НАСТРОЙКА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ PROFIBUS НА КОНТРОЛЛЕРАХ СЕРИИ REGUL RX00

Руководство пользователя

DPA-302.12

Версия документа 1.1

Версия ПО 1.7.2.0

Сентябрь 2024

История изменений руководства пользователя

Версия руководства пользователя	Описание изменения
1.0	Релизная версия документа
1.1	Реализована поддержка протокола версии Profibus DPV1. *Подраздел «Добавление ведомого устройства*»: добавлено описание о проверке конфигурации на превышение допустимых пределов входных/выходных областей данных. *Подраздел «Настройка параметров модуля коммуникационного процессора»*: добавлено описание новых параметров на вкладке редактора CP 01 031. *Подраздел «Настройка Profibus Master»:* добавлено описание новых параметров для настройки ведущего устройства, поддерживающего протокол версии Profibus DPV1. *Подраздел «Настройка DP Slave»:* добавлено описание новых параметров для настройки ведомого устройства, поддерживающего протокол версии Profibus DPV1. Добавлено приложение: «*Приложение А. Библиотека PsProfibus»*

АННОТАЦИЯ

Настоящий документ содержит сведения о настройке обмена данными по протоколу PROFIBUS DP на промышленных логических контроллерах серии Regul RX00. Настройка осуществляется с помощью программного обеспечения Astra.IDE.

Данное руководство предназначено для эксплуатационного персонала и инженеров-проектировщиков АСУ ТП, которые должны:

- иметь, как минимум, среднее техническое образование;
- приступить к работе только после изучения данного руководства.

Обновление информации в Руководстве

Производитель ООО «РегЛаб» оставляет за собой право изменять информацию в настоящем Руководстве и обязуется публиковать более новые версии с внесенными изменениями. Обновленная версия Руководства доступна для скачивания на официальном сайте Производителя: https://reglab.ru/.

Для своевременного отслеживания выхода новой версии Руководства рекомендуется оформить подписку на обновление документа. Для этого необходимо на сайте Производителя: https://reglab.ru/ кликнуть на кнопку «Подписаться на обновления» и оставить свои контактные данные.

В руководстве присутствуют знаки с предупреждающей и поясняющей информацией. Каждый знак обозначает следующее:

ПРЕДУПРЕЖДАЮЩИЕ ЗНАКИ

	**ВНИМАНИЕ!** Здесь следует обратить внимание на способы и приемы, которые необходимо в точности выполнять во избежание ошибок при эксплуатации или настройке.

ИНФОРМАЦИОННЫЕ ЗНАКИ

	**ИНФОРМАЦИЯ** Здесь следует обратить внимание на важную информацию

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Общие сведения

PROFIBUS DP – это шинная система передачи данных с протоколом DP (децентрализованные периферийные устройства ввода/вывода) для быстрого циклического обмена данными между ведущим и периферийными устройства (ведомыми). Технические характеристики шины определяются в соответствии со стандартами IEC 61158 («Промышленные коммуникационные сети. Спецификации на шинные линии связи») и IEC 61784 («Промышленные сети связи. Профили полевой шины»). Контроллер подключается к сети посредствам интерфейса RS-485.

Программное обеспечение контроллера Regul позволяет сконфигурировать его в качестве PROFIBUS Master. Master обменивается данными с ведомыми устройствами, диагностирует их, а также параметризирует и конфигурирует.

Доступен циклический и ациклический обмен данными через PROFIBUS DP:

- DP-V0 – для циклического обмена данными и диагностикой;
- DP-V1 – расширенная версия для ациклического обмена данными и управления авариями.

Перечень рекомендуемых документов

Для получения дополнительной информации по настройке других параметров контроллеров серии Regul RX00 в среде разработки Astra.IDE рекомендуется ознакомиться со следующими документами (доступны на сайте https://reglab.ru/):

- Программное обеспечение Astra.IDE. Руководство пользователя;
- Regul R500. Системное руководство.

Термины и определения

Программируемый логический контроллер (ПЛК) — микропроцессорное устройство в промышленном исполнении, используемое для сбора, преобразования, обработки, хранения информации и выработки команд управления, имеющее конечное количество входов и выходов, подключенных к ним датчиков, ключей, исполнительных механизмов к объекту управления, и предназначенное для работы в режимах реального времени.

Интегрированная среда разработки (IDE) — комплекс программных средств, используемый техническими специалистами для разработки прикладного программного обеспечения (ПО).

Astra.IDE — среда разработки с набором компонентов для настройки/программирования контроллеров серии Regul RX00, выпускается компанией «РегЛаб».

GSD (**General Station Data** — «Общие данные об устройстве») — файлы описания ведомых устройств, представленные в унифицированном виде. Использование GSD-файлов облегчает процедуру конфигурирования.

Ведомый (Slave) — подчиненное устройство, которое выполняет сбор информации или выдачу ее исполнительным устройствам, так же обменивается данными с ведущим устройством после получения от него запроса на передачу данных.

Ведущий (Master) — устройство управления обменом данными по шине. Ведущие устройства делятся на два класса:

- **1 класс**. Центральный контроллер, который циклически обменивается информацией с ведомыми устройствами с заранее определенным периодом;
- **2 класс**. Устройство, предназначенное для конфигурирования системы, наладки, обслуживания или диагностики.

Для передачи данных между ведущими станциями применяют маркер.

Маркер — выполняет роль арбитра, предоставляющего ведущему устройству право доступа. Маркер циркулирует в логическом кольце, состоящем из ведущих устройств. На время обладания маркером ведущее устройство становится «главным» по отношению к другим ведущим.

Начало работы

Установите на ПК программное обеспечение **Astra.IDE**. Описание процесса установки программы, а также инструкции по работе с программой приведены в документе «Программное обеспечение Astra.IDE. Руководство пользователя». Программа установки и документация доступны на сайте https://reglab.ru/.

Запустите программу **Astra.IDE**. Создайте проект в соответствии с аппаратной конфигурацией контроллера с помощью **Мастера конфигурации Regul** (инструкции приведены в документе «Программное обеспечение Astra.IDE. Руководство пользователя»).

Принцип добавления устройств в конфигурацию контроллера

Контроллер выступает в роли ведущего, опрашивающего устройства. Для начала работы в конфигурацию контроллера необходимо добавить модуль коммуникационного процессора CP 01 031.

При настройке режима необходимо задать параметры ведомых устройств, которые будут опрашиваться контроллером. Кроме того, требуется описать набор данных, который будет запрашиваться.

Модуль CP 01 031 предназначен для организации одного независимого канала связи по протоколу PROFIBUS DP через интерфейс RS-485.

Таблица 1 – Технические характеристики модуля коммуникационного процессора CP 01 031

Наименование параметра, единица измерения	**Значение**
Реализуемые протоколы	PROFIBUS DP V0/V1
Количество подключаемых устройств, не более (до 244 байт ввода, вывода, параметров, конфигурации или диагностических данных на ведомое устройство)	125
Скорость передачи данных, Кбит/с	от 9,6 до 12 000
Максимальный объем данных на все устройства, Кбайт: – входные данные – выходные данные – полный объем, не более	 до 1,4 до 1,4 2,5

Добавление модуля коммуникационного процессора

В окне дерева устройств поместите курсор на название крейта, нажмите правую кнопку мыши. В появившемся контекстном меню выберите пункт **Добавить устройство…** (Рисунок 1).

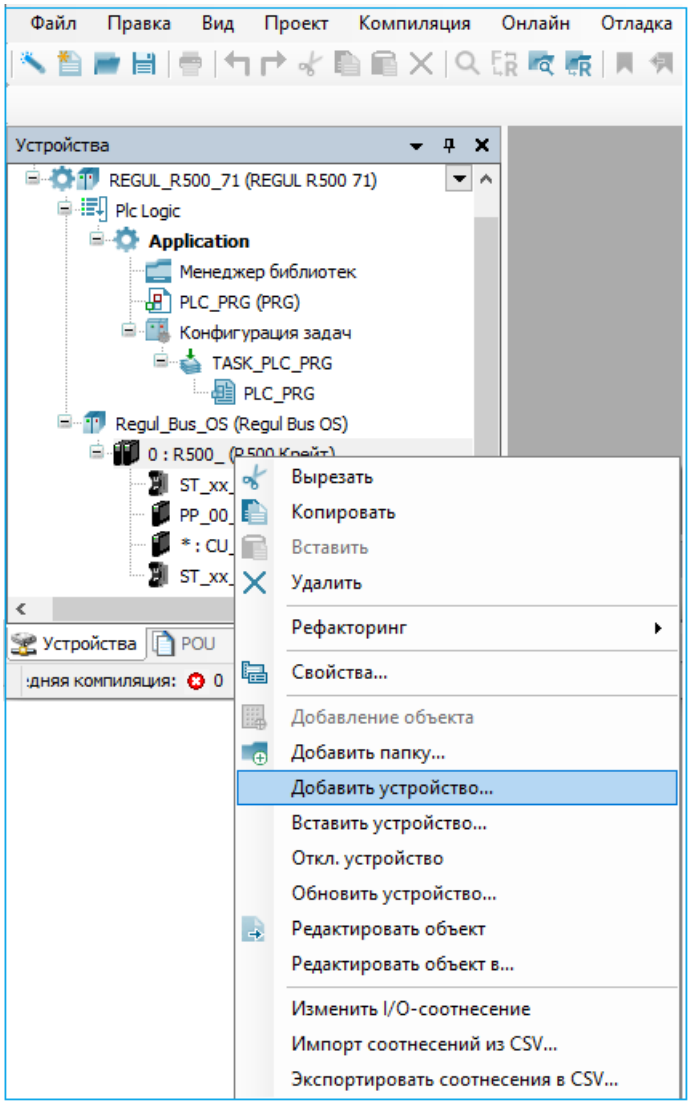


Рисунок 1 – Контекстное меню

Откроется окно с перечнем модулей для выбора (Рисунок 2).

Выберите: *Regul* ⇨ *Коммуникационные модули* ⇨ *CP 01 031*, дважды щелкните мышью по названию модуля или в нижней части окна нажмите кнопку ***Добавить устройство***. Модуль будет добавлен в крейт (появится в проекте в дереве устройств).

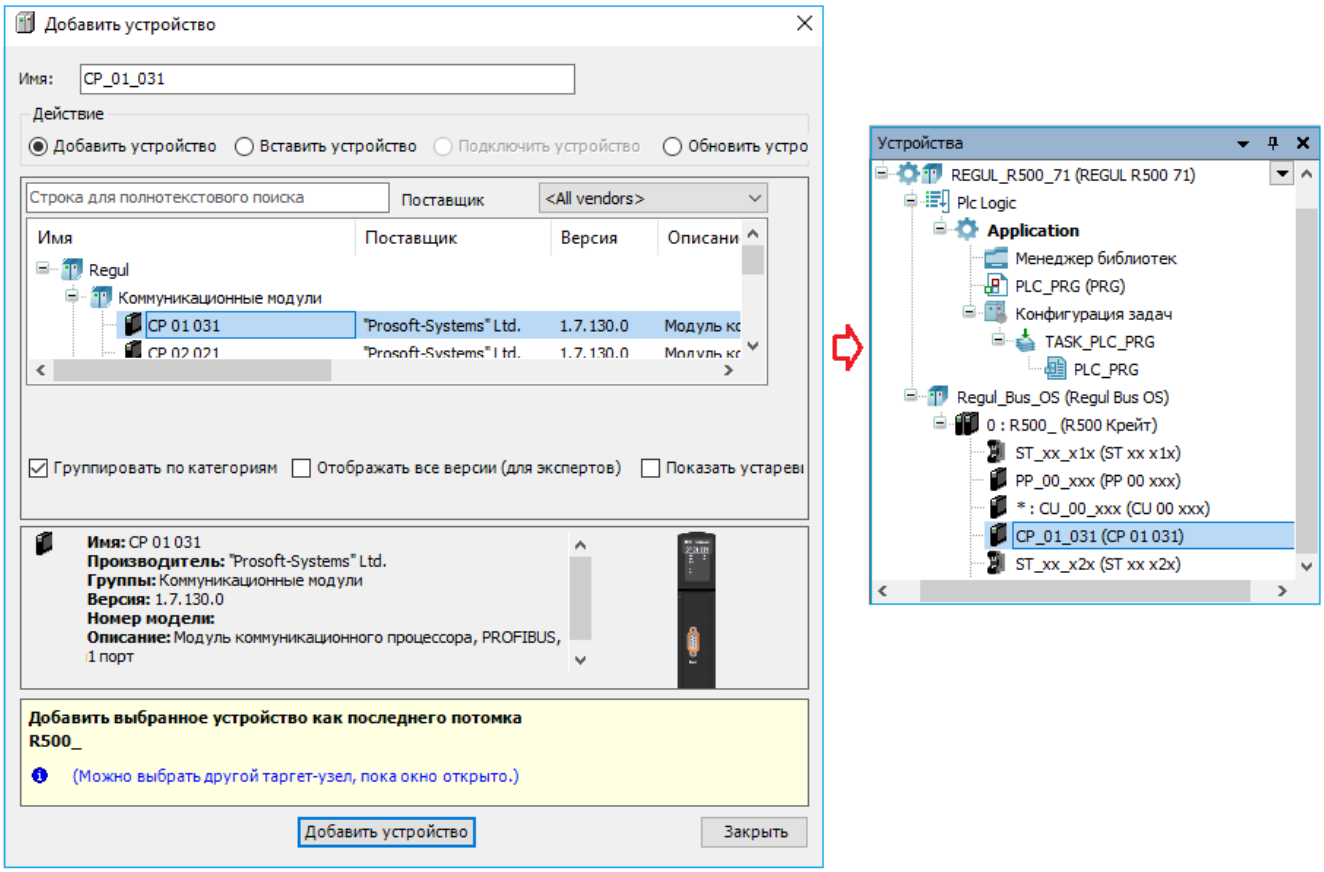


Рисунок 2 – Добавление в конфигурацию контроллера модуля CP 01 031

Добавление ведущего устройства

В дереве устройств к контроллеру требуется подключить не только устройство, соответствующее реально существующему модулю (CP 01 031), но и виртуальное устройство, у которого нет аппаратного эквивалента⇨***Profibus_Master***-устройство, являющееся виртуальным и фактически предназначенным для настройки параметров МЭК библиотеки, автоматически загружаемой в проект при подключении конкретного устройства.

В окне дерева устройств поместите курсор на название модуля коммуникационного процессора **CP 01 031**, нажмите правую кнопку мыши. В появившемся контекстном меню выберите пункт **Добавить устройство…** Откроется окно **Добавить устройство**, в котором выберите: *Промышленные сети ⇨ Profibus ⇨ DP Master ⇨ Profibus Master*. Нажмите кнопку ***Добавить устройство*** или дважды щелкните левой кнопкой мыши. Выбранное устройство появится в проекте в дереве устройств (Рисунок 3).

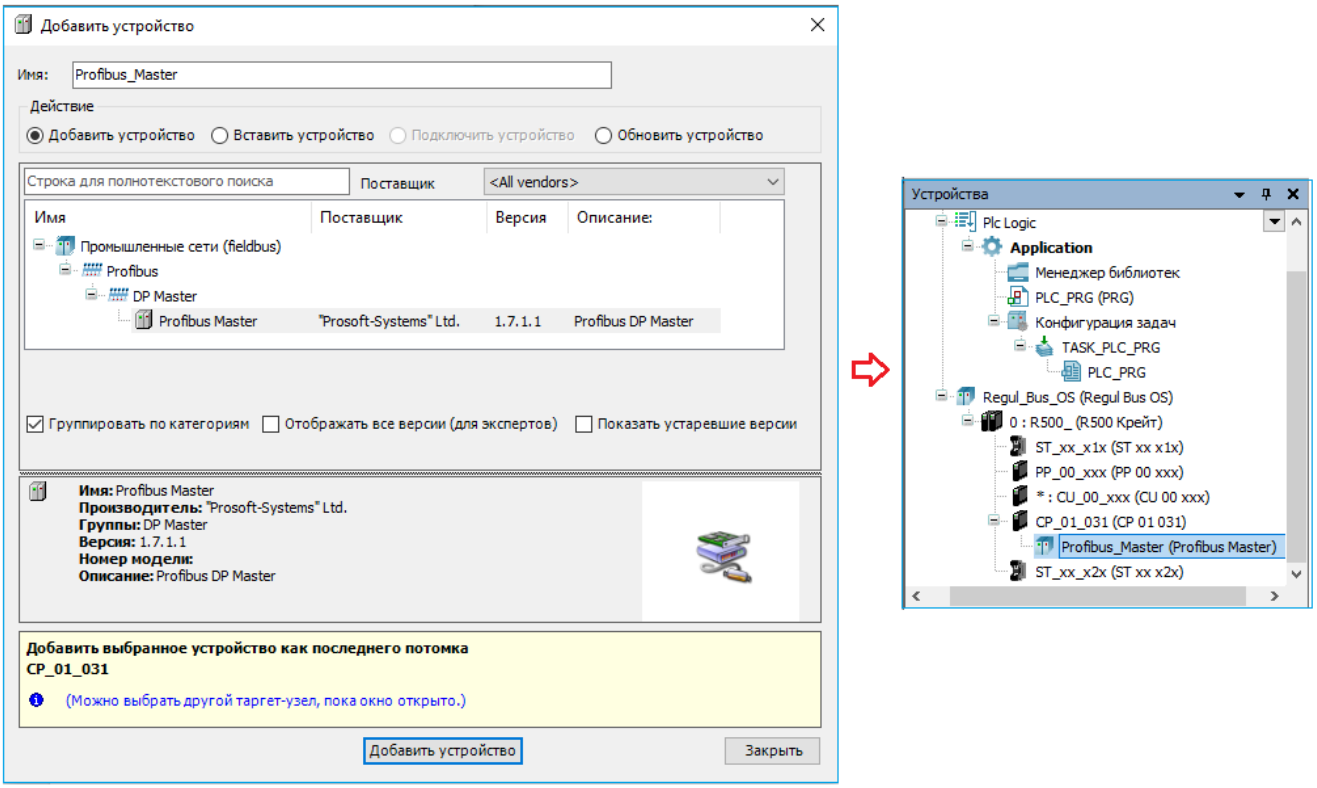


Рисунок 3 – Добавление в конфигурацию контроллера устройство Profibus Master

Добавление ведомого устройства

Для подключения ведомых устройств PROFIBUS DP, необходимо импортировать в среду разработки файлы описания устройств GSD, с расширением *gsd или *gs?. GSD-файл содержит идентификатор типа устройства, описание функций и характеристик протокола, поддерживаемых данным устройством, структуру передаваемой диагностической информации, размеры областей ввода-вывода циклического обмена, а также перечень идентификаторов конфигурации циклического, допустимых для использования применительно к данному подчиненному устройству при его конфигурировании каким-либо мастером.

Эти файлы поставляются производителями оборудования, выступающими в роле опрашиваемого устройства Profibus. GSD-файл создается и предоставляется пользователю на соответствующем языке. Файлы описания устройства, в зависимости от языка, будут помечаться соответствующей последней буквой в обозначении расширения (*.gs?). Версия по умолчанию на английском языке с расширением *gsd.

Скачайте на ПК файл описания. В среде разработки выберите в основном меню **Инструменты ⇨ Репозиторий устройств**. Откроется окно **Репозиторий устройств.** Нажмите кнопку ***Установить...*** Откроется окно **Установить описание устройства** (Рисунок 4).

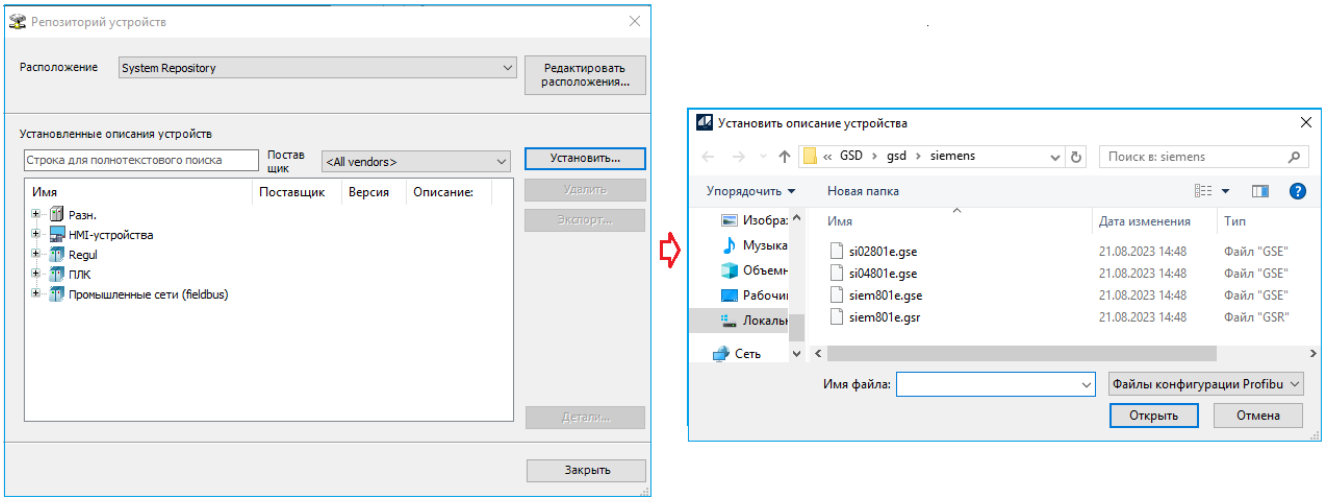


Рисунок 4 – Добавление файла описания устройства

Найдите сохраненный ранее файл описания на ПК и нажмите кнопку ***Открыть.*** В окне **Репозиторий устройств** В поле **Установленные описания устройств** в дереве устройств **DP Slave** появится добавленное устройство с описанием (Рисунок 5). Нажмите кнопку ***Закрыть***.

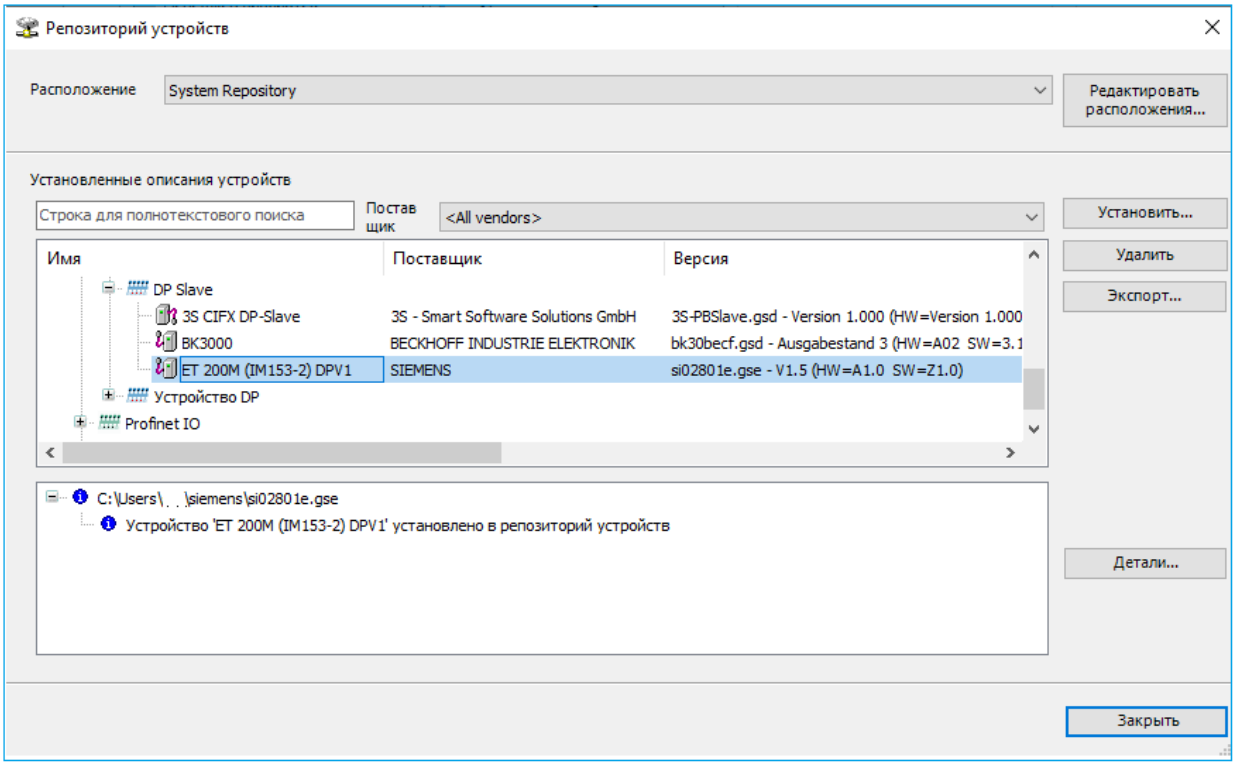


Рисунок 5 – Репозиторий устройств

Далее к устройству **Profibus_Master** нужно подключить необходимое количество ведомых устройств **(DP Slave)**, файл описание которых был добавлен ранее (см. описание выше).

В окне дерева устройств поместите курсор на название устройства **Profibus_Master**, нажмите правую кнопку мыши. В появившемся контекстном меню выберите пункт **Добавить устройство…** Откроется окно **Добавить устройство**, в котором выберите устройство необходимое для опроса контроллером: *Промышленные сети⇨ Profibus⇨ DP Slave ⇨…*(Рисунок 6).

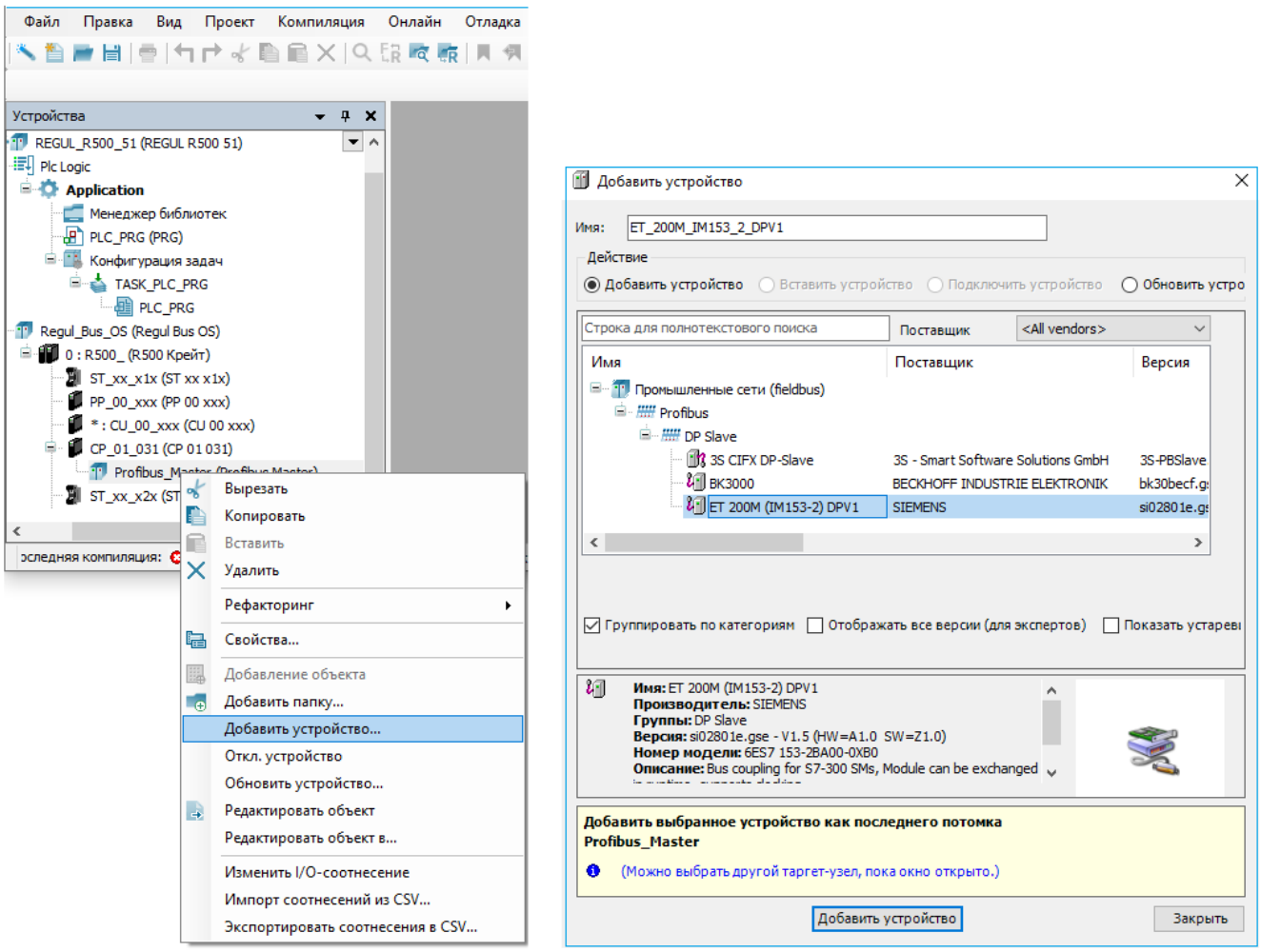


Рисунок 6 – Добавление в конфигурацию контроллера *Slave* - устройства

Нажмите кнопку ***Добавить устройство*** или дважды щелкните левой кнопкой мыши. Выбранное устройство появится в проекте в дереве устройств.

Далее необходимо сконфигурировать ведомые устройства. Конфигурируемые параметры зависят от конкретного устройства, но, в общем случае, это параметры сети и конфигурация модулей ввода/вывода данных.

Список возможных ведомых устройств (модулей ввода/вывода, датчиков, исполнительных устройств и т.д.) считывается из файла описания GSD. В окне дерева устройств поместите курсор на название ведомого устройства, нажмите правую кнопку мыши. В появившемся контекстном меню выберите пункт **Добавить устройство…** Откроется окно **Добавить устройство**, в котором выберите необходимое ведомое устройство (Рисунок 7).

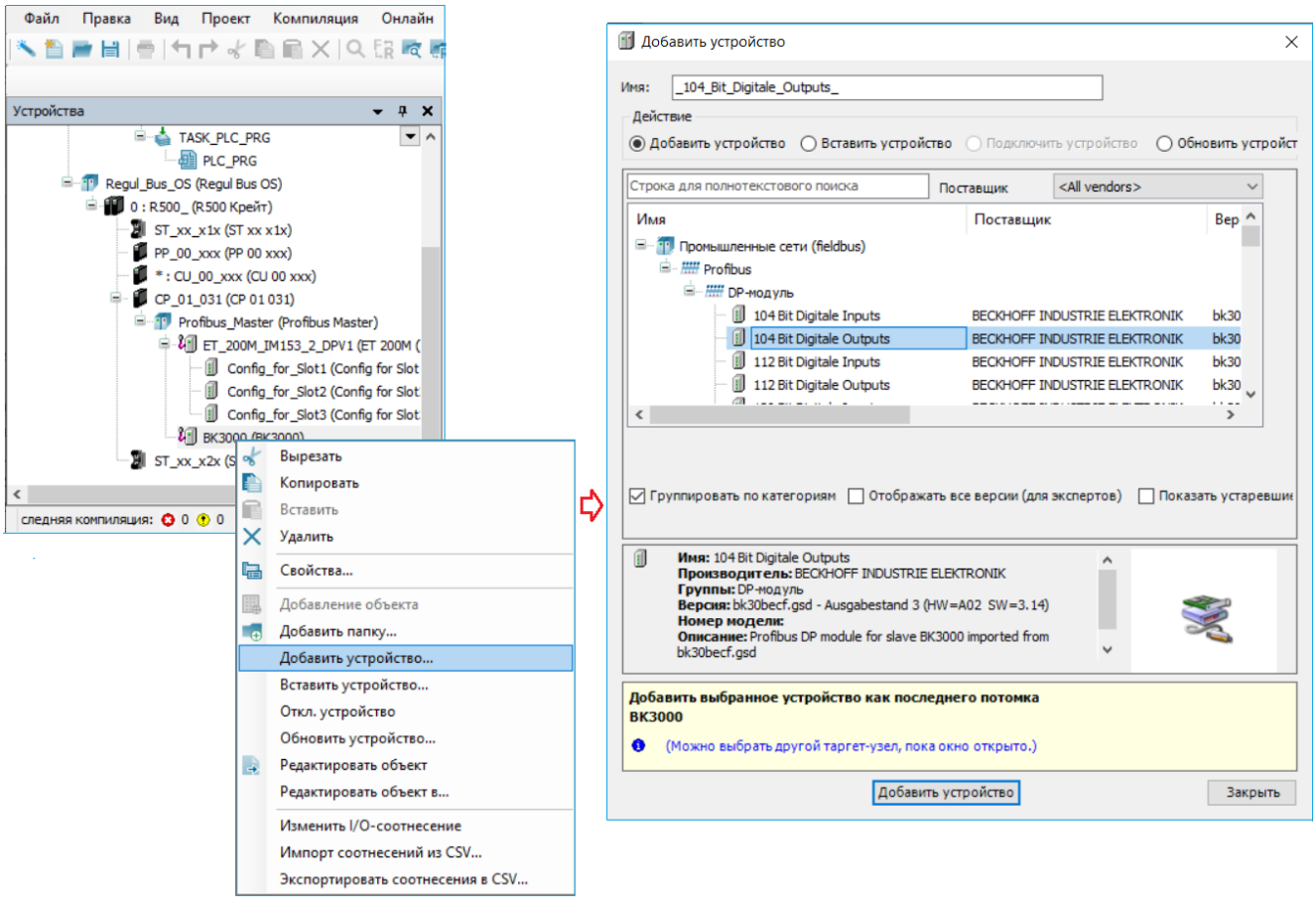


Рисунок 7 – Добавление ведомого устройства

Нажмите кнопку ***Добавить устройство*** или дважды щелкните левой кнопкой мыши. Выбранное ведомое устройство появится в проекте в дереве устройств.

Максимальные размеры данных (Max. length of input data, Max. length of output data, Max. length of in-/output data) и максимальное число модулей (Max. number of modules) определяются в GSD-файле. Для одного ведомого устройства объем входных и выходных данных не должен превышать 488 байт: 244 байта входных данных (Input Data) и 244 байта выходных данных (Output Data). Общий объем входных/выходных данных для **Profibus_Master** не должен превышать 1486 байт.

При компиляции проекта производится проверка конфигурации на превышение допустимых пределов входных/выходных областей данных. В случае превышения значения в окне сообщений отобразятся сообщения об ошибке, например:

```
«Размер входных данных (=208), заданных для слейва BK3000, превышает Max_Input_Len
= 64»;
«Общий размер I/O-данных (=209), заданный для слейва BK3000, превышает
Max_Data_Len»
```

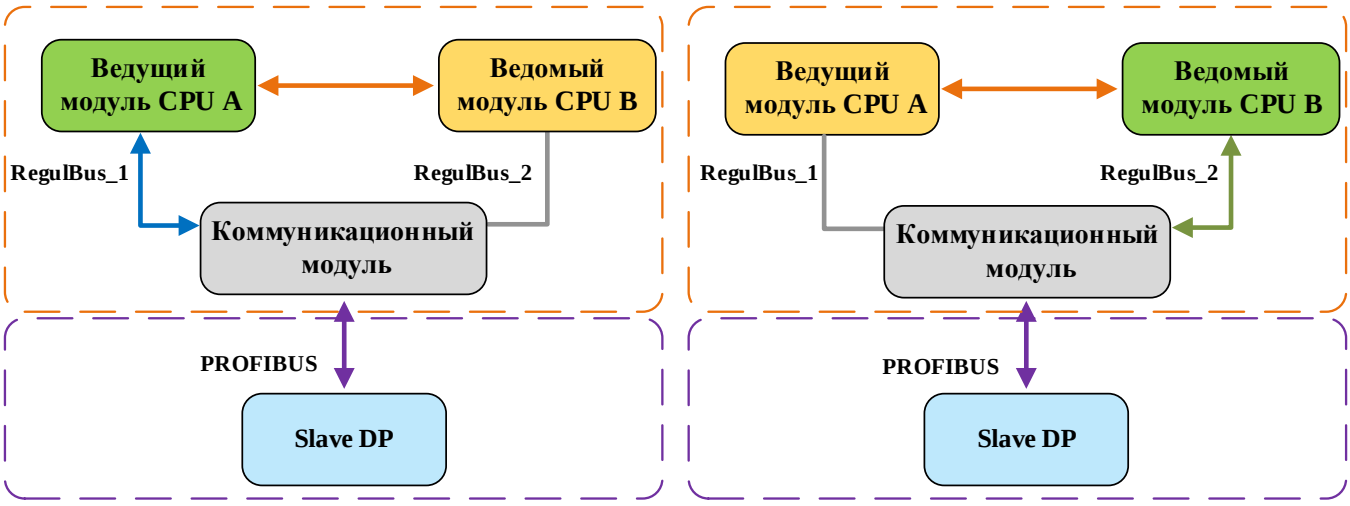
Конфигурирование в резервированной системе

Модуль СР 01 031 может работать в схеме частичного и комбинированного резервирования (более подробное описание о построении резервированной системы приведено в документе «Конфигурирование резервированной системы на контроллерах серии REGUL RX00»).

ИНФОРМАЦИЯ

При конфигурировании резервированной системы активируйте параметр **Влияние на ошибку шины** (см. раздел «Настройка параметров. Настройка параметров модуля коммуникационного процессора»)

На уровне шины RegulBus. При частичном/комбинированном резервировании на базе двух шин RegulBus данные транслируются с ведущего модуля ЦП через общий коммуникационный модуль в единую линию связи по протоколу PROFIBUS DP (Рисунок 8).



Условные обозначения:

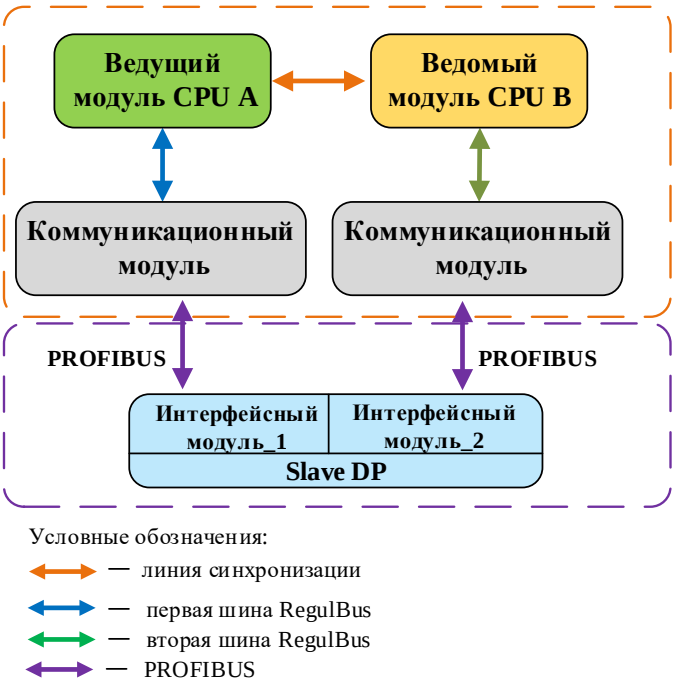
— линия синхронизации
— первая шина RegulBus
— вторая шина RegulBus
— PROFIBUS
— шина в резерве

Рисунок 8 – Трансляция данных с модулей ЦП в схеме частичного/комбинированного резервировании через коммуникационные модули СР 01 031

ВНИМАНИЕ!

Подключение модулей связи (Y-link), выполняющих функцию ведомого DP-устройства, оснащенного двумя интерфейсными модулями и способного автоматически переключаться на активную ветвь резервированной сети, не требуется! Данный функционал реализован в модуле СР 01 031

На уровне Profibus DP. Возможна реализация резервной линии связи по протоколу PROFIBUS DP и резервного ведомого устройства (Slave DP), где данные транслируются с обоих ЦП-партнеров через свой коммуникационный модуль СР 01 031 на каждый модуль-партнер Slave DP, только если на slave DP предусмотрен функционал поддержки двух линий связи (например, ET200M) (Рисунок 9).



Условные обозначения:

— линия синхронизации

— первая шина RegulBus

— вторая шина RegulBus

— PROFIBUS

Рисунок 9 – Трансляция данных с модулей ЦП через коммуникационные модули CP 01 031 при резервировании линии связи по протоколу PROFIBUS DP

НАСТРОЙКА ПАРАМЕТРОВ

Настройка параметров модуля коммуникационного процессора

Двойным щелчком по названию коммуникационного модуля откройте главную вкладку параметров (Рисунок 10). На вкладке присутствуют системные параметры, которые имеют атрибут *Только для чтения* и используются средой исполнения контроллера для идентификации модуля и его типа (см. описание в документе «Программное обеспечение Astra.IDE. Руководство пользователя»), а также общие параметры модуля:

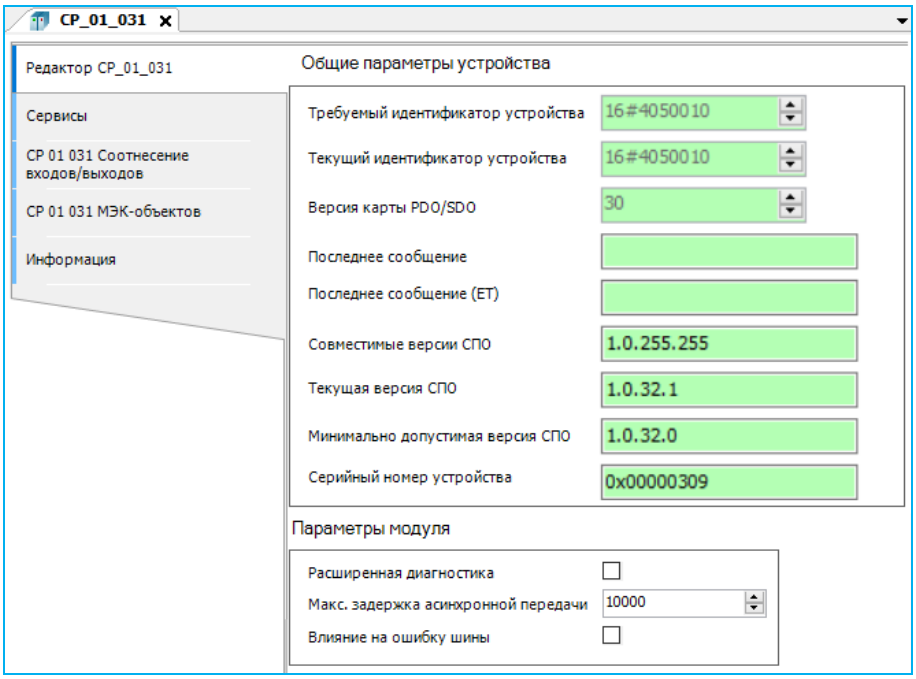


Рисунок 10 – Редактор коммуникационного модуля CP 01 031 в онлайн режиме

– **Расширенная диагностика (Extended diagnostic)** - установка флажка позволяет включить режим обновления диагностической информации сконфигурированных ведомых устройств, по умолчанию данный режим выключен. Полученная диагностическая информация будет доступна на вкладках: **Состояние** (представляет информацию в исходном виде) и **Diagnosis** (представляет информацию в отформатированном и доступном для чтения пользователю виде). Расширенная диагностика предоставляет информацию о текущем состоянии ведомого устройства согласно спецификации PROFIBUS DP (IEC 61158-6). Также диагностическая информация отображается в журнале **regul-bus-driver** (параллельно журналируя сообщения в лог-файл regul-bus-driver.log, который расположен в каталоге /logs/logger/user), например, сообщения вида:

```
«SlaveID1: Diag: S1: 2; S2: 5; S3: 0; MA: 255; INH: 128; INL: 30; EDL: 11
SlaveID1: ExtDiag: 67 0 0 8 130 0 0 0 0 0 0;»
```

– **Макс. задержка асинхронной передачи (Asynchronous delay)** - максимально допустимое время, в течение которого ведомые устройства должны сформировать ответ

на асинхронный запрос. По истечении данного времени осуществляется повтор запроса, количество повторов - 5, по исчерпании количества повторов выставляется ошибка асинхронного запроса. Задержка рассчитывается по следующей формуле:

*max (C1_Response_Timeout) * 10 мс,*

где *C1_Response_Timeout* - максимальное время обработки ациклической функции ведомым устройством (берется из GSD файла ведомого устройства). Выборка максимального значения *C1_Response_Timeout* производится по всем сконфигурированным ведомым устройствам.

Если указанное пользователем значение будет меньше, чем рассчитанное, то при компиляции, в окне сообщений, отобразится сообщение об ошибке:

`«Макс. задержка асинхронной передачи должна быть >= {рассчитанное значение}»;`

– **Влияние на ошибку шины («Impact on bus HwError»)** – можно принудительно включить/отключить формирование ошибки (HwError) для модуля CP 01 031, в случае выхода из строя одного из опрашиваемых ведомых устройств модулем. По умолчанию флажок отсутствует, неисправное ведомое устройство не влияет на формирование ошибки модуля и передачу ее по шине данных.

	**ИНФОРМАЦИЯ** При конфигурировании резервированной системы активируйте параметр **Влияние на ошибку шины**

При отсутствии флажка, в момент обрыва связи, появится знак  , информирующий об ошибке. Он отобразится только напротив устройства **Profibus Master** и ведомого устройства **DP Slave.** При наличии флажка, знак ошибки поднимется выше и дополнительно появится напротив модуля **CP 01 031 ⇨ Крейт ⇨ Regul_Bus_OS.**

Настройка Profibus Master

Для настройки параметров модуля, как ведущего устройства, необходимо перейти на вкладку **Profibus_Master**. Для перехода в редактор дважды щелкните левой кнопкой мыши по названию **Profibus_Master** в окне дерева устройств. Откроется вкладка (Рисунок 11).

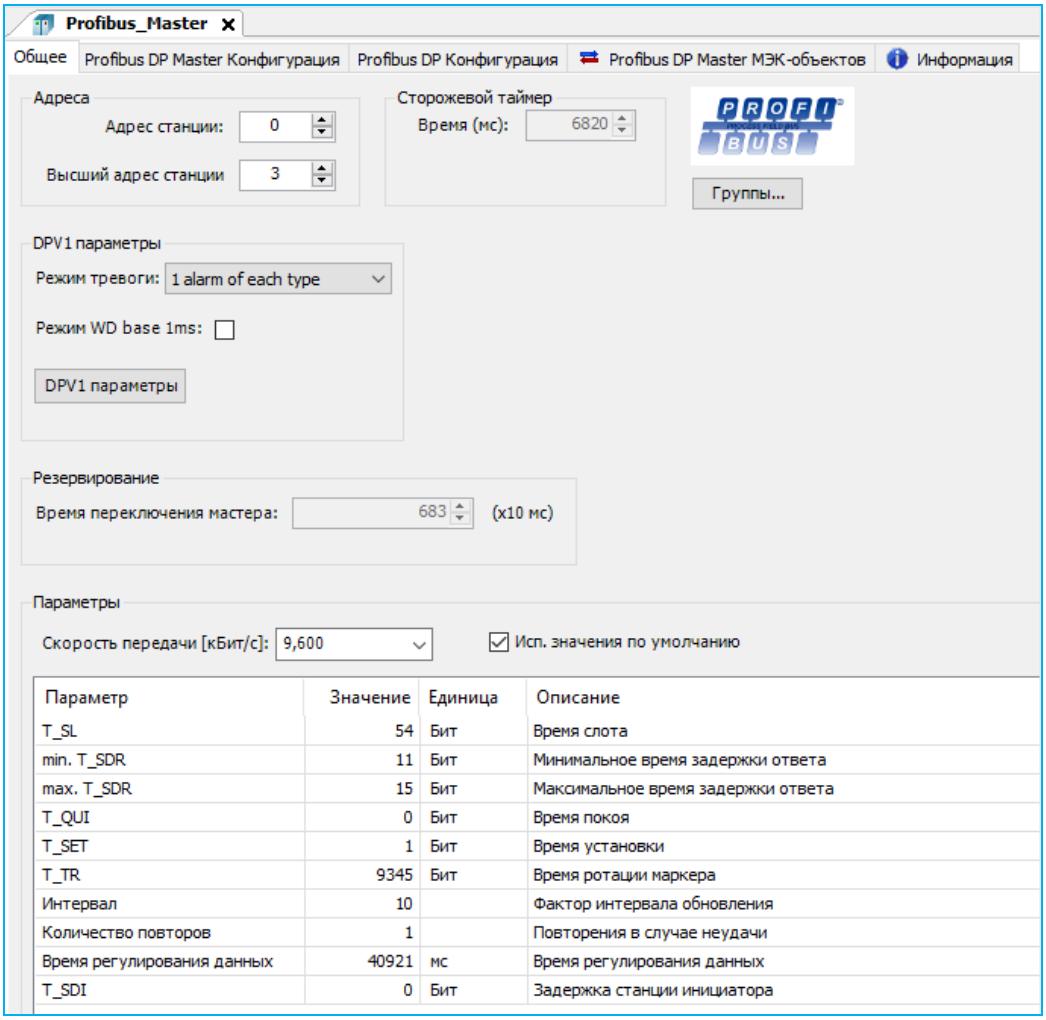


Рисунок 11 – Вкладка Profibus_Master

Каждое устройство сети PROFIBUS должно иметь уникальный адрес для связи (кроме повторителей/соединителей). Адреса закодированы одним байтом и содержат диапазон от 0 до 125. Каждому новому добавленному устройству на шине автоматически присваивается ***Адрес станции***.

В поле **Адрес станции** можно изменить адрес вручную с учетом следующего:

- **0...n** - адрес ведущего устройства должен начинаться с наименьшего адреса, таким образом., что сначала присваивают адреса ведущему устройству, а затем ведомым (**n...125**);
- **(n+1)...125** - адреса ведомых устройств должны продолжать нумерацию за ведущим устройством. Это означает, что в сети PROFIBUS с одним ведущим устройством (адрес **0**) для ведомых станций останется свободных 125 адресов (адреса от 1 до 125).

В поле **Высший адрес станции** указывается «старший» адрес станции, который должен быть равен наибольшему фактическому адресу PROFIBUS, присвоенному в сети.

Для обнаружения ошибок в передающих устройствах сети предусмотрен механизм временного мониторинга. В поле **Сторожевой таймер** указывается временной интервал, по истечении которого, если ведомое устройство не получает данные от ведущего, то все выходы

ведомого устройства переводятся в безопасное состояние. Для каждого ведомого устройства в сети можно включать/выключать параметр **Сторожевой таймер**.

В области **DPV1 параметры** осуществляются настройки параметров ведущего устройства, поддерживающего DPV1-расширения. Настройки параметров будут общими для всех добавленных ведомых устройств. Персональную настройку параметров ведомых устройств осуществляют на вкладке соответствующего **DP Slave** устройства, сняв флажок в строке **Наследовать настройки dvp1 мастера** (см. раздел «Настройка DP Slave»). Если ни одно добавленное ведомое устройство не поддерживает DPV1, то настройки **параметров DPV1** на ведущем устройстве будут заблокированы.

В строке **Режим тревоги (Alarm mode)** выбирается максимальное количество сигналов тревоги при их параллельной обработке. Несколько сигналов тревоги (2-32) одного или различных типов могут быть активны одновременно. Выберите из выпадающего списка необходимый вариант (Рисунок 12):

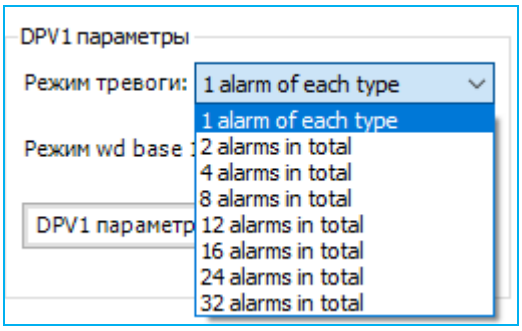


Рисунок 12 – Режим тревоги

- **1 alarm of each type -** один сигнал тревоги каждого типа (задан по умолчанию);
- **2 alarms is total** – всего два сигнала;
- **4 alarms is total** – всего четыре сигнала;
- **8 alarms is total** – всего восемь сигналов;
- **12 alarms is total** – всего двенадцать сигналов;
- **16 alarms is total** – всего шестнадцать сигналов;
- **24 alarms is total** – всего двадцать четыре сигнала;
- **32 alarms is total** – всего тридцать два сигнала.

Фактический режим тревоги для каждого ведомого устройства выбирается минимальным из комбинации параметров GSD файла Alarm_Type_Mode_supp / Alarm_Sequence_Mode_Count и установленного значения параметра **Режим тревоги**.

В строке **Режим WD_Base_1ms** – масштабирование, можно включить/отключить перерасчет сторожевого таймера ведомых устройств (Slave). При установке флажка, расчет осуществляется в 1 мс интервалах, по умолчанию флажок снят и применяется 10 мс интервалы. Если устройство не поддерживает **Режим WD_Base_1ms**, в журнале **regul-bus-driver** появится

запись, содержащая новые рассчитанные значения (округленные в большую сторону), например:

```
«…SlaveID1:10ms watchdog applied (wdFact1: 1, wdFact2: 95)»
```

Нажмите на кнопку *DPV1 параметров* и откроется окно с перечнем доступных для конфигурирования параметров (Рисунок 13):

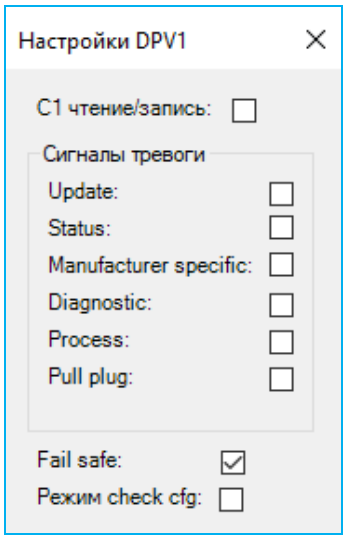


Рисунок 13 – Настройка DPV1 параметров мастера

- **C1 чтение/запись** – включение / отключение операции асинхронного обмена данными (Read/Write);
- **Сигналы тревоги,** выбор типа сигнала тревоги, установкой флажка в соответствующей строке:
  - **Update**, сигнализирующий об изменении параметров на устройстве (например, при локальной операции или удаленном доступе);
  - **Status**, сигнализирующий об изменении состояния устройства (например, о работе, остановке или готовности);
  - **Manufacturer Specific**, сигнализирующий о событии, специфическом для производителя ведомого устройства;
  - **Diagnostic**, сигнализирующий о событии внутри устройства (например, о перегревании, коротком замыкании и т.п.);
  - **Process**, сигнализирующий о возникновении важного события в процессе (например, о превышении верхнего предела некоторой величины);
  - **Pull plug**, сигнализирующий об изъятии/установке устройства (при горячей замене);
- **Failsafe – по умолчанию режим активирован (установлен флажок),** когда мастер в «Безопасном» состоянии, то он не отправляет нули в качестве выходных данных, а отправляет пустой пакет данных (отказоустойчивый пакет данных) ведомому устройству. Таким образом, ведомое устройство распознает, что оно должно разместить

безопасные выходные данные на выходах (значение безопасных выходных данных не обязательно равно нулю);

– **Режим check cfg - по умолчанию режим не активирован (флажок отсутствует),** в этом режиме ведомое устройство проверяет согласованность телеграмм Chk_Config от ведущего DP-V1 и отклоняет ошибочные конфигурации. Если **режим активирован (флажок установлен**), допускаются отклонения в конфигурации, зависящие от разработчика (пользователя) прикладного СПО. Например, можно принять в конфигурацию модуль, даже если он в данный момент не подключен. Поддержка этого бита указывается в GSD-файле ключевым словом «Check_Cfg_Mode = 1».

В PROFIBUS DP присутствуют команды синхронизации выходов (**Sync**) и входов (**Freeze**) ведомых устройств. Они реализуются отправкой широковещательного пакета с соответствующей командой. При получении команды ведомое устройство однократно обновляет свои выхода и/или входа. Т.к. пакет данных приходит одновременно на все устройства на шине, то синхронизация происходит одновременно для всех ведомых устройств. Следующее обновление выходов/входов происходит при очередном получении команды **Sync/Freeze**. При отправке команд **UnSync/UnFreeze** отменяется действие предыдущих команд синхронизации и ведомые устройства начинают обновлять выхода/входа по мере поступления новых данных. Команды синхронизации могут посылаться всем устройствам сети, группе или одному устройству.

Для задания групп ведомых устройств нажмите на кнопку ***Группы,*** откроется окно настройки **Групп** (Рисунок 14) для команд синхронизации.

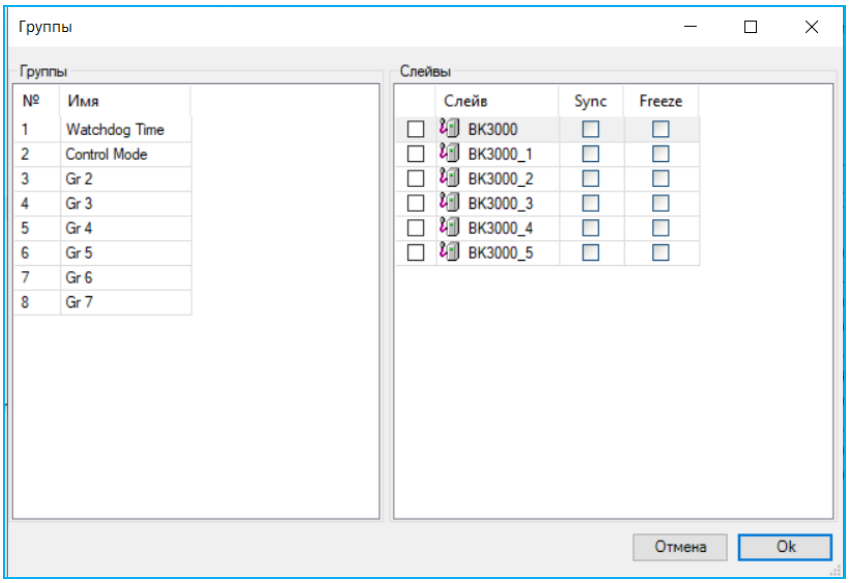


Рисунок 14 – Окно настройки групп ведомых устройств

Для определения ведомых устройств в группу, в области **Группы** наведите мышь на название одной из восьми групп (область подсветится синим цветом), а в области **Слейвы** установите флажок напротив выбранного ведомого устройства. Далее задайте параметры синхронизации, установив флажок в столбце **Sync** и/или **Freeze**. Ведомое устройство может

входить одновременно в несколько групп (максимально до 8 групп). Для определения параметров синхронизации одного ведомого устройства, без участия в группе, достаточно установить флажок напротив устройства в столбце **Sync** и/или **Freeze** (Рисунок 15).

Двойным щелчком левой кнопкой мыши по названию группы в ячейке **Имя** можно отредактировать название группы.

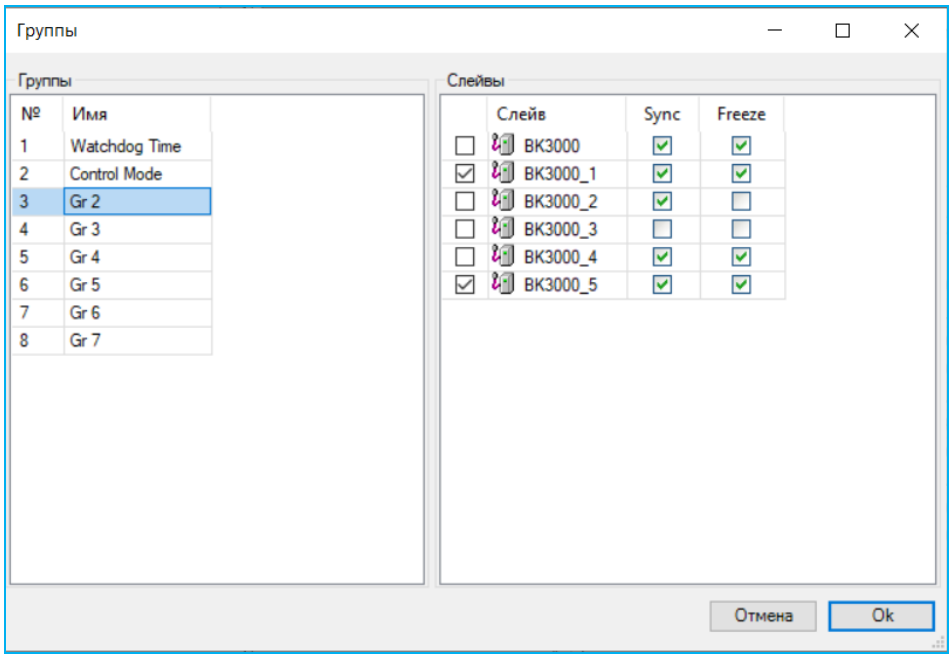


Рисунок 15 – Задание параметров для групп

Конфигурирование **Sync/Freeze**-группы должно выполняться после конфигурирования системы ведущего устройства, все ведомые устройства должны присутствовать в системе.

При наведении курсора мыши на название ведомого устройства появится всплывающая подсказка о принадлежности ведомого устройства к группам (Рисунок 16).

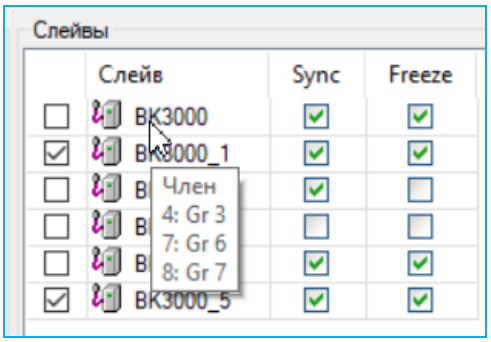


Рисунок 16 – Принадлежность ведомого устройства группе

В поле **Скорость передачи [кБит/с]** из выпадающего списка выберите скорость обмена данными по шине. Базовые модули обеспечивают передачу данных со скоростями от 9,6 кбит/с до 12 Мбит/с.

При указании значений скорости и задании параметров шины необходимо выбрать идентичные значения для всех коммуникационных модулей, подключенных к одной шине.

В области **Резервирование** представлен параметр для конфигурирования работы в резервированной системе:

– **Время переключения мастера** - временной интервал, по истечении которого Slave-устройство бракует коммуникационный модуль (Master) и переводит модуль ЦП из состояния **ведущий** в состояние **ведомый**. Значение определяет максимальный временной интервал, в течение которого Slave-устройства будут удерживать последнее значение, выдавая сигнал от коммуникационного модуля, с которым уже потеряна связь. Значение параметра рассчитывается и задается автоматически. Значение определяется следующим условием:

$$Toh * 10 > Twd,$$

где Toh - время переключения мастера (значение измеряется в интервалах по 10 мс), а Twd - значение сторожевого таймера (Рисунок 17).

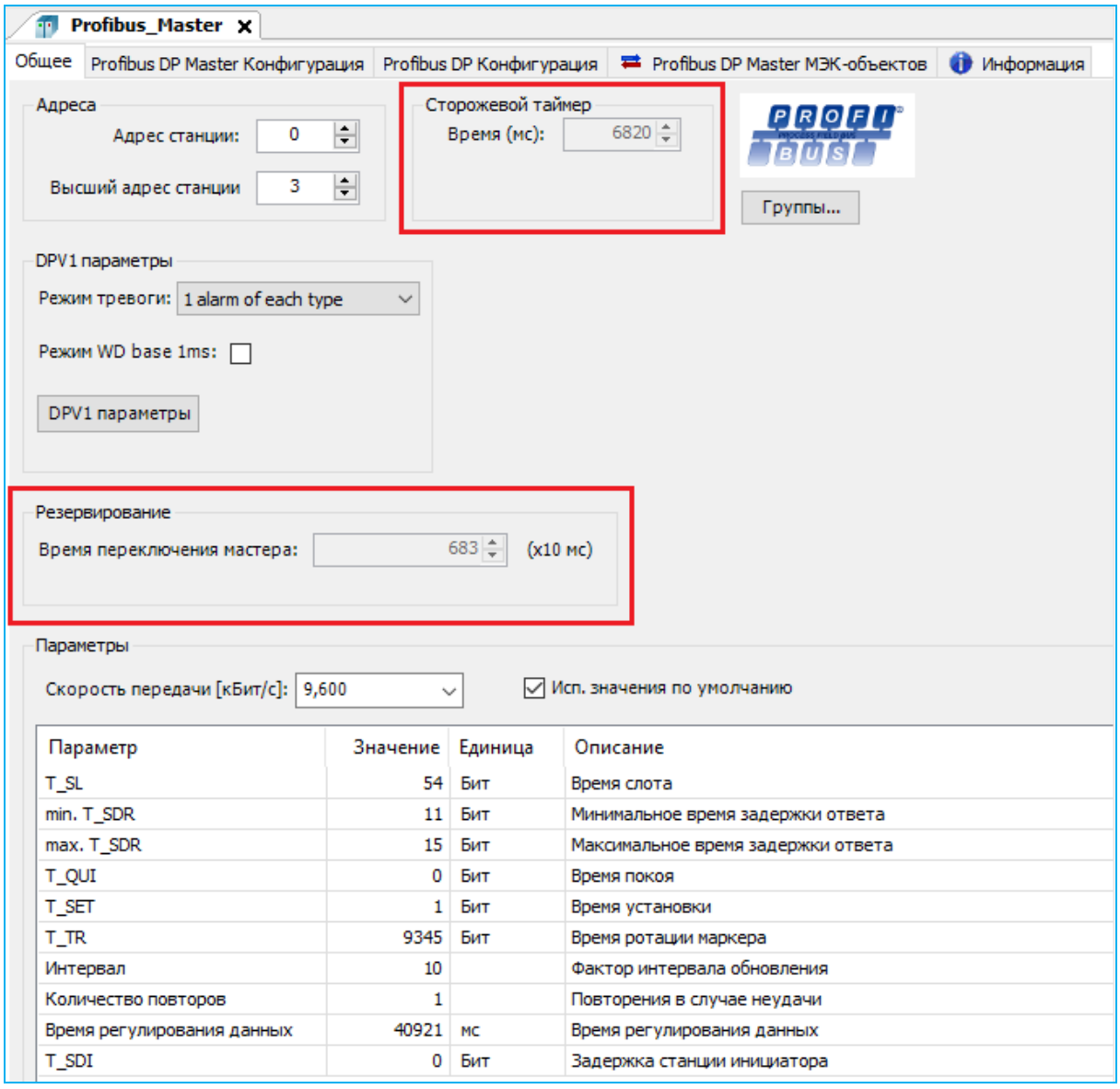
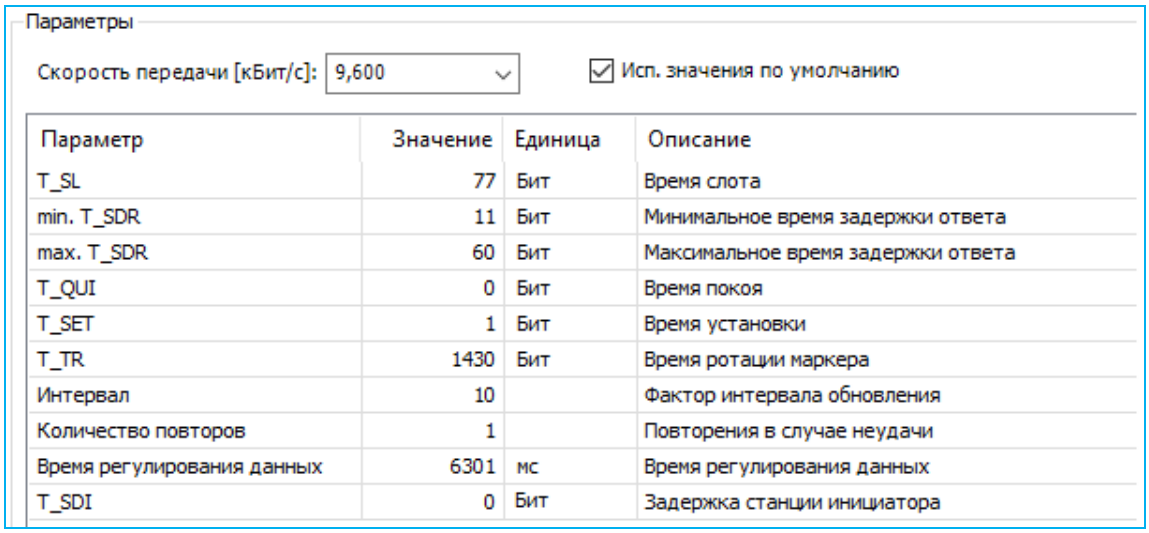


Рисунок 17 - Настройка общих параметров Profibus DP Master

В области **Параметры** в табличном виде представлены параметры шины, которые определяются таймаутами сети (Рисунок 18):


Параметры

Скорость передачи [кБит/с]: 9,600 Исп. значения по умолчанию

Параметр	Значение	Единица	Описание
T_SL	77	Бит	Время слота
min. T_SDR	11	Бит	Минимальное время задержки ответа
max. T_SDR	60	Бит	Максимальное время задержки ответа
T_QUI	0	Бит	Время покоя
T_SET	1	Бит	Время установки
T_TR	1430	Бит	Время ротации маркера
Интервал	10		Фактор интервала обновления
Количество повторов	1		Повторения в случае неудачи
Время регулирования данных	6301	мс	Время регулирования данных
T_SDI	0	Бит	Задержка станции инициатора

Рисунок 18 – Параметры шины

- **T_SL** – **Время слота (**в диапазоне **от 52 до 0xFFFF).** Время ожидания приема. Максимальный временной интервал, в течение которого ведущее устройство ожидает ответа на запрос;
- **min. T_SDR** – **Минимальное время задержки ответа (**в диапазоне **от 11 до 0xFF)**. Минимальный временной интервал, необходимый для формирования ответа на запрос. В течение этого времени ответ не может быть отправлен;
- **max. T_SDR** – **Максимальное время задержки ответа (**в диапазоне **от min.T_SDR до 0xFFFF)**. Максимальный временной интервал, необходимый для формирования ответа на запрос;
- **T_QUI** – **Время покоя (**в диапазоне **от 0 до 0xFF).** Время, необходимое на переключение, которое должно выждать передающее устройство, прежде чем разрешить работу приемного устройства (*T_QUI < min. T_SDR*);
- **T_SET** – **Время установки (**в диапазоне **от 1 до 0xFF).** Время установления, которое истекает с момента наступления события до момента выполнения необходимой реакции (например, между отправкой телеграммы и включением приемника);
- **T_TR** – **Время ротации маркера (**в диапазоне **от 1 до 0xFFFFFF)**. Промежуток времени, в течение которого гарантированно произойдет передача маркера каждому ведущему устройству. Маркер должен быть передан по кругу (логическое маркерное кольцо), за установленное время;
- **Интервал** – **Фактор интервала обновления (**в диапазоне **от 1 до 100).** Время обновления служит для инициализации обслуживания диапазона адресов ведущей станцией. После первого обновление, адреса циклически опрашиваются через каждый интервал (ведущий проводит поиск вновь включенных станций). Время обновления рассчитывается как **Интервал * T_TR**;

- **Количество повторов** – **Повторение в случае неудачи (**в диапазоне **от 1 до 15)**. Максимальное количество предпринимаемых попыток повторов передачи запроса ведущим устройством, если ведомое устройство не отвечает на запрос;
- **Время регулирования данных (Data Control Time)** – **Время регулирования данных (**не равен «0»**).** Период времени, в течение которого отправлена команда синхронизации, указывающая на режим работы ведущего устройства, и должно быть выполнено, по крайней мере, одно взаимодействие с каждым активированным ведомым устройством;
- **T_SDI** – **Задержка станции инициатора (**в диапазоне **от 0 до 0xFFFF).** Величина определяет время задержки станции инициатора.

По умолчанию установлен флажок в поле **Исп.значения по умолчанию** и значения параметров будут вычисляться автоматически, в зависимости от скорости, заданной пользователем, и параметров, установленных в GSD файлах. Редактирование параметров доступно, только если опция выключена (снят флажок с поля **Исп.значения по умолчанию**).

При редактировании параметров вручную и введении значений **T_SL**, **T_TR** меньше, чем рассчитанное минимальное значение, при компиляции проекта, в окне сообщений, отобразится сообщение об ошибке, например:

```
«Введенное значение tslot (54) меньше, чем рассчитанное минимальное значение для данной конфигурации (77)»; «Введенное значение ttr (1) меньше, чем рассчитанное минимальное значение для данной конфигурации (1055)».
```

	**ИНФОРМАЦИЯ** При наличии нескольких коммуникационных модулей на одной шине, после завершения добавления модулей ввода/вывода данных, на всех ведущих устройствах **Profibus_Master** необходимо проверить, что указаны идентичные значения скорости. Далее, вручную измените значения параметров шины T_SL и T_TR, согласно следующим формулам: Tsl = max(T_SL_0, T_SL_1, ...) - максимальное значение из всех T_SL; Ttr = Σ(T_TR_0, T_TR_1, ...) - сумма всех значений T_TR; где: T_SLi и T_TRi – значения T_SL и T_TR (рассчитанные по умолчанию) для каждого отдельного ведущего устройства; Tsl и Ttr – единые итоговые значения, которые должны быть выставлены вручную на всех ведущих устройствах **Profibus_Master**

Также при загрузке проекта осуществляется проверка всех параметров на соответствие указанным границам диапазона. В случае выхода за границы, в журнале **regul-bus-driver** появятся сообщения об ошибке в конфигурации с указанием конкретного параметра, например:

```
[E] ConfigFile: Parameter ttr: 0 is wrong (min: 1, max: 16777215 limits)
[E] CP_01_031: Cannot create config file
```

Настройка DP Slave

Для настройки параметров ведомого устройства, необходимо перейти на вкладку соответствующего **DP Slave** устройства. Для перехода в редактор дважды щелкните левой кнопкой мыши по названию ведомого устройства в окне дерева устройств. Откроется вкладка (Рисунок 19).

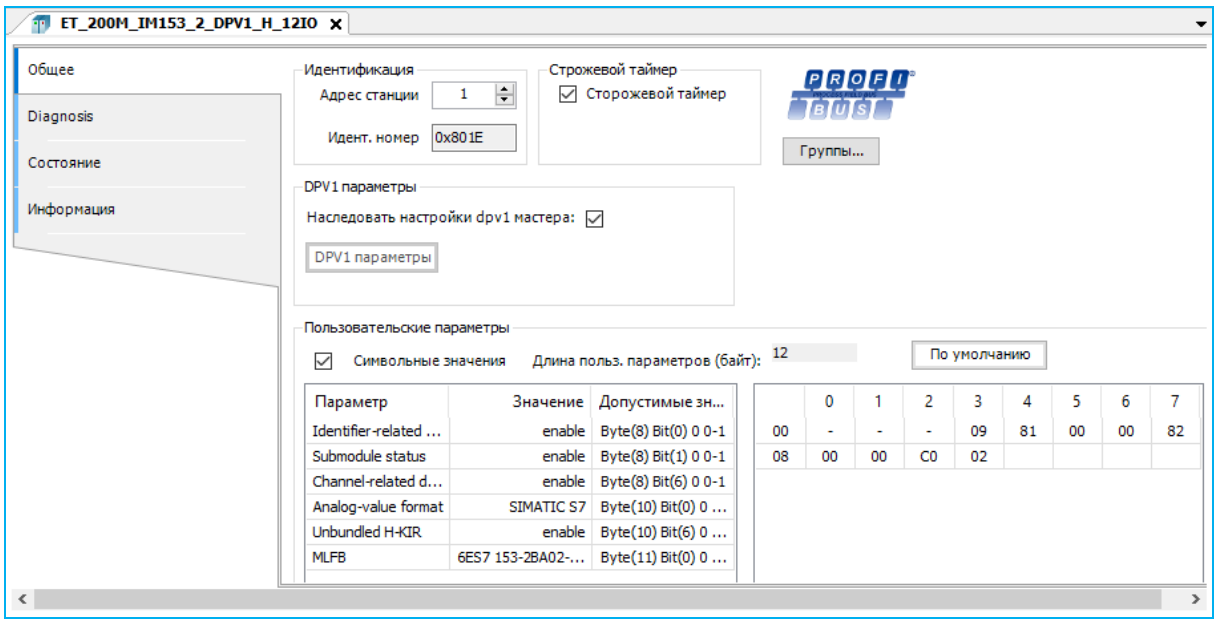


Рисунок 19 – Вкладка общих настроек ведомого устройства

На вкладке в поле **Идентификация** присутствуют следующие параметры:

- **Адрес станции.** Адрес ведомого устройства в сети PROFIBUS DP;
- **Идент.номер**. ID ведомого устройства (заданный в GSD-файле).

В поле **Сторожевой таймер** установлен по умолчанию флажок, означающий, что по истечении временного интервала (см. раздел «Настройка Profibus Master»), если ведомое устройство не получит данные от ведущего устройства, то все выходы ведомого устройства перейдут в безопасное состояние, что может свидетельствовать об отказе ведущего устройства или обрыве линии. Снятие флажка отключает сторожевой таймер на данном ведомом устройстве.

В области **DPV1 параметры** осуществляются индивидуальные настройки параметров ведомого устройства, поддерживающего DPV1-расширения. Для редактирования настроек снимите флажок в строке **Наследовать настройки dvp1 мастера**.

Если ведомое устройство не поддерживает DPV1, то настройки **параметров DPV1** будут заблокированы – кнопка ***DPV1 параметры*** будет неактивна.

Если кнопка ***DPV1 параметры*** активна, то при нажатии, открывается окно **Настройки DPV1**. В зависимости от содержащихся данных в GSD-файле ведомого устройства, будут доступны соответствующие для конфигурирования параметры DVP1 (Рисунок 20):

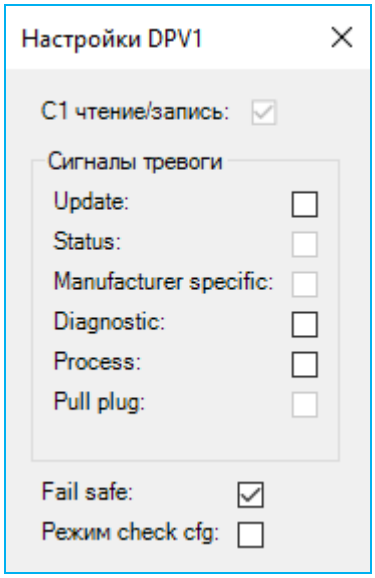


Рисунок 20 - Настройка DPV1 параметров ведомого устройства

– ☑: параметр активирован по умолчанию и недоступен для внесения изменения (заблокирован);
– ☐: параметр отсутствует и недоступен для внесения изменения (заблокирован);
– ☐: параметр присутствует и доступен для внесения изменений (установка флажка);
– ☑ : параметр активирован по умолчанию и доступен для внесения изменений (снятие флажка).

Описание параметров идентично параметрами **Настройки DPV1** Profibus Master (см. раздел «Настройка Profibus Master»).

В поле **Параметры пользователей (User parameters)**, над таблицей, в строке **Length of user parameters**, приведена информация о значении длины пользовательских параметров (байт), заданные в файле описания устройства.

В табличном виде слева приведены дополнительные параметры ведомого устройства для конфигурирования пользователем в доступном для чтения виде, определенные GSD-файлом. Колонка **Параметры** содержит наименование параметра. Значение параметра задается в столбце **Значение**. В ячейке **Значение** двойным щелчком левой кнопкой мыши можно изменить значение параметра (Рисунок 21).

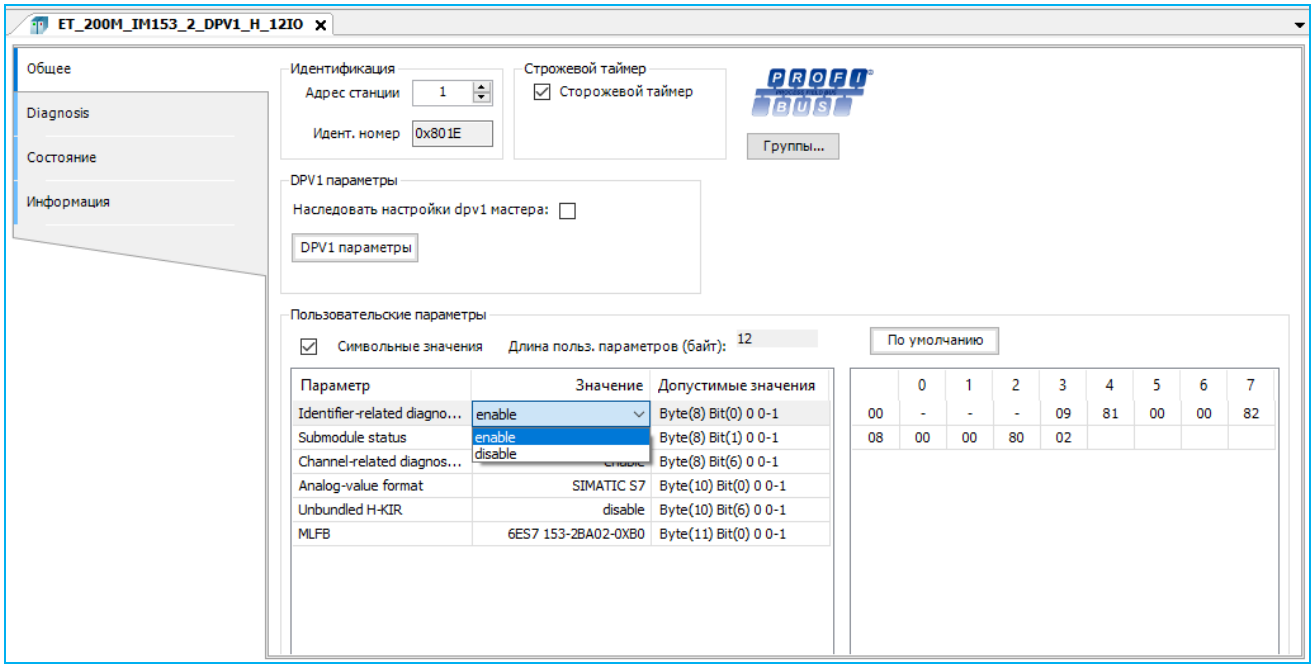


Рисунок 21 – Пользовательские параметры

В табличном виде справа представлены эти же параметры в шестнадцатеричном виде, которые также можно редактировать. Первые три байта в таблице устройств с поддержкой версии DPV0 (Рисунок 22) разблокированы и доступны для редактирования, для устройств с поддержкой DVP1 – заблокированы и определяются настройкой DPV1 параметров.

	0	1	2	3	4	5	6	7
00	00	00	00	09	81	00	00	82
08	00	00	80	02				

Рисунок 22 - Параметры в шестнадцатеричном виде для устройств версии DPV0

По умолчанию применяются *символьные имена* значений параметров. Если снять флажок в поле **Символьные значения**, то отображаются числовые значения параметров (Рисунок 23).

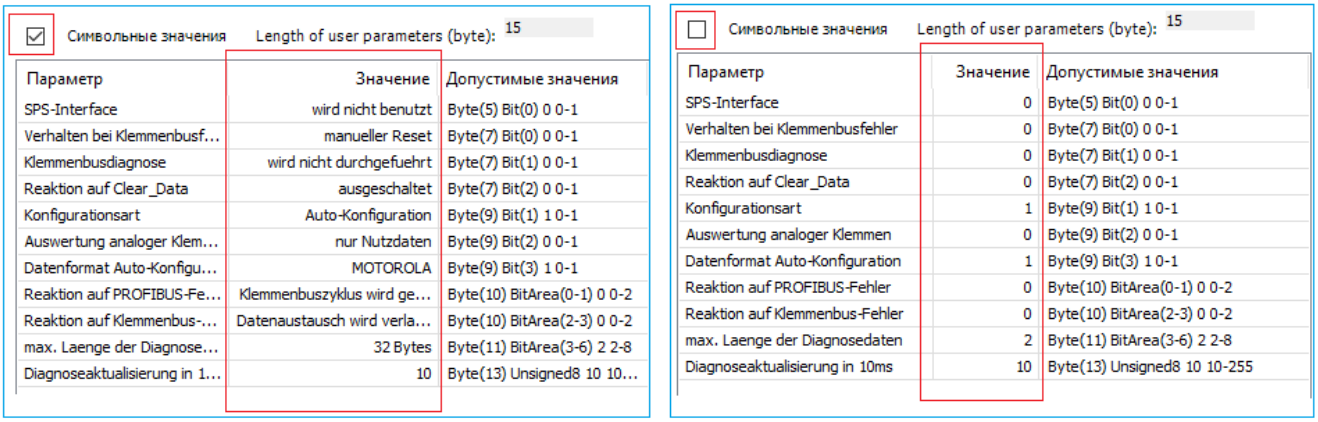


Рисунок 23 – Символьные значения

Для сброса настроек до исходных значений нажмите на кнопку ***По умолчанию***.

Описание пределов допустимых значений параметров приведены в столбце **Допустимые значения**. Например, запись: Byte(5) Bit(0) 0 0-1. Указывает на то, что данный параметр меняет

в 5 байте 0 бит. Значение по умолчанию 0. Диапазон возможных значений от 0 до 1. Аналогично с ***BitArea***, только в этом случае меняется сразу несколько бит (нумерация байт и бит с нуля). Если указан тип данных (Un)signed8, это значит, что параметр занимает весь указанный байт. Если указаны другие типы данных (Un)signed16/32 – параметр занимает следующие 2 или 4 байта, начиная с указанного.

Настройка параметров модуля ввода/вывода данных

В дереве устройств выберите необходимое ведомое устройство, двойным щелчком левой кнопкой мыши по названию откройте вкладку **Общее** (Рисунок 24).

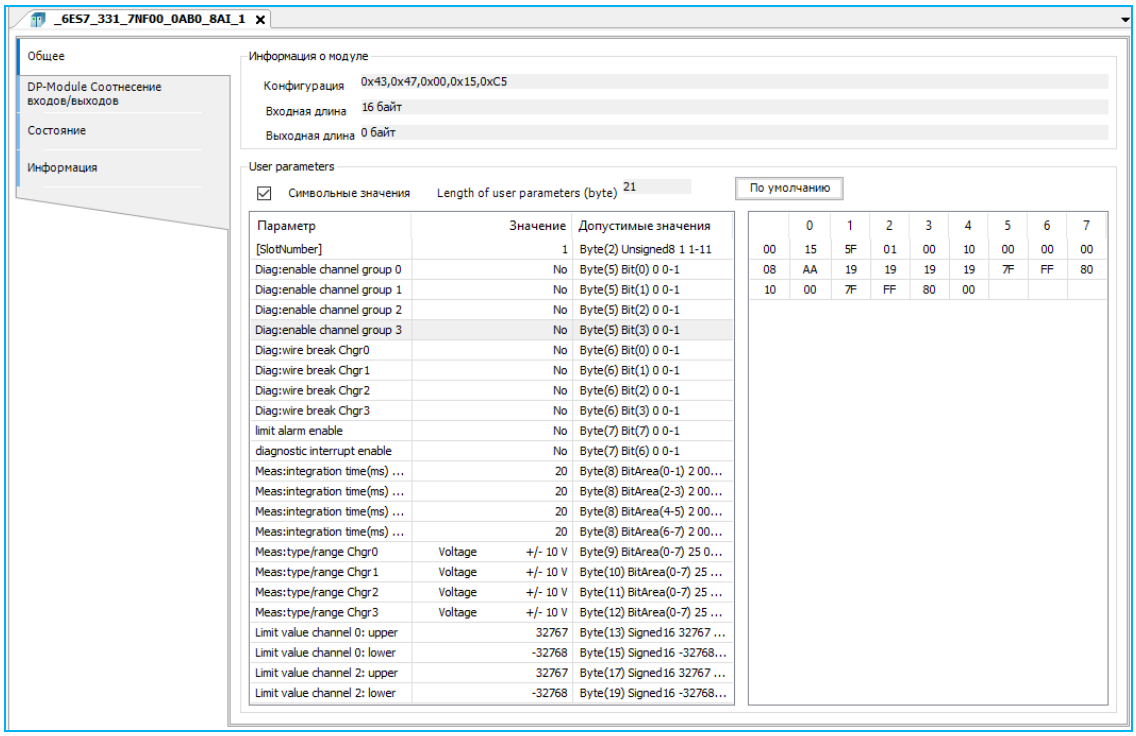


Рисунок 24 – Общие параметры ведомого устройства

На вкладке в поле **Информация о модуле** присутствуют следующие параметры:

- **Конфигурация.** Конфигурация входа/выхода ведомого модуля. Описание модуля по стандарту PROFIBUS;
- **Входная длина**. Длина входных данных (в байт) в соответствии с конфигурацией модуля в GSD-файле;
- **Выходная длина**. Длина выходных данных (в байт) в соответствии с конфигурацией модуля в GSD-файле.

Если в файле GSD описание модуля ввода/вывода имеет специфические параметры, то они будут отображены на вкладке в поле **Параметры пользователя** со своими значениями и диапазонами значений, в дополнение к параметрам ведомого устройства **(DP Slave).** Изменение параметров производится аналогично описанию, приведенному выше для ведомого устройства.

Расширенная диагностика

Диагностическая информации будет поступать со всех сконфигурированных для данного мастера ведомых устройств при условии, что активирован параметр **Расширенная диагностика** на вкладке редактора коммуникационного модуля **Редактор CP_01_031** (установлен флажок, см. раздел «Настройка параметров модуля коммуникационного процессора»). Текущая диагностическая информация будет отображаться на вкладке **Состояние** ведомого устройства (Рисунок 25).

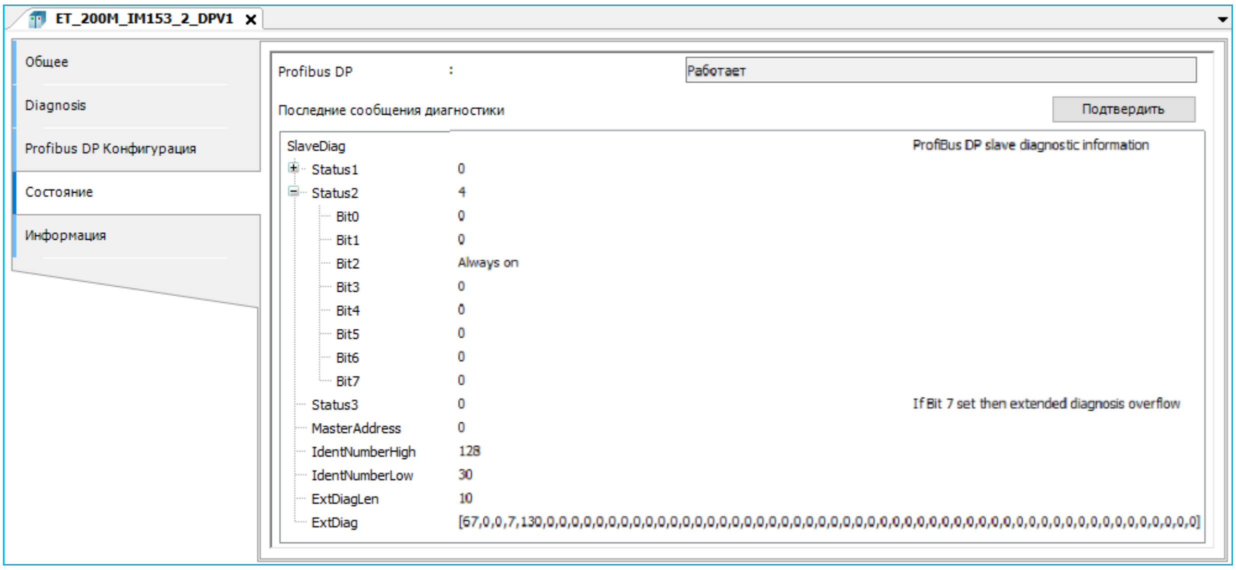


Рисунок 25 – Вкладка **Состояние**

Диагностика включает шесть диагностических байтов и дополнительные диагностические блоки. Первые шесть байт и их значение фиксированы и описаны в стандарте. После шести обязательных байт следуют один или несколько дополнительных диагностических блоков (расширенные диагностические данные – **ExtDiag,** длина расширенных диагностических данных – **ExtDiagLen**).

Детализация данных **ExtDiag** полученной расширенной диагностики будет представлена на вкладке **Diagnosis** (Рисунок 26).

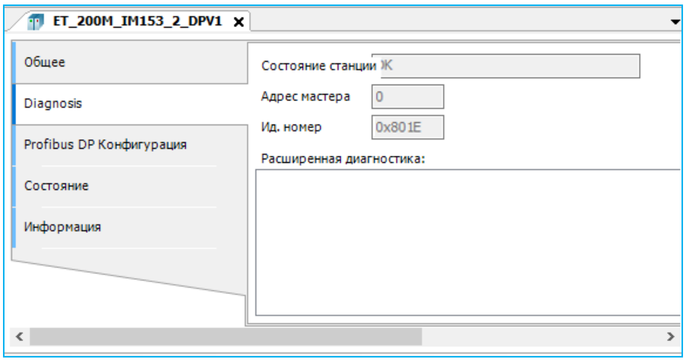


Рисунок 26 – Вкладка **Diagnosis**

Диагностические возможности Profibus DP зависят от версии (V0 или V1):

– для **DPV0** поддержана стандартная диагностика:

- **Device Related Diagnosis** (Блок расширенной диагностики, который состоит из специализированной диагностики, описание которой представлено в документации изготовителя на ведомое устройство);
- **Identifier related Diagnosis** (Блок расширенной диагностики, который предоставляет информацию о состоянии всех сконфигурированных модулей (OK, ERROR));
- **Channel Related Diagnosis** (Блок расширенной диагностики, предоставляющий детальную информацию по каналам модуля, например, для выявления короткого замыкания на аналоговом выходе модуля ввода-вывода. Каждый канал использует собственный диагностический блок);

– для **DPV1**, вместо блоков DPV0 Device Related Diagnosis, представлены расширенные диагностические блоки (Alarm, Status, Module_Status, PrmCmdAck, Red_Status и Manufacturer specific).

Привязка каналов к переменным программы

Для того, чтобы данные, получаемые от ведущего/ведомого устройства, использовать в программе контроллера, применяется инструмент **Соотнесение входов/выходов**. Он позволяет сопоставить данные PROFIBUS с пользовательскими переменными программы контроллера. На вкладке **Соотнесение входов/выходов** в методе IoDrvReadInputs выполняется передача значений из функционального блока компонента в его входные каналы, а в IoDrvWriteOutputs – выполняется передача значений из выходных каналов в переменные его функционального блока.

Более подробное описание, приведено в документе «Программное обеспечение Astra.IDE. Руководство пользователя. Привязка каналов к переменным программы».

На вкладке устройств/модулей **<Название…> Соотнесение входов/выходов** присутствует:

– для модуля коммуникационного процессора (Рисунок 27):

- присутствует параметр HwError, информирующий о статусе состояния самого модуля коммуникационного процессора. Значение параметра отображается в онлайн режиме. Параметр принимает значение TRUE (истина) или FALSE (ложь), т.е. при наличии ошибки отобразится: TRUE, а при отсутствии FALSE;

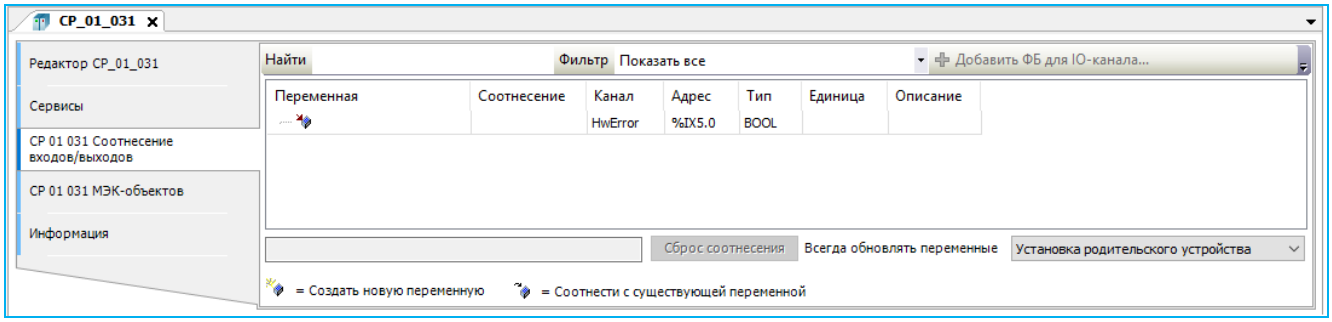


Рисунок 27 – Вкладка соотнесения входов/выходов модуля коммуникационного процессора

– для ведомого устройства (Рисунок 28):
  - отображаются значения/статусы, установленные по каждому из входов/выходов (характерные параметры канала Input_/Output_), и набор флагов состояния, в зависимости от файла описания устройства.

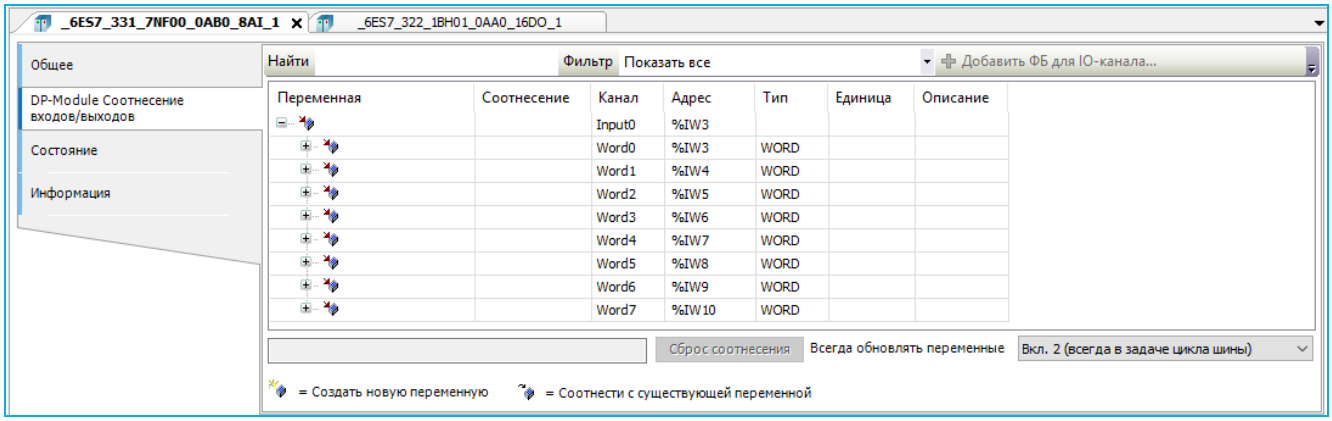


Рисунок 28 – Вкладка соотнесения входов/выходов ведомого устройства

Чтобы связать параметр ввода/вывода с переменной, на вкладке **<Название…> Соотнесение входов/выходов** дважды щелкните в строке нужного канала (Рисунок 29). Появится кнопка …, открывающая окно **Ассистент ввода** (Рисунок 30). Найдите нужную переменную. Если установлен флажок в поле **Структурированный вид**, то раскрывайте списки с помощью кнопки ⊞. Если флажок снят и переменные представлены одним большим списком, для удобства поиска воспользуйтесь фильтром.

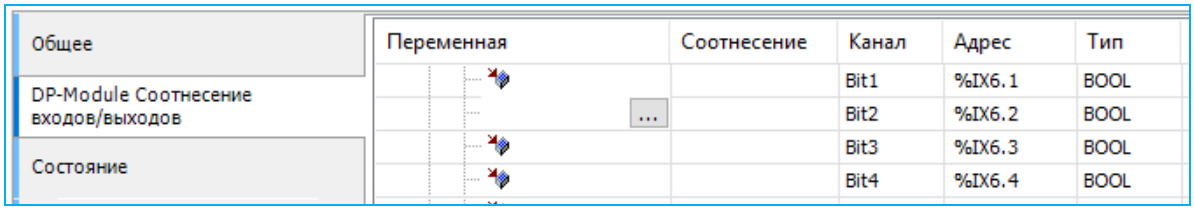


Рисунок 29 – Вызов Ассистента ввода

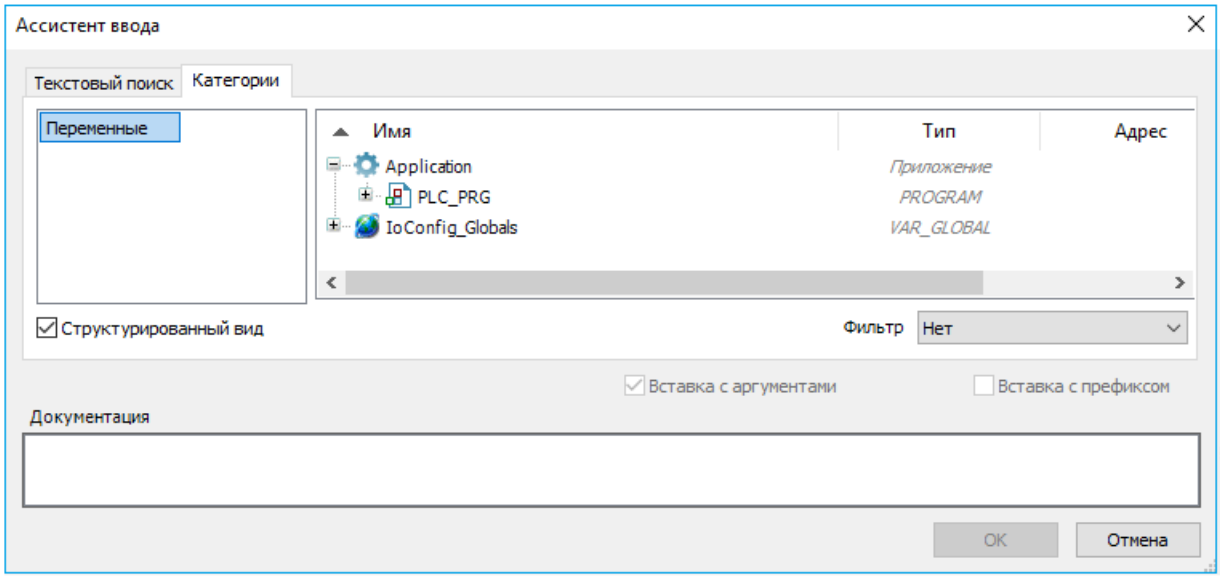


Рисунок 30 – Диалоговое окно «Ассистент ввода»

Для модуля коммуникационного процессора на вкладке **МЭК-Объектов** представлены объекты, позволяющие выполнить доступ к устройству из МЭК-приложения (Рисунок 31). С возможностью добавлять, редактировать и удалять экземпляры.

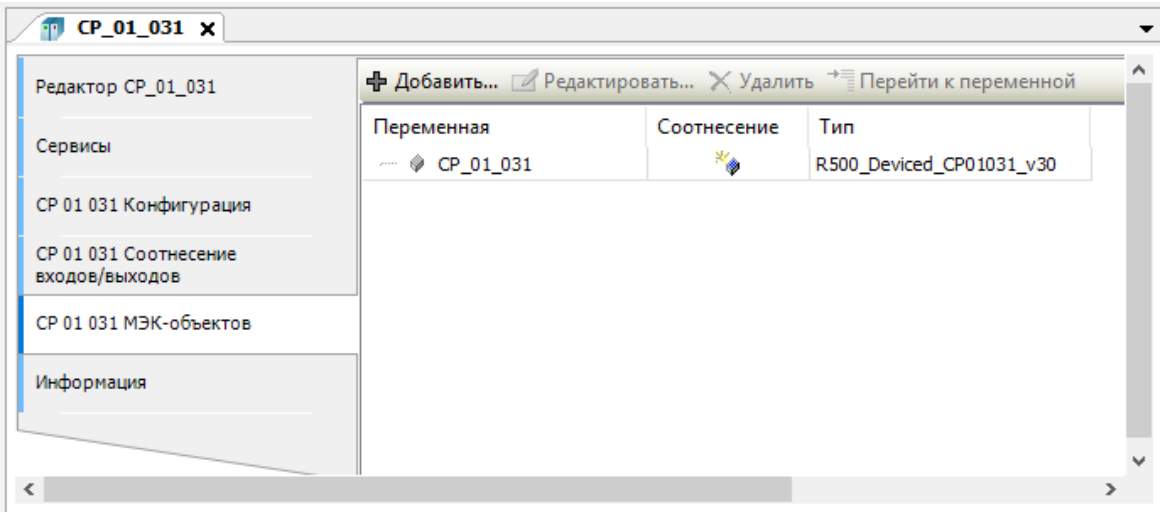


Рисунок 31 – Вкладка МЭК-объекты

ОБРАЩЕНИЕ В СЛУЖБУ ТЕХНИЧЕСКОЙ ПОДДЕРЖКИ

Для обращения в техническую поддержку Пользователю необходимо сформировать запрос на сайте технической поддержки: https://support.prosoftsystems.ru, либо отправить письмо по электронной почте: support@prosoftsystems.ru. В первом случае требуется предварительная регистрация.

Обращение обязательно должно содержать следующие сведения:

- подробное описание сложившейся ситуации;
- наименование объекта и его месторасположение;
- наименование системы автоматизации;
- модель ПЛК;
- серийный номер ПЛК;
- версия пакета обновления для среды разработки Astra.IDE;
- версия СПО-контроллера;
- архив с лог-файлами (см. документ «Astra.IDE User Guide DPA 302. Раздел «Журнал событий»);
- архив с лог-файлами, включающими в себя период времени, когда произошел отказ;
- дата и время возникновения отказа. А также периодичность и устойчивость повторения подобных отказов в случае, если такая информация имеется.

Желательно прислать проект для Astra.IDE, так как это может значительно упростить и ускорить процесс поиска причины отказа.

Для того, чтобы узнать, как получить необходимую информацию (сведений о версии Astra.IDE, версии СПО и так далее), ознакомьтесь с содержимым документа «Astra.IDE User Guide DPA 302».

ПРИЛОЖЕНИЕ А

Библиотека PsProfibus

Компонент содержит:

- функциональный блок **FB_ProfibusEventsCnt**, который предоставляет информацию о количестве полученных событий;
- функциональные блоки для работы со следующими типами сообщений:
  - «События»:
    FB_GetAlarmEventData, FB_GetStartedEventData FB_GetStoppedEventData;
  - «Запрос/Ответ»:
    FB_AlarmAck, FB_GetDiag, FB_SetGC, FB_Read, FB_Write, FB_FiCall;
- функциональный блок **FB_ExtDiagDetails** получения детализации расширенной диагностики.

Функциональный блок FB_ProfibusEventsCnt

Функциональный блок, который предоставляет информацию о количестве полученных событий:

- индикация аварии (uiAlarmsCnt);
- старт слейва (uiStartedCnt);
- останов слейва (uiStoppedCnt);
- новая диагностика слейва (uiNewDiagCnt).

Пример:

```
fbProfibusEventsCnt : FB_ProfibusEventsCnt;
```

Входные аргументы:

- интерфейсный вход, для предоставления модуля CP 01 031, ссылка на переменную *intProfibusModule* типа IProfibusModule ();
- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false).

Возвращаемое значение:

- результат счетчика очереди «Индикация аварии» *uiAlarmsCnt* типа UINT;
- результат счетчика очереди «Старт слейва» *uiStartedCnt* типа UINT;
- результат счетчика очереди «Останов слейва» *uiStoppedCnt* типа UINT;
- результат счетчика «Новая диагностика» *uiNewDiagCnt* типа UINT;

- переменная *enumStatus* типа E_StatusTypes, содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes, содержащая номер текущей ошибки.

E_StatusTypes

Содержит список статусов функциональных блоков (Attributes: qualified_only), указанные в таблице А.1.

Таблица А.1 – Статусы ФБ

Наименование	Значение	Описание
DONE	0	Функциональный блок успешно завершил работу, ошибок нет
RESET_DONE	65531	Функциональный блок успешно завершил сброс, переходит в ожидание
INIT	65532	Функциональный блок в стадии инициализации
ERROR	65533	Функциональный блок в ходе своей работы обнаружил ошибку, перешёл в состояние ошибка
BUSY	65534	Функциональный блок в состоянии исполнения
WAIT	65535	Функциональный блок в состоянии ожидания

E_ErrorTypes

Содержит список ошибок функциональных блоков (Attributes: qualified_only), указанные в таблице А.2.

Таблица А.2 – Ошибки ФБ

Наименование	Значение	Описание
ERR_OK	0	Ошибок нет
ERR_WRONG_CORE_HANDLER	10000	Неправильный указатель на обработчик данных
ERR_WRONG_SLAVE_ADR	10001	Неправильный адрес ведомого узла
ERR_CANNOT_GET_INTERNAL_DATA	10002	Невозможно получить доступ к внутренним данным
ERR_DETECTED_SECOND_CONNECTION	10003	Запущен второй экземпляр FB_ProfibusCore
ERR_EMPTY_EVENT_QUEUE	10005	Запрашиваемая очередь событий пустая
ERR_ANOTHER_CMD_EXECUTING	10006	Другой экземпляр функционального блока получения события уже запросил данные
ERR_WRITE_DATA_EXCEED_BUF	10007	Размер записываемых данных превышает внутренний буфер (240 байт)

Наименование	Значение	Описание
ERR_READ_DATA_EXCEED_BUF	10008	Размер запрашиваемых данных превышает внутренний буфер (240 байт)
ERR_STOP_IN_PROCESS	10009	Предыдущий запрос в процессе удаления
ERR_WRONG_CP_HANDLER	10010	Неправильный обработчик CP модуля
ERR_REQUEST_IN_PROCESS	10011	По данному ведомому узлу запрос уже в работе
ERR_ERROR_WRITE_DATA_ARRAY	10012	Не указан указатель на массив записи данных
ERR_ERROR_WRITE_DATA_ARRAY	10013	Не указан указатель на массив записи данных
ERR_CANNOT_SET_ERROR_DATA	10014	Ошибка копирования данных аварии, область данных не проинициализирована
ERR_INIT_LOAD_DATA_EXCEED_BUF	10015	Размер записываемых пользовательских данных превышает внутренний буфер (230 байт)
ERR_INIT_LOAD_DATA_ARRAY	10016	Не указан указатель на массив пользовательских данных
WARNING_CMD_STILL_EXECUTING	11000	Данные могут быть потеряны, если произвести останов
Диагностическая секция - 12000		
ERR_BLOCK_CANNOT_BE_REACHED	12000	Запрашиваемый блок диагностики не найден
ERR_BLOCK_LENGTH_EXCEED_OUT_BUF	12002	Считанное значение размера блока превышает выходной массив
ERR_DPV1_SHOULD_BE_ACTIVE	12003	Для данного типа блока диагностики должен у слейва быть активный DPV1 режим
ERR_EXT_DIAG_END_REACHED	12004	Достигли конец ExtDiag
ERR_EXT_DIAG_CANNOT_BE_ZERO_LENGTH	12005	ExtDiag не может быть нулевого размера
ERR_WRONG_CHANNEL_IDENT_NUMBER	12006	Channel related identifier_number превышает лимиты
ERR_DPV0_SHOULD_BE_ACTIVE	12007	Для данного типа блока диагностики должен у слейва быть активный DPV0 режим
ERR_REVISION_NUMBER_EXCEED_LIMITS	12008	Revision number превысил лимит
ERR_UNDEFINE_BLOCK_TYPE	12009	Тип выбранного блока не определен

Наименование	Значение	Описание
Диагностическая секция - 32000		
ERR_REQ_CREATE	32000	Ошибка создания запроса
ERR_REQ_SEND	32001	Ошибка посылки запроса
ERR_REQ_LIMIT_EXCEED	32002	Ошибка превышения количества повторов запроса
ERR_REQ_REPEAT_SEND	32003	Ошибка повторной посылки запроса
ERR_CANNOT_WRITE_ANSWER	32004	Ошибка записи данных во внутренний буфер
ERR_CANNOT_STOP_REQUEST	32005	Ошибка удаления запроса
ERR_REQUEST_REJECTED	32006	Ошибка обработки запроса стеком, запрос был сброшен
ERR_PROCESSING_REQUEST	32007	Ошибка обработки запроса слейвом, запрос был сброшен
ERR_STOP_REQUEST_WRONG_TYPE	32008	Ошибка удаления запроса. Запрос с неправильным типом

▸ Функциональные блоки типа «События» (EventsFB)

➢ FB_GetAlarmEventData

Функциональный блок извлекает полученное событие типа «Индикация аварии» из памяти контроллера.

Пример:

```
fbAlarmEvent : FB_GetAlarmEventData;
structAlarmData : S_AlarmInd;
```

Входные аргументы:

- интерфейсный вход, для предоставления модуля CP 01 031, ссылка на переменную *intProfibusModule* типа IProfibusModule ();
- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false).

Входные/выходные:

- структура *structData* типа S_AlarmInd, содержащая данные по типу события «Индикация аварии».

Возвращаемое значение:

- переменная *enumStatus* типа E_StatusTypes (список статусов приведен в таблице А.1), содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes (список ошибок приведен в таблице А.2), содержащая номер текущей ошибки.

➢ Структура S_AlarmInd

Структура данных события «Индикация аварии».

Содержит параметры, указанные в таблице А.3

Таблица А.3 – Параметры

Параметры	Тип	Описание
enumAlarmType	USINT	Тип аварии (E_AlarmType)
enumAlarmSpecifier	USINT	Спецификатор аварии (E_AlarmSpecifier)
bAddAck	BOOL	Необходимость дополнительной реакции пользователя, к примеру, чтение / запись каких-либо данных (по умолчанию FALSE)
usSlotNumber	USINT	Номер слота, если usSlotNumber = 0, значит аварийное сообщение относится ко всему слейву, не к конкретному модулю (по умолчанию 0)
usSequenceNumber	USINT	Порядковый номер ошибки. Нумерация происходит отдельно для каждого типа ошибки
usSlaveAddr	USINT	Адрес слейва
usLen	USINT	Фактическая длина принятых пользовательских данных аварийного сообщения
ausData	ARRAY [0..59] OF USINT	Массив пользовательских данных аварийного сообщения. Максимальный размер 60 байт

E_AlarmType

Содержит список типов аварии (Attributes: qualified_only), указанные в таблице А.4.

Таблица А.4 – Типы аварии

Наименование	Значение	Описание
EACH_TYPE	0	Зарезервировано
DIAG	1	Диагностическая авария
PROCESS	2	Авария модуля
PULL	3	Извлекли модуль из корзины
PLUG	4	Модуль вставили обратно в корзину

Наименование	Значение	Описание
STATUS	5	Статусные аварии
UPDATE	6	Аварии обновления
MAX_TYPE	7	Зарезервировано

➢ FB_GetStartedEventData

Функциональный блок извлекает полученное событие типа «Старт слейва» из памяти контроллера.

Пример:

```
fbStartedEvent : FB_GetStartedEventData;
structStartedData : S_StartedInd;
```

Входные аргументы:

- интерфейсный вход, для предоставления модуля CP 01 031, ссылка на переменную *intProfibusModule* типа IProfibusModule ();
- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false).

Входные/выходные:

- структура *structData* типа S_StartedInd, содержащая данные по типу события «Старт слейва»;

Возвращаемое значение:

- переменная *enumStatus* типа E_StatusTypes (список статусов приведен в таблице А.1), содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes (список ошибок приведен в таблице А.2), содержащая номер текущей ошибки.

➢ Структура S_StartedInd

Структура данных события «Старт слейва».

Содержит параметры, указанные в таблице А.5.

Таблица А.5 – Параметры

Параметры	Тип	Описание
usSlaveAddr	USINT	Адрес слейва
usAlarmLimit	USINT	Максимально возможное количество одновременно взведённых аварий слейва

➢ FB_GetStoppedEventData

Функциональный блок извлекает полученное событие типа «Останов слейва» из памяти контроллера.

Пример:

```
fbStoppedEvent : FB_GetStoppedEventData;
structStoppedData : S_StoppedInd;
```

Входные аргументы:

- интерфейсный вход, для предоставления модуля CP 01 031, ссылка на переменную *intProfibusModule* типа IProfibusModule ();
- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false).

Входные/выходные:

- структура *structData* типа S_StoppedInd, содержащая данные по типу события «Останов слейва»;

Возвращаемое значение:

- переменная *enumStatus* типа E_StatusTypes (список статусов приведен в таблице А.1), содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes (список ошибок приведен в таблице А.2), содержащая номер текущей ошибки.

➢ Структура S_StoppedInd

Структура данных события «Останов слейва».

Содержит параметры, указанные в таблице А.6

Таблица А.6 – Параметры

Параметры	Тип	Описание
usSlaveAddr	USINT	Адрес слейва

◗ Функциональные блоки типа «Запрос/Ответ» (ReqRespFB¶)

➢ FB_AlarmAck

Функциональный блок для отправки запроса на квитирование аварии.

Пример:

```
fbAlarmAck : FB_AlarmAck;
```

Входные аргументы:

- интерфейсный вход, для предоставления модуля CP 01 031, ссылка на переменную *intProfibusModule* типа IProfibusModule ();
- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false);
- переменная *enumType* типа E_AlarmType (список типов аварии приведен в таблице А.4), содержащая тип аварии;
- переменная *usSlotNum* типа USINT, содержащая номер слота;
- переменная *usSequenceNum* типа USINT, содержащая порядковый номер аларма;
- переменная *usSlaveAddr* типа USINT, содержащая номер слейва;

Возвращаемое значение:

- переменная *enumStatus* типа E_StatusTypes (список статусов приведен в таблице А.1), содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes (список ошибок приведен в таблице А.2), содержащая номер текущей ошибки.

➢ FB_GetDiag

Функциональный блок для отправки запроса на получение текущей диагностики.

Пример:

```
fbGetDiag : FB_GetDiag;
structDiagData : S_GetDiagData;
```

Входные аргументы:

- интерфейсный вход, для предоставления модуля CP 01 031, ссылка на переменную *intProfibusModule* типа IProfibusModule ();
- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false);
- переменная *usSlaveAddr* типа USINT, содержащая номер слейва.

Входные/выходные:

- структура *structData* типа S_GetDiagData, в которую будет помещена полученная расширенная диагностика по заданному слейву;

Возвращаемое значение:

- переменная *enumStatus* типа E_StatusTypes (список статусов приведен в таблице А.1), содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes (список ошибок приведен в таблице А.2), содержащая номер текущей ошибки.

➢ Структура S_GetDiagData

Структура, в которую будет помещена полученная расширенная диагностика по заданному слейву функциональным блоком FB_GetDiag.

Содержит параметры, указанные в таблице А.7.

Таблица А.7 – Параметры

Параметры	Тип	Описание
bDPV1Enable	BOOL	Наличие поддержки DPV1 режима
usModuleCnt	USINT	Количество модулей
usSlaveAddr	USINT	Адрес слейва
usStatus1	USINT	Station_status_1 - расшифровка iec61158-6-3 5.3.1
usStatus2	USINT	Station_status_2 - расшифровка iec61158-6-3 5.3.2
usStatus3	USINT	Station_status_3 - расшифровка iec61158-6-3 5.3.3
usMasterAddr	USINT	Diag_Master_Add - расшифровка iec61158-6-3 5.3.4
usIdentNumH	USINT	Ident_Number - расшифровка iec61158-6-3 5.3.5
usIdentNumL	USINT	Ident_Number - расшифровка iec61158-6-3 5.3.5
usDataLen	USINT	Фактическая длина полученных данных в массиве DiagnData
ausDiagData	ARRAY [0..MAX_DIAGN_LEN] OF USINT	Массив расширенной диагностики

➢ FB_SetGC

Функциональный блок для отправки специального запроса управления группами. Позволяет отправить специальный запрос как отдельному слейву, так и группе слейвов, или всем слейвам за раз. Посылаемые команды: sync, unsync, freeze, unfreeze.

Пример:

```
fbSetGC : FB_SetGC;
```

Входные аргументы:

- интерфейсный вход, для предоставления модуля CP 01 031, ссылка на переменную *intProfibusModule* типа IProfibusModule ();
- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false);
- команда *bSync* типа BOOL для заморозки выходных данных. Sync = 1, производит заморозку выходных данных модуля, повторные отправки sync = 1, приводят к синхронной выдачи данных с синхронизированных модулей, sync = 0 - это unsync команда, производится разморозка выходов;
- команда *bFreeze* типа BOOL для заморозки входных данных. Freeze = 1, производит заморозку входных данных модуля, повторные отправки freeze = 1, приводят к синхронному приёму данных синхронизированных модулей, freeze = 0 - это unfreeze команда, производится разморозка входов;
- команда *bSetClear* типа BOOL на переход в режим Clear. Не используется;
- переменная *usSlaveAddr* типа USINT, содержащая номер слейва. Может быть указан конкретный адрес слейва или 127, в этом случае запрос будет разослан всем слейвам;
- переменная *usGroup* типа USINT, содержащая номер группы. Это битовая переменная. Например, значение «1» - 1 группа, «4» - 3 группа, «5» - адресует 1 и 3 группу, «0» - ни какая группа не выбрана;

Возвращаемое значение:

- переменная *enumStatus* типа E_StatusTypes (список статусов приведен в таблице А.1), содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes (список ошибок приведен в таблице А.2), содержащая номер текущей ошибки.

➢ FB_Read

Функциональный блок для отправки запроса на чтение данных модуля. Адресация осуществляется по индексу и слоту.

Пример:

```
fbRead : FB_Read;
structReadData : S_ReadData;
```

Входные аргументы:

- интерфейсный вход, для предоставления модуля CP 01 031, ссылка на переменную *intProfibusModule* типа IProfibusModule ();

- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false);
- переменная *usSlaveAddr* типа USINT, содержащая номер слейва;
- переменная *usSlotNum* типа USINT, содержащая номер слота и используется для адресации к конкретному модулю;
- переменная *usIndex* типа USINT, содержащая номер индекса и используется для адресации к конкретным данным внутри выбранного модуля;
- переменная *usLen* типа USINT, содержащая длину считываемых данных.

Входные/выходные:

- структура *structData* типа S_ReadData, в которую будут помещены считанные данные.

Возвращаемое значение:

- переменная *enumStatus* типа E_StatusTypes (список статусов приведен в таблице А.1), содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes (список ошибок приведен в таблице А.2), содержащая номер текущей ошибки;
- структура *structErrorInfo* типа S_MS1Error с кодами ошибок запросов типа Read/Write.

➢ Структура S_ReadData

Структура, в которую будет помещены считанные данные функционального блока FB_Read.

Содержит параметры, указанные в таблице А.8.

Таблица А.8 – Параметры

Параметры	Тип	Описание
usSlaveAddr	USINT	Адрес слейва
usLen	USINT	Длина считанных данных
ausReadData	ARRAY [0..MAX_READ_WRITE_DATA_LEN] OF USINT	Массив содержащий запрошенные данные модуля

➢ Структура S_MS1Error

Структура с кодами ошибок запросов типа Read/Write.

Содержит параметры, указанные в таблице А.9.

Таблица А.9 – Параметры

Параметры	Тип	Описание
usSlaveAddr	USINT	Адрес слейва

Параметры	Тип	Описание
usErrorDecode	USINT	Задает описание для usErrorCode1 / usErrorCode2
usErrorCode1	USINT	Значение задается описанием usErrorDecode (см. IEC61158-6-3)
usErrorCode2	USINT	Значение задается описанием usErrorDecode (см. IEC61158-6-3)

➢ FB_Write

Функциональный блок для отправки запроса на запись данных. Адресация осуществляется по индексу и слоту.

Пример:

```
fbWrite : FB_Write;
structWriteData : S_WriteData;
```

Входные аргументы:

- интерфейсный вход, для предоставления модуля CP 01 031, ссылка на переменную *intProfibusModule* типа IProfibusModule ();
- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false);
- переменная *usSlaveAddr* типа USINT, содержащая номер слейва;
- переменная *usSlotNum* типа USINT, содержащая номер слота и используется для адресации к конкретному модулю;
- переменная *usIndex* типа USINT, содержащая номер индекса и используется для адресации к конкретным данным внутри выбранного модуля;
- переменная *usLen* типа USINT, содержащая длину записываемых данных;
- указатель *hData* типа RTS_IEC_HANDLE на массив записываемых данных.

Входные/выходные:

- структура *structData* типа S_WriteData, содержащая результат записи данных.

Возвращаемое значение:

- переменная *enumStatus* типа E_StatusTypes (список статусов приведен в таблице А.1), содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes (список ошибок приведен в таблице А.2), содержащая номер текущей ошибки;
- структура *structErrorInfo* типа S_MS1Error с кодами ошибок запросов типа Read/Write (см. таблицу А.9).

➢ Структура S_WriteData

Структура, содержащая результат записи данных функционального блока FB_Write.

Содержит параметры, указанные в таблице А.10.

Таблица А.10 – Параметры

Параметры	Тип	Описание
usSlaveAddr	USINT	Адрес слейва
usLen	USINT	Длина считанных данных

➢ FB_FiCall

Функциональный блок для отправки CALL запроса.

Пример:

```
fbGetDiag : FB_FiCall;
structCallInData : S_FiExtInData;
structCallOutData : S_FiCallOutData;
```

Входные аргументы:

- интерфейсный вход, для предоставления модуля CP 01 031, ссылка на переменную *intProfibusModule* типа IProfibusModule ();
- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false);
- структура *structInData* типа S_FiExtInData, содержащая входные данные.

Входные/выходные:

- структура *structOutData* типа S_FiCallOutData, в которую будет помещён результат выполнения функционального блока.

Возвращаемое значение:

- переменная *enumStatus* типа E_StatusTypes (список статусов приведен в таблице А.1), содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes (список ошибок приведен в таблице А.2), содержащая номер текущей ошибки;
- структура *structErrorInfo* типа S_MS1ExtError с кодами ошибок запроса.

➢ Структура S_FiExtInData

Входная структура данных Call запросов типа «Function Invocation».

Содержит параметры, указанные в таблице А.11.

Таблица А.11 – Параметры

Параметры	Тип	Описание
uiFiIndex	USINT	Индекс вызываемой функции
usEntityNumber	USINT	Номер объекта
usSlotNumber	USINT	Номер слота
usSlaveAddr	USINT	Адрес слейва
hUserData	RTS_IEC_HANDLE	Пользовательские данные
usUserDataLen	USINT	Длина передаваемых пользовательских данных

➢ Структура S_FiCallOutData

Структура, содержащая результирующие данные по CALL запросу.

Содержит параметры, указанные в таблице А.12.

Таблица А.12 – Параметры

Параметры	Тип	Описание
usAddr	USINT	Адрес слейва
usResLen	USINT	Фактическая длина полученных данных в массиве usaResData
usaResData	ARRAY [0..MAX_FI_ARG_ARRAY_LEN] OF USINT	Массив результирующих данных

➢ Структура S_MS1ExtError

Структура с кодом ошибки запросов типа LR/FI.

Содержит параметры, указанные в таблице А.13.

Таблица А.13 – Параметры

Параметры	Тип	Описание
usErrorCode	USINT	Код ошибки
usSlaveAddr	USINT	Адрес слейва

▸ Функциональный блок FB_ExtDiagDetails

Функциональный блок для детализации расширенной диагностики.

Пример:

```
fbExtDiagDetails : FB_ExtDiagDetails;
```

Входные аргументы:

- команда *bEnable* типа BOOL для включения/выключения функционального блока. Включается: bEnable = true и продолжают циклически выполнять свою работу до выключения: bEnable = false (по умолчанию bEnable = false);
- команда *bReset* типа BOOL на сброс текущей позиции;
- переменная *enumBlockType* типа E_BlockType, содержит тип блока расширенной диагностики;
- структура *structDiagData* типа S_GetDiagData, содержащая полученную расширенную диагностику по заданному слейву (список параметров приведен в таблице А.7);
- переменная *usDiagDataSize* типа USINT, содержащая размер расширенной диагностики;

Возвращаемое значение:

- переменная *enumStatus* типа E_StatusTypes (список статусов приведен в таблице А.1), содержащая текущий статус функционального блока;
- переменная *enumError* типа E_ErrorTypes (список ошибок приведен в таблице А.2), содержащая номер текущей ошибки;
- блок *unionExtDiagData* типа U_ExtDiagData, содержит блок данных выбранной расширенной диагностики. TYPE U_ExtDiagData : UNION.

  Union структура данных блоков расширенной диагностики:
  - структура *structAlarmBlock* тип S_AlarmBlock;
  - структура *structChannelBlock* тип S_ChannelDiagBlock;
  - структура *structDeviceBaseBlock* тип S_DeviceBaseBlock;
  - структура *structDxbLinkBlock* тип S_DxbLinkBlock;
  - структура *structIdentifierBlock* тип S_IdentifierBlock;
  - структура *structModulStatusBlock* тип S_ModulStatus;
  - структура *structRedDataBlock* тип S_RedData;
  - структура *structStatusBlock* тип S_StatusBlock;
  - переменная *usRevisionNum* тип USINT.

E_BlockType

Содержит типы блоков расширенной диагностики (Attributes: qualified_only), указанные в таблице А.14.

Таблица А.14 – Типы блоков

Наименование	Значение
DEVICE_RELATED_DIAGNOSIS	0
IDENTIFIER_RELATED_DIAGNOSIS	1
CHANNEL_RELATED_DIAGNOSIS	2
ALARM	3
STATUS	4
MODUL_STATUS	5
DXB_LINK_STATUS	6
PRM_CMD_ACK	7
RED_STATUS	8
REVISION_NUMBER	9

➢ Структура S_AlarmBlock

Структура данных блоков «Alarm» расширенной диагностики.

Содержит параметры, указанные в таблице А.15.

Таблица А.15 – Параметры

Параметры	Тип	Описание
enumAlarmType	E_ExtAlarmType (см. таблицу А.16)	Тип аварии
usSlotNumber	USINT	Номер слота в диапазоне 0-254, 255 - зарезервирован
enumAlarmSpecifier	E_AlarmSpecifier (см. таблицу А.17)	Спецификатор аварии
enumAddAck	E_AlarmAddAck (см. таблицу А.18)	Дополнительное подтверждение аварии
usSequenceNum	USINT	Порядковый номер статуса
ausDiagUserDat	ARRAY [0..(MAX_DIAG_USER_DATA - 1)] OF USINT	Массив дополнительной пользовательской информации

Таблица А.16 – Типы аварии

Наименование	Значение
RESERVED	0
DIAGNOSTIC_ALARM	1
PROCESS_ALARM	2
PULL_ALARM	3

Наименование	Значение
PLUG_ALARM	4
STATUS_ALARM	5
UPDATE_ALARM	6
MANUFACTURER_SPECIFIC	7
UNDEFINE	

Таблица А.17 – Спецификатор аварии

Наименование	Значение	Описание
NO_DIFF	0	Без изменений
SLOT_DISTURBED	1	Появилась ошибка и слот повреждён
SLOT_OK	2	Ошибка снята и слот в порядке
STILL_DISTURBED	3	Ошибка снята, но слот все ещё повреждён
UNDEFINE		Не определено

Таблица А.18 – Подтверждение аварии

Наименование	Значение	Описание
NO_ADDITIONAL_ACK	0	Без изменений
ADDITIONAL_ACK	1	Появилась ошибка и слот повреждён
UNDEFINE		Не определено

- **Структура S_ChannelDiagBlock**

Структура данных блоков «Channel_Related_Diagnosis» расширенной диагностики.

Содержит параметры, указанные в таблице А.19.

Таблица А.19 – Параметры

Параметры	Тип	Описание
usModuleNumber	USINT	Номер модуля
usChannelNumder	USINT	Номер канала
enumInOutSelection	E_ChannelInOutSelection (см. таблицу А.20)	Тип канала
enumErrorType	E_ChannelErrorType (см. таблицу А.21)	Тип ошибки канала
enumChannelType	E_Channel_Type (см. таблицу А.22)	Тип данных канала

Таблица А.20 – Тип канала

Наименование	Значение
RESERVED	0
INPUT	1
OUTPUT	2
INPUT_OUTPUT	3
UNDEFINE	4

Таблица А.21 – Тип ошибки канала

Наименование	Значение
RESERVED_0	0
SHORT_CIRCUIT	1
UNDERVOLTAGE	2
OVERVOLTAGE	3
OVERLOAD	4
OVERTEMPERATURE	5
LINE_BREAK	6
UPPER_LIMIT_EXCEEDED	7
LOWER_LIMIT_EXCEEDED	8
ERROR	9
SIMULATION_ACTIVE	10
RESERVED_1	11
RESERVED_2	12
RESERVED_3	13
RESERVED_4	14
PARAMETER_MISSING	15
PARAMETERIZATION_FAULT	16
POWER_SUPPLY_FAULT	17
FUSE_BLOWN_OPEN	18
COMMUNICATION_FAULT	19
GROUND_FAULT	20
REFERENCE_POINT_LOST	21
PROCESS_EVENT_LOST	22

Наименование	Значение
THRESHOLD_WARNING	23
OUTPUT_DISABLED	24
FUNCTIONAL_SAFETY_EVENT	25
EXTERNAL_FAULT	26
MANUFACTURER_SPECIFIC_1	27
MANUFACTURER_SPECIFIC_2	28
MANUFACTURER_SPECIFIC_3	29
MANUFACTURER_SPECIFIC_4	30
TEMPORARY_FAULT	31
UNDEFINE	

Таблица А.22 – Тип данных канала

Наименование	Значение
UNSPECIFIC_TYPE	0
ONE_BIT_TYPE	1
TWO_BIT_TYPE	2
FOUR_BIT_TYPE	3
OCTET_TYPE	4
WORD_TYPE	5
TWO_WORD_TYPE	6
RESERVED	7
UNDEFINE	

- **Структура S_DeviceBaseBlock**

Структура данных блоков “Device_Related_Diagnosis” расширенной диагностики.

Содержит параметр, указанные в таблице А.23.

Таблица А.23 – Параметр

Параметр	Тип	Описание
arrayDeviceBaseData	ARRAY [0..(DEVICE_BLOCK_DATA_LENGTH - 1)] OF USINT	Массив дополнительной пользовательской информации

➢ Структура S_DxbLinkBlock

Структура данных блоков «DXB_Link_Status» расширенной диагностики.

Содержит параметры, указанные в таблице А.24.

Таблица А.24 – Параметры

Параметры	Тип	Описание
enumStatusSpecifier	E_StatusSpecifier (см. таблицу 25)	Спецификатор статуса
ausDiagUserData	ARRAY [0..(MAX_DIAG_USER_DATA - 1)] OF USINT	Массив дополнительной пользовательской информации

Таблицу А.25 – Статусы

Наименование	Значение	Описание
NO_DIFF	0	Без изменений
STATUS_APPEARS	1	Появилась новый статус
STATUS_DISAPPEARS	2	Статус был снят
RESERVED	3	Зарезервировано
UNDEFINE		Неопределенно

➢ Структура S_IdentifierBlock

Структура данных блоков «Identifier_Related_Diagnosis» расширенной диагностики.

Содержит параметр, указанные в таблице А.26.

Таблица А.26 – Параметр

Параметр	Тип	Описание
arrayModules	ARRAY [0..MAX_MODULES] OF E_IdentifierState (см. таблицу А.27)	Массив со статусами модулей

Таблица А.27 – Идентификатор модуля

Наименование	Значение
EMPTY	0
ERROR	1
OK	2

➢ Структура S_ModulStatus

Структура данных блоков «Modul_Status» расширенной диагностики.

Содержит параметры, указанные в таблице А.28.

Таблица А.28 – Параметры

Параметры	Тип	Описание
enumStatusSpecifier	E_StatusSpecifier (см. таблицу А.29)	Спецификатор статуса
arrayModules	ARRAY [0..MAX_MODULES] OF E_ModulStatus (см. таблицу А.30)	Массив со статусами модулей

Таблица А.29 – Спецификатор статуса

Наименование	Значение	Описание
NO_DIFF	0	Без изменений
STATUS_APPEARS	1	Появился новый статус
STATUS_DISAPPEARS	2	Статус был снят
RESERVED	3	Зарезервировано
UNDEFINE		Неопределенно

Таблица А.30 – Статусное состояние модуля

Наименование	Значение
EMPTY	0
DATA_VALID	1
DATA_INVALID	2
DATA_INVALID_WRONG_MODULE	3
DATA_INVALID_NO_MODULE	4

➢ Структура S_RedData

Структура данных блоков «PrmCmdAck/Red_Status» расширенной диагностики.

Содержит параметры, указанные в таблице А.31.

Таблица А.31 – Параметры

Параметры	Тип	Описание
enumRedSpecifier	E_RedSpecifier (см. таблицу А.32)	Спецификатор блоков резервирования
usSequenceNum	USINT	Порядковый номер последней обработанной команды PrmCmd
structFunction	S_Function	Структура содержит статусы последних выполненных запросов
structRedStatus1	S_RedStatus	Структура, содержащая статусы компонентов
structRedStatus2	S_RedStatus	Структура, содержащая статусы компонентов

Параметры	Тип	Описание
usRedStatus3	USINT	User specific

Таблица А.32 – Спецификатор redundancy статуса

Наименование	Значение	Описание
NO_DIFF	0	Без изменений
RESERVED	1	Зарезервировано
UNDEFINE		Неопределенно

➢ Структура S_Function

Структура, содержащая описатели битов поля «Function».

Содержит описатели, указанные в таблице А.33.

Таблица А.33 – Описатели

Наименование	Тип	Описание
enumBit0	E_Reserved **Наименование** / **Значение** **RESERVED** / 0	Описатель статуса запроса «Reserved»
enumBit1	E_PrimaryReq **Наименование** / **Значение** **NO_PRIMARY_REQ_EXECUTED** / 0 **PRIMARY_REQ_EXECUTED** / 1	Описатель статуса запроса «Primary»
enumBit2	E_StartMS1Req **Наименование** / **Значение** **NO_START_MS1_REQ_EXECUTED** / 0 **START_MS1_REQ_EXECUTED** / 1	Описатель статуса запроса «Start MS1»
enumBit3	E_StopMS1Req **Наименование** / **Значение** **NO_STOP_MS1_REQ_EXECUTED** / 0 **STOP_MS1_REQ_EXECUTED** / 1	Описатель статуса запроса «Stop MS1»
enumBit4	E_CheckPropReq **Наименование** / **Значение** **NO_CHECK_PROP_REQ_EXECUTED** / 0 **CHECK_PROP_REQ_EXECUTED** / 1	Описатель статуса запроса «Check_Properties»

<table>
<tr><th>Наименование</th><th>Тип</th><th>Описание</th></tr>
<tr><td>enumBit5</td><td>E_Reserved<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>RESERVED</td><td>0</td></tr></table></td><td>Описатель статуса запроса «Reserved»</td></tr>
<tr><td>enumBit6</td><td>E_MasterStateClearReq<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>NO_MASTER_STATE_CLEAR_REQ_EXECUTED</td><td>0</td></tr><tr><td>MASTER_STATE_CLEAR_REQ_EXECUTED</td><td>1</td></tr></table></td><td>Описатель статуса запроса «MasterStateClear»</td></tr>
<tr><td>enumBit7</td><td>E_Reserved<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>RESERVED</td><td>0</td></tr></table></td><td>Описатель статуса запроса «Reserved»</td></tr>
</table>

➢ Структура S_RedStatus

Структура, содержащая описатели битов поля «Red_Status1/Red_Status2».

Содержит описатели, указанные в таблице А.34.

Таблица А.34 – Описатели

<table>
<tr><th>Наименование</th><th>Тип</th><th>Описание</th></tr>
<tr><td>enumBit0</td><td>E_Backup<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>NOT_BACKUP_STATE</td><td>0</td></tr><tr><td>BACKUP_STATE</td><td>1</td></tr></table></td><td>Описатель состояния «Backup»</td></tr>
<tr><td>enumBit1</td><td>E_Primary<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>NOT_PRIMARY_STATE</td><td>0</td></tr><tr><td>PRIMARY_STATE</td><td>1</td></tr></table></td><td>Описатель состояния «Primary»</td></tr>
<tr><td>enumBit2</td><td>E_HwDefect<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>NOT_HW_DEFECT</td><td>0</td></tr><tr><td>HW_DEFECT</td><td>1</td></tr></table></td><td>Описатель состояния «HW-Defect»</td></tr>
</table>

<table>
<tr><th>Наименование</th><th>Тип</th><th>Описание</th></tr>
<tr><td>enumBit3</td><td>E_DataExchange<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>NOT_DATA_EXCHANGE_STATE</td><td>0</td></tr><tr><td>DATA_EXCHANGE_STATE</td><td>1</td></tr></table></td><td>Описатель состояния «DataExchange»</td></tr>
<tr><td>enumBit4</td><td>E_FailSafe<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>NOT_FAIL_SAFE_STATE</td><td>0</td></tr><tr><td>FAIL_SAFE_STATE</td><td>1</td></tr></table></td><td>Описатель состояния «Fail Safe»</td></tr>
<tr><td>enumBit5</td><td>E_DatarateSet<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>RESERVED</td><td>0</td></tr></table></td><td>Описатель состояния «DatarateSet»</td></tr>
<tr><td>enumBit6</td><td>E_TohStarted<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>NOT_TOH_STATED</td><td>0</td></tr><tr><td>TOH_STATED</td><td>1</td></tr></table></td><td>Описатель состояния «TohStarted»</td></tr>
<tr><td>enumBit7</td><td>E_Reserved<table><tr><th>Наименование</th><th>Значение</th></tr><tr><td>RESERVED</td><td>0</td></tr></table></td><td>Описатель статуса «Reserved»</td></tr>
</table>

➢ Структура S_StatusBlock

Структура данных блоков «Status» расширенной диагностики.

Содержит параметры, указанные в таблице А.35.

Таблица А.35 – Параметры

Параметры	Тип	Описание
enumStatusType	E_StatusType (см. таблицу А.36)	Тип статуса
usSlotNumber	USINT	Номер слота в диапазоне 0-254, 255 - зарезервирован
enumStatusSpecifier	E_StatusSpecifier (см. таблицу А.29)	Спецификатор статуса
ausDiagUserData	ARRAY [0..(MAX_DIAG_USER_DATA - 1)] OF USINT	Массив дополнительной пользовательской информации

Таблица А.36 – Тип статуса

Наименование	Значение
RESERVED	0
STATUS_MESSAGE	1
MODUL_STATUS	2
DXB_LINK_STATUS	3
PRM_CMD_ACK	30
RED_STATUS	31
MANUFACTURER_SPECIFIC	32 (32…127)
UNDEFINE	128