN_MODE_START_K1_D1
#?3937=MAN_MODE_START_K2_D1
#?3938=MAN_MODE_START_K3_D1
#?3939=MAN_MODE_START_K4_D1
#?3940=MAN_MODE_START_K5_D1
#?3941=MAN_MODE_START_K6_D1
#?3942=MAN_MODE_START_K7_D1
#?3943=MAN_MODE_START_K8_D1

#?4016=MAN_MODE_BLOCK_K1_D1
#?4017=MAN_MODE_BLOCK_K2_D1
#?4018=MAN_MODE_BLOCK_K3_D1
#?4019=MAN_MODE_BLOCK_K4_D1
#?4020=MAN_MODE_BLOCK_K5_D1
#?4021=MAN_MODE_BLOCK_K6_D1
#?4022=MAN_MODE_BLOCK_K7_D1
#?4023=MAN_MODE_BLOCK_K8_D1

#?2096=MAN_MODE_BLOCK_D1
# ------------------------------
```

Дозатор 2

```
# ------------------------------
# Биты ручного режима Дозатор 2
# ------------------------------
#?3944=MAN_MODE_START_K1_D2
#?3945=MAN_MODE_START_K2_D2
#?3946=MAN_MODE_START_K3_D2
#?3947=MAN_MODE_START_K4_D2
#?3948=MAN_MODE_START_K5_D2
#?3949=MAN_MODE_START_K6_D2
#?3950=MAN_MODE_START_K7_D2
#?3951=MAN_MODE_START_K8_D2

#?4024=MAN_MODE_BLOCK_K1_D2
#?4025=MAN_MODE_BLOCK_K2_D2
#?4026=MAN_MODE_BLOCK_K3_D2
#?4027=MAN_MODE_BLOCK_K4_D2
#?4028=MAN_MODE_BLOCK_K5_D2
#?4029=MAN_MODE_BLOCK_K6_D2
```

```
#?4030=MAN_MODE_BLOCK_K7_D2
#?4031=MAN_MODE_BLOCK_K8_D2

#?2097=MAN_MODE_BLOCK_D2
# ------------------------------
```

Дозатор 3

```
# ------------------------------
# Биты ручного режима Дозатор 3
# ------------------------------
#?3952=MAN_MODE_START_K1_D3
#?3953=MAN_MODE_START_K2_D3
#?3954=MAN_MODE_START_K3_D3
#?3955=MAN_MODE_START_K4_D3
#?3956=MAN_MODE_START_K5_D3
#?3957=MAN_MODE_START_K6_D3
#?3958=MAN_MODE_START_K7_D3
#?3959=MAN_MODE_START_K8_D3

#?4032=MAN_MODE_BLOCK_K1_D3
#?4033=MAN_MODE_BLOCK_K2_D3
#?4034=MAN_MODE_BLOCK_K3_D3
#?4035=MAN_MODE_BLOCK_K4_D3
#?4036=MAN_MODE_BLOCK_K5_D3
#?4037=MAN_MODE_BLOCK_K6_D3
#?4038=MAN_MODE_BLOCK_K7_D3
#?4039=MAN_MODE_BLOCK_K8_D3

#?2098=MAN_MODE_BLOCK_D3
# ------------------------------
```

Дозатор 4

```
# ------------------------------
# Биты ручного режима Дозатор 4
# ------------------------------
#?3960=MAN_MODE_START_K1_D4
#?3961=MAN_MODE_START_K2_D4
#?3962=MAN_MODE_START_K3_D4
#?3963=MAN_MODE_START_K4_D4
#?3964=MAN_MODE_START_K5_D4
#?3965=MAN_MODE_START_K6_D4
#?3966=MAN_MODE_START_K7_D4
#?3967=MAN_MODE_START_K8_D4

#?4040=MAN_MODE_BLOCK_K1_D4
#?4041=MAN_MODE_BLOCK_K2_D4
#?4042=MAN_MODE_BLOCK_K3_D4
#?4043=MAN_MODE_BLOCK_K4_D4
#?4044=MAN_MODE_BLOCK_K5_D4
#?4045=MAN_MODE_BLOCK_K6_D4
#?4046=MAN_MODE_BLOCK_K7_D4
#?4047=MAN_MODE_BLOCK_K8_D4

#?2099=MAN_MODE_BLOCK_D4
# ------------------------------
```

Дозатор 5

```
# ------------------------------
# Биты ручного режима Дозатор 5
# ------------------------------
#?3968=MAN_MODE_START_K1_D5
#?3969=MAN_MODE_START_K2_D5
#?3970=MAN_MODE_START_K3_D5
#?3971=MAN_MODE_START_K4_D5
#?3972=MAN_MODE_START_K5_D5
#?3973=MAN_MODE_START_K6_D5
#?3974=MAN_MODE_START_K7_D5
#?3975=MAN_MODE_START_K8_D5

#?4048=MAN_MODE_BLOCK_K1_D5
#?4049=MAN_MODE_BLOCK_K2_D5
#?4050=MAN_MODE_BLOCK_K3_D5
#?4051=MAN_MODE_BLOCK_K4_D5
#?4052=MAN_MODE_BLOCK_K5_D5
#?4053=MAN_MODE_BLOCK_K6_D5
#?4054=MAN_MODE_BLOCK_K7_D5
#?4055=MAN_MODE_BLOCK_K8_D5

#?2100=MAN_MODE_BLOCK_D5
# ------------------------------
```

Дозатор 6

```
# ------------------------------
# Биты ручного режима Дозатор 6
# ------------------------------
#?3976=MAN_MODE_START_K1_D6
#?3977=MAN_MODE_START_K2_D6
#?3978=MAN_MODE_START_K3_D6
#?3979=MAN_MODE_START_K4_D6
#?3980=MAN_MODE_START_K5_D6
#?3981=MAN_MODE_START_K6_D6
#?3982=MAN_MODE_START_K7_D6
#?3983=MAN_MODE_START_K8_D6

#?4056=MAN_MODE_BLOCK_K1_D6
#?4057=MAN_MODE_BLOCK_K2_D6
#?4058=MAN_MODE_BLOCK_K3_D6
#?4059=MAN_MODE_BLOCK_K4_D6
#?4060=MAN_MODE_BLOCK_K5_D6
#?4061=MAN_MODE_BLOCK_K6_D6
#?4062=MAN_MODE_BLOCK_K7_D6
#?4063=MAN_MODE_BLOCK_K8_D6

#?2101=MAN_MODE_BLOCK_D6
# ------------------------------
```

Дозатор 7

```
# ------------------------------
# Биты ручного режима Дозатор 7
# ------------------------------
#?3984=MAN_MODE_START_K1_D7
#?3985=MAN_MODE_START_K2_D7
#?3986=MAN_MODE_START_K3_D7
#?3987=MAN_MODE_START_K4_D7
#?3988=MAN_MODE_START_K5_D7
#?3989=MAN_MODE_START_K6_D7
#?3990=MAN_MODE_START_K7_D7
#?3991=MAN_MODE_START_K8_D7

#?4064=MAN_MODE_BLOCK_K1_D7
#?4065=MAN_MODE_BLOCK_K2_D7
#?4066=MAN_MODE_BLOCK_K3_D7
#?4067=MAN_MODE_BLOCK_K4_D7
#?4068=MAN_MODE_BLOCK_K5_D7
#?4069=MAN_MODE_BLOCK_K6_D7
#?4070=MAN_MODE_BLOCK_K7_D7
#?4071=MAN_MODE_BLOCK_K8_D7

#?2102=MAN_MODE_BLOCK_D7
# ------------------------------
```

Дозатор 8

```
# ------------------------------
# Биты ручного режима Дозатор 8
# ------------------------------
#?3992=MAN_MODE_START_K1_D8
#?3993=MAN_MODE_START_K2_D8
#?3994=MAN_MODE_START_K3_D8
#?3995=MAN_MODE_START_K4_D8
#?3996=MAN_MODE_START_K5_D8
#?3997=MAN_MODE_START_K6_D8
#?3998=MAN_MODE_START_K7_D8
#?3999=MAN_MODE_START_K8_D8

#?4072=MAN_MODE_BLOCK_K1_D8
#?4073=MAN_MODE_BLOCK_K2_D8
#?4074=MAN_MODE_BLOCK_K3_D8
#?4075=MAN_MODE_BLOCK_K4_D8
#?4076=MAN_MODE_BLOCK_K5_D8
#?4077=MAN_MODE_BLOCK_K6_D8
#?4078=MAN_MODE_BLOCK_K7_D8
#?4079=MAN_MODE_BLOCK_K8_D8

#?2103=MAN_MODE_BLOCK_D8
# ------------------------------
```

Дозатор 9

```
# ------------------------------
# Биты ручного режима Дозатор 9
# ------------------------------
#?4000=MAN_MODE_START_K1_D9
#?4001=MAN_MODE_START_K2_D9
#?4002=MAN_MODE_START_K3_D9
#?4003=MAN_MODE_START_K4_D9
#?4004=MAN_MODE_START_K5_D9
#?4005=MAN_MODE_START_K6_D9
#?4006=MAN_MODE_START_K7_D9
#?4007=MAN_MODE_START_K8_D9

#?4080=MAN_MODE_BLOCK_K1_D9
#?4081=MAN_MODE_BLOCK_K2_D9
#?4082=MAN_MODE_BLOCK_K3_D9
#?4083=MAN_MODE_BLOCK_K4_D9
#?4084=MAN_MODE_BLOCK_K5_D9
#?4085=MAN_MODE_BLOCK_K6_D9
#?4086=MAN_MODE_BLOCK_K7_D9
#?4087=MAN_MODE_BLOCK_K8_D9

#?2104=MAN_MODE_BLOCK_D9
# ------------------------------
```

Дозатор 10

```
# ------------------------------
# Биты ручного режима Дозатор 10
# ------------------------------
#?4008=MAN_MODE_START_K1_D10
#?4009=MAN_MODE_START_K2_D10
#?4010=MAN_MODE_START_K3_D10
#?4011=MAN_MODE_START_K4_D10
#?4012=MAN_MODE_START_K5_D10
#?4013=MAN_MODE_START_K6_D10
#?4014=MAN_MODE_START_K7_D10
#?4015=MAN_MODE_START_K8_D10

#?4088=MAN_MODE_BLOCK_K1_D10
#?4089=MAN_MODE_BLOCK_K2_D10
#?4090=MAN_MODE_BLOCK_K3_D10
#?4091=MAN_MODE_BLOCK_K4_D10
#?4092=MAN_MODE_BLOCK_K5_D10
#?4093=MAN_MODE_BLOCK_K6_D10
#?4094=MAN_MODE_BLOCK_K7_D10
#?4095=MAN_MODE_BLOCK_K8_D10

#?2105=MAN_MODE_BLOCK_D10
# ------------------------------
```

Пример использования в Upak

В разделе описан пример использования ручного режима для Дозатора 1, настроенного на 5 компонент.

ВНИМАНИЕ!!! Учет ручного режима доступен только для тех компонентов, которые настроены в модуле Дозатора 1-10.

Учет ручного режима Дозатор 1 (5 компонент)

Описание работы работы примера.

В примере используется Дозатор 1 с настроенными 5 компонентами. Для загрузки в ручном режиме компонентов 1-5 используются регистры **SBL_K1_D1-SBL_K5_D1**.

Рассмотрим работу программы на примере первого компонента.

ВНИМАНИЕ!!! Для корректной работы модуля учета ручного режима, программа должна выставлять только один регистр начала загрузки в ручном режиме.

Такой режим обеспечивают флаги блокировки **MAN_MODE_BLOCK_K1_D1-MAN_MODE_BLOCK_K5_D1**, которые выставляет модуль дозатора контроллера ET-XX, после появления флагов **MAN_MODE_START_K1_D1-MAN_MODE_START_K5_D1.**

Формирование сборного регистра общей блокировки загрузки:

MAN_MODE_BLOCK_K1_D1+**MAN_MODE_BLOCK_K2_D1**+**MAN_MODE_BLOCK_K3_D1**+**MAN_MODE_BLOCK_K4_D1**+ **MAN_MODE_BLOCK_K5_D1**=**MAN_MODE_BLOCK_D1**

Таким образом флаг **MAN_MODE_BLOCK_D1** будет появляться сразу после появления одного из флагов **MAN_MODE_BLOCK_K1_D1-MAN_MODE_BLOCK_K5_D1**.

Обеспечение блокирования одновременной загрузки на примере 1-го компонента:

SBL_K1_D1*(**MAN_MODE_BLOCK_K1_D1**+~**MAN_MODE_BLOCK_D1**)=**MAN_MODE_START_K1_D1**

Таким образом, регистр **MAN_MODE_START_K1_D1** будет сформирован только в отсутствии регистра **MAN_MODE_BLOCK_D1**.

На следующем цикле программы модуль Дозатора сформирует флаг **MAN_MODE_BLOCK_K1_D1**, который заблокирует работу всех компонентов кроме 1-го (см. Полный текст примера).

После установки регистра **SBL_K1_D1** в 0, регистр **MAN_MODE_START_K1_D1** также перейдет в 0.

ВНИМАНИЕ!!! После установки регистра MAN_MODE_START_K1_D1 в 0, модуль дозатора контроллера ET-XX будет удерживать регистр MAN_MODE_BLOCK_K1_D1 в 1 еще 1500 мс для корректного учета веса материала.

Полный текст примера

Полный текст программы:

```
# Биты эмуляции загрузки / выгрузки Дозатора 1
#?3860=EMUL_Doz1LoadSp1
#?3861=EMUL_Doz1LoadSp2
#?3862=EMUL_Doz1UnloadSp1
#?3863=EMUL_Doz1UnloadSp2

# Кнопки ручного управления загрузкой
#?2721=SBL_K1_D1
#?2722=SBL_K2_D1
#?2723=SBL_K3_D1
#?2724=SBL_K4_D1
#?2725=SBL_K5_D1

# Выходы включения загрузки компонент 1-5
#?3000=K1_OUT
#?3001=K2_OUT
#?3002=K3_OUT
#?3003=K4_OUT
#?3004=K5_OUT

# Сборный флаг блокировки ручного режима в момент начала загрузки
MAN_MODE_BLOCK_K1_D1+MAN_MODE_BLOCK_K2_D1+MAN_MODE_BLOCK_K3_D1+MAN_MODE_BLOCK_K4_D1+
MAN_MODE_BLOCK_ K5_D1=MAN_MODE_BLOCK_D1

# Загрузка компонента 1 Дозатора 1
SBL_K1_D1*(MAN_MODE_BLOCK_K1_D1+~MAN_MODE_BLOCK_D1)=MAN_MODE_START_K1_D1
MAN_MODE_START_K1_D1*MAN_MODE_BLOCK_K1_D1=K1_OUT

# Загрузка компонента 2 Дозатора 1
SBL_K2_D1*(MAN_MODE_BLOCK_K2_D1+~MAN_MODE_BLOCK_D1)=MAN_MODE_START_K2_D1
MAN_MODE_START_K2_D1*MAN_MODE_BLOCK_K2_D1=K2_OUT

# Загрузка компонента 3 Дозатора 1
SBL_K3_D1*(MAN_MODE_BLOCK_K3_D1+~MAN_MODE_BLOCK_D1)=MAN_MODE_START_K3_D1
MAN_MODE_START_K3_D1*MAN_MODE_BLOCK_K3_D1=K3_OUT

# Загрузка компонента 4 Дозатора 1
SBL_K4_D1*(MAN_MODE_BLOCK_K4_D1+~MAN_MODE_BLOCK_D1)=MAN_MODE_START_K4_D1
MAN_MODE_START_K4_D1*MAN_MODE_BLOCK_K4_D1=K4_OUT

# Загрузка компонента 5 Дозатора 1
SBL_K5_D1*(MAN_MODE_BLOCK_K5_D1+~MAN_MODE_BLOCK_D1)=MAN_MODE_START_K5_D1
MAN_MODE_START_K5_D1*MAN_MODE_BLOCK_K5_D1=K5_OUT

K1_OUT+K2_OUT+K3_OUT+K4_OUT+K5_OUT=EMUL_Doz1LoadSp1
```

Совместная работа контроллера ET-XX и SCADA "Weigher"

Для работы с системой верхнего уровня "Weigher" в битовом адресном пространстве контроллера зарезервированы биты для чтения и записи.

Флаги / Регистры обмена модуля Дозаторов и SCADA "Weigher"

Дозатор 1

```
# ------------------------------
# Флаги / Регистры Дозатор 1
# ------------------------------
#Флаги дозатора
#?10=Doz1Error
#?11=Doz1ErrorZ
#?12=Doz1Wait
#?13=Doz1ZerroIgnore
#?14=Doz1Allow
#?15=Doz1Open
#?16=Doz1End
#?17=Doz1Empty
#?18=Doz1Pause
#?19=Doz1Res
#Скорости дозатора
#?20=Doz1Komp1Sp1
#?21=Doz1Komp1Sp2
#?22=Doz1Komp2Sp1
#?23=Doz1Komp2Sp2
#?24=Doz1Komp3Sp1
#?25=Doz1Komp3Sp2
#?26=Doz1Komp4Sp1
#?27=Doz1Komp4Sp2
#?28=Doz1Komp5Sp1
#?29=Doz1Komp5Sp2
#?30=Doz1Komp6Sp1
#?31=Doz1Komp6Sp2
#?32=Doz1Komp7Sp1
#?33=Doz1Komp7Sp2
#?34=Doz1Komp8Sp1
#?35=Doz1Komp8Sp2
#?36=Doz1Komp9Sp1
#?37=Doz1Komp9Sp2
#?38=Doz1Komp10Sp1
#?39=Doz1Komp10Sp2
#?40=Doz1Komp11Sp1
#?41=Doz1Komp11Sp2
#?42=Doz1Komp12Sp1
#?43=Doz1Komp12Sp2
#?44=Doz1Komp13Sp1
#?45=Doz1Komp13Sp2
#?46=Doz1Komp14Sp1
#?47=Doz1Komp14Sp2
#?48=Doz1Komp15Sp1
```

```
#?49=Doz1Komp15Sp2
#?50=Doz1Komp16Sp1
#?51=Doz1Komp16Sp2
#Весовые точки
#?52=Doz1Point1
#?53=Doz1Point2
#?54=Doz1Point3
#?55=Doz1Point4
#Флаги / Регистры связи с SDADA "Weigher"

# Флаги наличия активных скоростей
#?56=WS_Doz1Speed1
#?57=WS_Doz1Speed2

#Флаг окончания дозирования
#?3150=WS_Doz1End
#Флаг Дозатора работает
#?3160=WS_Doz1Working
#Флаг Дозатор выгружается
#?3170=WS_Doz1Unloading
#Флаг Дозатор пустой
#?3180=WS_Doz1Empty
#Флаг ошибка дозирования
#?3190=WS_Doz1Error
#Флаги коды ошибок
#?3200=WS_Doz1ErrNull
#?3201=WS_Doz1ErrDoz
#Флаг успешной установки паузы
#?3260=WS_Doz1PauseSet
#Флаги успешной инициализации
#?3270=WS_Doz1Reset

#Регистр сброса ошибки
#?3346=WC_Doz1ErrorRes
#Регистр сброса флага ожидания
#?3336=WC_Doz1WaitRes
#Регистр старта дозирования
#?3390=WC_Doz1Start
#Регистр разрешения дозирования на 2-й скорости
#?3400=WC_Doz1Sp2Allow
#Регистр установки паузы по дозатору
#?3426=WC_Doz1PauseSet
#Регистр инициализации дозатора
#?3436=WC_Doz1Reset
# ------------------------------
```

Дозатор 2

```
# ------------------------------
# Флаги / Регистры Дозатор 2
# ------------------------------
#Флаги дозатора
#?110=Doz2Error
#?111=Doz2ErrorZ
#?112=Doz2Wait
#?113=Doz2ZerroIgnore
#?114=Doz2Allow
#?115=Doz2Open
#?116=Doz2End
#?117=Doz2Empty
#?118=Doz2Pause
#?119=Doz2Res
#Скорости дозатора
#?120=Doz2Komp1Sp1
#?121=Doz2Komp1Sp2
#?122=Doz2Komp2Sp1
#?123=Doz2Komp2Sp2
#?124=Doz2Komp3Sp1
#?125=Doz2Komp3Sp2
#?126=Doz2Komp4Sp1
#?127=Doz2Komp4Sp2
#?128=Doz2Komp5Sp1
#?129=Doz2Komp5Sp2
#?130=Doz2Komp6Sp1
#?131=Doz2Komp6Sp2
#?132=Doz2Komp7Sp1
#?133=Doz2Komp7Sp2
#?134=Doz2Komp8Sp1
#?135=Doz2Komp8Sp2
#?136=Doz2Komp9Sp1
#?137=Doz2Komp9Sp2
#?138=Doz2Komp10Sp1
#?139=Doz2Komp10Sp2
#?140=Doz2Komp11Sp1
#?141=Doz2Komp11Sp2
#?142=Doz2Komp12Sp1
#?143=Doz2Komp12Sp2
#?144=Doz2Komp13Sp1
#?145=Doz2Komp13Sp2
#?146=Doz2Komp14Sp1
#?147=Doz2Komp14Sp2
#?148=Doz2Komp15Sp1
#?149=Doz2Komp15Sp2
#?150=Doz2Komp16Sp1
#?151=Doz2Komp16Sp2
#Весовые точки
#?152=Doz2Point1
#?153=Doz2Point2
#?154=Doz2Point3
#?155=Doz2Point4
#Флаги / Регистры связи с SDADA "Weigher"

# Флаги наличия активных скоростей
```

```
#?3104=WS_Doz2Speed1
#?3105=WS_Doz2Speed2

#Флаг окончания дозирования
#?3151=WS_Doz2End
#Флаг Дозатора работает
#?3161=WS_Doz2Working
#Флаг Дозатор выгружается
#?3171=WS_Doz2Unloading
#Флаг Дозатор пустой
#?3181=WS_Doz2Empty
#Флаг ошибка дозирования
#?3191=WS_Doz2Error
#Флаги коды ошибок
#?3203=WS_Doz2ErrNull
#?3204=WS_Doz2ErrDoz
#Флаг успешной установки паузы
#?3261=WS_Doz2PauseSet
#Флаги успешной инициализации
#?3271=WS_Doz2Reset

#Регистр сброса ошибки
#?3347=WC_Doz2ErrorRes
#Регистр сброса флага ожидания
#?3337=WC_Doz2WaitRes
#Регистр старта дозирования
#?3391=WC_Doz2Start
#Регистр разрешения дозирования на 2-й скорости
#?3401=WC_Doz2Sp2Allow
#Регистр установки паузы по дозатору
#?3427=WC_Doz2PauseSet
#Регистр инициализации дозатора
#?3437=WC_Doz2Reset
# ------------------------------
```

Дозатор 3

```
# ------------------------------
# Флаги / Регистры Дозатор 3
# ------------------------------
#Флаги дозатора
#?210=Doz3Error
#?211=Doz3ErrorZ
#?212=Doz3Wait
#?213=Doz3ZerroIgnore
#?214=Doz3Allow
#?215=Doz3Open
#?216=Doz3End
#?217=Doz3Empty
#?218=Doz3Pause
#?219=Doz3Res
#Скорости дозатора
#?220=Doz3Komp1Sp1
#?221=Doz3Komp1Sp2
#?222=Doz3Komp2Sp1
#?223=Doz3Komp2Sp2
#?224=Doz3Komp3Sp1
#?225=Doz3Komp3Sp2
#?226=Doz3Komp4Sp1
#?227=Doz3Komp4Sp2
#?228=Doz3Komp5Sp1
#?229=Doz3Komp5Sp2
#?230=Doz3Komp6Sp1
#?231=Doz3Komp6Sp2
#?232=Doz3Komp7Sp1
#?233=Doz3Komp7Sp2
#?234=Doz3Komp8Sp1
#?235=Doz3Komp8Sp2
#?236=Doz3Komp9Sp1
#?237=Doz3Komp9Sp2
#?238=Doz3Komp10Sp1
#?239=Doz3Komp10Sp2
#?240=Doz3Komp11Sp1
#?241=Doz3Komp11Sp2
#?242=Doz3Komp12Sp1
#?243=Doz3Komp12Sp2
#?244=Doz3Komp13Sp1
#?245=Doz3Komp13Sp2
#?246=Doz3Komp14Sp1
#?247=Doz3Komp14Sp2
#?248=Doz3Komp15Sp1
#?249=Doz3Komp15Sp2
#?250=Doz3Komp16Sp1
#?251=Doz3Komp16Sp2
#Весовые точки
#?252=Doz3Point1
#?253=Doz3Point2
#?254=Doz3Point3
#?255=Doz3Point4

#Флаги / Регистры связи с SDADA "Weigher"
#Флаги наличия активных скоростей
```

```
#?3106=WS_Doz3Speed1
#?3107=WS_Doz3Speed2

#Флаг окончания дозирования
#?3152=WS_Doz3End
#Флаг Дозатора работает
#?3162=WS_Doz3Working
#Флаг Дозатор выгружается
#?3172=WS_Doz3Unloading
#Флаг Дозатор пустой
#?3182=WS_Doz3Empty
#Флаг ошибка дозирования
#?3192=WS_Doz3Error
#Флаги коды ошибок
#?3206=WS_Doz3ErrNull
#?3207=WS_Doz3ErrDoz
#Флаг успешной установки паузы
#?3262=WS_Doz3PauseSet
#Флаги успешной инициализации
#?3272=WS_Doz3Reset

#Регистр сброса ошибки
#?3348=WC_Doz3ErrorRes
#Регистр сброса флага ожидания
#?3338=WC_Doz3WaitRes
#Регистр старта дозирования
#?3392=WC_Doz3Start
#Регистр разрешения дозирования на 2-й скорости
#?3402=WC_Doz3Sp2Allow
#Регистр установки паузы по дозатору
#?3428=WC_Doz3PauseSet
#Регистр инициализации дозатора
#?3438=WC_Doz3Reset
# ------------------------------
```

Дозатор 4

```
# ------------------------------
# Флаги / Регистры Дозатор 4
# ------------------------------
#Флаги дозатора
#?310=Doz4Error
#?311=Doz4ErrorZ
#?312=Doz4Wait
#?313=Doz4ZerroIgnore
#?314=Doz4Allow
#?315=Doz4Open
#?316=Doz4End
#?317=Doz4Empty
#?318=Doz4Pause
#?319=Doz4Res
#Скорости дозатора
#?320=Doz4Komp1Sp1
#?321=Doz4Komp1Sp2
#?322=Doz4Komp2Sp1
#?323=Doz4Komp2Sp2
#?324=Doz4Komp3Sp1
#?325=Doz4Komp3Sp2
#?326=Doz4Komp4Sp1
#?327=Doz4Komp4Sp2
#?328=Doz4Komp5Sp1
#?329=Doz4Komp5Sp2
#?330=Doz4Komp6Sp1
#?331=Doz4Komp6Sp2
#?332=Doz4Komp7Sp1
#?333=Doz4Komp7Sp2
#?334=Doz4Komp8Sp1
#?335=Doz4Komp8Sp2
#?336=Doz4Komp9Sp1
#?337=Doz4Komp9Sp2
#?338=Doz4Komp10Sp1
#?339=Doz4Komp10Sp2
#?340=Doz4Komp11Sp1
#?341=Doz4Komp11Sp2
#?342=Doz4Komp12Sp1
#?343=Doz4Komp12Sp2
#?344=Doz4Komp13Sp1
#?345=Doz4Komp13Sp2
#?346=Doz4Komp14Sp1
#?347=Doz4Komp14Sp2
#?348=Doz4Komp15Sp1
#?349=Doz4Komp15Sp2
#?350=Doz4Komp16Sp1
#?351=Doz4Komp16Sp2
#Весовые точки
#?352=Doz4Point1
#?353=Doz4Point2
#?354=Doz4Point3
#?355=Doz4Point4

#Флаги / Регистры связи с SDADA "Weigher"
#Флаги наличия активных скоростей
```

```
#?3108=WS_Doz4Speed1
#?3109=WS_Doz4Speed2

#Флаг окончания дозирования
#?3153=WS_Doz4End
#Флаг Дозатора работает
#?3163=WS_Doz4Working
#Флаг Дозатор выгружается
#?3173=WS_Doz4Unloading
#Флаг Дозатор пустой
#?3183=WS_Doz4Empty
#Флаг ошибка дозирования
#?3193=WS_Doz4Error
#Флаги коды ошибок
#?3209=WS_Doz4ErrNull
#?3210=WS_Doz4ErrDoz
#Флаг успешной установки паузы
#?3263=WS_Doz4PauseSet
#Флаги успешной инициализации
#?3273=WS_Doz4Reset

#Регистр сброса ошибки
#?3349=WC_Doz4ErrorRes
#Регистр сброса флага ожидания
#?3339=WC_Doz4WaitRes
#Регистр старта дозирования
#?3393=WC_Doz4Start
#Регистр разрешения дозирования на 2-й скорости
#?3403=WC_Doz4Sp2Allow
#Регистр установки паузы по дозатору
#?3429=WC_Doz4PauseSet
#Регистр инициализации дозатора
#?3439=WC_Doz4Reset
# ------------------------------
```

Дозатор 5

```
# ------------------------------
# Флаги / Регистры Дозатор 5
# ------------------------------
#Флаги дозатора
#?410=Doz5Error
#?411=Doz5ErrorZ
#?412=Doz5Wait
#?413=Doz5ZerroIgnore
#?414=Doz5Allow
#?415=Doz5Open
#?416=Doz5End
#?417=Doz5Empty
#?418=Doz5Pause
#?419=Doz5Res
#Скорости дозатора
#?420=Doz5Komp1Sp1
#?421=Doz5Komp1Sp2
#?422=Doz5Komp2Sp1
#?423=Doz5Komp2Sp2
#?424=Doz5Komp3Sp1
#?425=Doz5Komp3Sp2
#?426=Doz5Komp4Sp1
#?427=Doz5Komp4Sp2
#?428=Doz5Komp5Sp1
#?429=Doz5Komp5Sp2
#?430=Doz5Komp6Sp1
#?431=Doz5Komp6Sp2
#?432=Doz5Komp7Sp1
#?433=Doz5Komp7Sp2
#?434=Doz5Komp8Sp1
#?435=Doz5Komp8Sp2
#?436=Doz5Komp9Sp1
#?437=Doz5Komp9Sp2
#?438=Doz5Komp10Sp1
#?439=Doz5Komp10Sp2
#?440=Doz5Komp11Sp1
#?441=Doz5Komp11Sp2
#?442=Doz5Komp12Sp1
#?443=Doz5Komp12Sp2
#?444=Doz5Komp13Sp1
#?445=Doz5Komp13Sp2
#?446=Doz5Komp14Sp1
#?447=Doz5Komp14Sp2
#?448=Doz5Komp15Sp1
#?449=Doz5Komp15Sp2
#?450=Doz5Komp16Sp1
#?451=Doz5Komp16Sp2
#Весовые точки
#?452=Doz5Point1
#?453=Doz5Point2
#?454=Doz5Point3
#?455=Doz5Point4

#Флаги / Регистры связи с SDADA "Weigher"
#Флаги наличия активных скоростей
```

```
#?3110=WS_Doz5Speed1
#?3111=WS_Doz5Speed2

#Флаг окончания дозирования
#?3154=WS_Doz5End
#Флаг Дозатора работает
#?3164=WS_Doz5Working
#Флаг Дозатор выгружается
#?3174=WS_Doz5Unloading
#Флаг Дозатор пустой
#?3184=WS_Doz5Empty
#Флаг ошибка дозирования
#?3194=WS_Doz5Error
#Флаги коды ошибок
#?3212=WS_Doz5ErrNull
#?3213=WS_Doz5ErrDoz
#Флаг успешной установки паузы
#?3264=WS_Doz5PauseSet
#Флаги успешной инициализации
#?3274=WS_Doz5Reset

#Регистр сброса ошибки
#?3350=WC_Doz5ErrorRes
#Регистр сброса флага ожидания
#?3340=WC_Doz5WaitRes
#Регистр старта дозирования
#?3394=WC_Doz5Start
#Регистр разрешения дозирования на 2-й скорости
#?3404=WC_Doz5Sp2Allow
#Регистр установки паузы по дозатору
#?3430=WC_Doz5PauseSet
#Регистр инициализации дозатора
#?3440=WC_Doz5Reset
# --------------------------------
```

Дозатор 6

```
# ------------------------------
# Флаги / Регистры Дозатор 6
# ------------------------------
#Флаги дозатора
#?510=Doz6Error
#?511=Doz6ErrorZ
#?512=Doz6Wait
#?513=Doz6ZerroIgnore
#?514=Doz6Allow
#?515=Doz6Open
#?516=Doz6End
#?517=Doz6Empty
#?518=Doz6Pause
#?519=Doz6Res
#Скорости дозатора
#?520=Doz6Komp1Sp1
#?521=Doz6Komp1Sp2
#?522=Doz6Komp2Sp1
#?523=Doz6Komp2Sp2
#?524=Doz6Komp3Sp1
#?525=Doz6Komp3Sp2
#?526=Doz6Komp4Sp1
#?527=Doz6Komp4Sp2
#?528=Doz6Komp5Sp1
#?529=Doz6Komp5Sp2
#?530=Doz6Komp6Sp1
#?531=Doz6Komp6Sp2
#?532=Doz6Komp7Sp1
#?533=Doz6Komp7Sp2
#?534=Doz6Komp8Sp1
#?535=Doz6Komp8Sp2
#?536=Doz6Komp9Sp1
#?537=Doz6Komp9Sp2
#?538=Doz6Komp10Sp1
#?539=Doz6Komp10Sp2
#?540=Doz6Komp11Sp1
#?541=Doz6Komp11Sp2
#?542=Doz6Komp12Sp1
#?543=Doz6Komp12Sp2
#?544=Doz6Komp13Sp1
#?545=Doz6Komp13Sp2
#?546=Doz6Komp14Sp1
#?547=Doz6Komp14Sp2
#?548=Doz6Komp15Sp1
#?549=Doz6Komp15Sp2
#?550=Doz6Komp16Sp1
#?551=Doz6Komp16Sp2
#Весовые точки
#?552=Doz6Point1
#?553=Doz6Point2
#?554=Doz6Point3
#?555=Doz6Point4

#Флаги / Регистры связи с SDADA "Weigher"
#Флаги наличия активных скоростей
```

```
#?3112=WS_Doz6Speed1
#?3113=WS_Doz6Speed2

#Флаг окончания дозирования
#?3155=WS_Doz6End
#Флаг Дозатора работает
#?3165=WS_Doz6Working
#Флаг Дозатор выгружается
#?3175=WS_Doz6Unloading
#Флаг Дозатор пустой
#?3185=WS_Doz6Empty
#Флаг ошибка дозирования
#?3195=WS_Doz6Error
#Флаги коды ошибок
#?3215=WS_Doz6ErrNull
#?3216=WS_Doz6ErrDoz
#Флаг успешной установки паузы
#?3265=WS_Doz6PauseSet
#Флаги успешной инициализации
#?3275=WS_Doz6Reset

#Регистр сброса ошибки
#?3351=WC_Doz6ErrorRes
#Регистр сброса флага ожидания
#?3341=WC_Doz6WaitRes
#Регистр старта дозирования
#?3395=WC_Doz6Start
#Регистр разрешения дозирования на 2-й скорости
#?3405=WC_Doz6Sp2Allow
#Регистр установки паузы по дозатору
#?3431=WC_Doz6PauseSet
#Регистр инициализации дозатора
#?3441=WC_Doz6Reset
# ------------------------------
```

Дозатор 7

```
# ------------------------------
# Флаги / Регистры Дозатор 7
# ------------------------------
#Флаги дозатора
#?610=Doz7Error
#?611=Doz7ErrorZ
#?612=Doz7Wait
#?613=Doz7ZerroIgnore
#?614=Doz7Allow
#?615=Doz7Open
#?616=Doz7End
#?617=Doz7Empty
#?618=Doz7Pause
#?619=Doz7Res
#Скорости дозатора
#?620=Doz7Komp1Sp1
#?621=Doz7Komp1Sp2
#?622=Doz7Komp2Sp1
#?623=Doz7Komp2Sp2
#?624=Doz7Komp3Sp1
#?625=Doz7Komp3Sp2
#?626=Doz7Komp4Sp1
#?627=Doz7Komp4Sp2
#?628=Doz7Komp5Sp1
#?629=Doz7Komp5Sp2
#?630=Doz7Komp6Sp1
#?631=Doz7Komp6Sp2
#?632=Doz7Komp7Sp1
#?633=Doz7Komp7Sp2
#?634=Doz7Komp8Sp1
#?635=Doz7Komp8Sp2
#?636=Doz7Komp9Sp1
#?637=Doz7Komp9Sp2
#?638=Doz7Komp10Sp1
#?639=Doz7Komp10Sp2
#?640=Doz7Komp11Sp1
#?641=Doz7Komp11Sp2
#?642=Doz7Komp12Sp1
#?643=Doz7Komp12Sp2
#?644=Doz7Komp13Sp1
#?645=Doz7Komp13Sp2
#?646=Doz7Komp14Sp1
#?647=Doz7Komp14Sp2
#?648=Doz7Komp15Sp1
#?649=Doz7Komp15Sp2
#?650=Doz7Komp16Sp1
#?651=Doz7Komp16Sp2
#Весовые точки
#?652=Doz7Point1
#?653=Doz7Point2
#?654=Doz7Point3
#?655=Doz7Point4

#Флаги / Регистры связи с SDADA "Weigher"
#Флаги наличия активных скоростей
```

```
#?3114=WS_Doz7Speed1
#?3115=WS_Doz7Speed2

#Флаг окончания дозирования
#?3156=WS_Doz7End
#Флаг Дозатора работает
#?3166=WS_Doz7Working
#Флаг Дозатор выгружается
#?3176=WS_Doz7Unloading
#Флаг Дозатор пустой
#?3186=WS_Doz7Empty
#Флаг ошибка дозирования
#?3196=WS_Doz7Error
#Флаги коды ошибок
#?3218=WS_Doz7ErrNull
#?3219=WS_Doz7ErrDoz
#Флаг успешной установки паузы
#?3266=WS_Doz7PauseSet
#Флаги успешной инициализации
#?3276=WS_Doz7Reset

#Регистр сброса ошибки
#?3352=WC_Doz7ErrorRes
#Регистр сброса флага ожидания
#?3342=WC_Doz7WaitRes
#Регистр старта дозирования
#?3396=WC_Doz7Start
#Регистр разрешения дозирования на 2-й скорости
#?3406=WC_Doz7Sp2Allow
#Регистр установки паузы по дозатору
#?3432=WC_Doz7PauseSet
#Регистр инициализации дозатора
#?3442=WC_Doz7Reset
# ------------------------------
```

Дозатор 8

```
# ------------------------------
# Флаги / Регистры Дозатор 8
# ------------------------------
#Флаги дозатора
#?710=Doz8Error
#?711=Doz8ErrorZ
#?712=Doz8Wait
#?713=Doz8ZerroIgnore
#?714=Doz8Allow
#?715=Doz8Open
#?716=Doz8End
#?717=Doz8Empty
#?718=Doz8Pause
#?719=Doz8Res
#Скорости дозатора
#?720=Doz8Komp1Sp1
#?721=Doz8Komp1Sp2
#?722=Doz8Komp2Sp1
#?723=Doz8Komp2Sp2
#?724=Doz8Komp3Sp1
#?725=Doz8Komp3Sp2
#?726=Doz8Komp4Sp1
#?727=Doz8Komp4Sp2
#?728=Doz8Komp5Sp1
#?729=Doz8Komp5Sp2
#?730=Doz8Komp6Sp1
#?731=Doz8Komp6Sp2
#?732=Doz8Komp7Sp1
#?733=Doz8Komp7Sp2
#?734=Doz8Komp8Sp1
#?735=Doz8Komp8Sp2
#?736=Doz8Komp9Sp1
#?737=Doz8Komp9Sp2
#?738=Doz8Komp10Sp1
#?739=Doz8Komp10Sp2
#?740=Doz8Komp11Sp1
#?741=Doz8Komp11Sp2
#?742=Doz8Komp12Sp1
#?743=Doz8Komp12Sp2
#?744=Doz8Komp13Sp1
#?745=Doz8Komp13Sp2
#?746=Doz8Komp14Sp1
#?747=Doz8Komp14Sp2
#?748=Doz8Komp15Sp1
#?749=Doz8Komp15Sp2
#?750=Doz8Komp16Sp1
#?751=Doz8Komp16Sp2
#Весовые точки
#?752=Doz8Point1
#?753=Doz8Point2
#?754=Doz8Point3
#?755=Doz8Point4

#Флаги / Регистры связи с SDADA "Weigher"
#Флаги наличия активных скоростей
```

```
#?3116=WS_Doz8Speed1
#?3117=WS_Doz8Speed2

#Флаг окончания дозирования
#?3157=WS_Doz8End
#Флаг Дозатора работает
#?3167=WS_Doz8Working
#Флаг Дозатор выгружается
#?3177=WS_Doz8Unloading
#Флаг Дозатор пустой
#?3187=WS_Doz8Empty
#Флаг ошибка дозирования
#?3197=WS_Doz8Error
#Флаги коды ошибок
#?3221=WS_Doz8ErrNull
#?3222=WS_Doz8ErrDoz
#Флаг успешной установки паузы
#?3267=WS_Doz8PauseSet
#Флаги успешной инициализации
#?3277=WS_Doz8Reset

#Регистр сброса ошибки
#?3353=WC_Doz8ErrorRes
#Регистр сброса флага ожидания
#?3343=WC_Doz8WaitRes
#Регистр старта дозирования
#?3397=WC_Doz8Start
#Регистр разрешения дозирования на 2-й скорости
#?3407=WC_Doz8Sp2Allow
#Регистр установки паузы по дозатору
#?3433=WC_Doz8PauseSet
#Регистр инициализации дозатора
#?3443=WC_Doz8Reset
# ------------------------------
```

Дозатор 9

```
# ------------------------------
# Флаги / Регистры Дозатор 9
# ------------------------------
#Флаги дозатора
#?810=Doz9Error
#?811=Doz9ErrorZ
#?812=Doz9Wait
#?813=Doz9ZerroIgnore
#?814=Doz9Allow
#?815=Doz9Open
#?816=Doz9End
#?817=Doz9Empty
#?818=Doz9Pause
#?819=Doz9Res
#Скорости дозатора
#?820=Doz9Komp1Sp1
#?821=Doz9Komp1Sp2
#?822=Doz9Komp2Sp1
#?823=Doz9Komp2Sp2
#?824=Doz9Komp3Sp1
#?825=Doz9Komp3Sp2
#?826=Doz9Komp4Sp1
#?827=Doz9Komp4Sp2
#?828=Doz9Komp5Sp1
#?829=Doz9Komp5Sp2
#?830=Doz9Komp6Sp1
#?831=Doz9Komp6Sp2
#?832=Doz9Komp7Sp1
#?833=Doz9Komp7Sp2
#?834=Doz9Komp8Sp1
#?835=Doz9Komp8Sp2
#?836=Doz9Komp9Sp1
#?837=Doz9Komp9Sp2
#?838=Doz9Komp10Sp1
#?839=Doz9Komp10Sp2
#?840=Doz9Komp11Sp1
#?841=Doz9Komp11Sp2
#?842=Doz9Komp12Sp1
#?843=Doz9Komp12Sp2
#?844=Doz9Komp13Sp1
#?845=Doz9Komp13Sp2
#?846=Doz9Komp14Sp1
#?847=Doz9Komp14Sp2
#?848=Doz9Komp15Sp1
#?849=Doz9Komp15Sp2
#?850=Doz9Komp16Sp1
#?851=Doz9Komp16Sp2
#Весовые точки
#?852=Doz9Point1
#?853=Doz9Point2
#?854=Doz9Point3
#?855=Doz9Point4

#Флаги / Регистры связи с SDADA "Weigher"
#Флаги наличия активных скоростей
```

```
#?3118=WS_Doz9Speed1
#?3119=WS_Doz9Speed2

#Флаг окончания дозирования
#?3158=WS_Doz9End
#Флаг Дозатора работает
#?3168=WS_Doz9Working
#Флаг Дозатор выгружается
#?3178=WS_Doz9Unloading
#Флаг Дозатор пустой
#?3188=WS_Doz9Empty
#Флаг ошибка дозирования
#?3198=WS_Doz9Error
#Флаги коды ошибок
#?3224=WS_Doz9ErrNull
#?3225=WS_Doz9ErrDoz
#Флаг успешной установки паузы
#?3268=WS_Doz9PauseSet
#Флаги успешной инициализации
#?3278=WS_Doz9Reset

#Регистр сброса ошибки
#?3354=WC_Doz9ErrorRes
#Регистр сброса флага ожидания
#?3344=WC_Doz9WaitRes
#Регистр старта дозирования
#?3398=WC_Doz9Start
#Регистр разрешения дозирования на 2-й скорости
#?3408=WC_Doz9Sp2Allow
#Регистр установки паузы по дозатору
#?3434=WC_Doz9PauseSet
#Регистр инициализации дозатора
#?3444=WC_Doz9Reset
# ------------------------------
```

Дозатор10

```
# ------------------------------
# Флаги / Регистры Дозатор 10
# ------------------------------
#Флаги дозатора
#?910=Doz10Error
#?911=Doz10ErrorZ
#?912=Doz10Wait
#?913=Doz10ZerroIgnore
#?914=Doz10Allow
#?915=Doz10Open
#?916=Doz10End
#?917=Doz10Empty
#?918=Doz10Pause
#?919=Doz10Res
#Скорости дозатора
#?920=Doz10Komp1Sp1
#?921=Doz10Komp1Sp2
#?922=Doz10Komp2Sp1
#?923=Doz10Komp2Sp2
#?924=Doz10Komp3Sp1
#?925=Doz10Komp3Sp2
#?926=Doz10Komp4Sp1
#?927=Doz10Komp4Sp2
#?928=Doz10Komp5Sp1
#?929=Doz10Komp5Sp2
#?930=Doz10Komp6Sp1
#?931=Doz10Komp6Sp2
#?932=Doz10Komp7Sp1
#?933=Doz10Komp7Sp2
#?934=Doz10Komp8Sp1
#?935=Doz10Komp8Sp2
#?936=Doz10Komp9Sp1
#?937=Doz10Komp9Sp2
#?938=Doz10Komp10Sp1
#?939=Doz10Komp10Sp2
#?940=Doz10Komp11Sp1
#?941=Doz10Komp11Sp2
#?942=Doz10Komp12Sp1
#?943=Doz10Komp12Sp2
#?944=Doz10Komp13Sp1
#?945=Doz10Komp13Sp2
#?946=Doz10Komp14Sp1
#?947=Doz10Komp14Sp2
#?948=Doz10Komp15Sp1
#?949=Doz10Komp15Sp2
#?950=Doz10Komp16Sp1
#?951=Doz10Komp16Sp2
#Весовые точки
#?952=Doz10Point1
#?953=Doz10Point2
#?954=Doz10Point3
#?955=Doz10Point4

#Флаги / Регистры связи с SDADA "Weigher"
#Флаги наличия активных скоростей
```

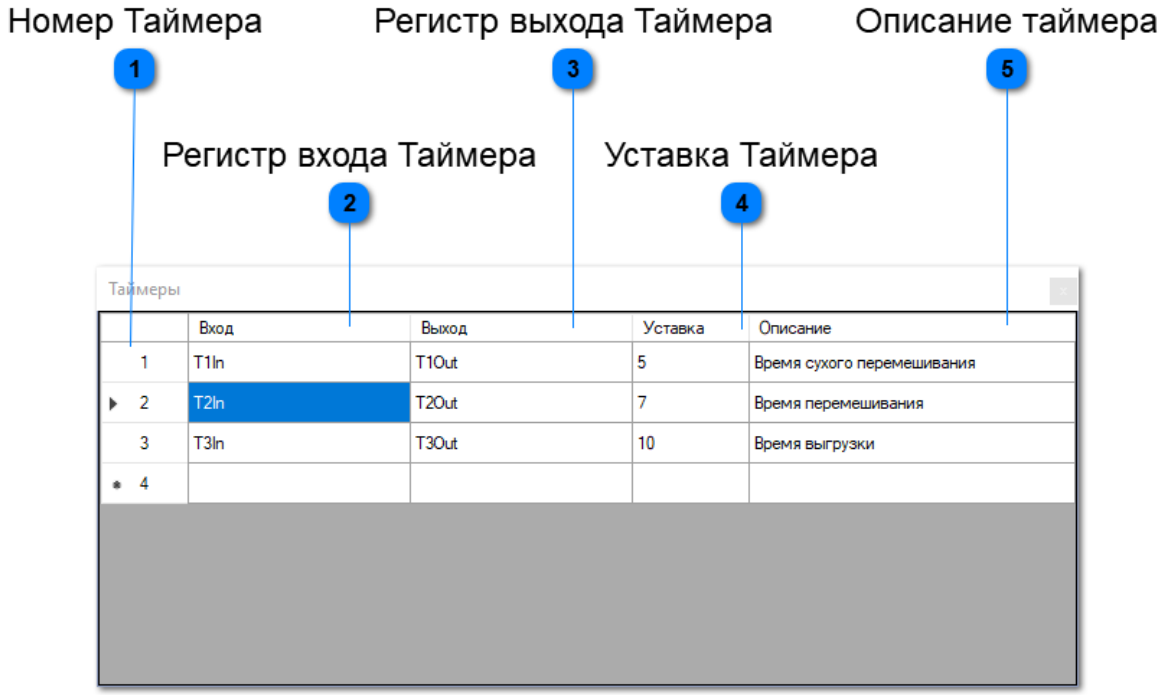
```
#?3120=WS_Doz10Speed1
#?3121=WS_Doz10Speed2

#Флаг окончания дозирования
#?3159=WS_Doz10End
#Флаг Дозатора работает
#?3169=WS_Doz10Working
#Флаг Дозатор выгружается
#?3179=WS_Doz10Unloading
#Флаг Дозатор пустой
#?3189=WS_Doz10Empty
#Флаг ошибка дозирования
#?3199=WS_Doz10Error
#Флаги коды ошибок
#?3227=WS_Doz10ErrNull
#?3228=WS_Doz10ErrDoz
#Флаг успешной установки паузы
#?3269=WS_Doz10PauseSet
#Флаги успешной инициализации
#?3279=WS_Doz10Reset

#Регистр сброса ошибки
#?3355=WC_Doz10ErrorRes
#Регистр сброса флага ожидания
#?3345=WC_Doz10WaitRes
#Регистр старта дозирования
#?3399=WC_Doz10Start
#Регистр разрешения дозирования на 2-й скорости
#?3409=WC_Doz10Sp2Allow
#Регистр установки паузы по дозатору
#?3435=WC_Doz10PauseSet
#Регистр инициализации дозатора
#?3445=WC_Doz10Reset
# ------------------------------
```

Таймеры

Функциональный блок "Таймеры" предназначен для организации временных задержек, кратных 1 сек. Контроллер позволяет использовать до 40 таймеров.



1 Номер Таймера

Порядковый номер Таймера.

2 Регистр входа Таймера

Для запуска таймера регистр входа таймера надо уставновить в 1. 1 должна удерживаться на все время работы таймера. При подаче 0 на вход таймера он сбрасывается в начальное положение.

3 Регистр выхода Таймера

Регистр выхода Таймера будет установлен контроллером в 1, после отсччета временной уставки.

4 Уставка Таймера

Временная уставка в секундах, после которой произойдет срабатывание таймера.

> ⚠ Для каждого таймера предусмотрена коррекция, для изменения настроек в процессе работы. Коррекция может быть произведена из Меню контроллера либо интерфейсу. При этом значение уставки будет изменено на величину коррекции.

5 **Описание таймера**

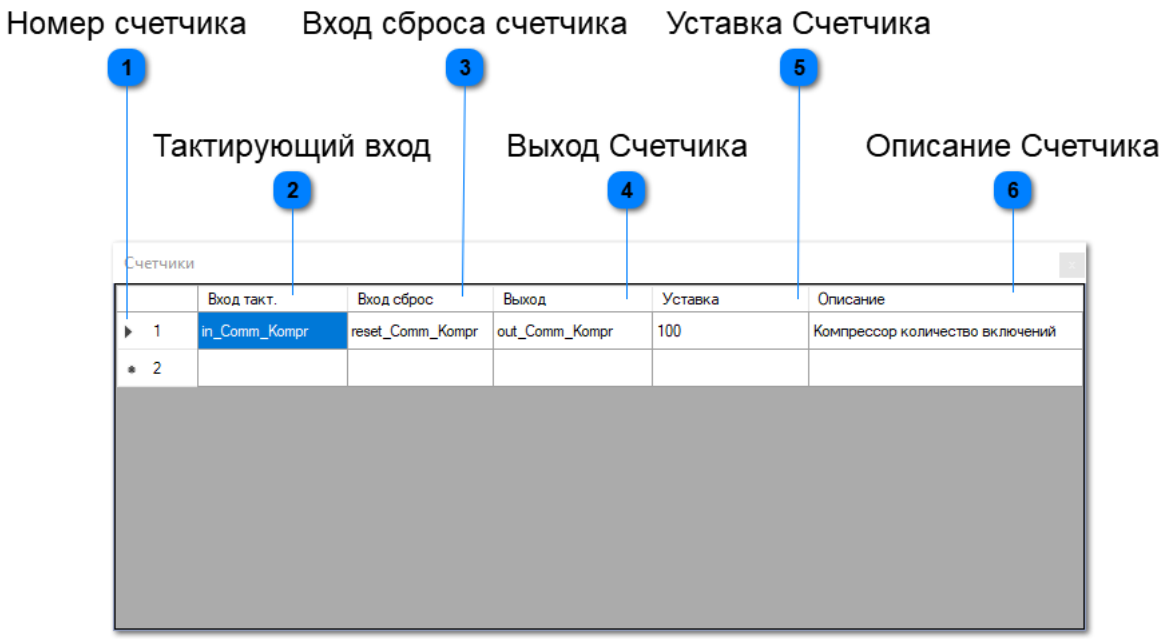
Текстовое поле для описания таймера.

Таймеры мс.

Функциональный блок "Таймеры мс" предназначен для организации временных задержек, кратных 100 мс. Контроллер позволяет использовать до 20 таймеров. Принцип настройки аналогичен функциональному блоку "Таймеры".

Счетчики

Функциональный блок "Счетчики" предназначен для подсчета количества импульсов и выдаче сигнала по достижению заданного значения. Контроллер позволяет использовать до 40 счетчиков.



1 **Номер счетчика**

Порядковый номер Счетчика.

2 **Тактирующий вход**

Тактирующий вход Счетчика, счет происходит по переднему фронту.

3 **Вход сброса счетчика**

При подаче 1 на вход сброса Счетчика, внутренний реситр счета сбрасывается в 0.

4 **Выход Счетчика**

При достижении значения Уставки, контроллер выставлят 1.

5 **Уставка Счетчика**

Количество импульсов до срабатывания таймера.

6 **Описание Счетчика**

Текстовое описание Счетчика.

Поддерживаемые контроллеры

ПЛК ET-XX

Введение

Настоящее руководство по эксплуатации предназначено для ознакомления обслуживающего персонала с устройством, конструкцией, работой и техническим обслуживанием программируемого логического контроллера ПЛК ЕТ-ХХ, в дальнейшем по тексту именуемого «прибор» или «контроллер».
Подключение, регулировка и техобслуживание прибора должны производиться только
квалифицированными специалистами после прочтения настоящего руководства по эксплуатации.
Контроллер изготавливается в нескольких модификациях: ПЛК ЕТ-ХХ и ПЛК ЕТ-ХХ Lite.
Основные характеристики ПЛК приведены в таблице 1.1

Таблица 1.1

Наименование	ПЛК ЕТ-ХХ	ПЛК ЕТ-ХХ Lite
Номинальное напряжение питания	24 В	24 В
Аналоговые выходы 0-20 мА	4 шт.	Отсутствуют.
Аналоговые входы 0-10 В	4 шт.	4 шт.
Дискретные входы 12-28 В	4 шт.	4 шт.
Дискретные выходы (открытый коллектор) 12-28 В	8 шт.	8 шт.
Порт программирования Ethernet	1 шт.	1 шт.
RS-485 (свободно конфигурируемые)	5 шт.	3 шт.

Используемые термины и сокращения

Modbus – открытый протокол обмена по сети RS-485, разработан компанией ModiCon,
в настоящий момент поддерживается независимой организацией Modbus-IDA (www.modbus.org).

Modbus-TCP – версия протокола Modbus, адаптированная к работе в сети TCP/IP.

EEPROM – электрически стираемое перепрограммируемое ПЗУ.

RAM – оперативная память для хранения значений переменных пользовательской программы.

FRAM – энергонезависимая память для хранения значений переменных пользовательской программы.

ОЗУ – оперативное запоминающее устройство, оперативная память.

ОС – операционная система.

ПЛК – программируемый логический контроллер.

Пользовательская программа – программа, созданная в среде UpakNet пользователем
контроллера (или лицом, производящим его начальное программирование).

ПО – программное обеспечение.

ПК – персональный компьютер.

МП – монтажная панель.

РП – руководство пользователя «Программирование программируемых логических контроллеров ЕТ-ХХ».

ШИМ – широтно-импульсная модуляция.

Категория используемой нагрузки (по ГОСТ IEC 60947-1-2017) для типичной области применения:

- **DC-13** – для постоянного тока: управление электромагнитами постоянного тока;
- **AC-15** – для переменного тока: управление электромагнитными нагрузками.

Назначение

Контроллер предназначен для использования в составе различных автоматизированных систем контроля и управления на промышленных предприятиях. Программная среда активно развивается и модернизируется.

Контроллер может управлять:

• выделенными локальными объектами;

• локальным объектом в составе комплексной информационной сети;

• группой локальных объектов в составе комплексной информационной сети.

Контроллер можно применить на промышленных объектах.

Логика работы прибора программируется с помощью UpakNet.

Поддерживаются языки программирования:

• Upak (ELL) – язык релейной логики структурно аналогичен языку LAD.

• Logic (LC) – язык высокого уровня общего назначения по синтаксису схожий с языком C.

Технические характеристики и условия эксплуатации

Технические характеристики

Основные технические характеристики контроллера представлены в таблицах 2.1-2.10.

Таблица 2.1

Напряжение		
Параметр	Значение (свойства)	
	ПЛК ЕТ-XX	ПЛК ЕТ-XX Lite
Напряжение питания	от 9 до 30 В постоянного тока при Т > минус 20 °С от 9 до 26 В постоянного тока при минус 40 °С > Т > минус 20 °С (номинальное 12 или 24 В)	
Потребляемая мощность, не более.	0.5 Вт.	
Пусковой ток, не более: при напряжении 24 В	0.4 мА	
Длительность переходного процесса, не более: при напряжении 24 В	100 мс	
Выходное напряжение встроенного источника питания энергонезависимой памяти	3.3 ± 0.1 В	

Таблица 2.2

Дискретные выходы (открытый коллектор)		
Параметр	Значение (свойства)	
	ПЛК ЕТ-XX	ПЛК ЕТ-XX Lite
Подключаемые входные устройства	Коммутационные устройства (контакты кнопок, выключателей, герконов, реле и т. п.), датчики, имеющие на выходе транзистор n-p-n или p-n-p типа с открытым коллектором, дискретные сигналы 24 ± 3 В	
Количество выходных каналов	8 шт.	
Максимальный коммутируемый NPN транзистора ток не более:	500 мА	
Время изменения состояния из состояния «лог. 0» в «лог. 1» и обратно, не более:	0.25 мкс (XT4-XT5 выходы DO1–DO8)	
Время наработки на отказ NPN транзистора, не менее:	60000 часов	

Таблица 2.3

Цифровые (дискретные) входы	
Параметр	Значение (свойства)
	ПЛК ЕТ-XX и ПЛК ЕТ-XX Lite
Количество входов из них быстродействующих	4 шт. 4 (XT2 входы DI1–DI4)
Тип входов по ГОСТ Р 52931	1 и 2
Напряжение питания дискретных входов	24 ± 3 В
Максимальный входной ток дискретного входа, не более	2.5 мА при питании 24 В
Сигнал «логической единицы», соответствующий состоянию «Включено», дискретных входов для постоянного напряжения (ток в цепи)	От 15 до 28 В (ток от 3 до 15 мА)
Сигнал «логического нуля», соответствующий состоянию «Выключено», дискретных входов для постоянного напряжения (ток в цепи)	От минус 3 до 5 В (ток до 15 мА)
Минимальная длительность импульса, воспринимаемого дискретным входом: для быстродействующих	0,02 мс
Подключаемые входные устройства	Коммутационные устройства (контакты кнопок, выключателей, герконов, реле и т. п.), датчики, имеющие на выходе транзистор n-p-n или p-n-p типа с открытым коллектором, дискретные сигналы 24 ± 3 В

Таблица 2.4

Аналоговые выходы	
Параметр	Значение (свойства)
	ПЛК ЕТ-ХХ
Количество аналоговых выходов	4 шт. (ХТ3 входы АО1–АО4)
Тип выходного сигнала	ток от 4 до 20 мА
Сопротивление нагрузки	Не более 500 Ом для 4…20 мА
Предел основной приведенной погрешности ЦАП	± 0,5 %
Разрядность ЦАП	10 бит
Минимальный период обновления выходов	10 мс
Питание аналоговых выходов, внутреннее	24 ± 1 В, длина линии от источника питания не должна превышать 30 м
Предел допускаемой дополнительной приведенной погрешности аналоговых выходов, вызванной изменением температуры окружающего воздуха от нормальной на каждые 10 °С изменения температуры	Не более 0,5 предела допускаемой основной приведенной погрешности аналоговых выходов

Таблица 2.5

Аналоговые входы		
Параметр	Значение (свойства)	
	ПЛК ЕТ-XX	ПЛК ЕТ-XX Lite
Количество входов	4 шт. (XT1 входы AI1–AI4)	
Тип поддерживаемых унифицированных сигналов	напряжение от 0 до 10 В	
Разрядность АЦП	12 бит	
Входное сопротивление, не более: в режиме измерения напряжения, не менее	200 кОм	
Период опроса одного входа	10 мс	
Предел основной приведенной погрешности преобразования	± 0,25 %	
Предел дополнительной приведенной погрешности преобразования на каждые 10 градусов изменения температуры	± 0,05 %	

Таблица 2.6

Вычислительные ресурсы		
Параметр	Значение (свойства)	
	ПЛК ЕТ-XX	ПЛК ЕТ-XX Lite
Центральный процессор	Архитектура ARM Cortex M4 GD32F450ZKT6	
Частота процессора	120 Mhz	
Объем оперативной памяти RAM	65 KB	
Объем энергонезависимой памяти (EEPROM)	2 MB	
Объем памяти для программы	256 KB	
Хранение переменных (FRAM)	2 KB	
Количество сокетов	1	
Время выполнения пустого цикла	Установленное по умолчанию (стабилизированное) – 10 мс (настраивается в окне "Настройки контроллера") «UpakNet». Настоятельно не рекомендуется устанавливать время цикла, равное 0-4 мс	
Встроенные часы реального времени	1 шт.	
Часы реального времени с собственным батарейным питанием. Погрешность хода, не более: при температуре плюс 25 °С 5 с в сутки при температуре минус 40 °С	5 с в сутки 20 с в сутки	

Таблица 2.7

Заводские сетевые настройки	
IP-адрес	192. 168. 80. 80
Маска IP-адреса	255.255.255.0

Таблица 2.8

Общие сведения	
Габаритные размеры, не более	(99,5 x 22,7 x 112) ± 1 мм
Степень защиты корпуса	IP20
Индикация на передней панели Индикация обмена: Индикация выполнения программы:	LED (светодиодная) Да (Желт. Зел.) Да (Красный)
Средняя наработка на отказ*	80000 ч.
Средний срок службы	15 лет.
Вид монтажа	Монтаж на DIN-рейку.

Таблица 2.9

Интерфейсы связи и программирования					
Интерфейсы связи	Протоколы (тип связи и особенности работы)	Формат передачи данных	Скорости передачи*	Длина кабеля**, не более	Тип рекомендуемого кабеля
RS-485 (Uart1)(X1), (Uart2, Uart3)(XT6)	ModBus-RTU	конфигурация 8N1	1200, 9600, 19200, 57600, 115200 бит/с	100 м	КИПЭВ 1 × 2 × 0,6 ТУ 16.К99-008 или аналогичный (витые пары в экране)
Ethernet 100 Base-T (X3)	Modbus TCP	–	10, 100 Мбит/с	100 м	Категория 5 тип UTP (витые пары без экрана), STP или FTP (витые пары в экране)
RS-485 Ethernet (X4) (Uart0"pin1,2), (Uart4"pin3,6)	ModBus-RTU	конфигурация 8N1	1200, 9600, 19200, 57600, 115200 бит/с	100-1000 м	КИПЭВ 1 × 2 × 0,6 ТУ 16.К99-008 или аналогичный (витые пары в экране)
* Критерий правильного функционирования интерфейсов связи контроллера – не более 1 % ошибок на любой из скоростей. ** Максимальная длина зависит от скорости обмена.					

Таблица 2.10

Ток подключения интерфейсов		
Интерфейс	Максимальный отдаваемый предельный ток	Максимальный принимаемый предельный ток
RS-485	90 мА	20 мкА

Изоляция узлов прибора

Типы изоляции представлены в таблице 3.1- 3.8

Таблица 3.1

Тип	Описание
Основная (О)	Изоляция для защиты частей оборудования, находящихся под напряжением, от поражения электрическим током. Электрическая прочность основной изоляции прибора проверяется типовыми испытаниями: приложением испытательного переменного напряжения, величина которого различна для различных цепей прибора
Двойная (Д)	Включает основную и дополнительные изоляции
Функциональная (Ф)	Изоляция для исправной работы оборудования. Не обеспечивает защиту от поражения электрическим током. Электрическая прочность функциональной изоляции прибора проверяется приложением испытательного переменного напряжения 1000 В (действующее значение)

ПРИМЕЧАНИЕ
Время испытания – 1 минута.

Таблица 3.2

Принадлежность к типу изоляции			
Входы/выходы	**Основная**	**Функциональная**	**Двойная**
DO (XT4/XT5)		(Ф)	
DI (XT2)	(О)		
AO (XT3)		(Ф)	
AI (XT1)	(О)		
RS-485 (XT6)			(Д)
Пит. +24 В и (RS-485)(X1)	(О)		(Д)
Ethernet (X3)		(Ф)	
RS-485 Ethernet (X4) в составе Пит. +24 В			(Д)

Таблица 3.3

Параметр	**Значения (свойства)**
Питание ПЛК ЕТ-ХХ и ЕТ-ХХ Lite	
Гальваническая развязка	Между выводами
	Между группами остальных цепей
Электрическая прочность изоляции	500 В (Д)

Таблица 3.4

Параметр	**Значения (свойства)**
Цифровые (дискретные) входы	
Гальваническая развязка	Между выводами
	Между группами (входных сигналов)
Электрическая прочность изоляции	500 В (О)

Таблица 3.5

Параметр	**Значения (свойства)**
Дискретные выходы (открытый коллектор)	
Гальваническая развязка	Групповая между выводами
	Между группами (выходных сигналов)
Электрическая прочность изоляции	500 В (Ф)

Таблица 3.6

Параметр	**Значения (свойства)**
Аналоговые входы	
Гальваническая развязка	Между выводами
	Между группами (входных сигналов)
Электрическая прочность изоляции	500 В (О)

Таблица 3.7

Параметр	**Значения (свойства)**
Аналоговые выходы	
Гальваническая развязка	Групповая между выводами
	Между группами (выходных сигналов)
Электрическая прочность изоляции	500 В (Ф)

Таблица 3.8

Параметр	**Значения (свойства)**					
Интерфейсы связи						
Интерфейсы	Ethernet	RS-485(0)	RS-485(1)	RS-485(2)	RS-485(3)	RS-485(4)
Гальваническая развязка	Групповая между выводами	Между выводами				
	Между группами остальных цепей	Между группами (интерфейса). Между группами остальных цепей				
Электрическая прочность изоляции	500 В (Ф)	1500 В (Д)	1500 В (Д)	1500 В (Д)	1500 В (Д)	1500 В (Д)

Условия эксплуатации

Условия эксплуатации ПЛК ET-XX соответствуют требованиям ГОСТ IEC 61131-2.
Прибор соответствует второму классу электробезопасности в соответствии с ГОСТ IEC61131-2.
Прибор предназначен для эксплуатации в следующих условиях:

- закрытые взрывобезопасные помещения или шкафы электрооборудования без агрессивных паров и газов;
- температура окружающего воздуха от минус 40 до плюс 55 °С;
- относительная влажность от 10 % до 95 % при плюс 35 °С (без образования конденсации);
- высота над уровнем моря не более 2000 м;
- допустимая степень загрязнения 1 (несущественные загрязнения или наличие только сухих
непроводящих загрязнений).

По устойчивости к климатическим воздействиям во время эксплуатации прибор соответствует группе исполнения В4 ГОСТ Р 52931.
По устойчивости к механическим воздействиям при эксплуатации прибор соответствует группе исполнения N2 ГОСТ Р 52931 (частота вибрации от 10 до 55 Гц).
По устойчивости к воспламенению и распространению пламени FV1 корпус прибора соответствует ГОСТ Р 51841.

Помехоустойчивость и помехоэмиссия

Прибор отвечает требованиям по устойчивости к воздействию помех в соответствии с ГОСТ IEC 61131-2.

По уровню излучения радиопомех (помехоэмиссии) контроллер соответствует нормам, установленным для оборудования класса Б по ГОСТ 30805.22.

Прибор устойчив к колебаниям и провалам напряжения питания:

• для переменного тока в соответствии с требованиями ГОСТ 30804.4.11;

• для постоянного тока в соответствии с ГОСТ IEC 61131-2 длительность прерывания до 10мс

включительно, длительность интервала от 1 с и более.

По устойчивости к электромагнитным воздействиям и по уровню излучаемых радиопомех прибор соответствует оборудованию класса А по ГОСТ Р 51522.

Прибор устойчив к воздушному электростатическому разряду ± 8 кВ.

Прибор устойчив к радиочастотному электромагнитному полю напряженностью до 10 В/м в полосе частот от 80 до 1000 МГц.

Устройство и особенности конструкции

Конструкция

Контроллер ПЛК ET-XX выпускается в конструктивном исполнении для крепления на DIN-рейке 35 мм.

Параметры корпуса и выводов ПЛК ET-XX представлены на рисунка 1.1-1.4.

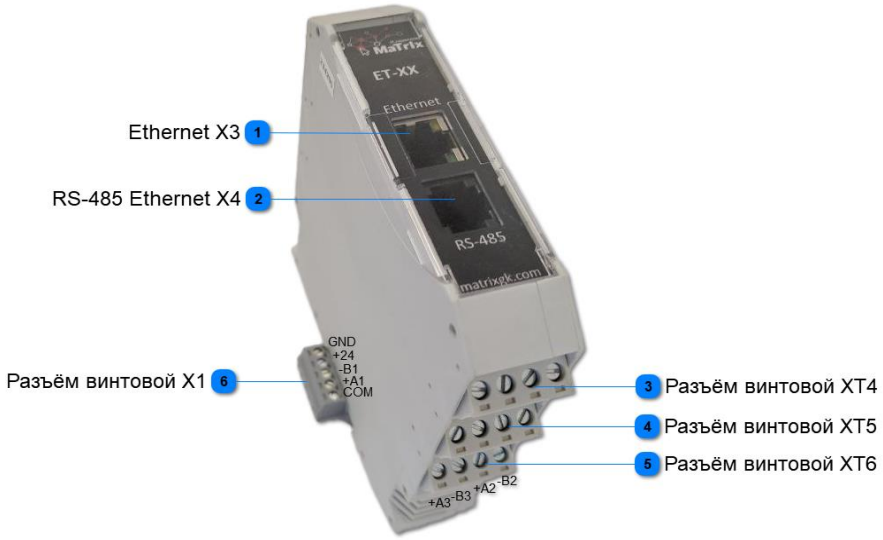
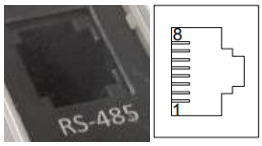


Рисунок 1.1 ПЛК ET-XX Вводы/выводы

1 Ethernet X3

RJ45 - сетевой разъём связи с пользовательской программой UpakNet.

2 **RS-485 Ethernet X4**



RJ45 - сетевой разъём имеет в составе два интерфейса RS-485 и наличие питания +24В

- ("pin8" **GND**);
- ("pin7" **GND**);
- ("pin6" **-B4**);
- ("pin5" **+24В**);
- ("pin4" **+24В**);
- ("pin3" **+A4**);
- ("pin2" **-B0**);
- ("pin1"**+A0**).

3 **Разъём винтовой XT4**

DO - вывод дискретных сигналов (нумерация слева направо от разъема питания X1)

4 **Разъём винтовой XT5**

DO - вывод дискретных сигналов (нумерация слева направо от разъема питания X1)

5 **Разъём винтовой XT6**

RS-485 - интерфейсы связи (+A3, -B3) (+A2, -B2) (нумерация справа налево от выводов шины X1)

6 **Разъём винтовой X1**

Разъём шины питания и основного интерфейса связи RS-485 (GND, +24, -B1, +A1, COM) (нумерация сверху вниз)

Рисунок 1.2 ПЛК ET-XX Вводы/выводы

1 Выводы основной шины

Выводы шины питания и основного интерфейса RS-485 предназначены для присоединения дополнительных модулей расширения (вводы/выводы) с аналогичной шиной.

2 Разъём винтовой XT1

AI - ввод аналоговых сигналов (нумерация слева направо от разъема питания X1)

3 Разъём винтовой XT2

DI - ввод цифровых сигналов (нумерация слева направо от разъема питания X1)

4 Разъём винтовой XT3

AO - вывод аналоговых сигналов (нумерация слева направо от разъема питания X1)

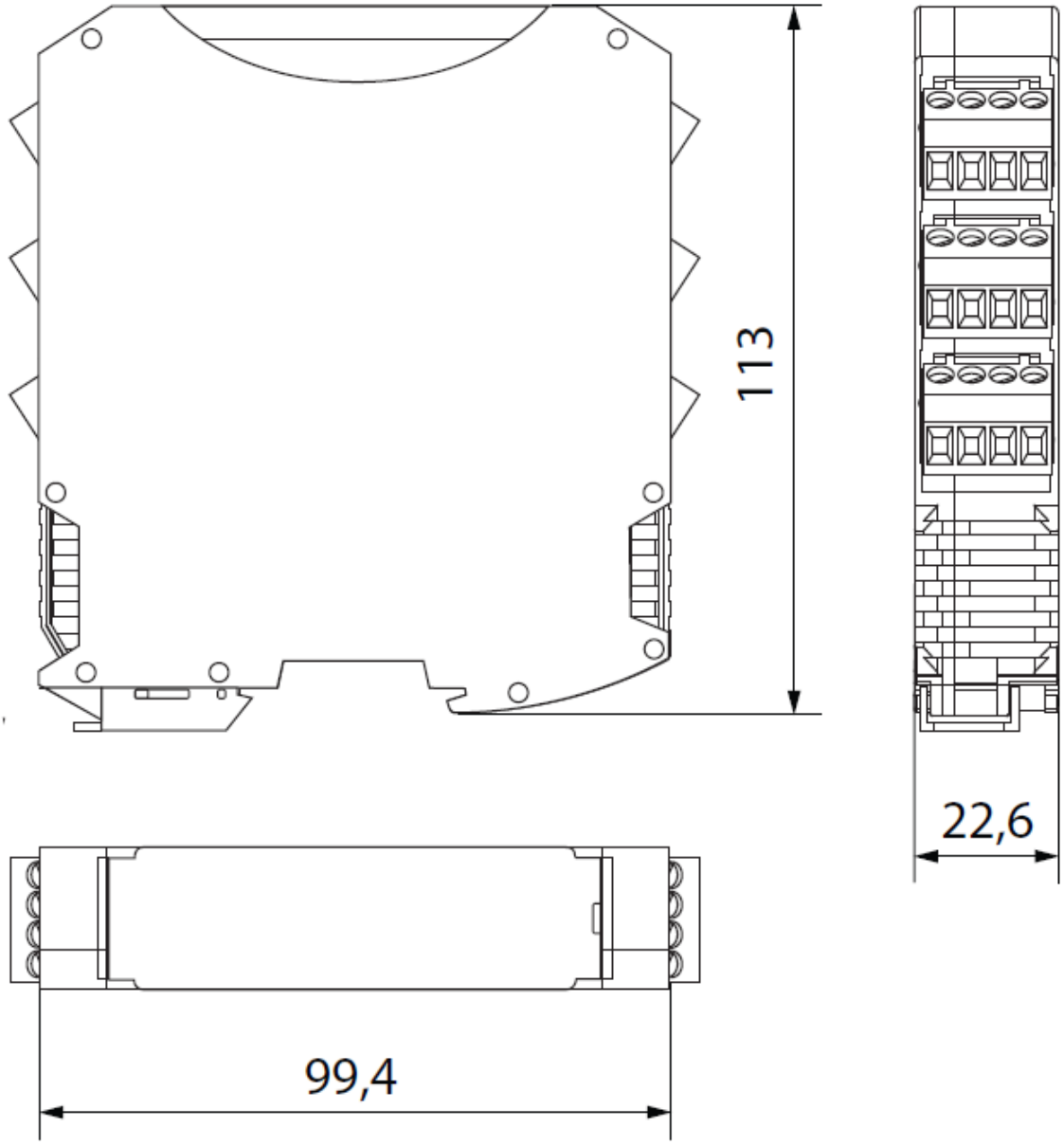


Рисунок 1.3 ПЛК ET-XX Габаритные размеры корпуса прибора

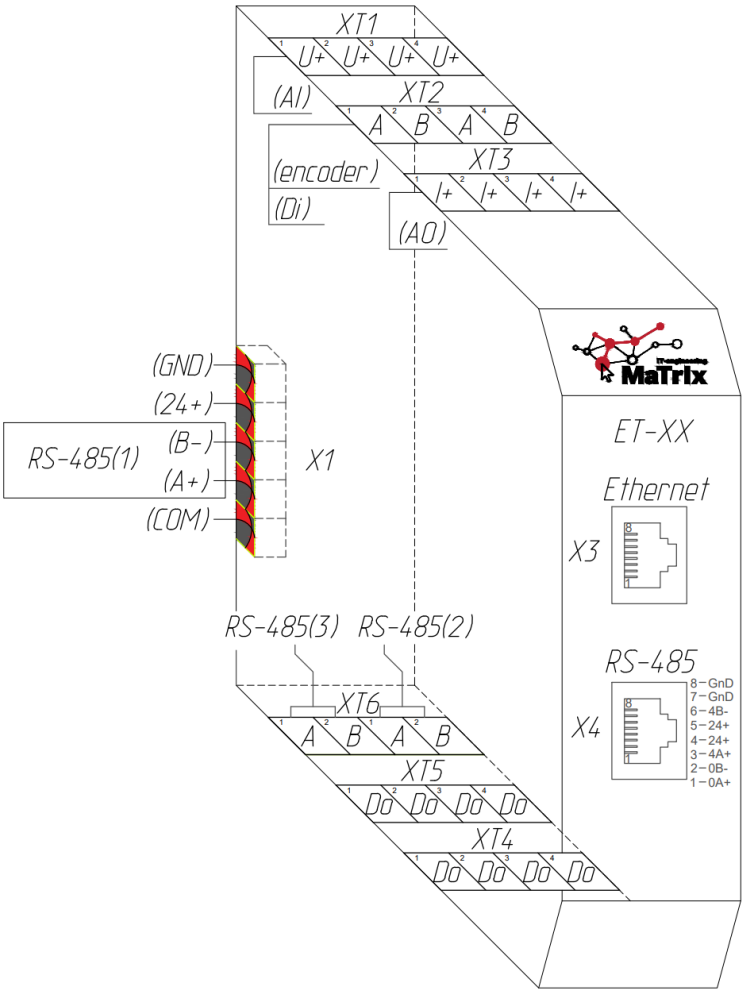


Рисунок 1.4 ПЛК ET-XX Схема структурная

Цифровые входы

Прибор имеет 4 цифровых (дискретных) входов. Обработка значений с входов осуществляется пользовательской программой контроллера.

Входы DI1– DI4 можно использовать в качестве счетчиков импульсов, на работу с энкодерами или перевести в режим обработки по прерыванию высокочастотного таймера.

Максимальные частоты входных сигналов, которые могут воспринимать эти входы, приведены в таблице 5.1, характеристики входов в таблице 2.3

Таблица 5.1

Режим работы дискретного входа	**Минимальная длит-ость импульса, воспринимаемого дискретным входом**	**Комментарий**
Программная обработка	5000 мкс (200 Гц)	Определяется длительностью цикла прибора
Счетчик импульсов	5 мкс (100 кГц)	При коэффициенте заполнения 0,5 (50 %) и отключенном фильтре
Энкодер	5 мкс (до 100 кГц*)	При коэффициенте заполнения 0,5 (50 %) и отключенном фильтре
* Частота на контактах энкодера 1A, 1B и 2A.		

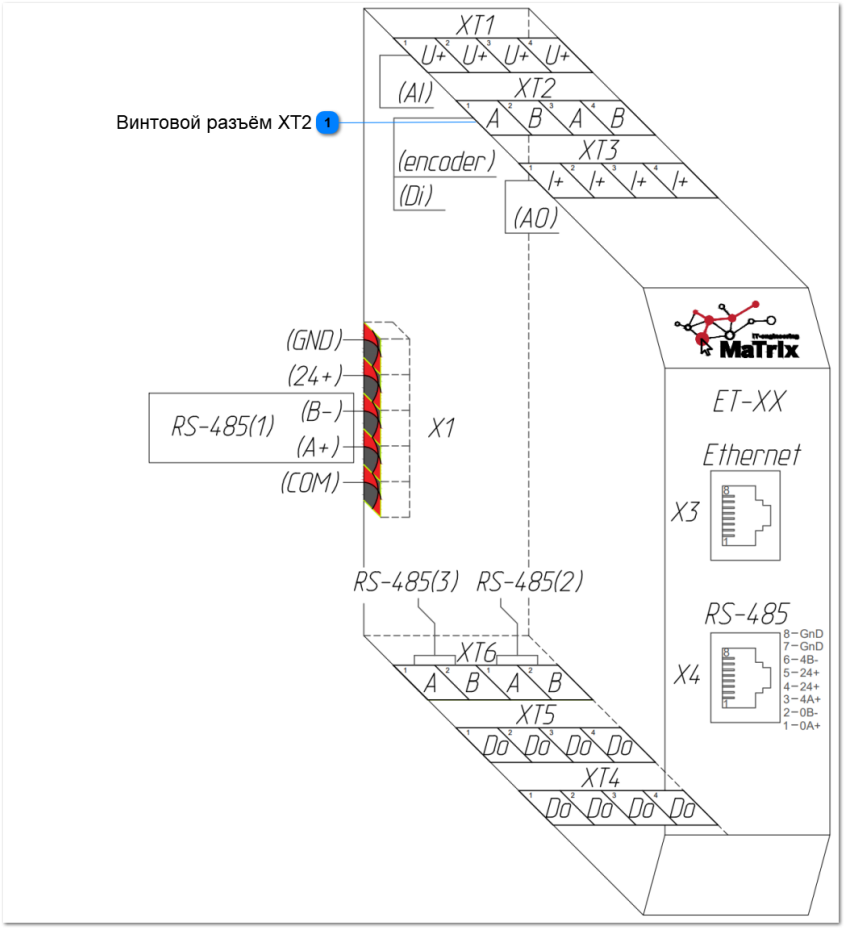


Рисунок 5.1 ПЛК ET-XX Схема структурная.

1 **Винтовой разъём XT2**

DI - ввод цифровых сигналов энкодера или дискретный сигналов +24 В.

Цифровые выходы

Прибор имеет 8 цифровых (дискретных) выходов типа NPN открытый коллектор.
Выходы управляются пользовательской программой UpakNet.
Частота изменения сигнала не более 0.25 мкс. определяется длительностью цикла ПЛК.
Максимальные частоты выходных сигналов приведены в таблице 2.2

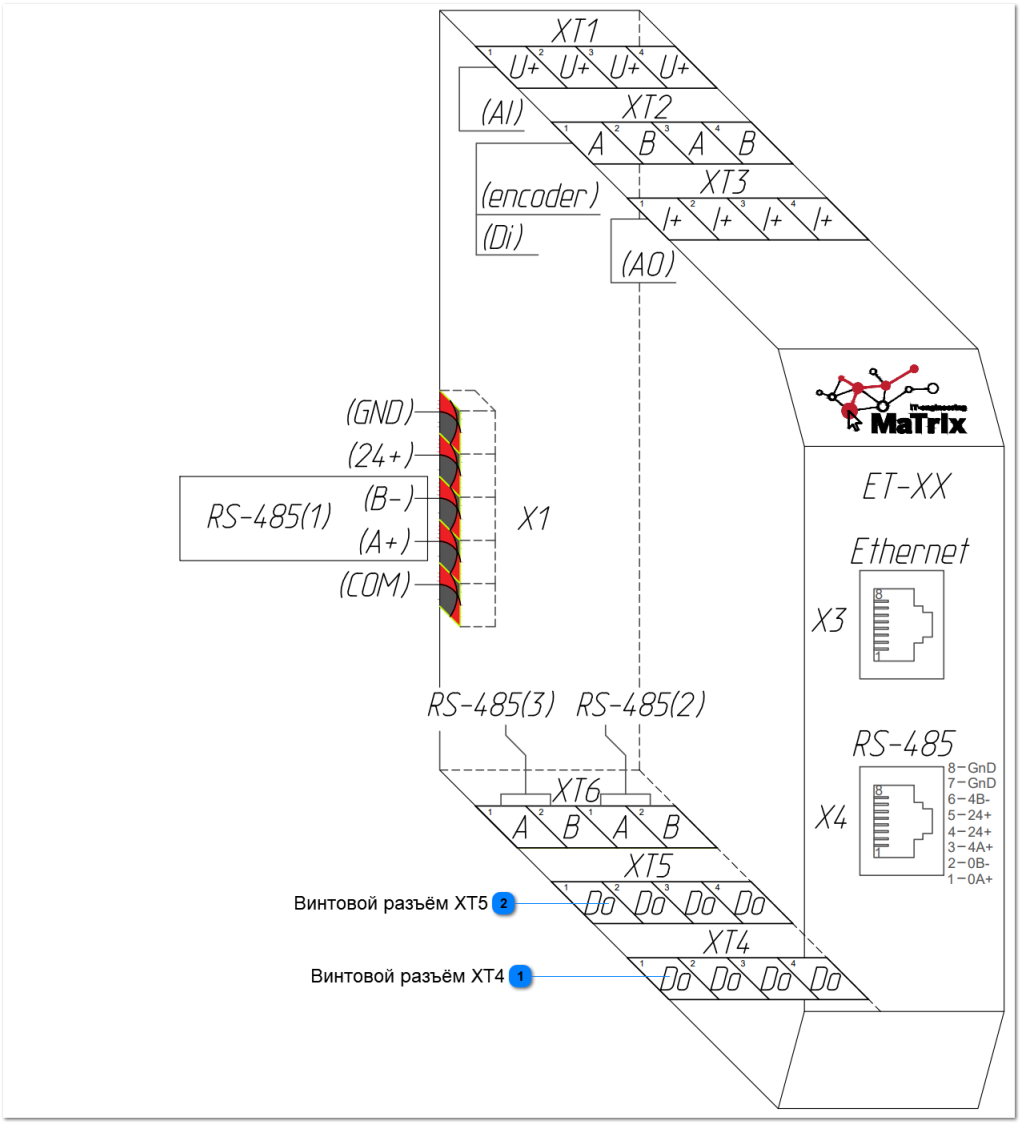


Рисунок 6.1 ПЛК ET-XX Схема структурная.

1 **Винтовой разъём XT4**

DO - вывод дискретных сигналов NPN открытый коллектор -24 В.

2 **Винтовой разъём XT5**

DO - вывод дискретных сигналов NPN открытый коллектор -24 В.

Аналоговые входы

Прибор имеет 4 аналоговых входа. Значения с входов обрабатываются пользовательской
программой UpakNet . Входы можно настроить независимо друг от друга на работу в одном из следующих режимов: измерение напряжения от 0 до 10 В.
Период обновления результатов измерения по каждому входу равен 10 мс.
Результаты измерения каждого канала можно отфильтровать независимо друг от друга помощью цифровых фильтров. Более подробные сведения о настройке аналоговых входов приведены в РП UpakNet, характеристики входов таблица 2.5.

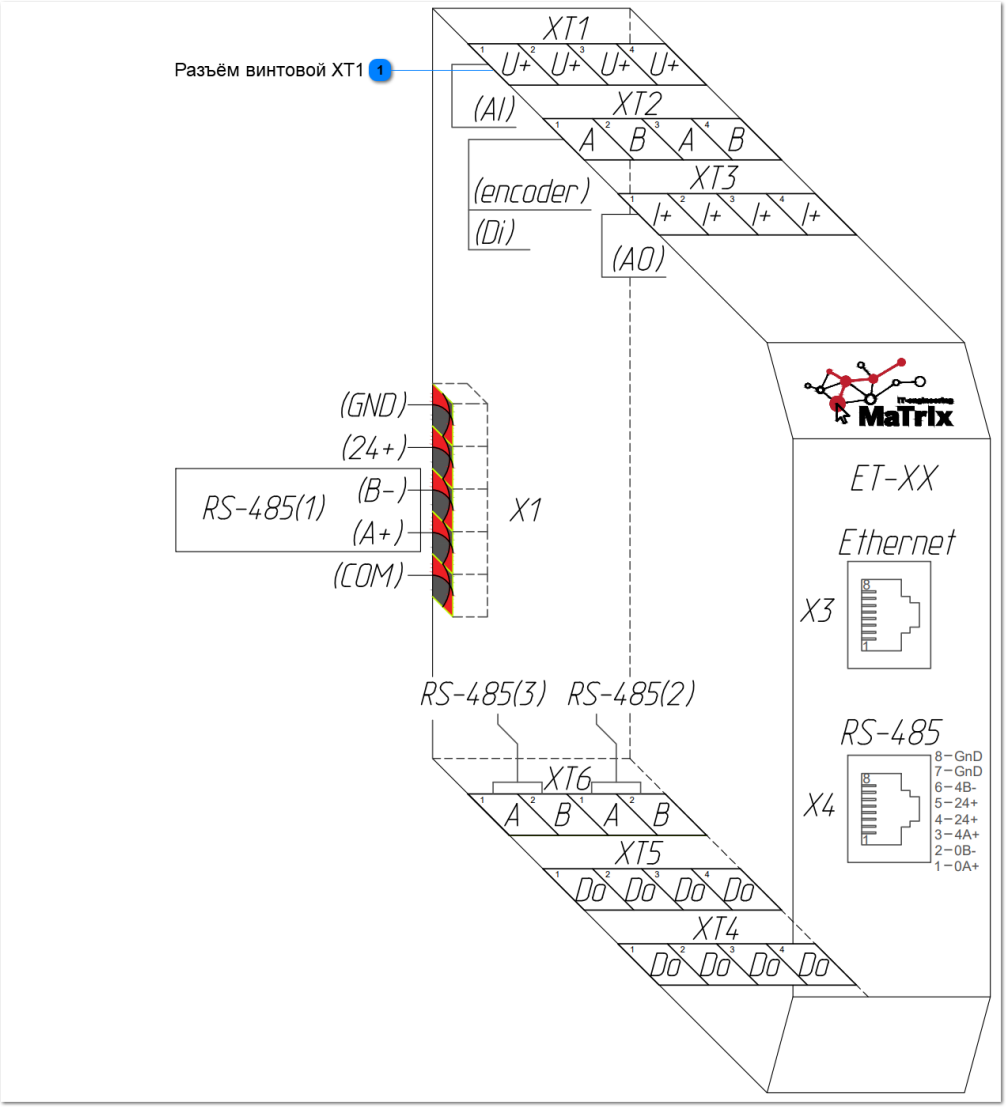


Рисунок 7.1 ПЛК ET-XX Схема структурная.

1 **Разъём винтовой XT1**

AI - ввод аналоговых сигналов 0-10 В.

Аналоговые выходы

Прибор имеет 4 аналоговых выхода ток от 4 до 20 мА.

Более подробные сведения о настройке аналоговых выходов приведены в РП UpakNet, характеристики выходов таблица 2.4.

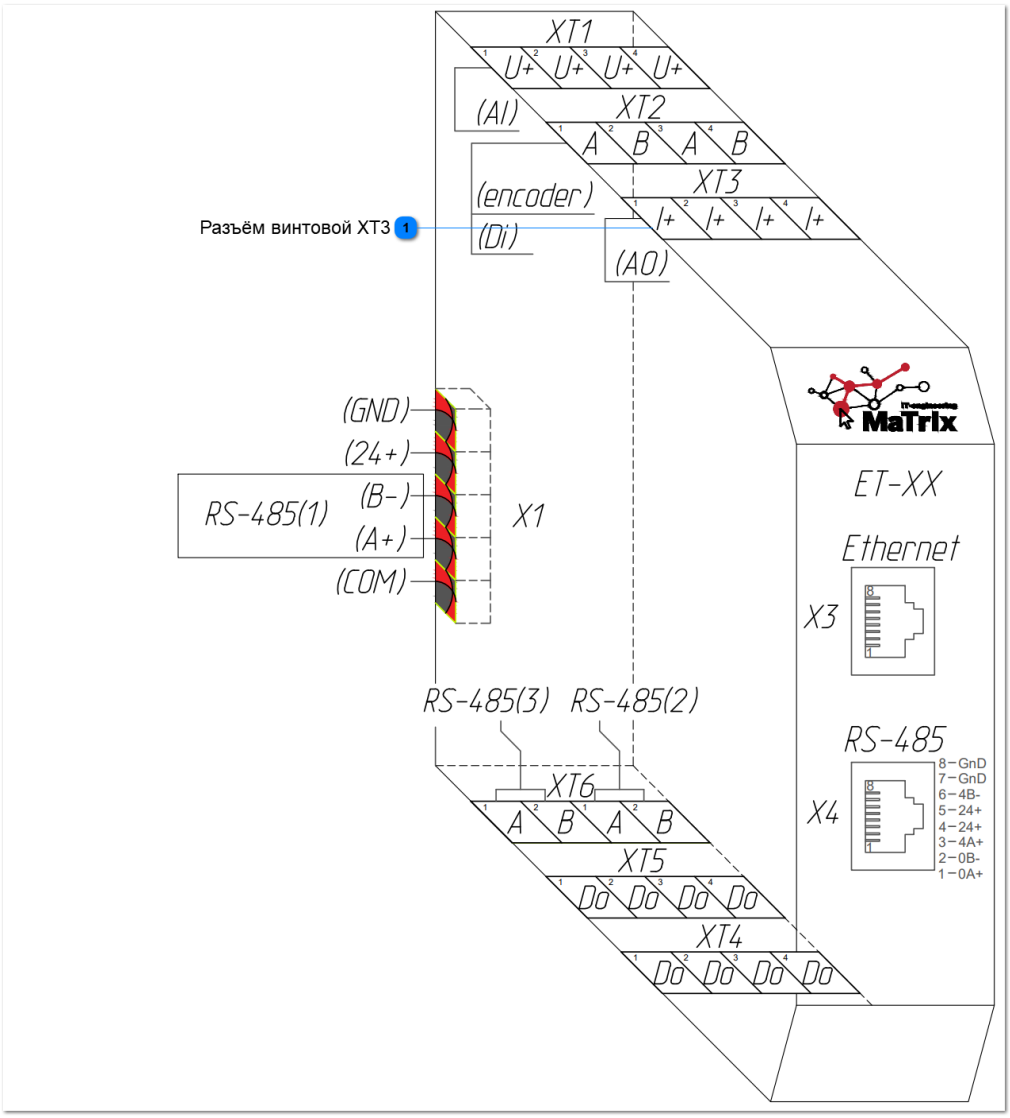


Рисунок 8.1 ПЛК ET-XX кроме ПЛК ET-XX Lite Схема структурная.

1 **Разъём винтовой XT3**

AO - вывод аналоговых сигналов 4-20мА.

Индикация

На передней панели прибора по месту коммутации порта (Ethernet) светодиоды показывают следующую информацию.

Список 9.1

Светодиоды (свойства) Ethernet

• Зеленый - 100 Мбит/с светодиод горит,10 Мбит/с светодиод не горит;

• Оранжевый - передача данных светодиод горит;

• Красный - инициализация медленное мерцание, программа остановлена медленное мерцание спустя 10 секунд со старта, программа выполняется быстрое мерцание.

Ethernet

Контроллер оснащён коммуникационным портом Ethernet 100 Мбит/с, который обеспечивает сетевое взаимодействие контроллера с другими устройствами.
Коммуникационный стандарт **Стандарт EIA/TIA 568B** кабель для подключения витая пара именуемый в дальнейшей "Патч-корд RJ45 Male".
Контакты порта Ethernet показаны на рисунке 10.1. Назначение контактов описано в таблице 10.1.

Рисунок 10.1

Таблица 10.1

№ контакта	Описание
8	Unused
7	Unused
6	RX-
5	Unused
4	Unused
3	RX+
2	TX-
1	TX+

Интерфейс RS-485

В приборе есть один интерфейс RS-485 для связи последовательных устройств по протоколам Modbus RTU, в режимах Master и Slave.
Для соединения приборов по интерфейсу RS-485 применяется экранированная витая пара проводов, согласно требованиям таблицы 2.9.
Общая длина линии RS-485 не должна превышать 1000 м.
Линии связи следует подключать с соблюдением полярности. Линия связи А подключается к клемме А прибора, аналогично подключается линия связи В к клемме В.
Кабель для подключения витая пара именуемый в дальнейшей "Патч-корд RJ45 Male".
Максимальная нагрузка сквозной линии питания разъёма "J45 Famele" (max 50 мА).
Подробную схему подключения см. в разделе "Подключение устройств ПЛК ET-XX".
Контакты порта RS-485 Ethernet показаны на рисунке 11.1. Назначение контактов описано в таблице 11.1.

Рисунок 11.1

Таблица 11.1

№ контакта	Описание
8	GND (max 50 мА)
7	GND (max 50 мА)
6	RS-485(4) -B
5	24+ (max 50 мА)
4	24+ (max 50 мА)
3	RS-485(4) +A
2	RS-485(0) -B
1	RS-485(0) +A

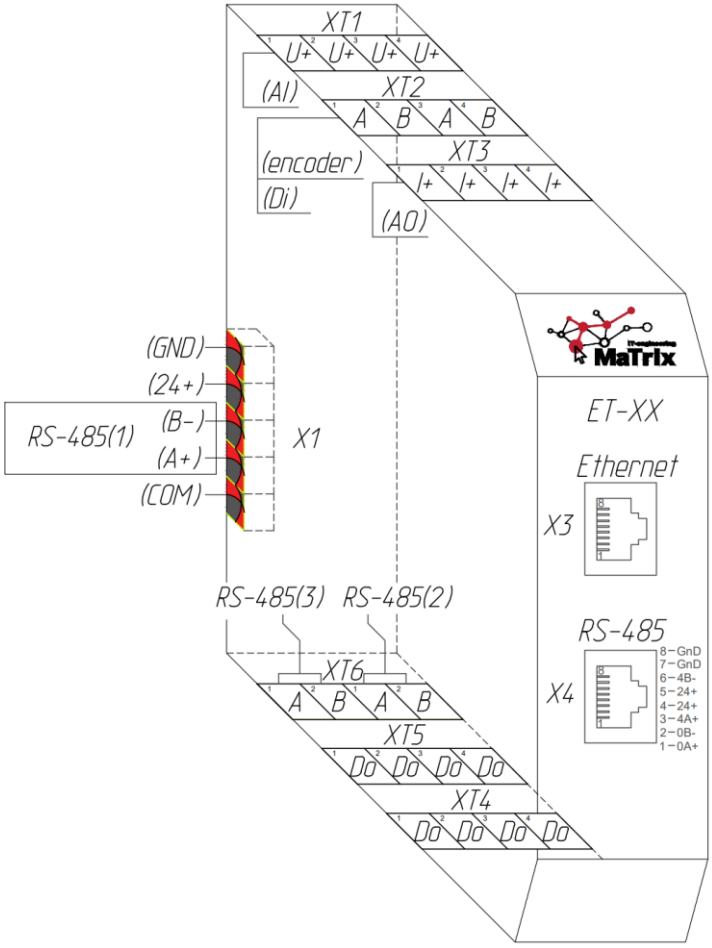


Рисунок 11.2ПЛК ET-XX Схема структурная.

Часы реального времени

Прибор оснащен встроенными часами реального времени (RTC), которые могут питаться от батареи. Энергии полностью заряженной батареи хватает на непрерывную работу часов реального времени в течение 15 лет.
В случае эксплуатации контроллера при температуре на границах рабочего диапазона время работы часов сокращается.

Меры безопасности

По способу защиты от поражения электрическим током контроллер ПЛК160-24.X(M02) соответствует классу III, а ПЛК160-220. X (M02) соответствует классу II по ГОСТ 12.2.007.0-75. При работе с контроллером ПЛК160-24. X (M02) согласно ГОСТ Р 51841 следует использовать только источник питания со сверхнизким безопасным напряжением.

Во время эксплуатации и технического обслуживания прибора следует соблюдать требования

ГОСТ 12.3.019-80, «Правил эксплуатации электроустановок потребителей» и «Правил охраны труда при эксплуатации электроустановок потребителей».

Открытые контакты клемм прибора во время эксплуатации находятся под напряжением величиной до 250 В. Любые подключения к прибору и работы по его техническому обслуживанию следует производить только при отключенном питании контроллера и подключенных к нему исполнительных механизмов.

Не допускается попадание влаги на контакты выходных соединителей и внутренние элементы

контроллера. Запрещено использовать прибор при наличии в атмосфере кислот, щелочей, масел и иных агрессивных веществ.

В случае применения прибора на объектах, подконтрольных Федеральной службе по экологическому, технологическому и атомному надзору (ФСЭТАН), объектах органов безопасности и охраны правопорядка или иных объектах, потенциально представляющих опасность для жизни и здоровья окружающих, требуется обязательная защита паролем ПЛК.

Требования к паролю:

• длина пароля должна составлять не менее 8 символов и не более 32 символов;

• пароль должен содержать буквы латинского алфавита и цифры.

Пароль рекомендуется менять не реже 1 раза в 3 месяца. Не допускается подключать прибор к

локальной сети Ethernet с выходом в сеть Internet без обеспечения надежных средств межсетевого экранирования. Физический доступ к прибору должен быть разрешен только квалифицированному обслуживающему персоналу.

Монтаж

Установка контроллера

Во время монтажа прибора следует учитывать меры безопасности из раздела "Меры безопасности". Для обеспечения электробезопасности при монтаже прибора следует руководствоваться подразделом "Изоляция узлов прибора".
Перед монтажом прибора следует подготовить место в шкафу электрооборудования. Конструкция шкафа должна защищать прибор от попадания в него влаги, грязи и посторонних предметов.
Контроллер закрепляется на DIN-рейку или внутреннюю стену шкафа защелками вверх.

Установка на DIN-рейке

Для установки прибора на DIN-рейке следует:
Подготовить на DIN-рейке место для установки прибора в соответствии с размерами, предварительно установить шину питания и интерфейса RS-485. Количество приборов может быть увеличено с помощью дополнительного подсоединения шины.
Установить прибор на DIN-рейку в соответствии с рисунком 12.4 по стрелке (1) и с усилием прижать его к DIN-рейке в направлении стрелки (2) до фиксации защелки.
Степень затяжки - Белых винтовых разъёмов (0.4±1 N/m).
Степень затяжки - Серых винтовых разъёмов (0.2±1 N/m).

• **Рисунок 12.1** - Первая точка установки шины под номером (1) устанавливается на кромку DIN-рейки, вторая точка с фиксатором под номером (2) защелкивается с не большим усилием, прижмите точку (2).
• **Рисунок 12.2 и 12.3** - Удерживая шину стрелка(2), сдвигайте шину по направлению стрелка(1) до упора в сторону шины стрелка(2). Серый разъём питания подключить по направлению стрелки (3) придерживая шины (1),(2).
• **Рисунок 12.4 Рисунок 12.5** - Установка прибора на DIN-рейку в соответствии с рисунком по стрелке (1) и с усилием прижать его к DIN-рейке в направлении стрелки (2) до фиксации защелки.
• **Рисунок 12.6** - Снятие прибора и шины с DIN-рейки производится в обратном порядке. Поднимите фиксатор отверткой и снимите прибор.

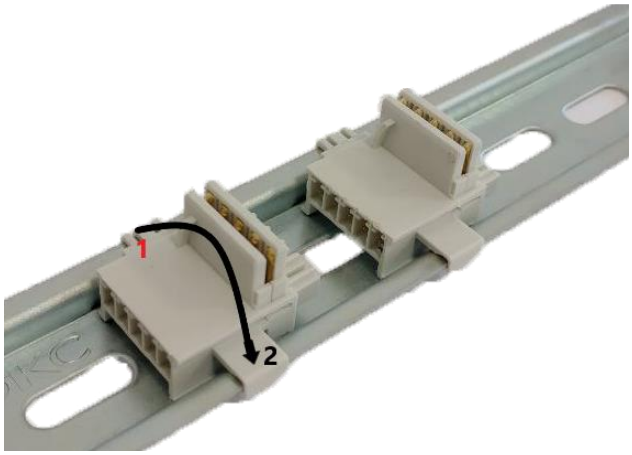


Рисунок 12.1

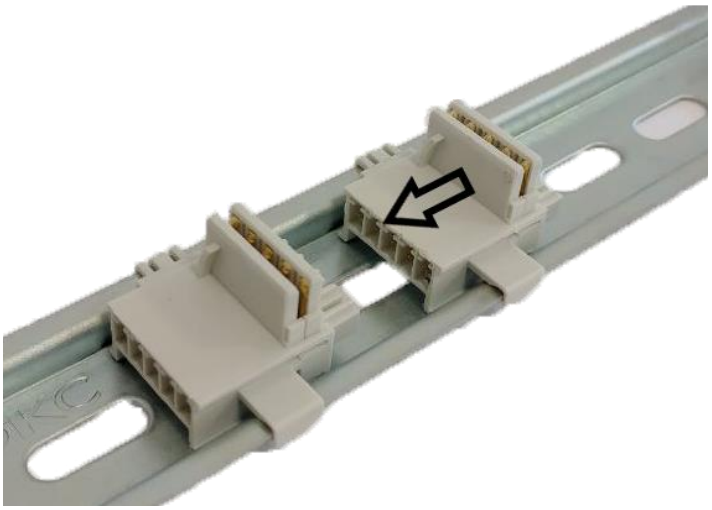


Рисунок 12.2

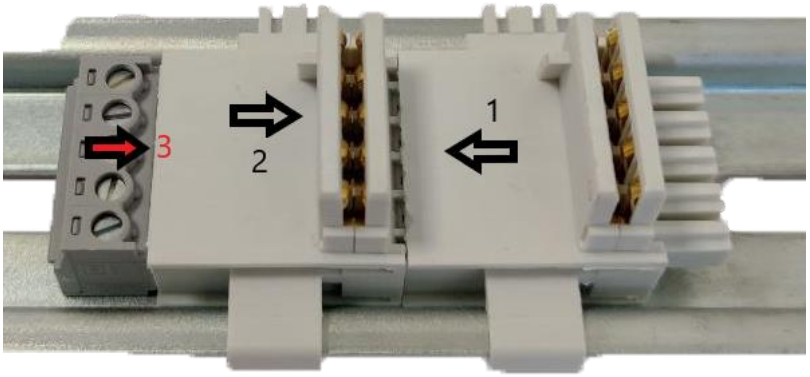


Рисунок 12.3

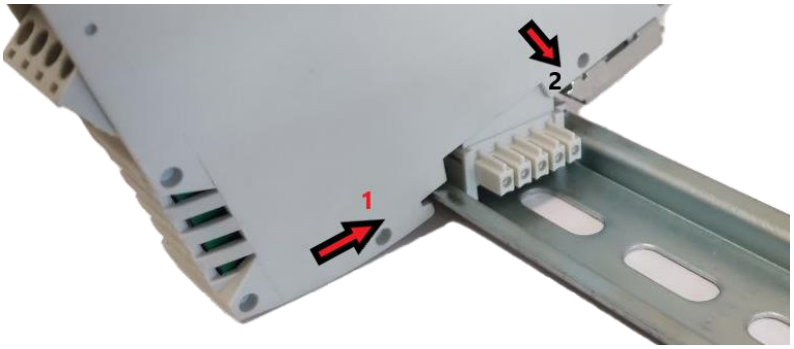


Рисунок 12.4

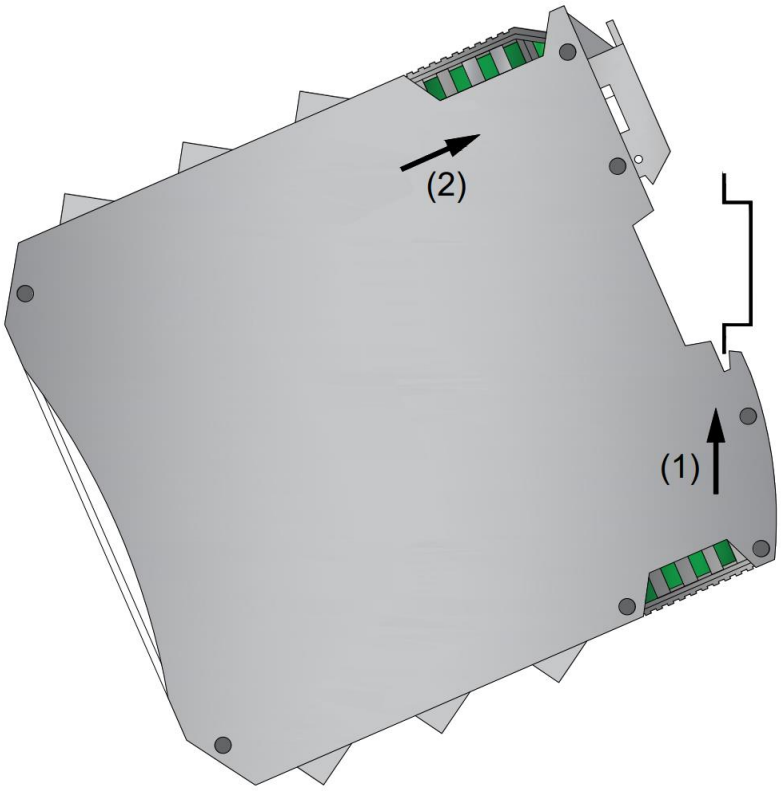


Рисунок 12.5 Схема структурная вид сбоку

Рисунок 12.6

Установка на МП

Для установки прибора на стену следует:

Подготовить место на стене для установки прибора в соответствии с размерами.

Для крепления используется DIN-рейка смонтированная на монтажной панели.

Во время монтажа следует оставить зазоры между стенками и корпусом прибора не менее 30 мм.

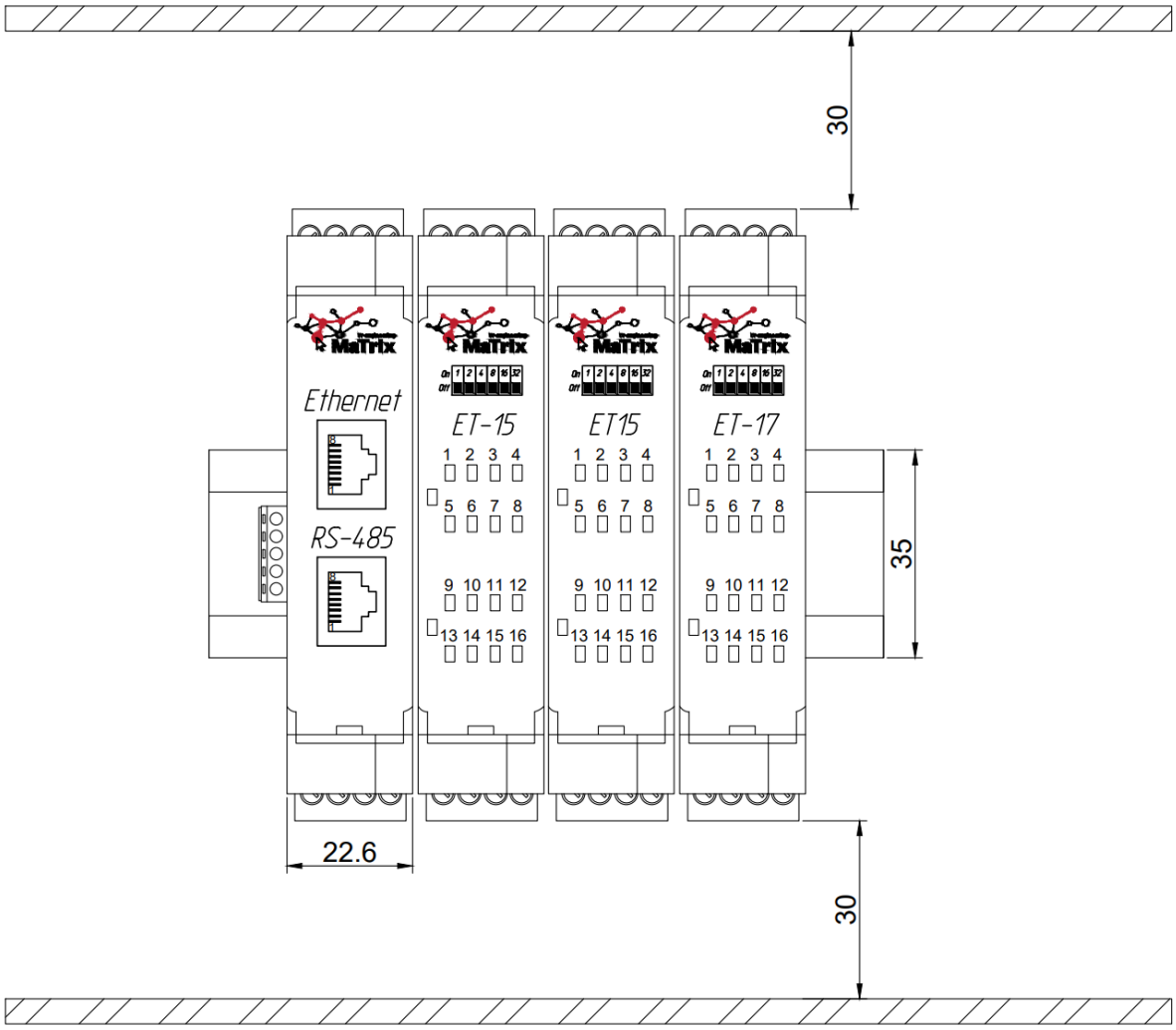


Рисунок 13.1

Подключение

Рекомендации по подключению

Для обеспечения надежности электрических соединений рекомендуется использовать медные многожильные кабели, концы которых перед подключением следует тщательно зачистить, залудить или использовать кабельные наконечники. Жилы кабелей следует зачистить так, чтобы их оголенные концы после подключения к прибору не выступали за пределы клеммника. Сечение жил кабелей должно быть не более 1 мм2.

Общие требования к линиям соединений:

• во время прокладки кабелей следует выделить линии связи, соединяющие прибор с датчиком, в самостоятельную трассу (или несколько трасс), располагая ее (или их) отдельно от силовых кабелей, а также от кабелей, создающих высокочастотные и импульсные помехи;

• для защиты входов прибора от влияния промышленных электромагнитных помех линии связи прибора с датчиком следует экранировать. В качестве экранов могут быть использованы как специальные кабели с экранирующими оплетками, так и заземленные стальные трубы подходящего диаметра. Экраны кабелей с экранирующими оплетками следует подключить к контакту функционального заземления (FE) в щите управления;

• фильтры сетевых помех следует устанавливать в линиях питания прибора;

• искрогасящие фильтры следует устанавливать в линиях коммутации силового оборудования.

При монтаже системы, в которой работает прибор, следует учитывать правила организации
эффективного заземления:

• все заземляющие линии следует прокладывать по схеме «звезда» с обеспечением хорошего контакта с заземляемым элементом;

• все заземляющие цепи должны быть выполнены проводами наибольшего сечения;

Подключение питания

Прибор следует питать от распределенной питающей сети, не связанной непосредственно с питанием мощного силового оборудования. Во внешней цепи рекомендуется установить выключатель, обеспечивающий отключение прибора от сети. Следует использовать автоматический выключатель, рассчитанный на ток 1 А, характеристика С.

Не следует осуществлять питание каких-либо устройств от сетевых контактов контроллера.

Схема подключения питания прибора представлена на рисунке 14.1.

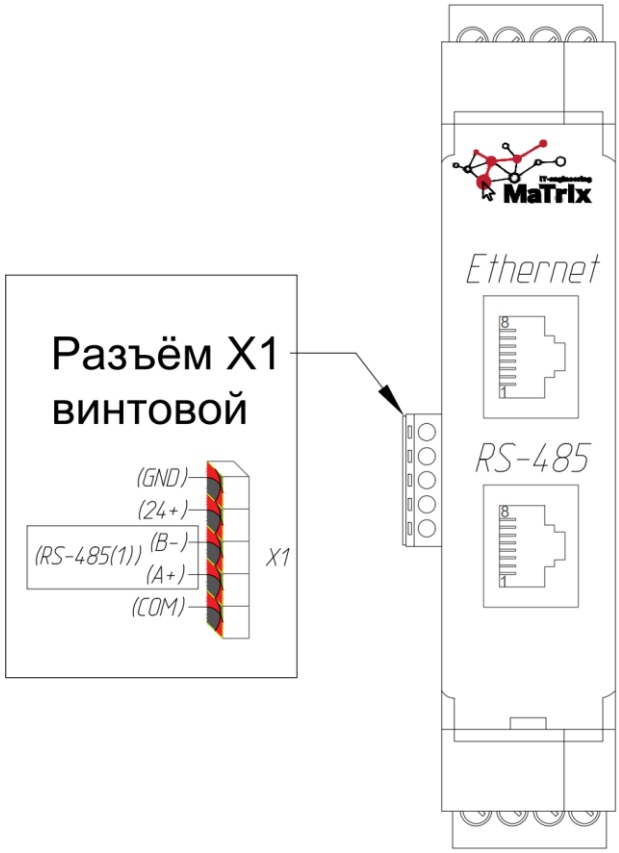


Рисунок 14.1

Подключение входов и выходов

Подключение источников сигналов к дискретным входам, а также подключение исполнительных механизмов к дискретным выходам осуществляются в соответствии со схемами, приведенными на рисунках 15.1-15.4.

Для индуктивных нагрузок, например, для работы с контакторами или магнитными клапанами,

управляемыми постоянным напряжением, следует всегда использовать безынерционные диоды. Эти диоды часто устанавливаются в управляемые устройства заранее. Если диоды не установлены, то их необходимо смонтировать.

Если индуктивные нагрузки включаются релейными выходами с переменным напряжением, следует предусмотреть RC-цепочку, снижающую пиковое напряжение при включении нагрузки и, благодаря этому, защищающую контакты реле от повреждений при искровом разряде.

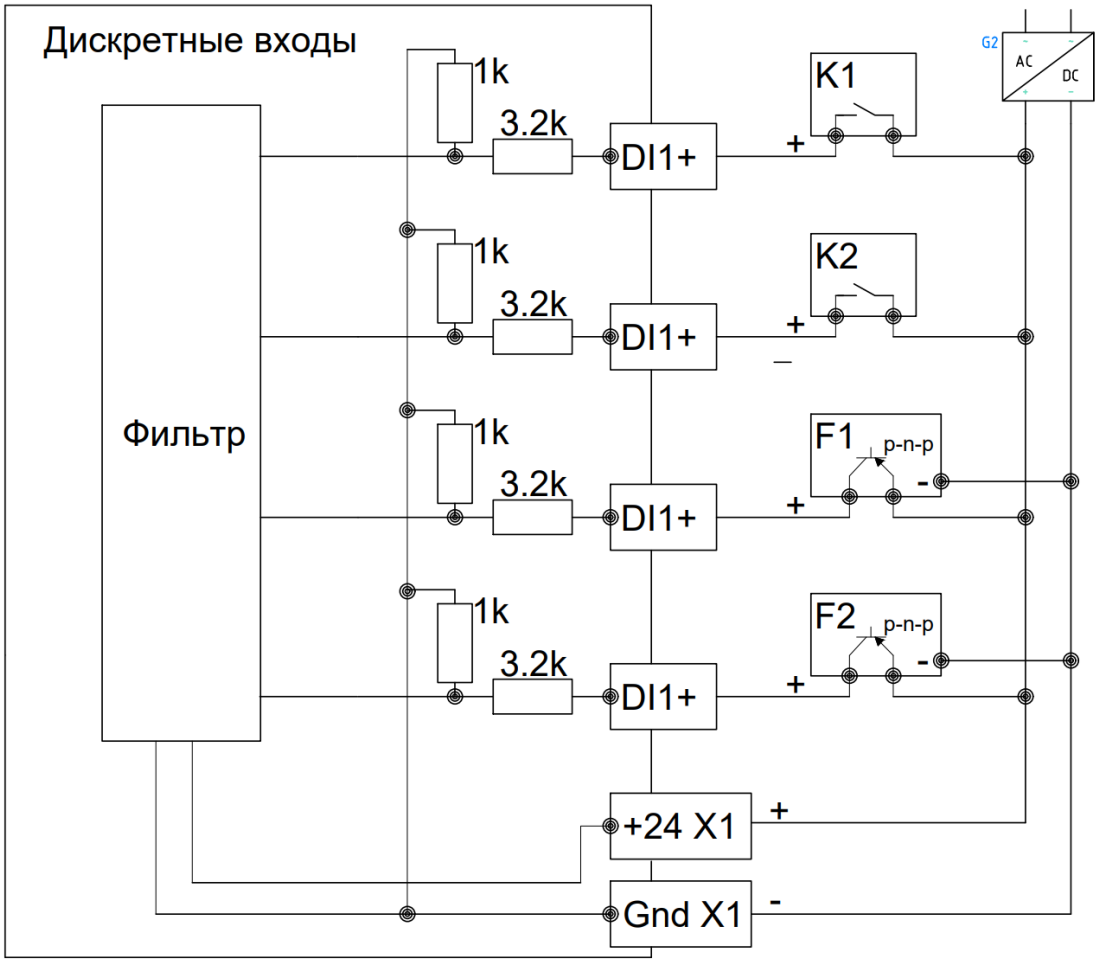


Рисунок 15.1 Дискретный входы.

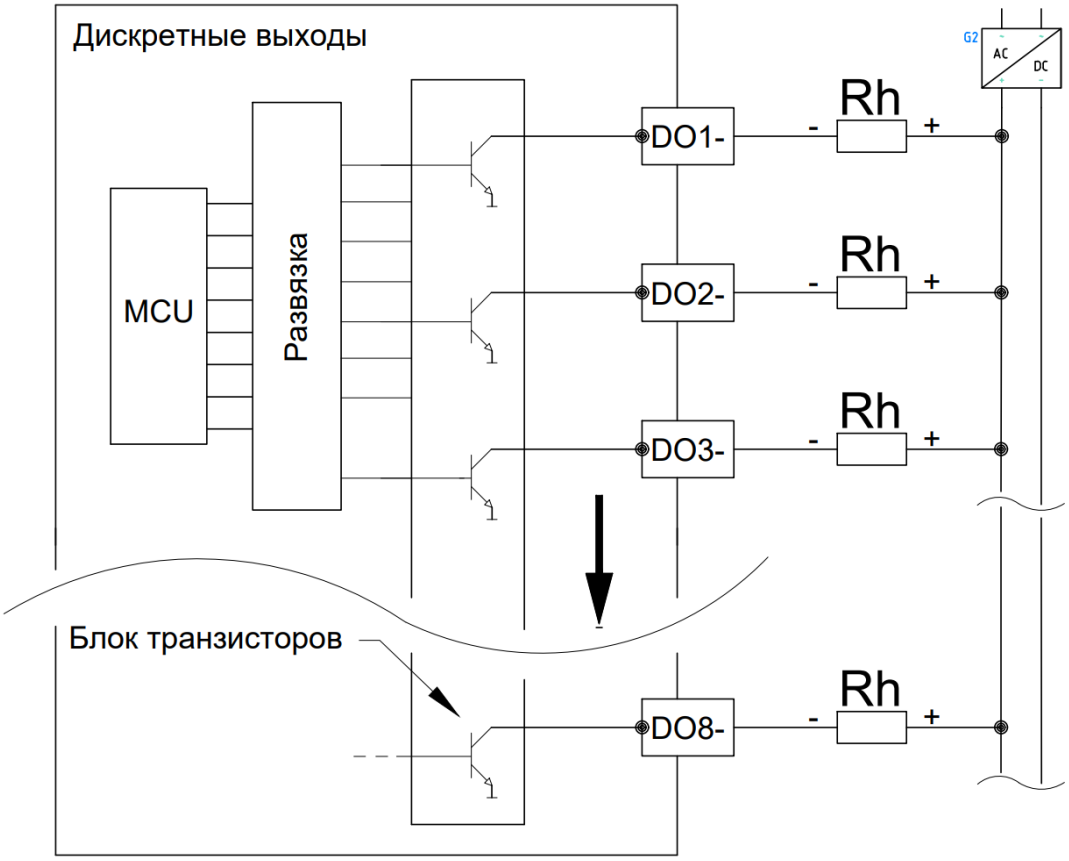


Рисунок 15.1 Дискретный выходы (открытый коллектор)

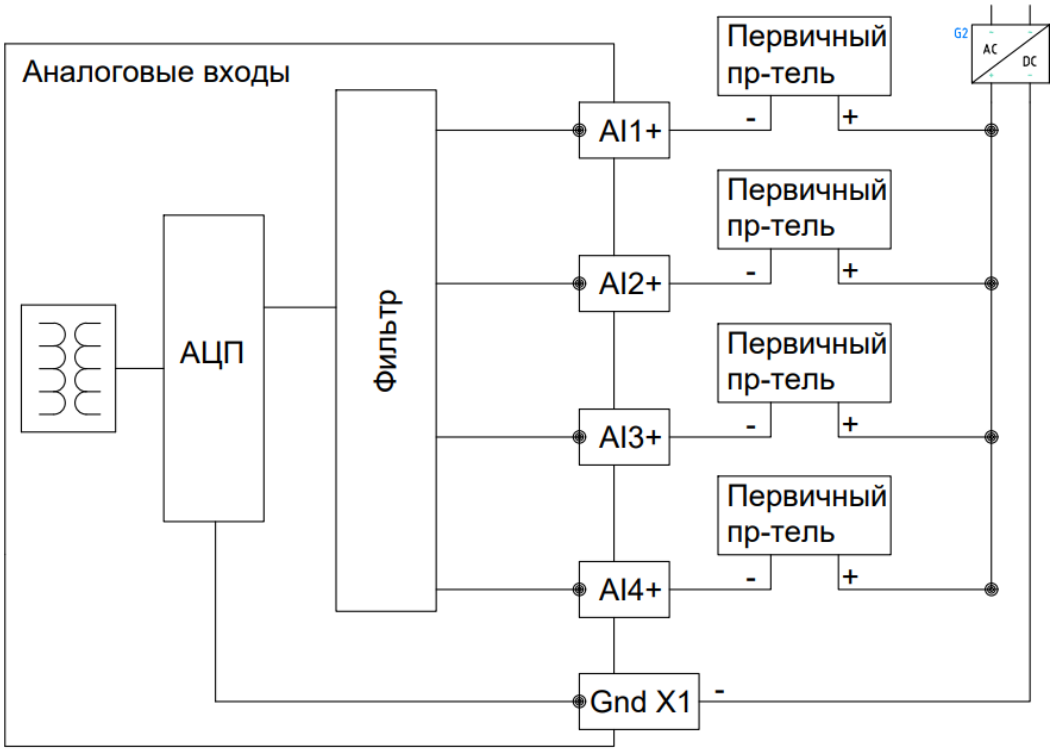


Рисунок 15.1 Аналоговые входы

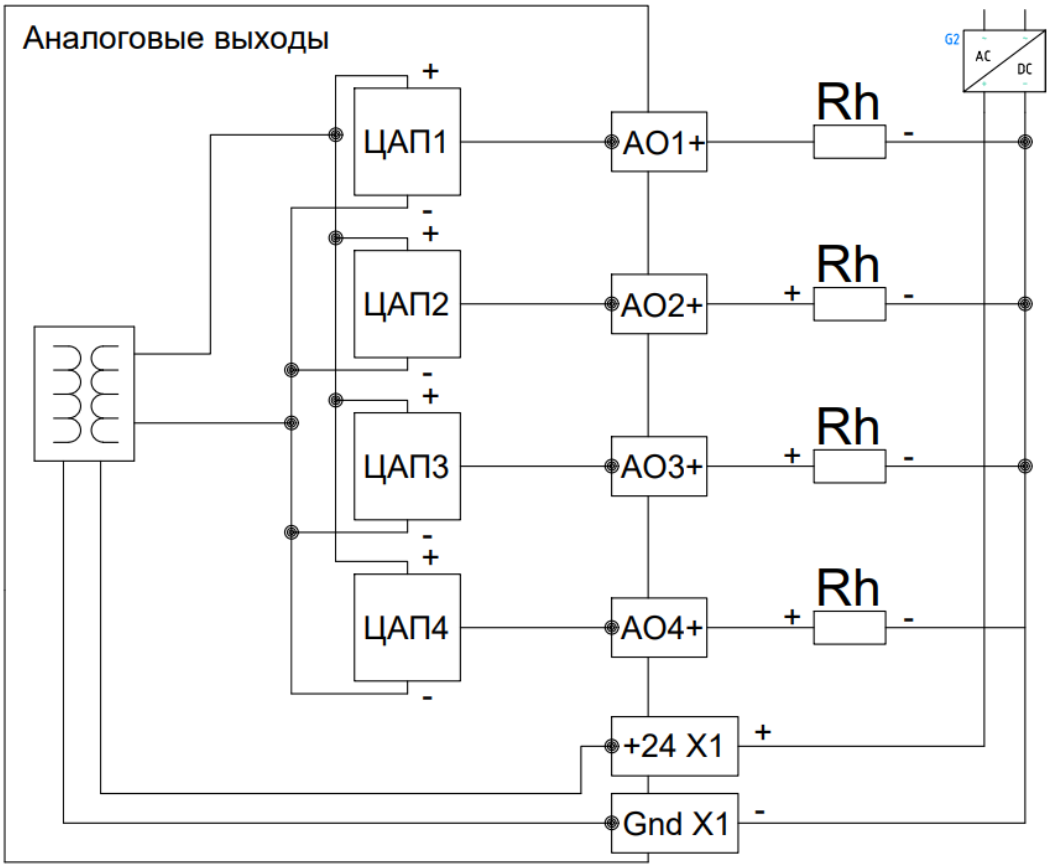


Рисунок 15.1 Аналоговые выходы

Подключение устройств к ПЛК ЕТ-ХХ

В таблице 16.1 перечислены устройства, которые можно подключить к контроллеру, а также способы подключения.

Таблица 16.1 – Способы подключения

Подключаемое устройство	**Порт ПЛК ЕТ-ХХ**	**Кабель**	**Номер рисунка**	**Комментарий**
Модули ввода/ вывода серии ЕТ, панели индикации, также любое, поддерживающее RS-485 и протоколы из таблицы 2.9	RS-485(0-4)	Витая пара	Рисунок 11.2	Подключение следует производить при отключенном напряжении питания всех устройств сети RS485*. Соблюдать полярность
ПК, связь со SCADA-системой, модули ввода/ вывода серии ЕТ, «UpakNet»	Ethernet	Ethernet UTP Cat 5	Рисунок 10.1	Патч-корд RJ45 Male
• кабель подключается к прибору с помощью разъема на передней панели; • другой конец кабеля подключается к Ethernet-порту компьютера. • перезапуск контроллера осуществляется с помощью отключения питания разъём (X1).				

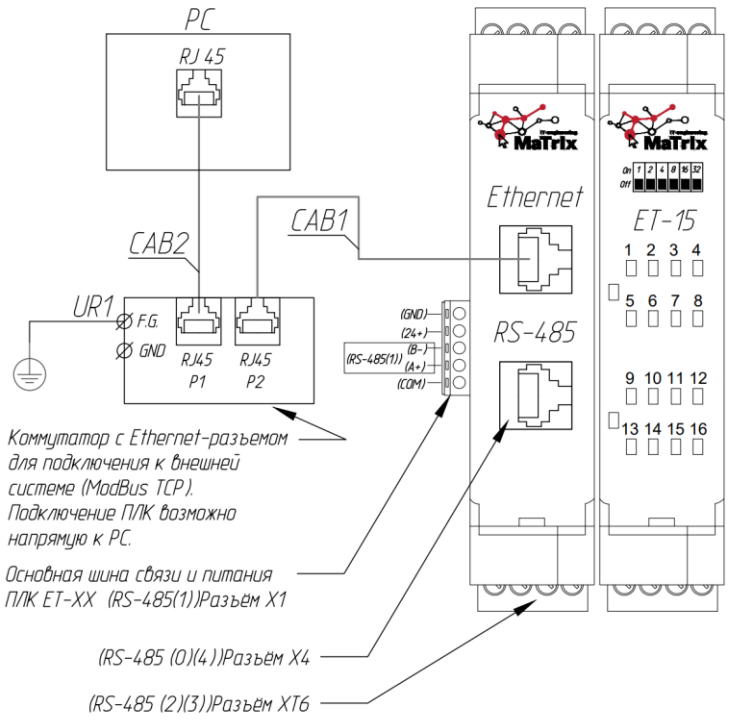

PC
RJ 45
CAB2
CAB1
UR1
F.G.
GND
RJ45
P1
RJ45
P2
(GND)
(24+)
(RS-485(1))
(B-)
(A+)
(COM)
Ethernet
RS-485
ET-15
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Коммутатор с Ethernet-разъемом для подключения к внешней системе (ModBus TCP). Подключение ПЛК возможно напрямую к PC.
Основная шина связи и питания ПЛК ET-XX (RS-485(1))Разъём X1
(RS-485 (0)(4))Разъём X4
(RS-485 (2)(3))Разъём XT6

Рисунок 16.1

Пример комплексной архитектуры системы управления с применением ПЛК ЕТ-XX приведен на рисунке 16.2.

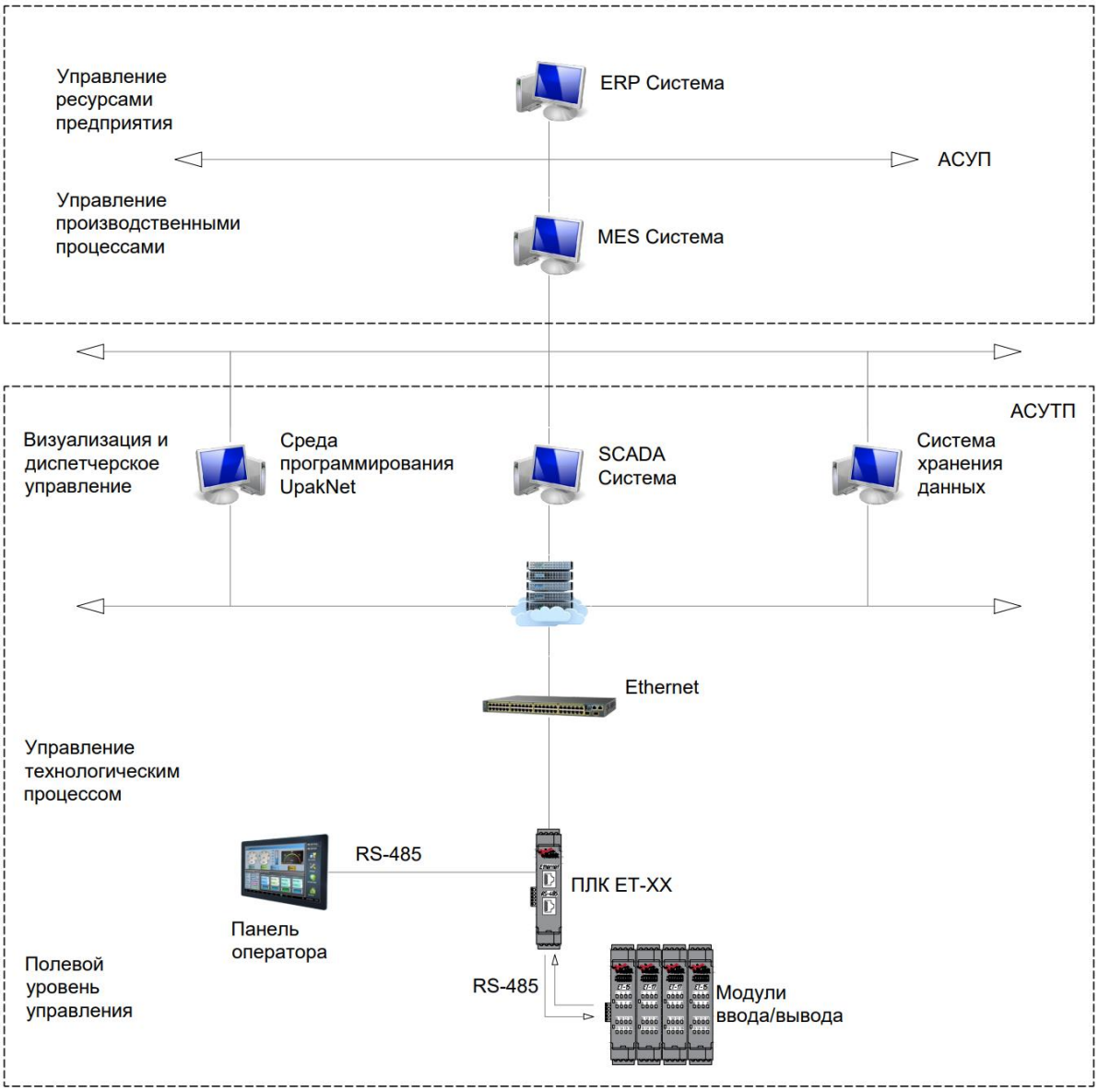


Рисунок 16.2

Эксплуатация

Использование по назначению

Перед использованием прибор следует запрограммировать, то есть создать пользовательскую программу. После создания пользовательскую программу можно сохранить в энергонезависимой Flash-памяти прибора и запускаться на выполнение после включения питания или перезагрузки. Прибор программируется с помощью UpakNet. С помощью интерфейса контроллера: Ethernet.
Прибор подключается к ПК через интерфейс Ethernet с помощью кабеля "Патч-корд". Кабель включается в гнездо (Ethernet) на лицевой панели прибора. Ответная часть кабеля подключается к Ethernet-порту ПК.

Пробный пуск

Если прибор находился длительное время при температуре ниже рабочей, то перед включением и началом работ с прибором необходимо выдержать его в помещении с температурой, соответствующей рабочему диапазону, в течение 30 мин.
Перед подачей питания на прибор следует проверить правильность подключения напряжения и его уровень.
Для ET-XX с питанием от источника постоянного напряжения:
• если напряжение ниже 9 В, то работа контроллера не гарантируется (контроллер прекращает функционировать, однако из строя не выходит);
• если напряжения питания выше уровня 30 В, то возможен выход прибора из строя.
Во время подачи на прибор напряжения питания допустимого диапазона на лицевой стороне корпуса начинает светиться желтым и красным светом индикаторы приведенные в список 9.1. Если напряжение питания ниже допустимого, индикатор светиться не будет.
После включения питания контроллер загрузится и элементы индикации начнут светиться. Если в контроллер загружена пользовательская программа, то пользовательская программа
сразу начинает исполняться.

Техническое обслуживание

Во время выполнения работ по техническому обслуживанию контроллера следует соблюдать правила из раздела «Меры безопасности».
Технический осмотр контроллера проводится обслуживающим персоналом не реже одного раза в 6 месяцев и включает в себя выполнение следующих операций:
• очистка корпуса и клеммных колодок контроллера от пыли, грязи и посторонних предметов;

• проверка качества подключения внешних связей.

Обнаруженные во время осмотра недостатки следует немедленно устранить.

Маркировка

На корпус прибора нанесены:

• наименование прибора;

• заводской номер прибора.

На потребительскую тару нанесены:

• наименование прибора;

• страна-изготовитель;

• заводской номер прибора и год выпуска.

Упаковка

Упаковка прибора производится в соответствии с ГОСТ 23088-80 в потребительскую тару,

выполненную из коробочного картона по ГОСТ 7933-89.

Упаковка прибора при пересылке почтой производится по ГОСТ 9181-74.

Транспортирование и хранение

Прибор должен транспортироваться в закрытом транспорте любого вида. В транспортных средствах тара должна крепиться согласно правилам, действующим на соответствующих видах транспорта. Условия транспортирования должны соответствовать условиям 5 по ГОСТ 15150-69 при температуре окружающего воздуха от минус 25 до плюс 55 °С с соблюдением мер защиты от ударов и вибраций. Прибор следует перевозить в транспортной таре поштучно или в контейнерах.

Условия хранения в таре на складе изготовителя и потребителя должны соответствовать условиям 1 по ГОСТ 15150-69. В воздухе не должны присутствовать агрессивные примеси.

Прибор следует хранить на стеллажах.

Комплектность

Наименование	Количество
Контроллер ПЛК ET-XX	1шт.
Патч-корд RJ45 Male	1шт.
Шина связи и питания	1шт.
Разъём 5pin	1шт.

Изготовитель оставляет за собой право внесения дополнений в комплектность прибора без уведомления или согласования с пользователем.