УТВЕРЖДЕН
КДСА.426471.004 РП УЛ

КОНТРОЛЛЕР ПРОГРАММИРУЕМЫЙ ЛОГИЧЕСКИЙ *MKLogic-500®*

Руководство по программированию
КДСА.426471.004 РП 1.0_02

Уфа 2020 г.

Содержание

Введение

Настоящее руководство по программированию (далее – РП) содержит сведения, необходимые для конфигурирования и программирования изделия **Контроллер программируемый логический MKLogic-500** (далее – изделие) на языках МЭК-61131-3 специалистами АСУТП.

В РП приведены сведения об установке и настройке среды разработки, конфигурировании изделия и особенностях работы с модулями ввода-вывода и коммуникационными модулями при написании технологических программ в среде разработки.

Настоящее РП распространяется на изделие **Контроллер программируемый логический MKLogic-500**.

Конфигурирование и программирование изделия должно осуществляться специально обученным и изучившим настоящее РП обслуживающим персоналом.

В РП приняты следующие условные обозначения:

Обозначение	Комментарий
⚠ **Внимание**	Информация, на которую следует обратить особое внимание
`Шрифт Courier New`	Названия функций, параметров, переменных, папок, примеры листинга, названия папок, документов
Подчёркнутое начертание	Ссылки на разделы руководства, рисунки и таблицы
«Текст в кавычках»	Названия пунктов меню, кнопок

Глава 1 Подготовка к работе с изделием

Перед началом конфигурирования и программирования изделия следует обязательно ознакомиться со следующими документами:

- КДСА.426471.004 РЭ_00 Контроллер программируемый логический MKLogic-500 Руководство по эксплуатации;
- Спецификация прикладного уровня и коммуникационного профиля CANopen CiA 301 (CANopen_CiA_301_spec_rus);
- ISaGRAF 6 (cam_ISa6_ru);
- ISaGRAF 6 - Начальное руководство (gst_ISa6_ru);
- Браузер перекрестных ссылок (crb_ISa6_ru);
- Глоссарий ISaGRAF (gloss_ISa6_ru);
- Лицензирование (lmr_ISa6_ru);
- Механизм обеспечения отказоустойчивости (fvr_ISa6_ru);
- Монтаж ввода-вывода (iow_ISa6_ru);
- Руководство по языкам (lrf_ISa6_ru);
- Связывания (bnd_ISa6_ru);
- Словарь (dct_ISa6_ru);
- Стандартные операции (lrso_ISa6_ru);
- Функции (lrsf_ISa6_ru);
- Функциональные блоки (lrsb_ISa6_ru);
- Язык ST (stx_ISa6_ru).

Ознакомление со следующими документами является желательным, но не обязательным:

- Единая платформа автоматизации. Параметры среды разработки. (ode_ru);
- Генерация документов (dgn_ISa6_ru);
- Язык SAMA (sama_ISa6_ru);
- Язык SFC (sfc_ISa6_ru);
- Язык LD (ldr_ISa6_ru);
- Язык FBD (fbd_ISa6_ru);
- Язык IEC 61499 (iec_ISa6_ru);
- Нормативные функциональные блоки (lrnb_ISa6_ru).

Все вышеперечисленные документы поставляются вместе со средой разработки ACP Workbench ISaGRAF 6.5 и располагаются на диске в папке `Docs` (здесь и далее, все пути и имена файлов приводятся относительно папки с установочным комплектом ACP Workbench ISaGRAF 6.5).

Глава 2 Работа в среде разработки ACP Workbench ISaGRAF 6.5

2.1 Установка и настройка среды разработки ACP Workbench ISaGRAF 6.5

2.1.1 Установка предварительно требуемых приложений

Среда разработки ACP Workbench ISaGRAF 6.5 (далее среда разработки ACP) в ходе своей установки устанавливает сама почти все требуемые приложения, кроме Microsoft .NET Framework 4.5.1. Перед началом установки настоятельно рекомендуется установить его самостоятельно, открыв папку `Prerequisites\Microsoft .net\` и запустив файл `NDP451-KB2858728-x86-x64-AllOS-ENU.exe`.

2.1.2 Установка среды разработки ACP Workbench ISaGRAF 6.5

Для установки среды разработки ACP следует запустить программу–инсталлятор (файл `Setup.exe`) и следовать указаниям мастера установки. При выборе языка рекомендуется выбрать английский язык.

Установка может занять достаточно много времени в силу того, что устанавливается среда программирования MS Visual Studio 2013 Shell.

После завершения установки среды разработки ACP следует исправить её конфигурационные файлы, согласно рекомендациям документа: `Особенность работы ACP.pdf`.

2.1.3 Регистрация среды разработки ACP Workbench ISaGRAF 6.5

После установки среды разработки ACP необходимо её зарегистрировать согласно рекомендациям документа `Активация ISaGRAF ACP 6.pdf`. Без активации среда разработки не позволяет взаимодействовать с модулями центрального процессора изделия.

2.1.4 Установка плагина MK500 IODevice

Для конфигурирования модулей центрального процессора изделия необходимо установить плагин MK500 IODevice для среды разработки ACP. Для этого следует запустить файл `Plugin\MK500_IODevice_Setup_2.0.0.6.exe` и следовать указаниям мастера установки.

2.2 Создание проекта в среде разработки ACP Workbench ISaGRAF 6.5

Для создания проекта в среде разработки ACP следует выполнить следующие действия:

1) Запустить экземпляр приложения среды разработки ACP, дважды кликнув по ярлыку «ISaGRAF 6.5» на рабочем столе либо выполнив «Пуск»-«Все программы»-«ISaGRAF 6.5»-«Automation Collaborative Platform 6.5 - eng».

2) После запуска среды разработки ACP следует создать проект, выполнив «FILE»-«New»-«Project…» (Рис. 1).

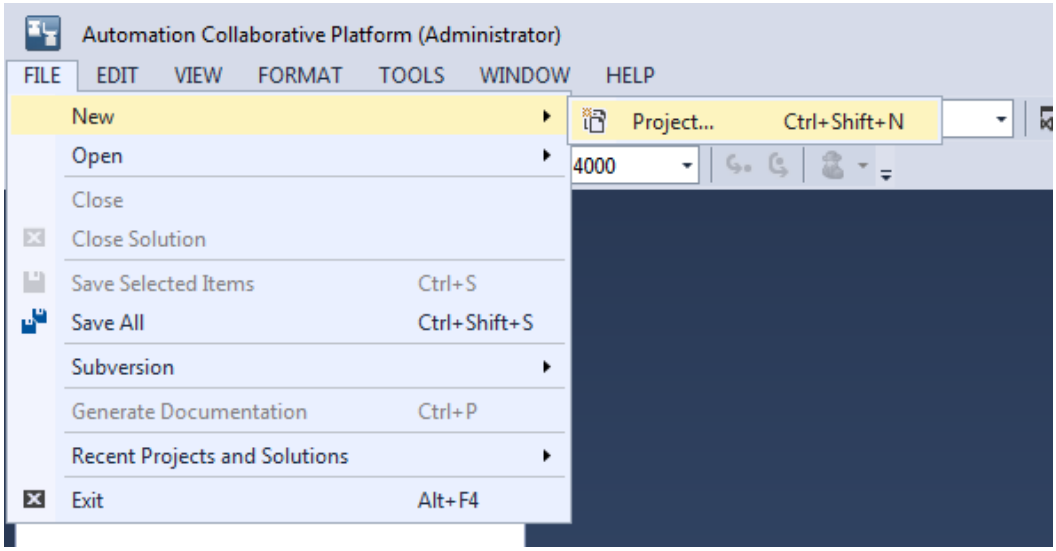


Рис. 1 – Создание нового проекта

3) В окне создания нового проекта следует выбрать шаблон ISaFREE_TPL, задать имя проекта, папку размещения нового проекта и имя решения. Если планируется использовать систему контроля версий, следует выбрать опцию «Add to Subversion» (Рис. 2). Затем следует нажать кнопку «OK».

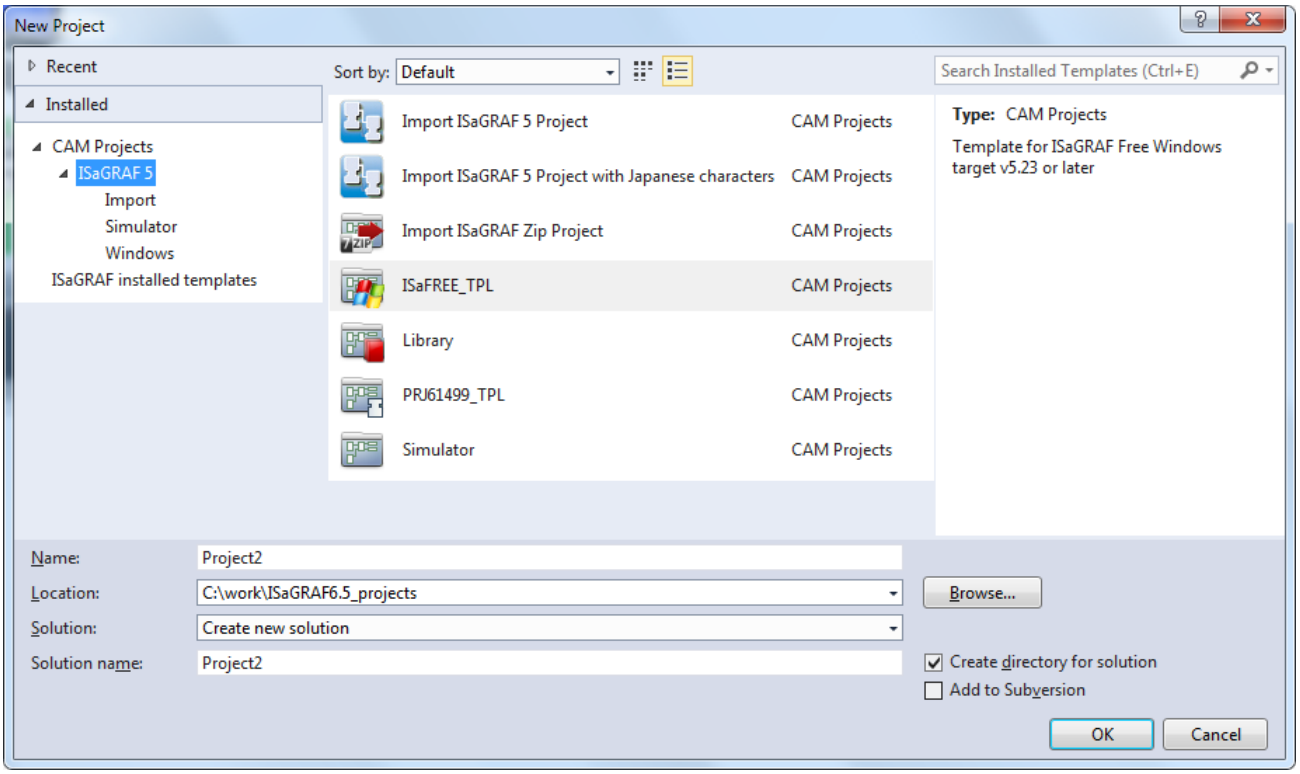


Рис. 2 – Окно создания нового проекта

4) Далее следует импортировать файл определения свойств модулей изделия (далее – tdb-файл). См. п. 2.5. Для этого следует в контекстном меню проекта выбрать «Import»-«Import Target Definitions» (Рис. 3).

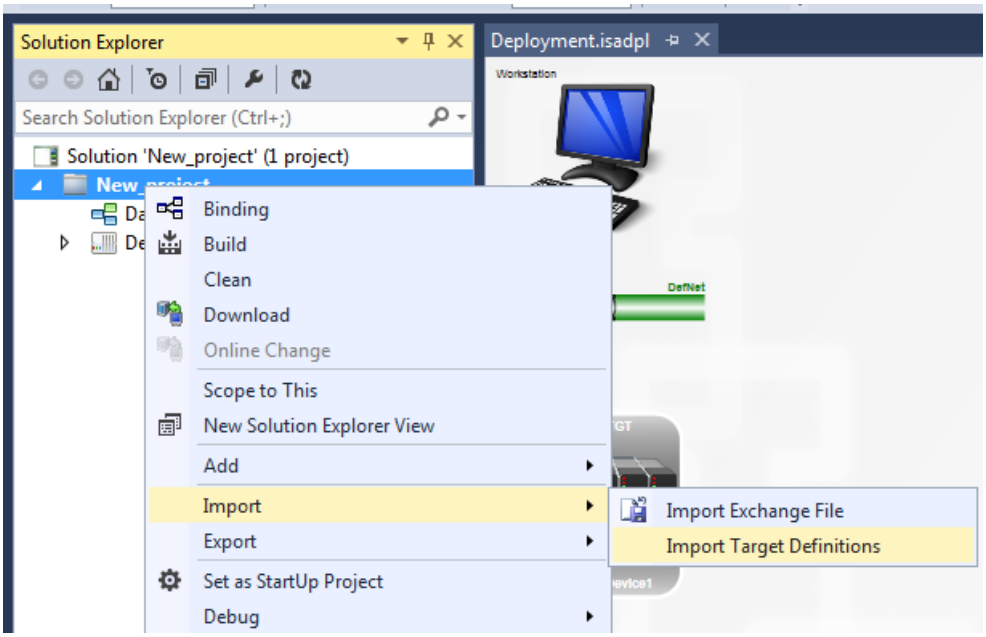


Рис. 3 – Импорт tdb-файла

В диалоговом окне следует выбрать актуальный tdb-файл (п. 2.3) и нажать кнопку «Открыть» (Рис. 4).

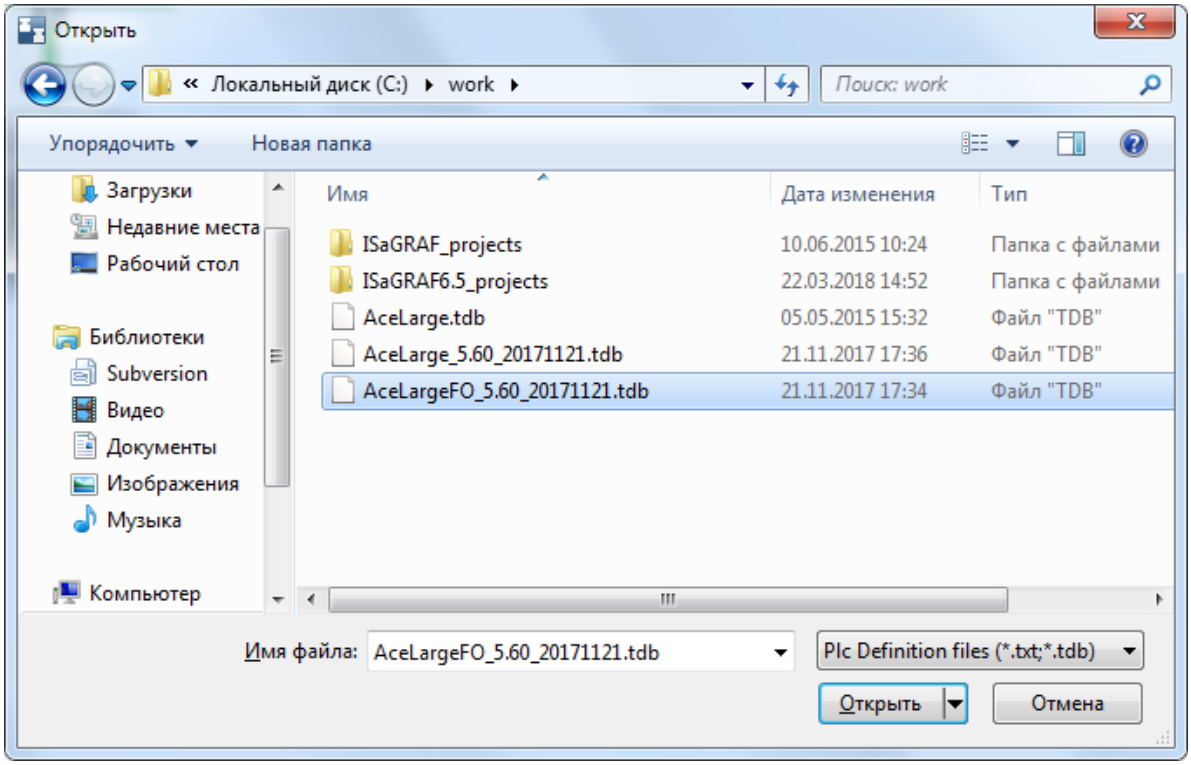


Рис. 4 – Окно импорта tdb-файла

Процесс импорта tdb-файла занимает несколько минут и сильно загружает процессор персонального компьютера.

После завершения импорта tdb-файла в блокноте будет открыт журнал импорта. Его можно закрывать сразу.

5) Для применения импортированного tdb-файла следует выбрать в дереве проекта устройство, и в окне «Properties» в секции «Hardware» для параметра «Target» выбрать «NFT_MK501_2-FO» (Рис. 5). В ходе применения новых настроек будет выведено окно с перечнем отличий между старыми и новыми параметрами, для окончательного применения параметров следует нажать кнопку «Yes».

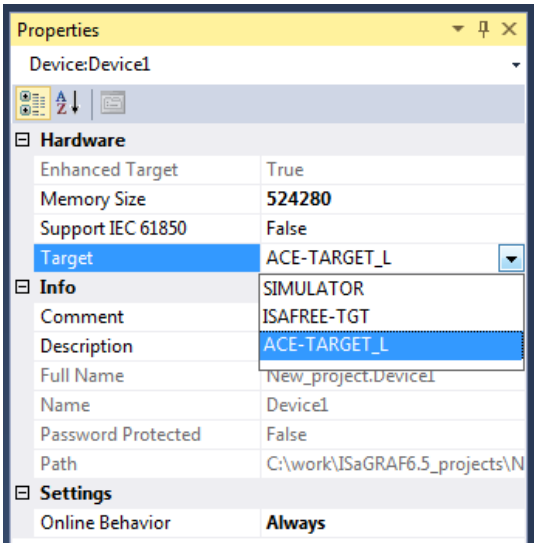


Рис. 5 – Применение импортированного tdb-файла

2.3 Настройка и диагностика модулей центрального процессора изделия

Модули центрального процессора изделия выпускаются с предустановленными сетевыми настройками. Для установки необходимых сетевых настроек, а также для настройки временных параметров и для выполнения диагностики следует использовать плагин MK500 IODevice.

Для вызова окна плагина MK500 IODevice следует выбрать в дереве проекта устройство и в его контекстном меню выбрать пункт «MK500 IODevice» (Рис. 6). Исходный вид окна плагина MK500 IODevice приведён на Рис. 7.

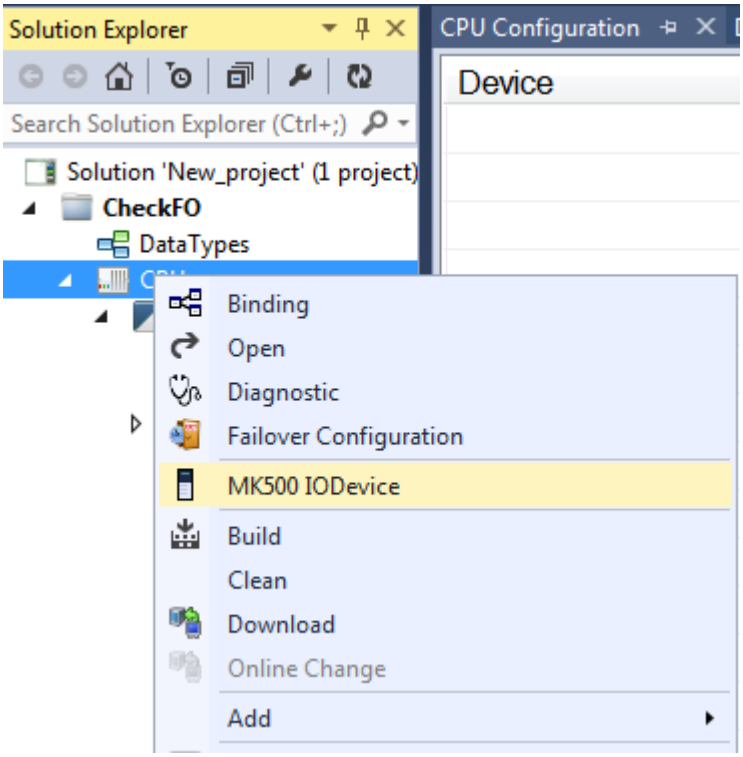


Рис. 6 – Вызов плагина MK500 IODevice

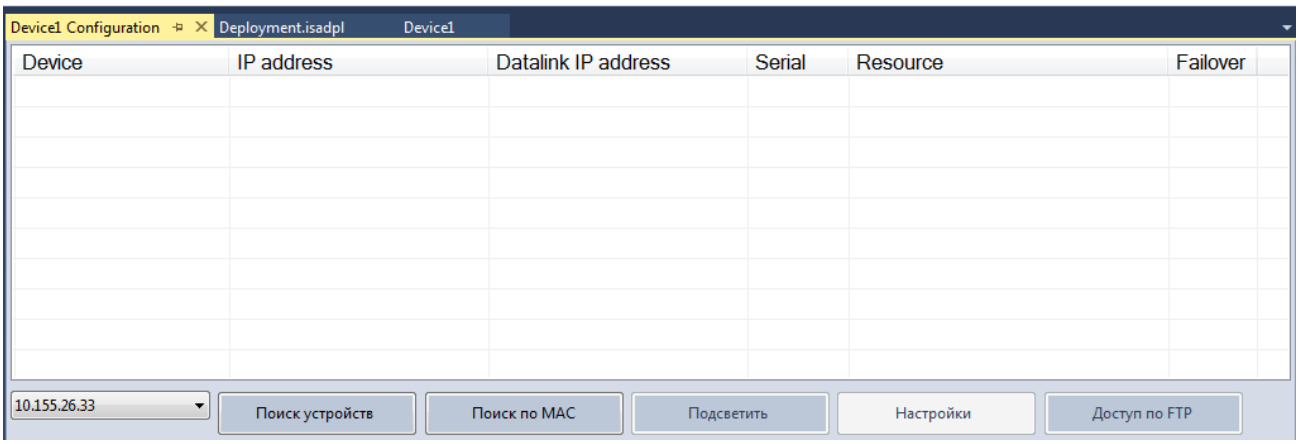


Рис. 7 – Исходный вид окна плагина MK500 IODevice

2.3.1 Подключение к модулю центрального процессора изделия с помощью плагина MK500 IODevice

Для начала настройки модуля центрального процессора изделия (далее модуль CPU) следует выполнить следующие операции:

1) С помощью сетевого кабеля подключить порт ETH1 настраиваемого модуля CPU непосредственно к порту рабочего ПК (с установленной средой разработки ACP), либо к общему с рабочим ПК сетевому коммутатору.

2) В среде разработки ACP открыть окно плагина MK500 IODevice и выбрать IP-адрес сетевого интерфейса, к которому подключён настраиваемый модуль CPU; затем нажать кнопку «Поиск устройства». В течение 10 секунд будет выполняться сканирование сети с целью найти и опознать все доступные модули центрального процессора изделия. По окончании поиска таблица будет заполнена перечнем модулей CPU, доступных для

настройки и диагностики (Рис. 8). В таблицу выводятся следующая информация для каждого обнаруженного модуля CPU:

- Наименование типа модуля CPU;
- IP-адрес+шлюз для сетевого интерфейса в роли IP (п. 2.3.2);
- IP-адрес+шлюз для сетевого интерфейса в роли Datalink IP (п. 2.3.2);
- Серийный номер модуля CPU;
- Имя ресурса выполняемой в модуле CPU программы пользователя;
- Поддержка модулем CPU режима резервирования Failover.

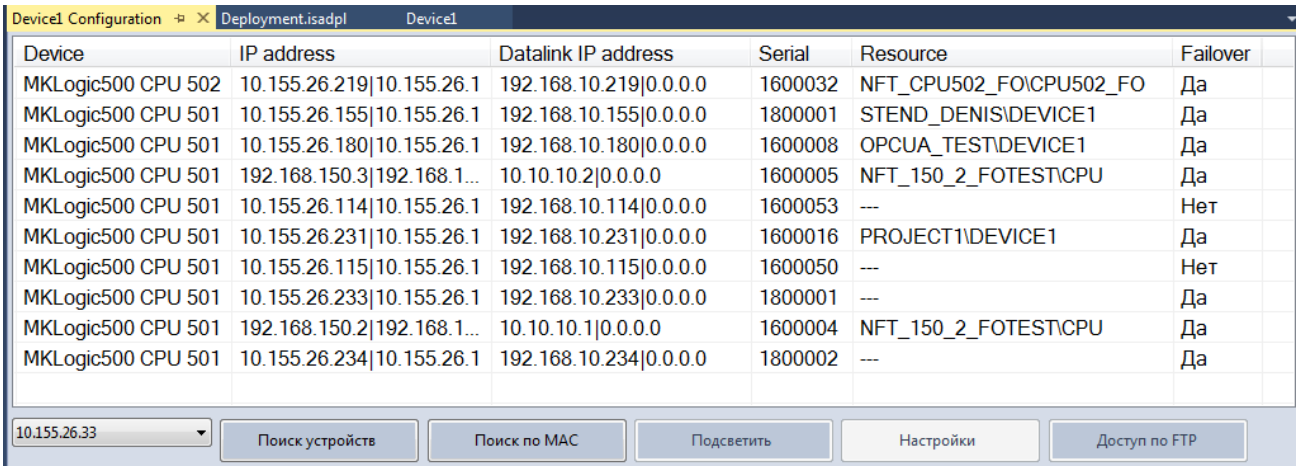


Рис. 8 – Окно плагина MK500 IODevice с найденными модулями центрального процессора

3) Если поиск не дал результатов, следует провести поиск модуля по MAC-адресу. Для этого в окне плагина MK500 IODevice следует нажать кнопку «Поиск по MAC», в открывшемся окне ввести MAC-адрес порта ETH1 желаемого модуля (MAC-адреса портов модулей CPU нанесены на боковую стенку корпуса) и нажать кнопку «OK» (Рис. 9).

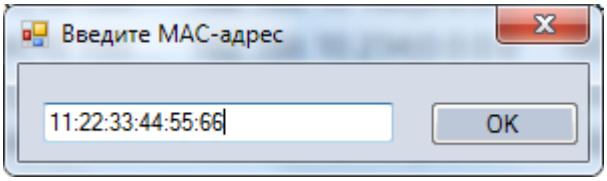


Рис. 9 – Окно ввода MAC-адреса модуля CPU

4) При выборе в таблице строки модуля и нажатии на кнопку «Подсветить» модуль в течение 5 секунд мигает всеми индикаторами лицевой панели. Это позволяет идентифицировать модуль в случае, когда сетевые настройки модулей CPU не позволяют этого сделать.

5) При необходимости скопировать из модуля CPU актуальные tdb-файлы или записать в модуль настроечные файлы следует выбрать в таблице строку с модулем и нажать кнопку «Доступ по FTP». В открывшемся окне проводника можно найти актуальные tdb-файлы для данного модуля CPU, в него же можно скопировать настроечные файлы (Рис. 10).

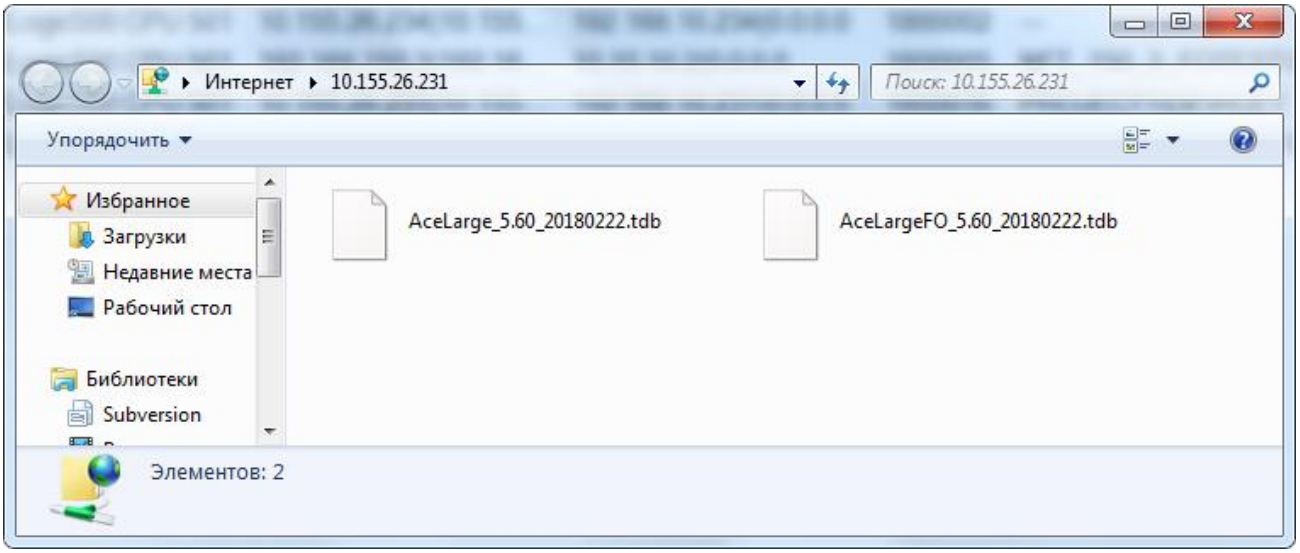


Рис. 10 – Окно проводника в режиме доступа по FTP к модулю CPU

6) Для выполнения непосредственно настройки и/или диагностики модуля CPU следует выбрать в таблице строку с модулем и нажать кнопку «Настройки» либо дважды кликнуть на этой строке.

2.3.2 Настройка и диагностика модуля CPU с помощью плагина MK500 IODevice

Открывшееся окно настроек модуля CPU имеет 4 раздела, открывающиеся при нажатии соответствующего пункта списка в левой части окна (Рис. 11). Общими для всех разделов являются кнопки «Прочитать из CPU» и «Закрыть» в нижнем правом углу окна. Кнопку «Прочитать из CPU» следует использовать для обновления значений в текущем разделе окна настроек.

Раздел «Сетевые настройки» предназначен для настройки сетевых интерфейсов модуля CPU, а также для назначения ролей сетевым интерфейсам и определения параметров режима резервирования Failover (п. 2.4).

Раздел «Настройки времени» предназначен для настройки времени и часового пояса часов реального времени модуля CPU, а также для настройки NTP-сервера модуля CPU.

Раздел «Статус приложений» предназначен для диагностики работы системного программного обеспечения модуля CPU, а также для перезапуска системного программного обеспечения модуля CPU в случае непредвиденного отказа.

Раздел «Программа пользователя» предназначен для работы с журналом сообщений системы исполнения ISaGRAF 5.60, а также для удаления программы пользователя с модуля CPU.

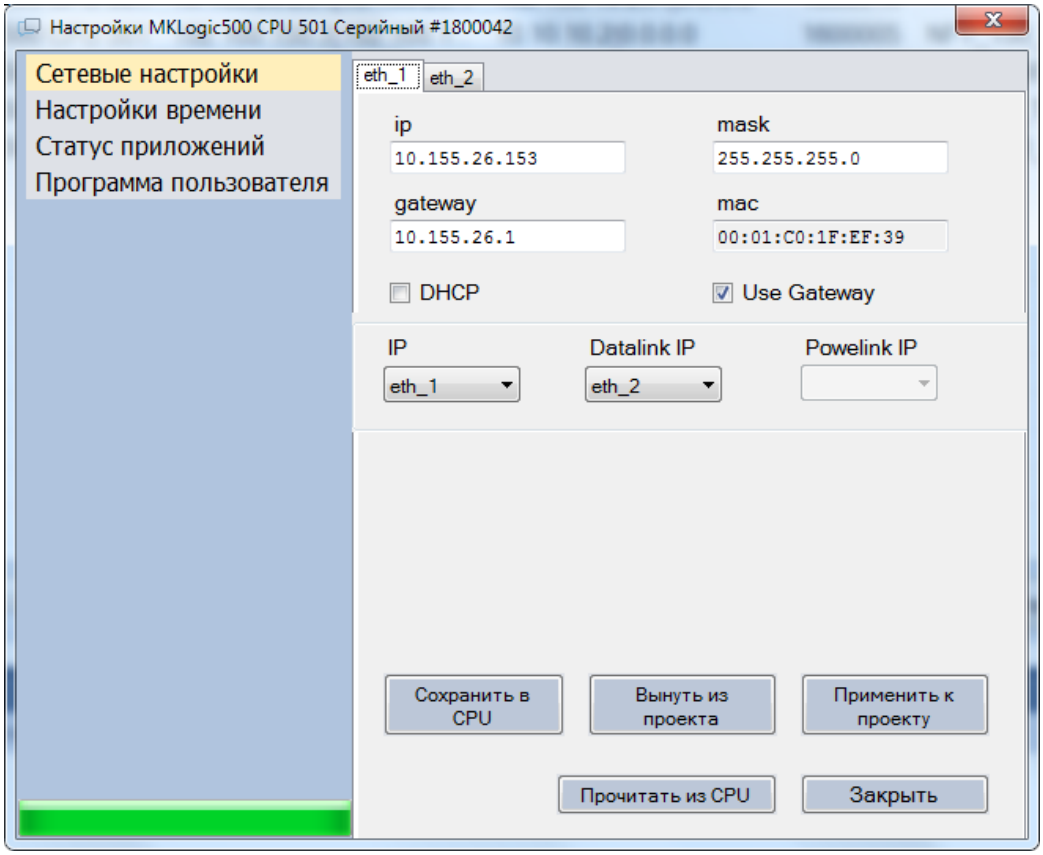


Рис. 11 – Окно сетевых настроек модуля CPU без поддержки Failover

Ниже будут более подробно разобраны особенности работы во всех четырёх разделах.

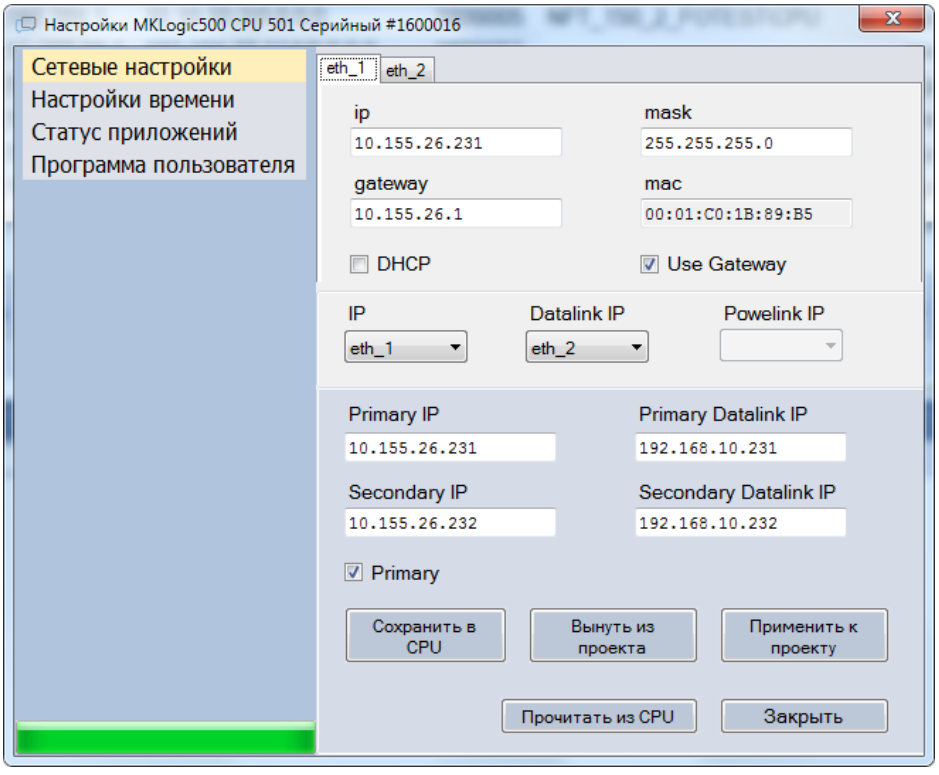


Рис. 12 – Окно сетевых настроек модуля CPU с поддержкой Failover

В верхней части окна настроек модуля CPU в разделе «Сетевые настройки» (Рис. 11, Рис. 12) расположены вкладки с параметрами сетевых интерфейсов модуля CPU. С их помощью выполняется настройка сетевых интерфейсов модуля CPU.

В средней части окна расположены выпадающие списки для задания ролей сетевых интерфейсов модуля CPU с поддержкой режима Failover (Рис. 12). Необходимо выбрать интерфейсы для роли IP-интерфейса (для связи со средой разработки ACP) и для роли Datalink IP-интерфейса (связь с резервным модулем CPU в режиме резервирования Failover). Роль Powerlink IP интерфейса в настоящее время не реализована.

В нижней части окна расположены настройки режима резервирования Failover (для модулей CPU с поддержкой этого режима) и кнопки «Сохранить в CPU», «Вынуть из проекта» и «Применить к проекту».

Настройки режима резервирования Failover должны быть идентичны для основного и резервного модулей CPU, единственное отличие должно заключаться в значении флага Primary (включён для основного модуля CPU и выключен для резервного). Значения Primary IP и Secondary IP должны соответствовать IP-адресам интерфейсов в роли IP основного и резервного модулей CPU соответственно, значения Primary Datalink IP и Secondary Datalink IP должны соответствовать IP-адресам интерфейсов в роли Datalink IP основного и резервного модулей CPU соответственно.

Кнопка «Вынуть из проекта» копирует текущие сетевые настройки проекта в поля IP-адреса интерфейса в роли IP и в поля настроек режима резервирования Failover, что упрощает конфигурацию модуля CPU при уже настроенном проекте.

Кнопка «Применить к проекту» копирует значения поля IP-адреса интерфейса в роли IP и настройки режима резервирования Failover в сетевую конфигурацию проекта, что может быть полезным при адаптации уже существующих проектов под новые сетевые настройки.

Кнопка «Сохранить в CPU» служит для непосредственно применения введённых настроек сетевых интерфейсов, ролей сетевых интерфейсов и режима резервирования Failover в модуле CPU.

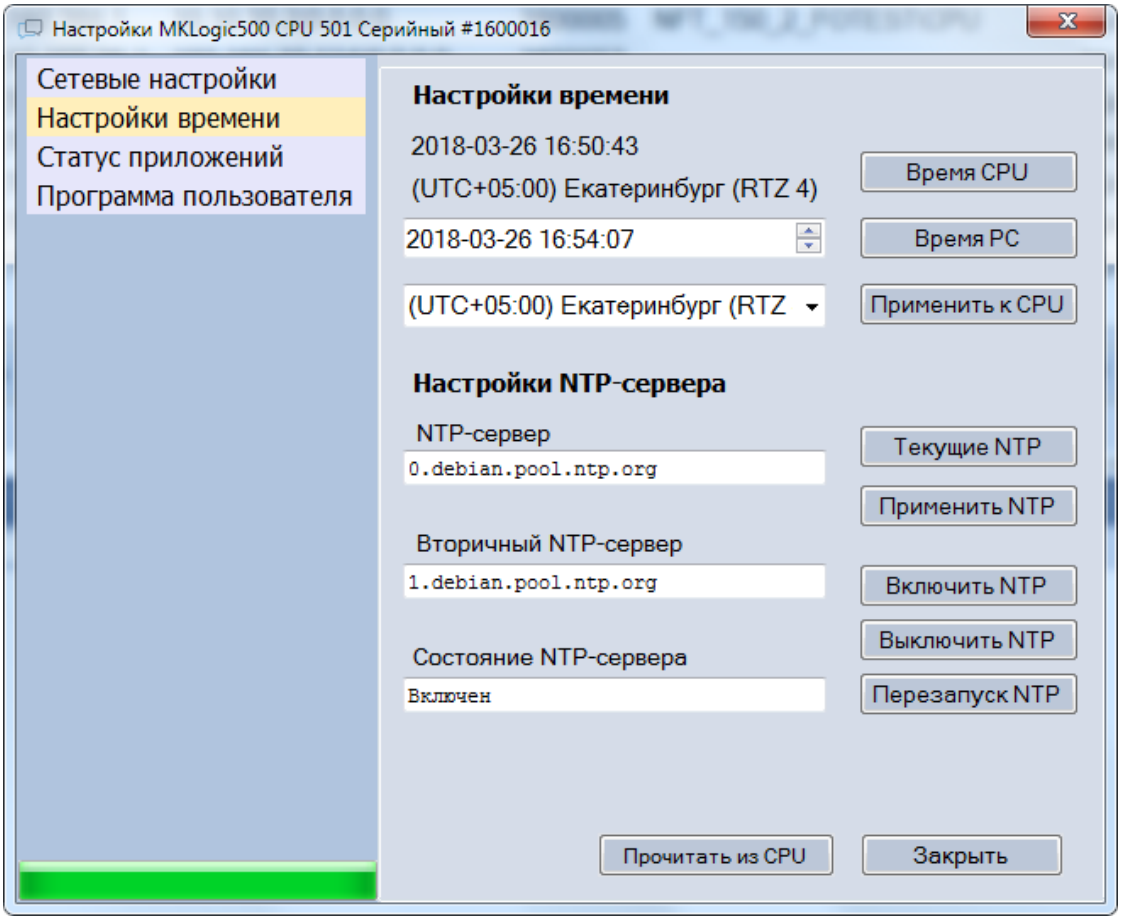


Рис. 13 – Окно настроек времени модуля CPU

⚠ ВНИМАНИЕ

Нажатие кнопки «Сохранить в CPU» приводит к перезапуску системного ПО модуля CPU с остановкой и перезапуском программы пользователя, и требует подтверждения в диалоговом окне.

В верхней части окна настроек модуля CPU в разделе «Настройки времени» (Рис. 13) расположены поля ввода и кнопки для настройки временных параметров модуля CPU.

Текущие время и часовой пояс часов реального времени модуля CPU расположены в верхней части группы настроек и могут быть обновлены по нажатию кнопки «Время CPU». Под ними находятся поля ввода времени и часового пояса. нажатием кнопки «Время PC» их значения можно синхронизировать со значениями ПК, на котором установлена среда разработки ACP. Нажатием кнопки «Применить к CPU» эти значения применяются к модулю CPU.

В нижней части окна расположены поля ввода для настройки параметров NTP-сервера модуля CPU и кнопки для управления NTP-сервером модуля CPU. Текущие значения адресов NTP-серверов и статус NTP-сервера могут быть обновлены нажатием кнопки «Текущие NTP», кнопка «Применить NTP» служит для записи значений адресов NTP-серверов в модуль CPU.

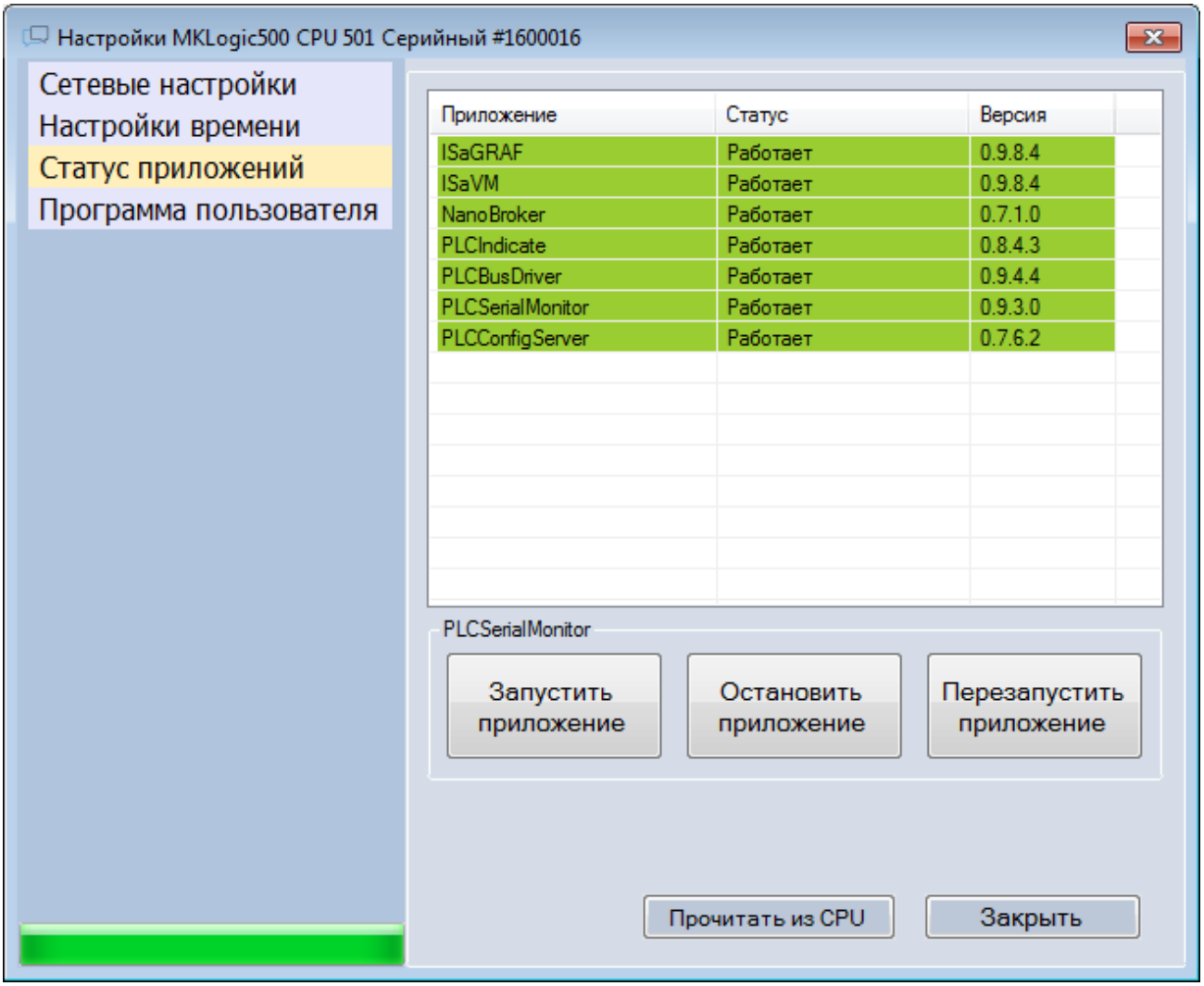


Рис. 14 – Окно статуса приложения модуля CPU

В верхней части окна настроек модуля CPU в разделе «Статус приложений» (Рис. 14) расположен список объектов системного программного обеспечения модуля CPU с указанием их имени, номера версии и текущего статуса.

В нижней части окна расположены кнопки управления службой PLCSerialMonitor из пакета системного программного обеспечения модуля CPU.

⚠ ВНИМАНИЕ

Остановка или перезапуск службы PLCSerialMonitor влечёт остановку программы пользователя.

Ручное управление работой службой PLCSerialMonitor следует использовать исключительно по согласованию с сервисными службами организации-производителя изделия.

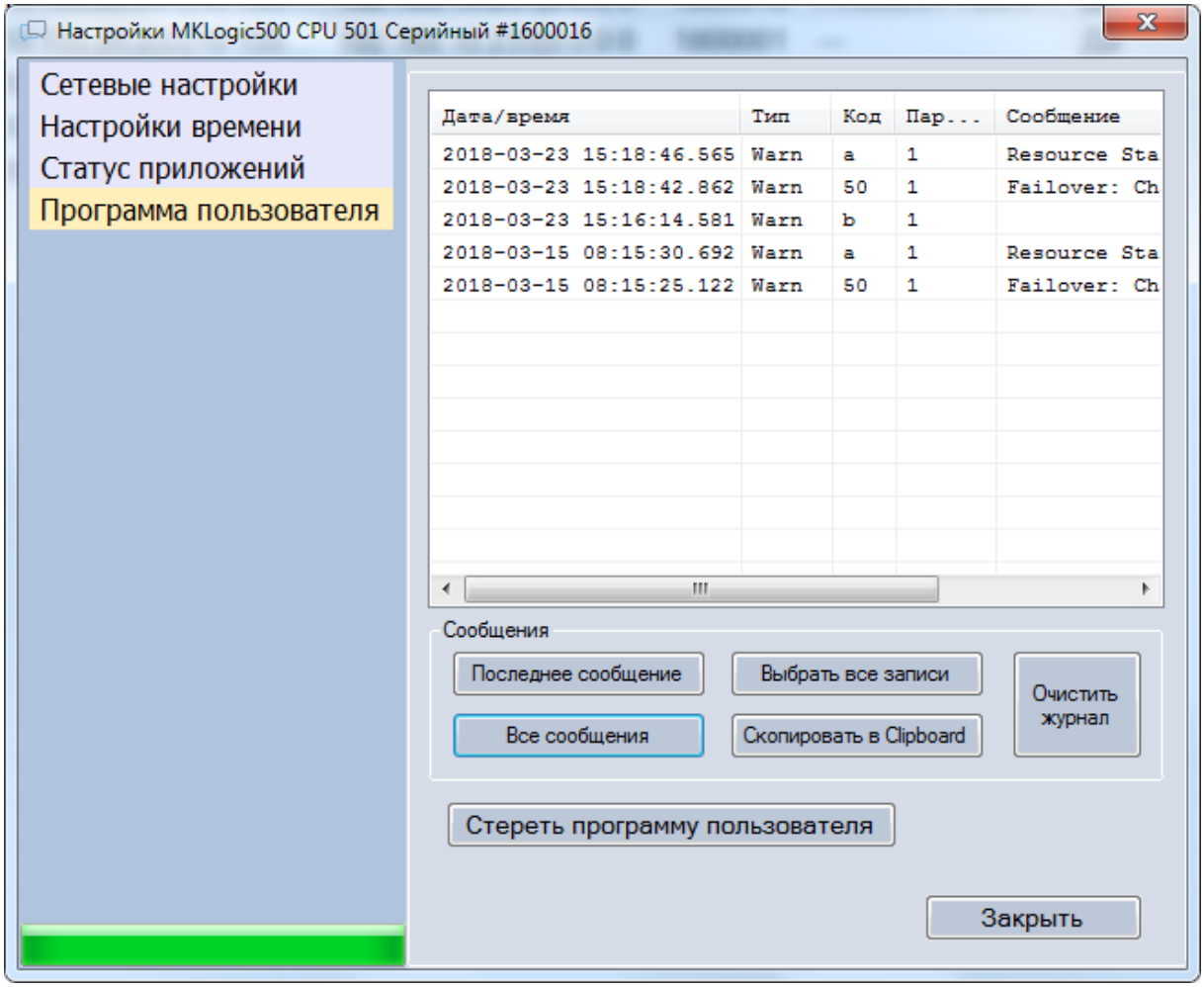


Рис. 15 – Окно программы пользователя модуля CPU

В верхней части окна настроек модуля CPU в разделе «Программа пользователя» (Рис. 15) расположен журнал сообщений программы пользователя. По умолчанию при открытии окна показывается только самое последнее сообщение журнала.

В нижней части окна расположены кнопки управления работой журнала, а также кнопка стирания программы пользователя.

Кнопка стирания программы пользователя предназначена либо для очистки модуля CPU перед его передачей третьим лицам, либо для восстановления нормальной работоспособности модуля CPU при сбое в ходе загрузки проекта в модуль CPU (п. 2.7).

2.4 Настройка сетевых параметров проекта

2.4.1 Настройка сетевых параметров проекта без поддержки режима Failover

При работе с модулем CPU без поддержки режима Failover сетевые настройки проекта включают в себя только IP-адрес интерфейса для связи со средой разработки ACP.

Для его настройки следует открыть окно Deployment, выполнив «VIEW»-«Deployment View» (Рис. 16).

Рис. 16 – Вызов окна Deployment

В открывшемся окне (Рис. 17) следует выбрать (нажав левую кнопку мыши) линию, соединяющую интересующее нас устройство с шиной ETCP (имя по умолчанию – DefNet), после чего в окне Properties можно будет задать IP-адрес интерфейса модуля CPU.

Альтернативный вариант задания IP-адреса проекта (путём копирования из модуля CPU) был описан в п. 2.3.2.

Для применения сетевых настроек проекта следует его пересобрать, лучше всего с предварительной очисткой проекта. Для этого следует последовательно выполнить «BUILD»-«Clean Solution» и «BUILD»-«Build Solution».

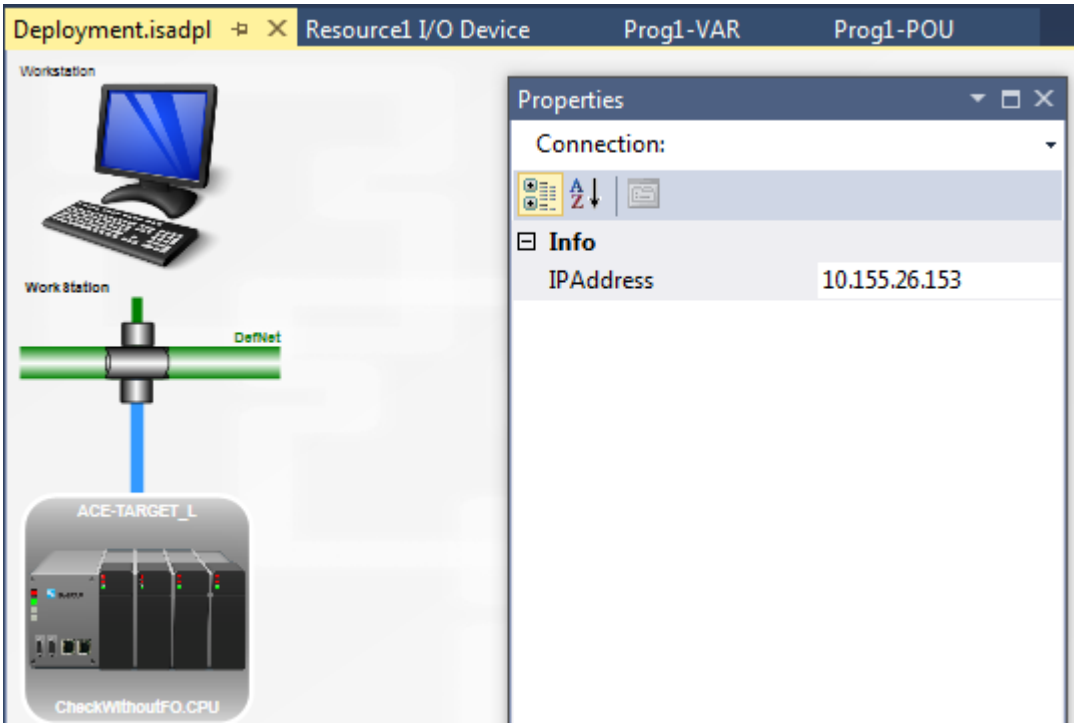


Рис. 17 – Настройка сетевых параметров проекта без поддержки режима Failover из окна Deployment

2.4.2 Настройка сетевых параметров проекта с поддержкой режима Failover

При работе с модулем CPU с поддержкой режима Failover сетевые настройки проекта включают в себя IP-адрес интерфейса для связи со средой разработки ACP, а

также IP-адреса IP-интерфейса и Datalink IP-интерфейса основного и резервного модулей CPU (Primary и Secondary Device соответственно).

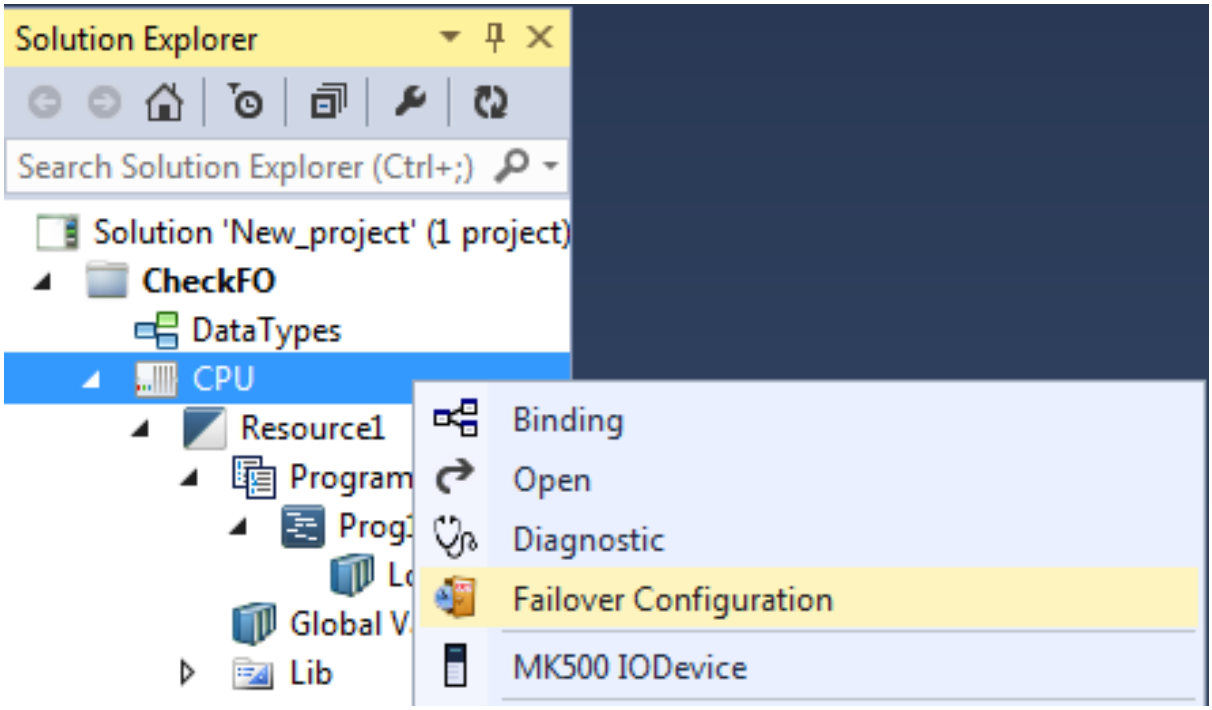


Рис. 18 – Вызов окна Failover Configuration

Для настройки IP-адресов следует открыть окно Failover Configuration, выбрав в дереве проекта устройство и в его контекстном меню выбрав пункт «Failover Configuration» (Рис. 18).

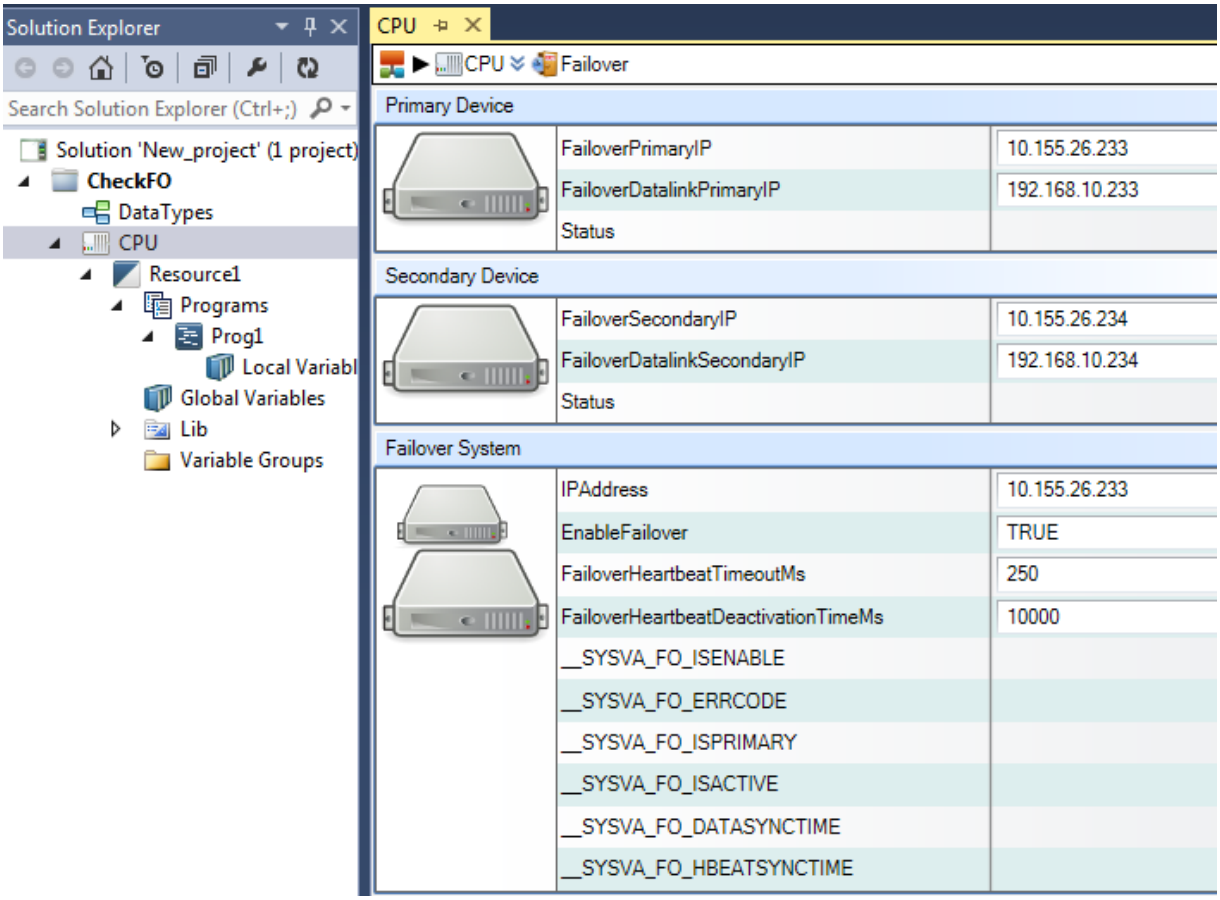


Рис. 19 – Настройка сетевых параметров проекта с поддержкой режима Failover из окна Failover Configuration

В открывшемся окне (Рис. 19) следует ввести IP-адреса для основного и для резервного модулей CPU, а также IP-адрес интерфейса для связи со средой разработки ACP (обычно вводится IP-адрес, совпадающий с FailoverPrimaryIP).

Также в окне Failover Configuration следует включить режим Failover (ввести TRUE в поле EnableFailover), а также настроить временные параметры режиме Failover.

Альтернативный вариант задания IP-адресов проекта (путём копирования из модуля CPU) был описан в п. 2.3.2.

Также следует отметить то, что при старте модулей CPU происходит сравнение их сетевых настроек, в котором стартовавший первым модуль CPU применяет свои сетевые настройки в режиме Secondary на стартовавший вторым модуль CPU (п. 5.2.1).

Для применения сетевых настроек проекта следует его пересобрать, лучше всего с предварительной очисткой проекта. Для этого следует последовательно выполнить «BUILD»-«Clean Solution» и «BUILD»-«Build Solution», либо «BUILD»-«Rebuild Solution».

2.5 Структура проекта

В рамках среды исполнения ISaGRAF проект (Solution) создаётся как коллекция для устройств (п. 2.5.1), имеющих общую аппаратную платформу и среду исполнения, и описываемых общим tdb-файлом.

Также общими для всех устройств являются типы данных (массивы, структуры) и константы. Ознакомиться с имеющимися типами данных, а также добавить свои типы данных, можно выбрав в дереве проектов раздел DataTypes.

Tdb-файл содержит в себе описание всех модулей изделия, описание дополнительных констант, типов, функций и функциональных блоков, а также настройки и ограничения создаваемого с его помощью проекта. Каждый модуль CPU содержит в себе актуальный tdb-файл, который можно извлечь из модуля CPU способом, описанным в п. 2.3.1.

2.5.1 Устройства (Device) проекта

В рамках среды исполнения ISaGRAF устройством называется представление оборудования (ПЛК), выполняющего виртуальные машины IsaVM (п. 2.5.2).

На уровне устройства определяются сетевые параметры (п. 2.4), а также размер оперативной памяти в модуле CPU изделия, отводимый для хранения пользовательских переменных.

Для настройки размера оперативной памяти в модуле CPU для хранения пользовательских переменных следует открыть окно свойств устройства. Для этого необходимо выбрать в дереве проектов интересующее нас устройство и в его контекстном меню выбрать пункт Properties (Рис. 20). Далее в окне Properties следует изменить (обычно требуется увеличить) значение Memory Size, по умолчанию равное 524280.

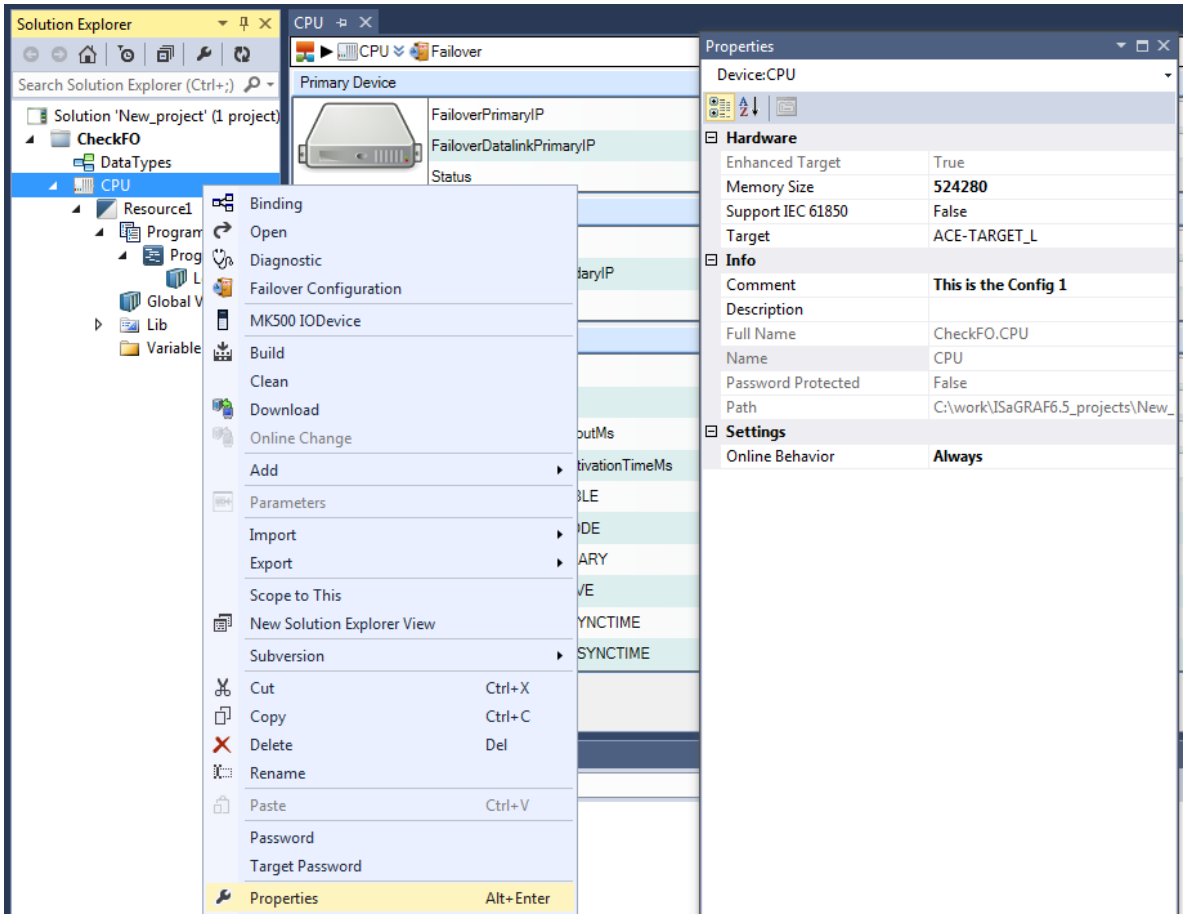


Рис. 20 – Настройка параметров устройства

2.5.2 Ресурсы (Resource) проекта

В рамках среды исполнения ISaGRAF ресурсом называется совокупность программных модулей и определений, составляющая виртуальную машину IsaVM. Виртуальные машины IsaVM выполняются в модуле CPU как отдельные процессы под общим контролем и управлением среды исполнения ISaGRAF, и фактически позволяют параллельно выполнять несколько разных независимых программ пользователя.

Каждый ресурс содержит в себе конфигурацию устройств ввода-вывода, переменные и программные модули, не пересекающиеся с конфигурациями, переменными и программными модулями других ресурсов.

⚠ ВНИМАНИЕ

В модулях CPU настоящего изделия разрешено выполнять только один ресурс, поэтому создание дополнительных ресурсов в проектах, предназначенных для работы с модулями CPU изделия, настоятельно не рекомендуется. Проекты с более чем одним ресурсом, не будут нормально запускаться в модулях CPU изделия.

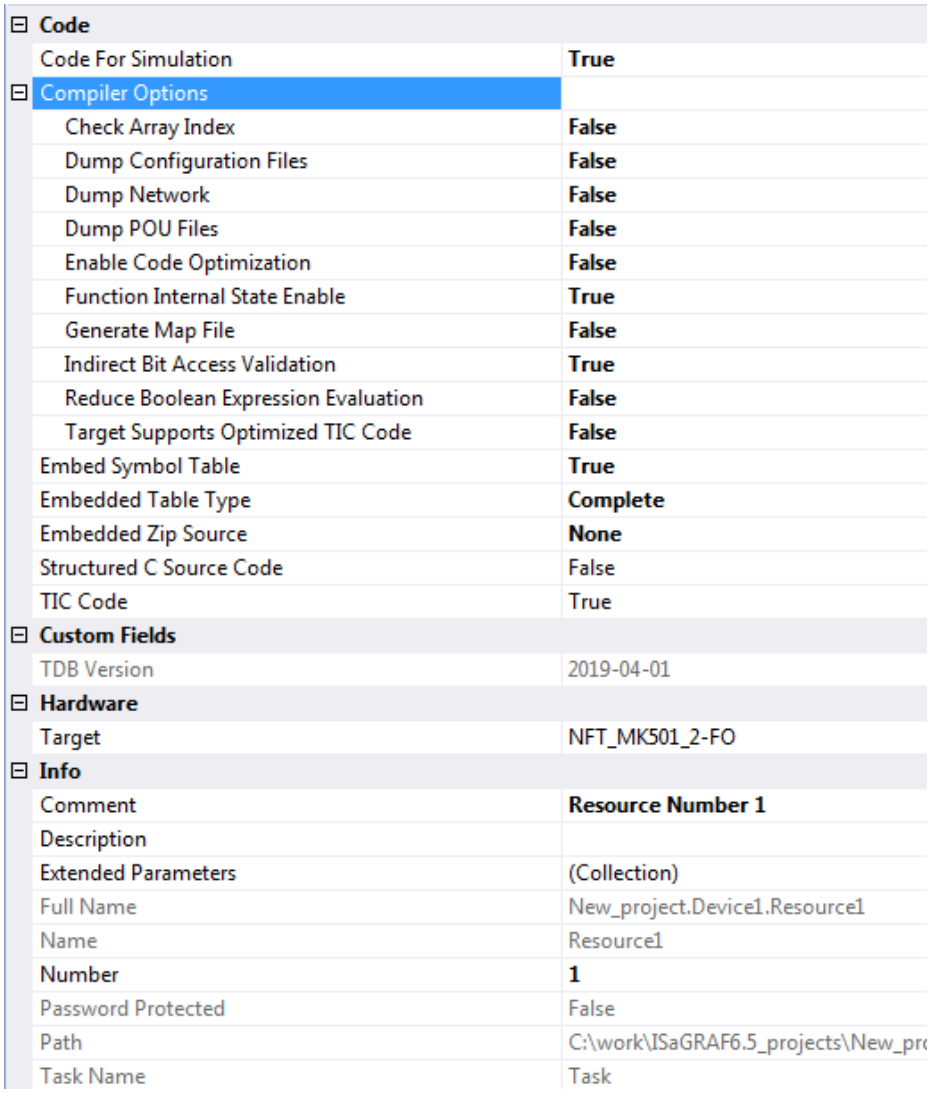
Добавление, настройка и удаление модулей ввода-вывода в ресурс будут рассмотрены в Главе 3.

Для настройки параметров ресурса следует выбрать в дереве проектов интересующий нас ресурс и в его контекстном меню выбрать пункт Properties. В

открывшемся окне Properties (Рис. 21, Рис. 22). Далее будут перечислены самые значимые параметры ресурса и даны рекомендации по их настройке.

Секция **Code** (Рис. 21):

- Code for simulation – отвечает за сборку версии проекта для запуска на локальном симуляторе. В рабочем проекте настоятельно рекомендуется отключать (false), это ускорит сборку и позволит избежать сообщений компилятора о нехватке памяти при использовании в проекте больших объёмов пользовательских данных (у симулятора небольшой объём памяти для хранения пользовательских данных).
- Check Array Index – отвечает за добавление в генерируемый код проверок на выход за пределы массивов. Если установить в true, при выходе за пределы массива рабочая программа останавливается с ошибкой «21Kernel TIC: Boundary check error.».
- Dump Configuration Files – отвечает за генерацию отладочных файлов конфигурации ресурса. Если установить его в true, то при сборке проекта в папке ресурса будут созданы файлы-дампы конфигурации ресурса с именами вида RESOURCE1_Conf.ttc, RESOURCE1_Constants.ttc и RESOURCE1_DwlOrder.ttc.
- Dump Network – отвечает за генерацию отладочных файлов сетевой конфигурации проекта. Если установить его в true, то при сборке проекта в папке проекта будет создан файл-дамп сетевой конфигурации NetworkConf.ttc.
- Dump POU Files – отвечает за генерацию отладочных файлов POU (программных модулей) ресурса. Если установить его в true, то при сборке проекта в папке ресурса будут созданы файлы-дампы программных модулей ресурса с именами вида RESOURCE1_Pou_PROG1.ttc. Число создаваемых файлов определяется числом программных модулей (программы, функции, функциональные блоки) в составе ресурса.
- Enable Code Optimization – отвечает за оптимизацию результирующего TIC-кода по скорости. Рекомендуется установить в true, так как его включение приводит к существенному (до 2 раз) ускорению времени исполнения программы пользователя.
- Function Internal State Enable – отвечает за сохранение функциями значений своих локальных переменных между вызовами. Подробнее см. п. 5.1.2.
- Generate Map File – отвечает за генерацию отладочных файлов пользовательских и системных переменных ресурса. Если установить его в true, то при сборке проекта в папке ресурса будут созданы файлы-дампы переменных с именами RESOURCE1_SymbolsDebug.ttc и RESOURCE1_SymbolsTarget.ttc.
- Indirect Bit Access Validation – отвечает за добавление в генерируемый код проверок на ошибки при косвенной адресации отдельных битов переменной. Если установить в true, при выходе за пределы переменной рабочая программа останавливается с ошибкой.



Code	
Code For Simulation	**True**
Compiler Options	
Check Array Index	**False**
Dump Configuration Files	**False**
Dump Network	**False**
Dump POU Files	**False**
Enable Code Optimization	**False**
Function Internal State Enable	**True**
Generate Map File	**False**
Indirect Bit Access Validation	**True**
Reduce Boolean Expression Evaluation	**False**
Target Supports Optimized TIC Code	**False**
Embed Symbol Table	**True**
Embedded Table Type	**Complete**
Embedded Zip Source	**None**
Structured C Source Code	False
TIC Code	True
Custom Fields	
TDB Version	2019-04-01
Hardware	
Target	NFT_MK501_2-FO
Info	
Comment	**Resource Number 1**
Description	
Extended Parameters	(Collection)
Full Name	New_project.Device1.Resource1
Name	Resource1
Number	**1**
Password Protected	False
Path	C:\work\ISaGRAF6.5_projects\New_pr
Task Name	Task

Рис. 21 – Настройка параметров ресурса – 1

- Reduce Boolean Expression Evaluation– отвечает за сокращённое вычисление булевых выражений. Если установить его в true, то булевы выражения будут вычисляться не полностью, а только до момента, когда становится очевидным конечный результат. Рекомендуется установить значение в true.
- Target Supports Optimized TIC Code – отвечает за поддержку оптимизированного TIC-кода средой исполнения. Рекомендуется установить значение в true.
- Embedded Symbol Table – отвечает за встраивание таблицы символов в проект. Всегда принимает значение true, изменение запрещено на уровне среды разработки.
- Embedded Table Type – отвечает за объём встраиваемой таблицы символов в проект. Всегда принимает значение Complete, изменение запрещено на уровне среды разработки.
- Embedded Zip Source – отвечает за генерацию файла--архива ресурса, устройства или всего проекта (в зависимости от значения). Если установить его в Project, Device или Resource, то при сборке проекта в папке ресурса будет создан файл IDS00103, представляющий собой 7z-архив проекта, устройства

или ресурса. В рабочем проекте настоятельно не рекомендуется использовать для этого параметра значение, отличное от None, так как среда разработки в ходе загрузки проекта в модуль CPU устройства пытается загрузить и этот файл, что в версии среды разработки ACP 6.50 всегда заканчивается неудачей.

– Structured C Source Code – отвечает за генерацию проекта в программу на языке C. Если установить его в true, то при сборке проекта в папке ресурса будет создан набор файлов с расширениями *.h и *.c, которые теоретически можно скомпилировать непосредственно в машинный код и запустить в модуле CPU изделия. Значение данного параметра всегда противоположно значению параметра TIC Code. Рекомендуется установить значение параметра в false.
– TIC Code – отвечает за генерацию проекта в исполняемый средой isagraf TIC-код. Если установить его в true, то при сборке проекта в папке ресурса будет создан набор файлов, предназначенных для загрузки и исполнения в модуле CPU изделия. Значение данного параметра всегда противоположно значению параметра Structured C Source Code. Для нормальной работы настоятельно рекомендуется установить значение параметра в true.

Секция **Hardware** (Рис. 21):

– Target – отвечает за тип целевого устройства, для которого собирается ресурс. Для работы с модулями CPU изделия должен быть установлен в «ACE-TARGET_L».

Секция **Info** (Рис. 21):

– Comment – текстовый комментарий к ресурсу.
– Number – порядковый номер ресурса.

Секция **Memory Size for Online Changes** (Рис. 22):

– Code Size – объём памяти (в байтах), который выделяется при онлайн-обновлении секции кода.
– Maximum Extra POUs – максимальное число программных модулей, которые могут быть добавлены при онлайн-обновлении.
– SFC States Mem Size – объём памяти (в байтах), который выделяется при онлайн-обновлении программных модулей на языке SFC.
– User Variable Size – объём памяти (в байтах), который выделяется при онлайн-обновлении переменных.

Memory Size for Online Changes	
Code Size	**14000**
Maximum Extra POUs	**20**
SFC States Mem size	**4096**
User Variable Size	**2048**
Memory Usage Info	
Biggest Free User Variable Me	2048
Data Space Usage	9868
Memory Usage (Code)	200
Memory Usage (Compressed	
Memory Usage (Data)	536200
Memory Usage (Retain Space)	0
Memory Usage (Symbolic Tat	17086
Memory Usage (Temporary V	4
User Variable Data Space Incre	0
Settings	
Cycle Time	**100**
Cycle Time Units	**ms**
Detect Errors	**True**
Execution Mode	**Real Time**
Memory For Retain	RETAIN
Nb Stored Errors	**16**
Online Behavior	**Always**
Trigger Cycles	**True**
SFC Dynamic Behavior Limits	
Gain Factor	**8**
Offset Factor	**18**

Рис. 22 – Настройка параметров ресурса – 2

Секция **Settings** (Рис. 22):

– Cycle Time – заданное время цикла выполнения программы пользователя, в единицах Cycle Time Units.

⚠ ВНИМАНИЕ

Установка заданного времени цикла, равного или меньшего реального времени выполнения программы пользователя, приводит к существенному росту загрузки процессора модуля CPU изделия.

– Cycle Time Units – единицы времени Cycle Time, для модулей CPU изделия могут принимать только значение ms (миллисекунды).
– Detect Errors – отвечает за хранение ошибок средой исполнения.

- Execution Mode – указывает на режим выполнения программы пользователя. Для непрерывной работы (нормальный режим работы модуля CPU) должен принимать значение «Real Time», для пошагового выполнения программы пользователя цикл за циклом должен принимать значение «Cycle to cycle».
- Memory For Retain – указывает на место хранения сохраняемых переменных. Для работы с модулями CPU изделия должен быть установлен в «RETAIN».
- Nb Stored Errors – число ошибок, которые хранятся в среде исполнения при параметре Detect Errors равном true.
- Online Behavior – указывает на режим работы программы пользователя (Design, Simulator, Online или Always, то есть в любом режиме), в котором допускается подключение к программе пользователя отладчиком («DEBUG»-«Start Debugging»). Рекомендуется установить в Always.
- Trigger Cycles – указывает на режим выполнения программы пользователя. Если параметр установлен в false, то следующий цикл выполнения программы пользователя стартует немедленно после завершения предыдущего цикла; если параметр установлен в true, то если реальное время выполнения программы пользователя меньше значения Cycle Time, очередной цикл выполнения программы будет запущен только по истечении Cycle Time. В большинстве случаев этот параметр должен быть установлен в true.

Секция **SFC Dynamic Behavior Limits** (Рис. 22):

- Gain Factor и Offset Factor – параметры, определяющие объём выделяемой памяти при добавлении одного программного модуля на языке SFC.

2.6 Сборка проекта

Сборка проекта (Solution) выполняется с помощью команд «BUILD»-«Build Solution». Собрать отдельный программный блок либо устройство можно, выделив его в дереве проекта и выполнив команду «BUILD»-«Build Selection».

В случае внесения изменений в конфигурацию ввода-вывода проекта (Глава 3) либо в сетевые параметры проекта рекомендуется перед сборкой выполнить команду очистки «BUILD»-«Clean Solution» либо выполнить команду «BUILD»-«Rebuild Solution».

Если в ходе сборки возникли ошибки, они будут выведены в окно Error List (Рис. 23). Проект при этом собран не будет.

Следует отметить, что компилятор среды разработки ACP Workbench 6.5 не всегда указывает на одну лишь первопричину ошибки, что может затруднить локализацию проблемы. В примере, приведённом на Рис. 23, синтаксическая ошибка (лишние символы `aaa` перед идентификатором массива) порождает сразу 4 ошибки сборки, из которых следует обратить внимание только на самую первую («AAA: undeclared identifier»). С её устранением исчезают и остальные три ошибки.

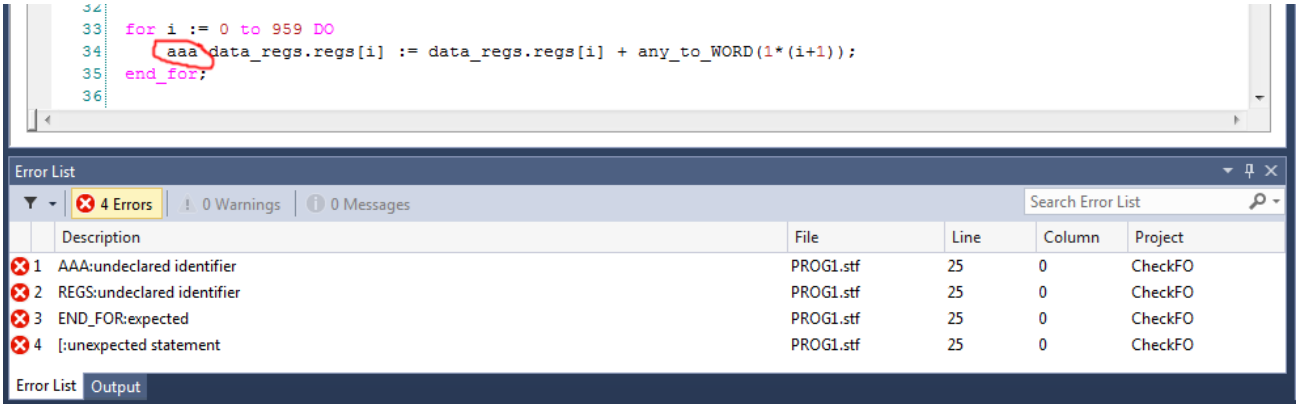


Рис. 23 – Пример вывода ошибок при сборке проекта

Если в ходе сборки возникли предупреждения, они также будут выведены в окно Error List. В примере на Рис. 24 компилятор предупреждает об использовании функции, могущей вызвать зависание программы пользователя. Проект при этом будет собран и готов к загрузке в модуль CPU.

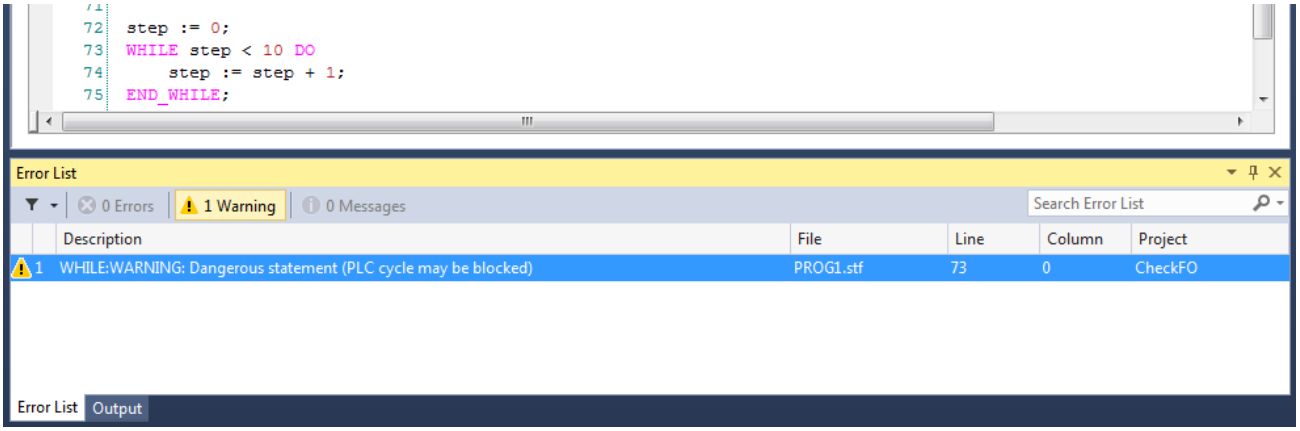


Рис. 24 – Пример вывода предупреждений при сборке проекта

2.7 Загрузка проекта в процессорные модули изделия

После успешной сборки проекта появляется возможность загрузить его в модуль CPU (в случае проекта с поддержкой Failover – в пару модулей CPU). Для этого следует в дереве проектов выбрать соответствующее устройство и в его контекстном меню (либо на панели инструментов) выбрать пункт «Download» (Рис. 25).

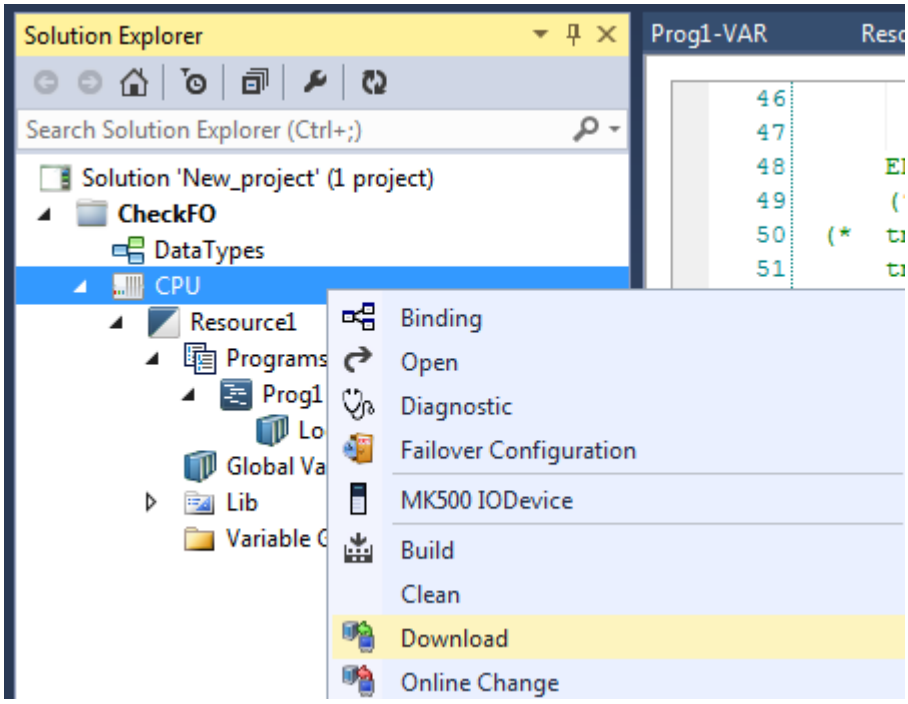


Рис. 25 – Загрузка собранного проекта в модуль CPU

Если в момент загрузки проекта в модуле CPU уже выполняется программа пользователя, будет выдано окно подтверждения (Рис. 26). Для продолжения загрузки проекта следует нажать кнопку «Да», нажатие кнопки «Нет» прекратит загрузку и никак не повлияет на выполнение уже запущенной в модуле CPU программы.

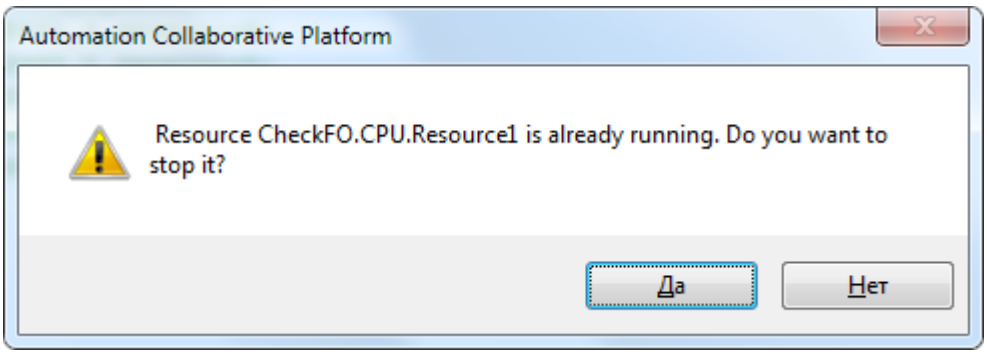


Рис. 26 – Окно подтверждения загрузки проекта в модуль CPU с уже запущенной программой пользователя

Если загрузка проекта завершилась успешно, в окно Output будет выведено соответствующее сообщение (Рис. 27).

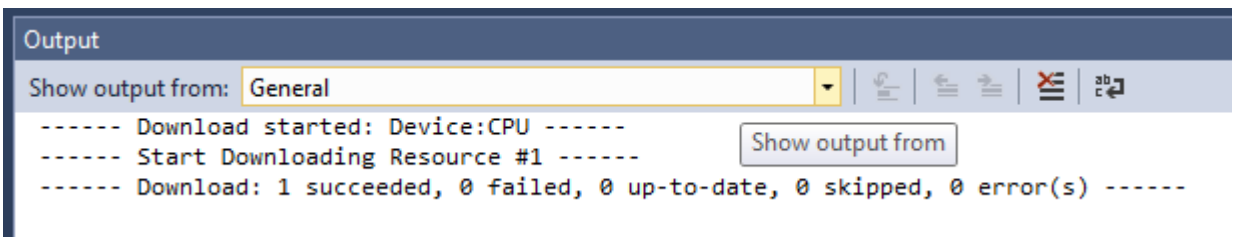


Рис. 27 – Вывод сообщения об успешной загрузке проекта в модуль CPU

Также при старте программы пользователя модуль CPU в течение одной секунды мигает всей индикацией на лицевой панели, если он не Secondary-модуль CPU (п. 5.2.1).

В случае наличия ошибки (несовместимость версии tdb-файла и среды исполнения модуля, отсутствия связи между средой разработки и модулем CPU и т.п.) в окно Output будет выведено сообщение об ошибке. На Рис. 28 приведено сообщение, выводимое средой разработки при отсутствии связи с модулем CPU.

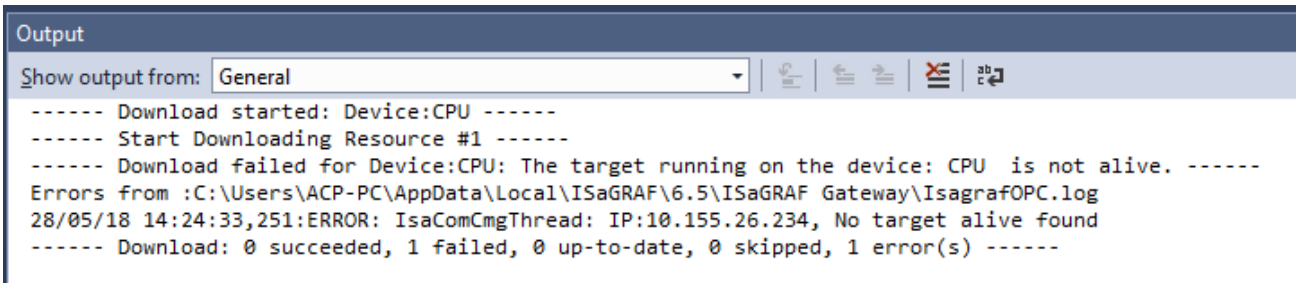


Рис. 28 – Вывод сообщения об ошибке при отсутствии связи с модулем CPU

Также в среде разработки ACP могут быть доступными два пункта меню:

– «Upload» – выгрузка проекта пользователя из модуля CPU. В версии среды разработки ACP 6.50 это действие не может быть реализовано, так требует для своей работы значения параметра ресурса Embedded Zip Source отличное от None (п. 2.5.2).
– «Online Change» – применение изменений в проекте без остановки проекта. Позволяет применять незначительные изменения (не затрагивающие конфигурацию ввода-вывода проекта) без перезапуска проекта. Следует использовать этот вариант обновления с осторожностью, так как в случае несовместимости внесённых изменений может потребоваться перезапуск модуля/модулей CPU для возврата их в рабочее состояние.

2.8 Мониторинг и управление работой программы в процессорном модуле изделия

Среда разработки ACP 6.50 предоставляет следующие возможности мониторинга и управления работой программы пользователя в модуле CPU:

– диагностика работы программы пользователя;
– просмотр и модификация значения переменных программы пользователя;
– управление работой ресурса программы пользователя.

Все эти возможности доступны только в онлайн-режиме среды разработки. Для перехода в онлайн-режим следует на панели инструментов «Debug» выбрать режим работы «Online» и нажать кнопку «Start Debugging» (Рис. 29).

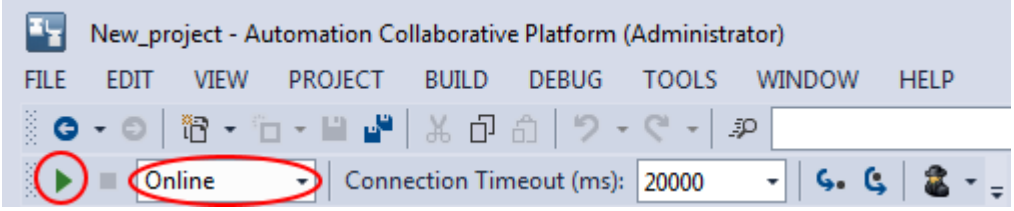


Рис. 29 – Перевод среды разработки ACP в онлайн-режим

Для перехода в онлайн-режим, модуль CPU должен быть доступен среде разработки по сети. Таймаут подключения к модулю CPU определяется параметром Connection Timeout (по умолчанию 20000 мс, см. Рис. 29) на панели инструментов «Debug».

После успешного перехода в онлайн-режим строка статуса среды разработки становится оранжевого цвета, редактирование программы пользователя становится невозможным (Рис. 30). При этом становятся доступными функции онлайн-мониторинга состояния программы пользователя, функции управления работой ресурсов программы пользователя и функции просмотра и модификации переменных программы пользователя.

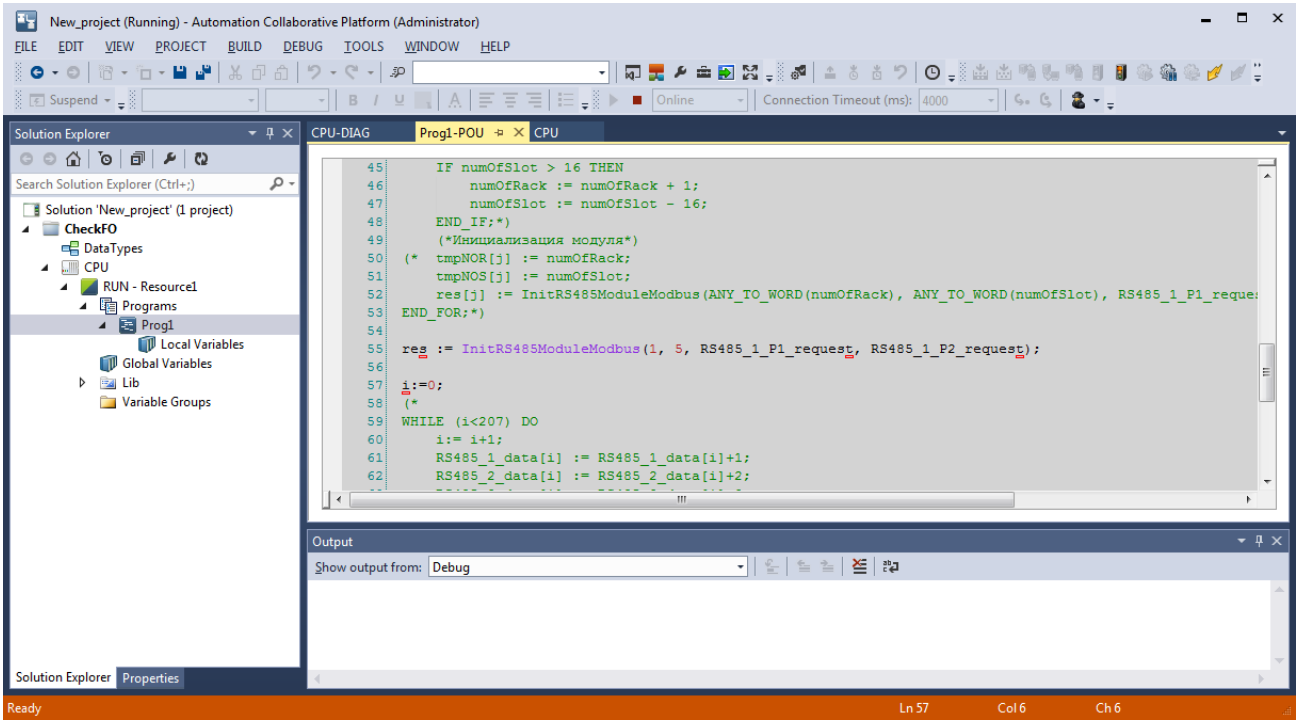


Рис. 30 – Внешний вид среды разработки ACP в онлайн-режиме

2.8.1 Диагностика работы программы пользователя

Для просмотра диагностической информации ресурса или всего устройства следует в онлайн-режиме выбрать в дереве проекта ресурс или устройство, нажать правую кнопку мыши и выбрать в контекстном меню пункт «Diagnostic» (Рис. 31).

В открывшемся окне диагностики (Рис. 32) выводится следующая информация обо всех ресурсах устройства (в случае, если в дереве проекта был выбран ресурс – только об этом ресурсе):

- число ресурсов устройства, запущены ли они и сохранены ли они в энергонезависимую память модуля CPU;
- контрольные суммы, номера версий (в рамках проекта) и даты загрузки ресурса в модуль CPU;
- использованием ресурсом памяти модуля CPU.

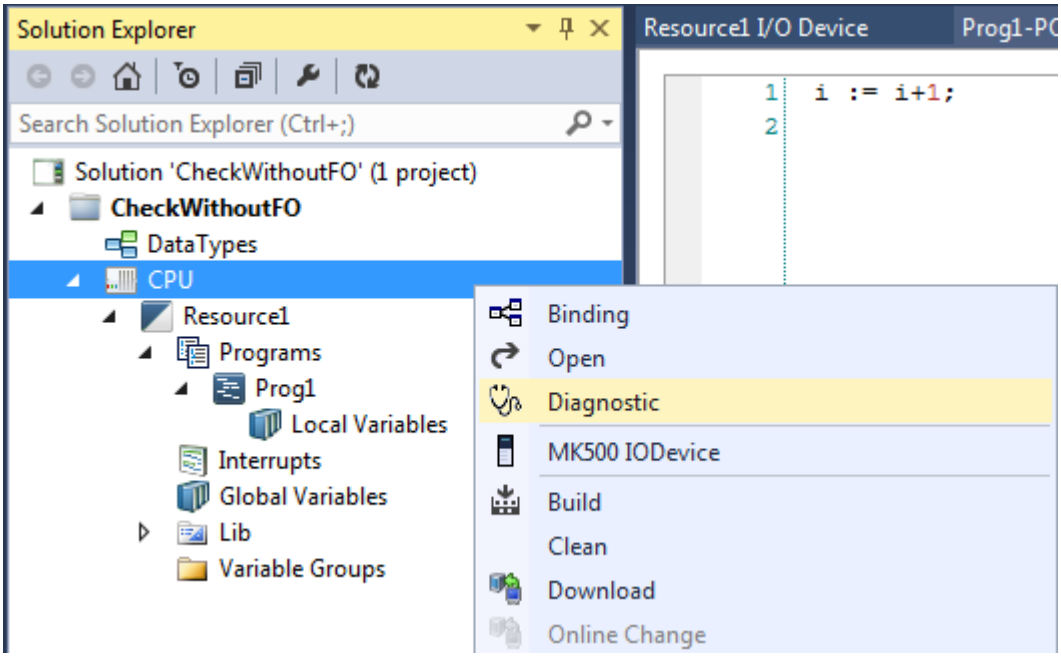


Рис. 31 – Открытие окна диагностики

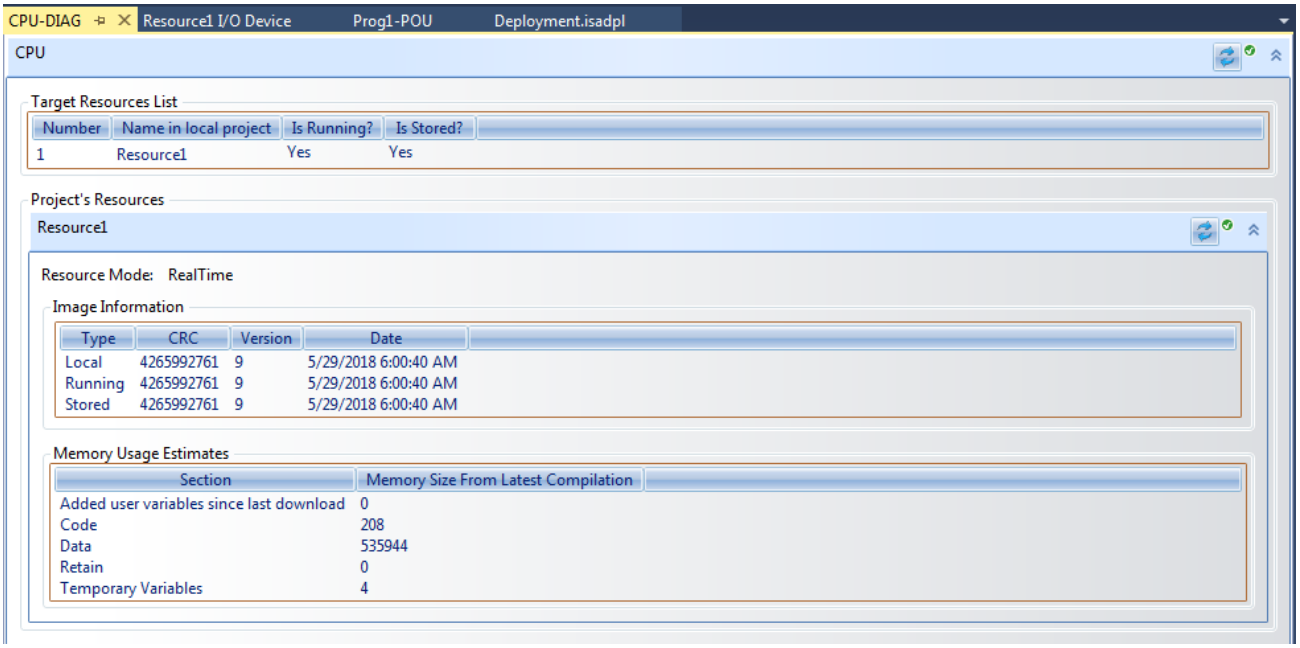


Рис. 32 – Окно диагностики устройства

Для модулей CPU с поддержкой режима Failover также доступно окно онлайн-диагностики работы системы резервирования. Для его просмотра следует в онлайн-режиме выбрать в дереве проекта устройство, и в его контекстном меню выбрать пункт «Failover Configuration» (Рис. 18). Откроется окно Failover Configuration, ранее описанное в п. 2.4.2.

В онлайн-режиме редактирование сетевых параметров недоступно, вместо этого в окне Failover Configuration (Рис. 33) выводится информация о состоянии программы пользователя в ведущем и ведомом модулях CPU резервированной пары, а также подробная информация о текущем состоянии системы резервирования в модуле CPU, назначенном Primary при настройке сетевых параметров.

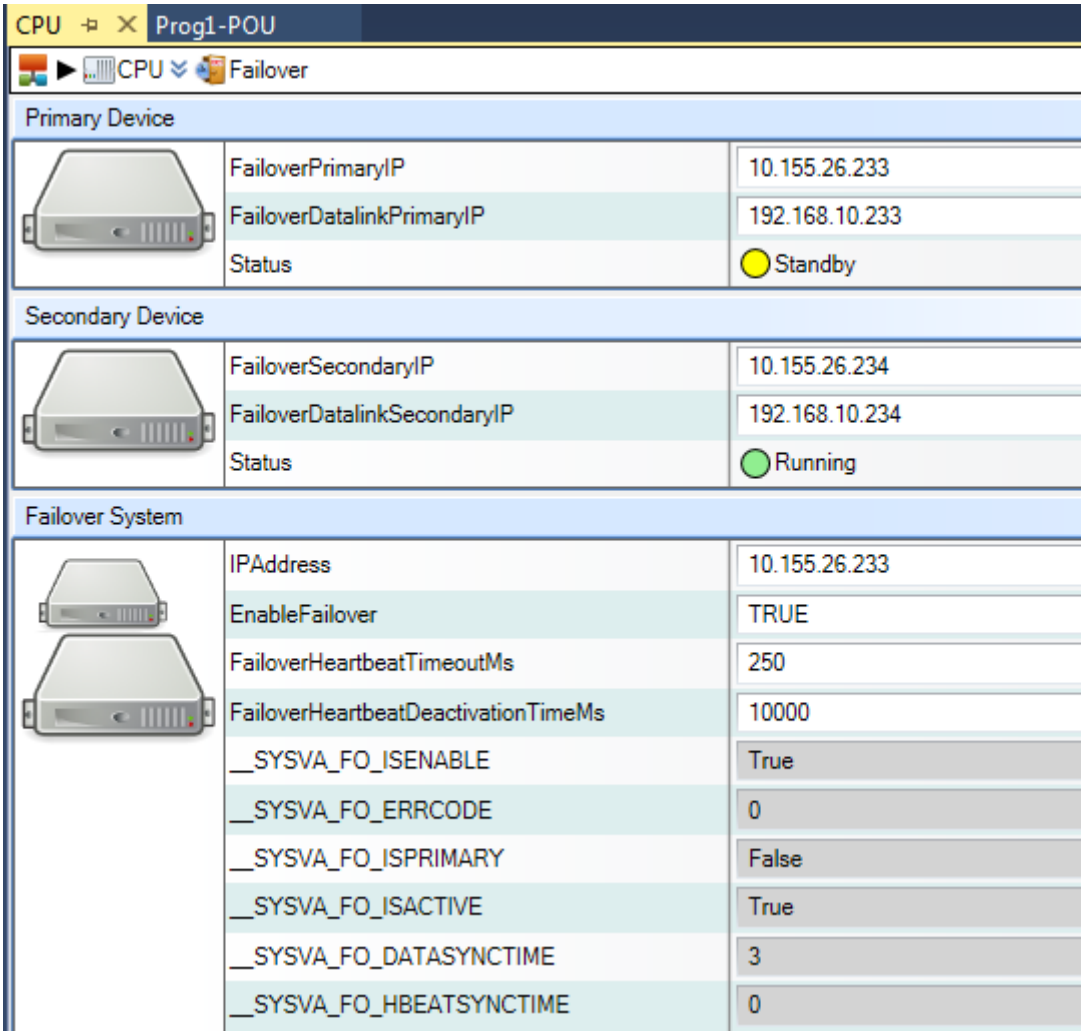


Рис. 33 – Окно сетевых параметров проекта с поддержкой режима `Failover` в онлайн-режиме

Более подробную информацию о параметрах системы резервирования можно узнать из документа «Механизм обеспечения отказоустойчивости» (fvr_ISa6_ru).

2.8.2 Просмотр и модификация значения переменных программы пользователя

Для просмотра и модификации переменных программы пользователя в онлайн-режиме предназначены так называемые «Spy list».

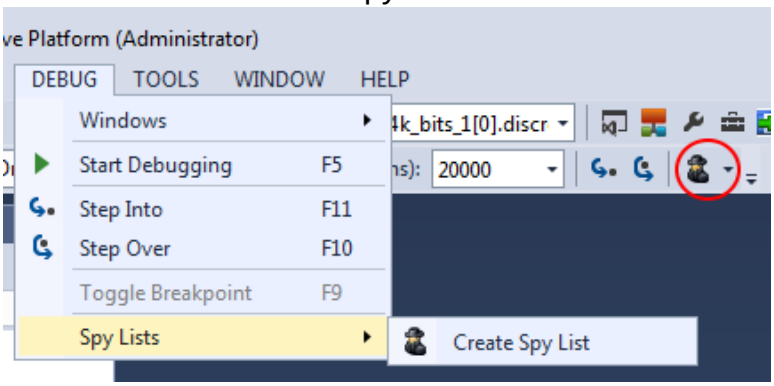


Рис. 34 – Создание экземпляра «`Spy list`»-а

Для создания нового экземпляра «Spy list»-а следует в меню среды разработки выбрать «DEBUG»-«Spy Lists»-«Create Spy List» либо нажать кнопку «Create Spy List» на панели инструментов (Рис. 34).

Для добавления интересующей нас переменной следует поместить курсор на свободное поле колонки «Name» окна «Spy list» и начать вводить имя переменной. Появится выпадающий список с именами доступных переменных, из которых можно выбрать нужную. Если окно «Spy list» открыто не в онлайн-режиме, в нём можно также указать имя окна (для удобства пользования) и период обновления данных в онлайн-режиме (Рис. 35).

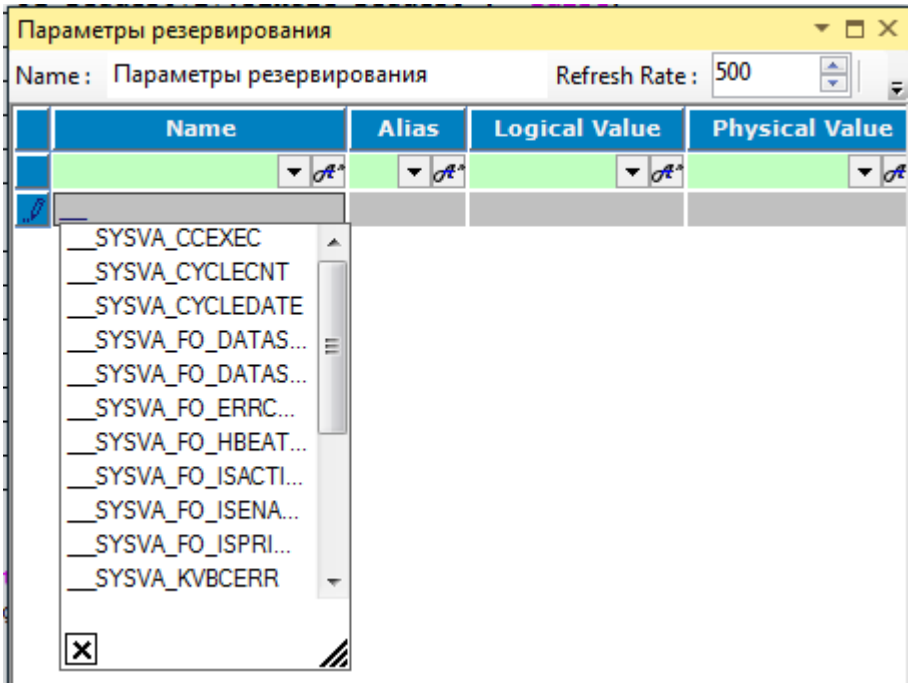


Рис. 35 – Настройки «Spy list»-а и формирование списка переменных

В онлайн-режиме в открытом окне «Spy list»-а выводятся текущие значения переменных программы пользователя (Рис. 36).

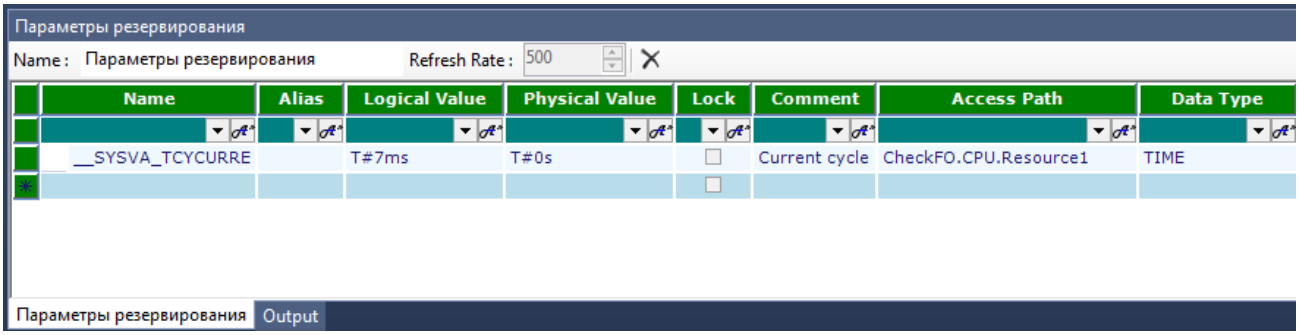


Рис. 36 – Просмотр переменных программы пользователя в онлайн-режиме

Для модификации значений переменных программы пользователя следует поместить курсор в колонку «Logical Value» соответствующей переменной в окне «Spy list», ввести необходимое значение и нажать «Enter» (Рис. 37).

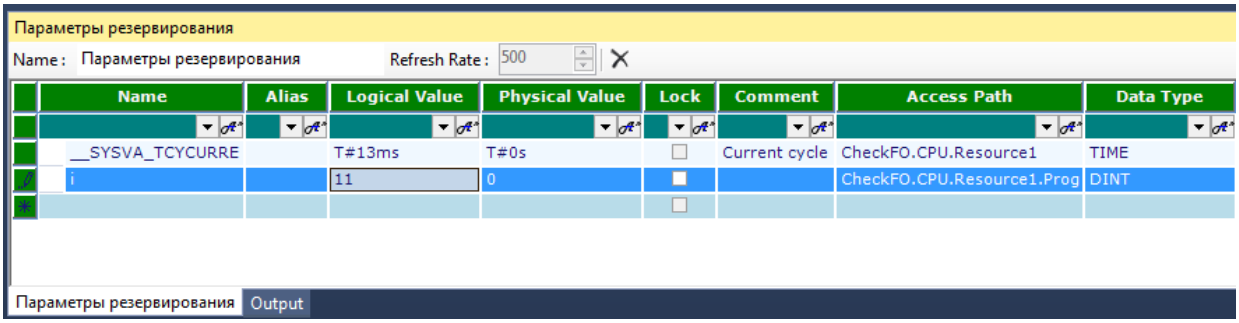


Рис. 37 – Модификация переменных программы пользователя в онлайн-режиме

Альтернативный вариант модификации переменной – выполнить на строке с интересующий переменной в окне «Spy list» двойной клик либо из контекстного меню выбрать пункт «Write Variable…». Затем в открывшемся окне (Рис. 38) можно ввести и записать новое значение переменной, либо защитить переменную от дальнейших изменений (опция «Lock»). Защищать переменную без необходимости не рекомендуется, так как при попытке записи в неё может произойти зависание ресурса, выйти из которого можно только перезапуском модуля CPU.

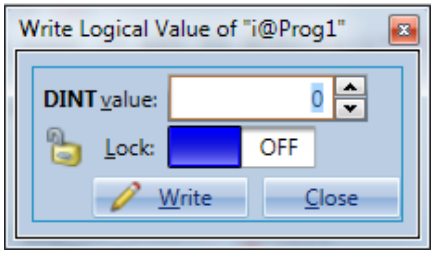


Рис. 38 – Окно записи значения переменной программы пользователя в онлайн-режиме

2.8.3 Управление работой ресурса программы пользователя

В онлайн-режиме среда разработки ACP 6.50 позволяет выполнять следующие операции над исполняемым в модуле CPU ресурсом:

- останавливать и запускать исполнение ресурса;
- переводить режим исполнения ресурса в режим пошагового исполнения и обратно в непрерывный режим;
- включать и отключать мониторинг системных сообщений среды исполнения ISaGRAF.

Рис. 39 – Кнопки «Stop Resource(s)» (сверху) и «Start Resource(s)» (снизу) панели инструментов «Target Execution»

Для остановки и запуска ресурса служат кнопки «Stop Resource(s)» и «Start Resource(s)» панели инструментов «Target Execution» (Рис. 39). Их нажатие в онлайн-режиме позволяет остановить и заново запустить исполнение ресурсов в текущем

устройстве. Данная операция может быть полезна для выполнения перезапуска программы пользователя без перезагрузки модуля CPU.

Рис. 40 – Кнопки «Cycle-to-Cycle Mode» (сверху), «One Cycle» (в центре) и «Real Time Mode» (снизу) панели инструментов «Target Execution»

Для перевода режима исполнения ресурса в пошаговый режим, исполнения одного цикла программы пользователя и обратно в непрерывный режим работы служат кнопки «Cycle-to-Cycle Mode», «One Cycle» и «Real Time Mode» панели инструментов «Target Execution» (Рис. 40). В пошаговом режиме удобно отлаживать программу пользователя.

Рис. 41 – Кнопки «Start System Events» (сверху) и «Stop System Events» (снизу) панели инструментов «Target Execution»

Для просмотра системных сообщений среды исполнения ISaGRAF служат кнопки «Start System Events» (сверху) и «Stop System Events» (снизу) панели инструментов «Target Execution» (Рис. 41). При включённом режиме вывода системных сообщений в окно Output среды исполнения выводятся в режиме реального времени системные сообщения среды исполнения ISaGRAF (Рис. 42), что может быть полезным для диагностики системных сбоев среды исполнения (если возникнет подозрение на их наличие).

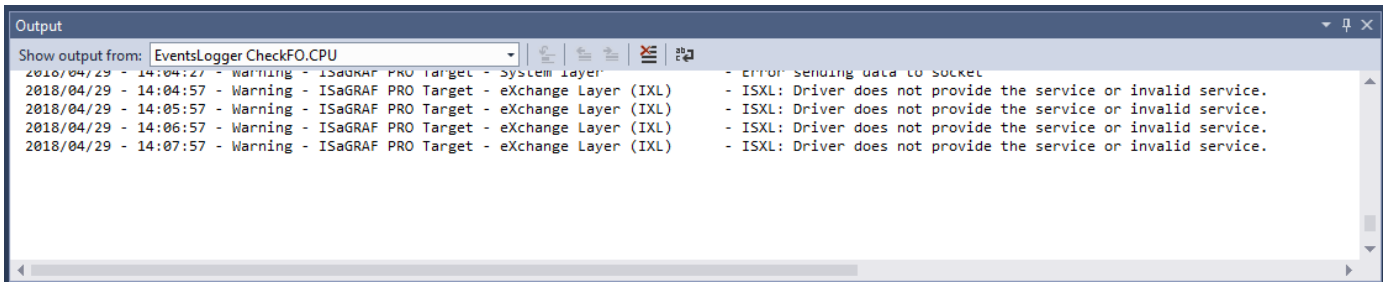


Рис. 42 – Пример вывода системных сообщений среды исполнения ISaGRAF

Глава 3 Работа с модулями изделия в среде разработки АСР Workbench ISaGRAF 6.5

3.1 Общие принципы работы с модулями изделия

Далее в главе 3 под модулями изделия в среде разработки АСР Workbench ISaGRAF 6.50 подразумеваются устройства ввода-вывода (I/O Device), соответствующие реальным модулям контроллера программируемого логического MKLogic-500 (см. раздел 1.3 КДСА.426471.004 РЭ).

Согласно документа «Монтаж ввода-вывода» (iow_ISa6_ru), все модули изделия являются комплексными устройствами ввода-вывода, т.е. состоят их нескольких простых устройств ввода-вывода.

Все прочие устройства ввода-вывода, доступные для добавления в проект, далее именуются дополнительными устройствами ввода-вывода.

3.1.1 Добавление и удаление модулей изделия

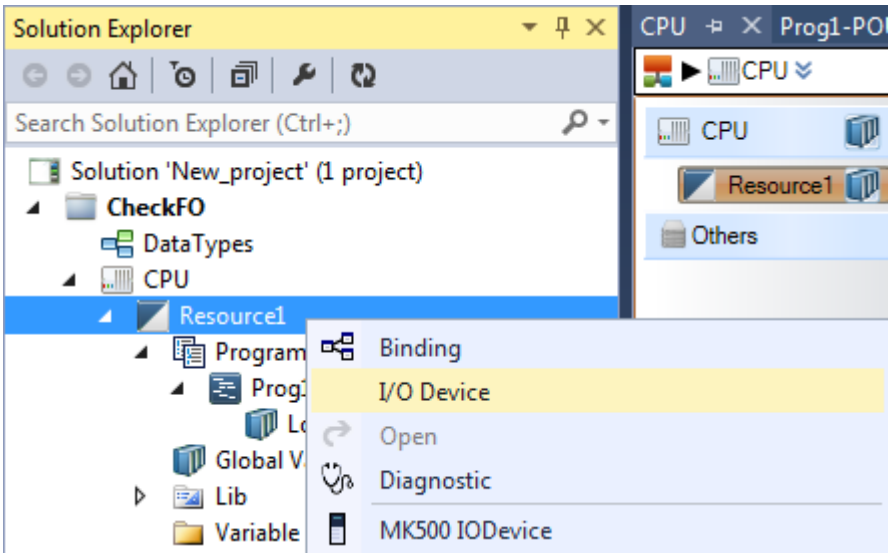


Рис. 43 – Вызов окна `I/O Device`

При работе в среде разработки АСР Workbench ISaGRAF 6.50 добавление, конфигурация и удаление модулей изделия выполняется в окне I/O Device. Для его открытия следует в дереве проекта выбрать требуемый ресурс, нажать правую кнопку мыши и из контекстного меню выбрать пункт «I/O Device» (Рис. 43).

В открывшемся окне (Рис. 44) можно увидеть панель инструментов (слева), окно устройств (в центре), окно каналов (справа сверху) и окно параметров каналов (справа снизу).

Добавление и удаление модулей изделия в проекте выполняется с помощью кнопок «Add Device» и «Delete» панели инструментов (Рис. 45).

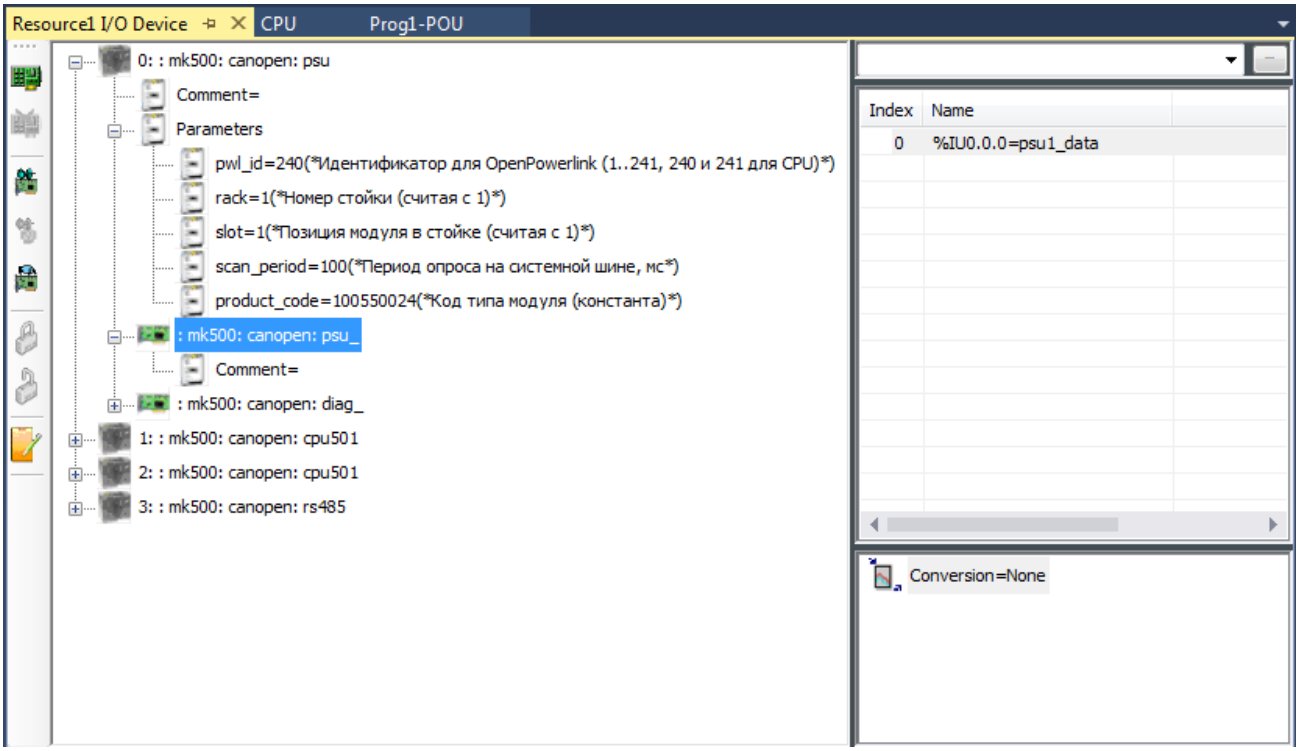


Рис. 44 – Пример окна `I/O Device`

Рис. 45 – Кнопки «Add Device» и «Delete» панели инструментов окна `I/O Device`

Нажатие на кнопку «Add Device» вызывает окно «Device Selector» (Рис. 46), в котором следует выбрать необходимый модуль изделия, задать ему индекс в списке устройств, выбрать (если это возможно) число каналов ввода-вывода, добавить псевдоним (Alias), описание (Description) и комментарий, после чего нажать кнопку «ОК» для добавления устройства.

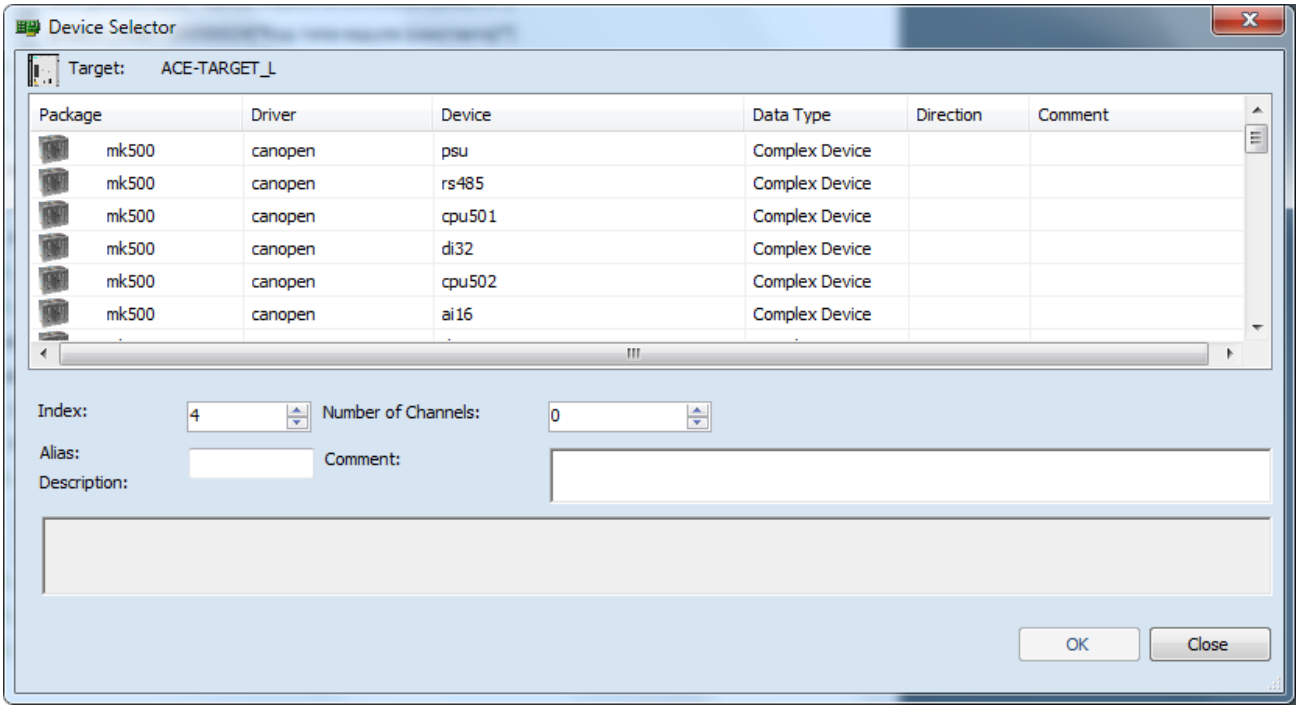


Рис. 46 – Окно «Device Selector»

Табл. 1 – Обозначения модулей изделия в окне «Device Selector» и их соответствия наименованиям модулей изделия

Обозначения модулей изделия в окне «Device Selector»	Название модуля изделия	Назначение
psu	МК-550-024	Модуль питания напряжения постоянного тока с интерфейсом CAN
cpu501	МК-501-022	Модуль центрального процессора с 2 интерфейсами Ethernet 100/1000 Base-T и 2 интерфейсами RS-485
cpu502	МК-502-142	Модуль центрального процессора с 1 оптоволоконным интерфейсом резервирования, 4 интерфейсами Ethernet 100/1000 Base-T и 2 интерфейсами RS-485
cpu503	МК-503-120	Модуль центрального процессора с 1 оптоволоконным интерфейсом резервирования, 2 интерфейсами Ethernet 100/1000 Base-T
cn545	МК-545-010	Коммуникационный модуль с 1 портом Ethernet 100/1000 Base-T с поддержкой Powerlink с двумя хаб-выходами
ai8, ai8a	МК-516-008	Модуль аналогового ввода с 8 изолированными аналоговыми входами 0-20 (4 - 20) мА
ai8a_hart	МК-576-008 А	Модуль аналогового ввода с 8 изолированными аналоговыми входами 0 - 20 (4 - 20) мА с поддержкой HART
ai16	МК-513-016	Модуль аналогового ввода с 16 аналоговыми входами 0 - 20 (4 - 20) мА
ao8, ao8a	МК-514-008	Модуль аналогового вывода с 8 аналоговыми

Обозначения модулей изделия в окне «Device Selector»	Название модуля изделия	Назначение
	МК-514-008 А	выходами 0 - 20 (4 - 20) мА
ao8a_hart	МК-574-008 А	Модуль аналогового вывода с 8 аналоговыми выходами 0 - 20 (4 - 20) мА с поддержкой HART
di32, di32history	МК-521-032	Модуль дискретного ввода напряжения постоянного тока с 32 дискретными входами
do32	МК-531-032	Модуль дискретного вывода напряжения постоянного тока с 32 дискретными выходами
rs485, rs485new	МК-541-002	Коммуникационный модуль с 2 интерфейсами RS-485

Нажатие на кнопку «Delete» панели инструментов или клавиатуры удалит выбранный модуль изделия из списка устройств.

⚠ ВНИМАНИЕ

Нажатие кнопки «Delete» удаляет выбранный модуль изделия без дополнительных подтверждений, поэтому следует быть внимательным при использовании этой кнопки, и в случае ошибочного удаления использовать команду «Undo» панели инструментов среды разработки, или комбинацию клавиш «Ctrl+Z».

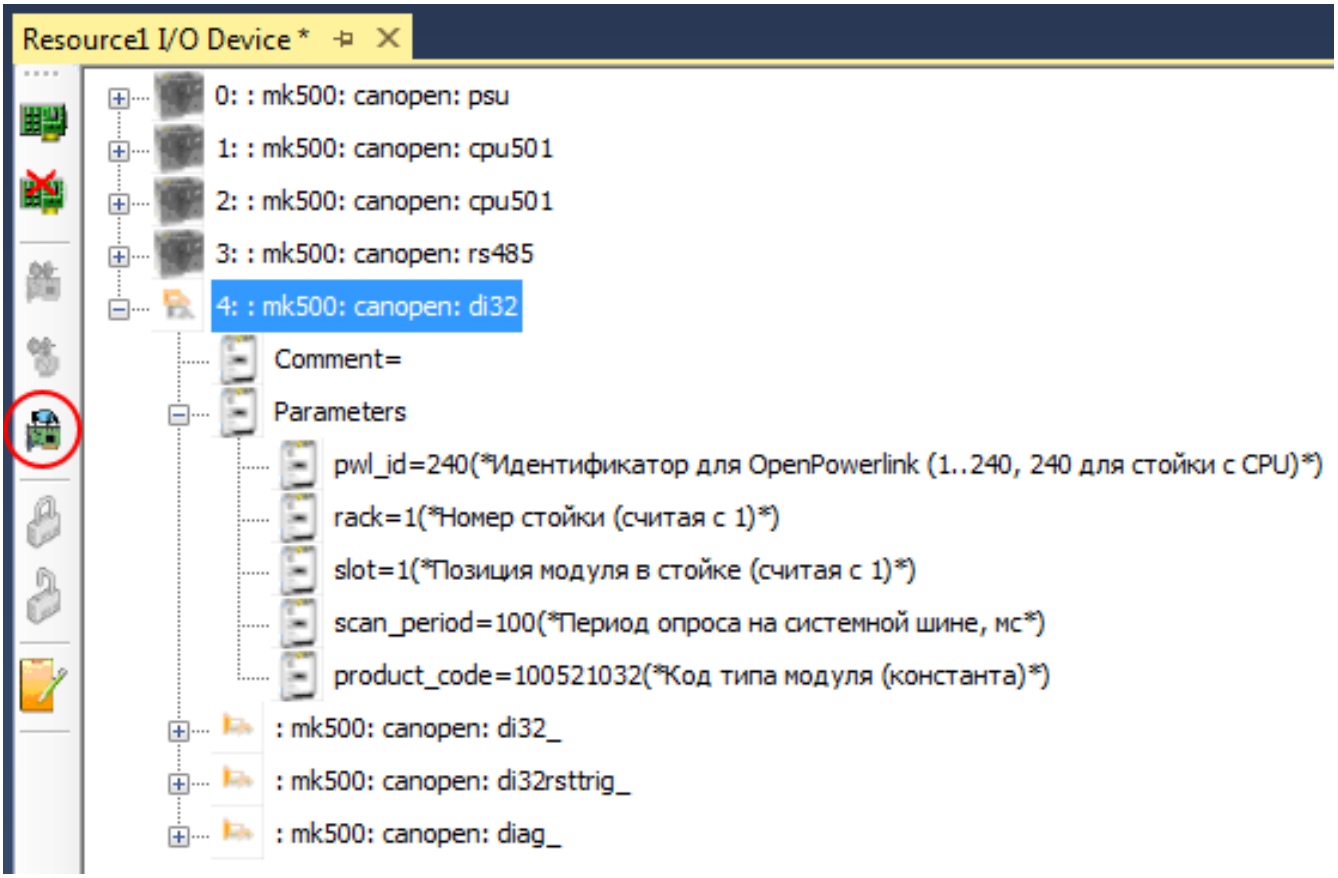


Рис. 47 – Кнопка «Real/Virtual I/O Device»

Если необходимо временно исключить модуль изделия из проекта, но не удалять, можно перевести его в виртуальное состояние. Для этого следует выбрать в окне устройств необходимый модуль изделия и нажать на панели инструментов окна кнопку «Real/Virtual I/O Device». В результате (Рис. 47) выбранный модуль изделия «побледнеет» в окне устройств, и не будет обрабатываться средой исполнения ISaGRAF.

Для вывода модуля изделия из виртуального состояния следует выбрать его в окне устройств и повторно нажать кнопку «Real/Virtual I/O Device».

3.1.2 Ранжирование модулей изделия

Под ранжированием модулей изделия понимается изменение положения модулей изделия в списке путём изменения индекса модулей. Индекс модуля изделия в списке (отображается справа от иконки устройства в окне устройств, Рис. 44) важен, так как он определяет порядок (от меньшего индекса к большему) инициализации, вызова функций ввода-вывода и деинициализации модуля в ходе работы среды исполнения.

При проектировании рекомендуется назначать индексы модулям не подряд, а разделяя модули на группы по принадлежности к стойкам (например, первая стойка – индексы с 0 по 99, вторая стойка – индексы с 200 по 299 и т.п.), оставляя интервалы, как между отдельными модулями изделия, так и между стойками. Этот подход позволит относительно легко добавить новые либо ранжировать ранее добавленные модули изделия.

⚠ ВНИМАНИЕ

При назначении индексов модулям нельзя превышать значение в 32767, возможно неопределённое поведение CPU.

Для ранжирования модуля изделия (а также для редактирования псевдонима и/или комментария к модулю) следует дважды кликнуть левой кнопкой мыши на требующем корректировки модуле. Откроется окно «Device Selector», без возможности добавлять новые модули, но с возможностью изменить индекс модуля, псевдоним либо комментарий. После введение нового индекса (не совпадающего с уже существующим) следует нажать «OK».

3.1.3 Настройка параметров модуля изделия

Все модули изделия имеют унифицированные параметры (Рис. 48):

- pwl_id – идентификатор на шине Powerlink. Для модулей ввода-вывода, находящихся на одной шине CAN с процессорными модулями следует оставлять значение 240.
- rack – номер стойки модуля (считая с 1), должен совпадать с положением переключателя «ADDRESS» на блоке питания стойки;
- slot – номер слота модуля в стойке (считая с 1), следует указывать порядковый номер модуля в стойке, учитывая особые случаи, описанные в разделе 11.3 КДСА.426471.004 РЭ;

– scan_period – период опроса устройства на шине CAN, в мс, следует выбирать сообразно требованиям к скорости обновления данных в программе пользователя. Не рекомендуется устанавливать период меньше 10 мс;
– product_code – константа, соответствует коду модуля устройства.

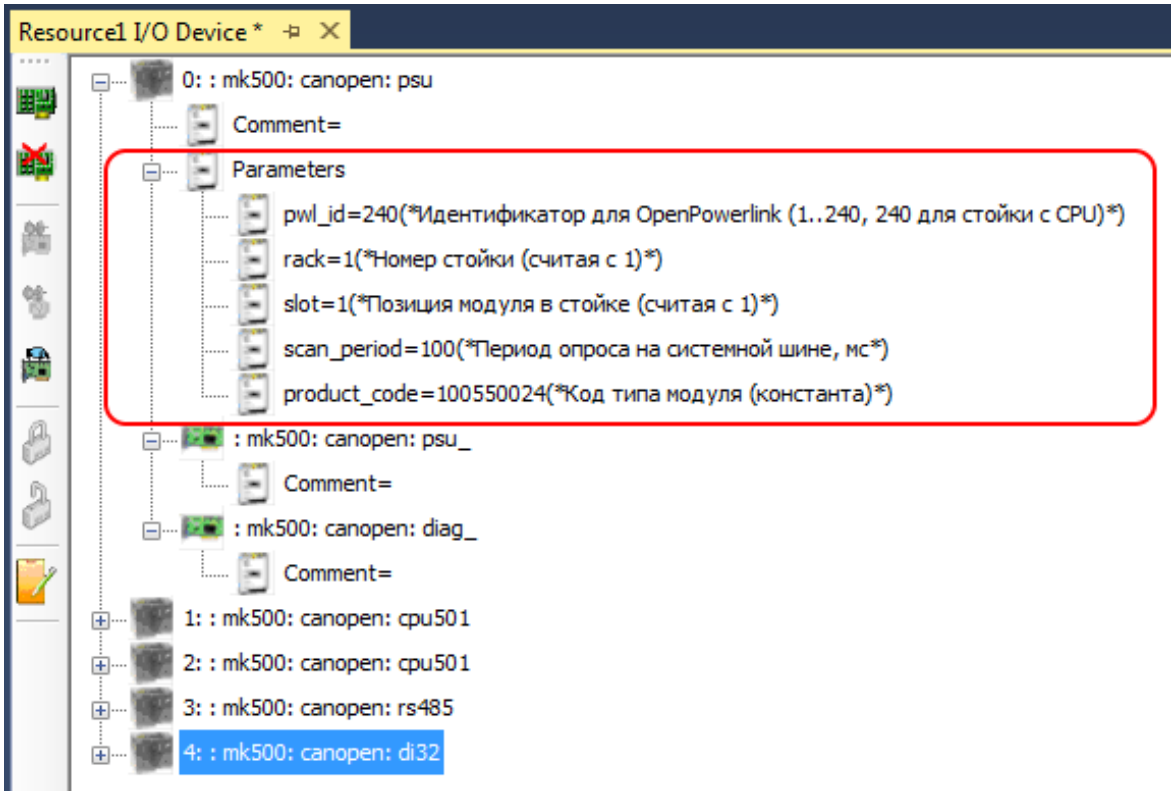


Рис. 48 – Параметры модуля изделия

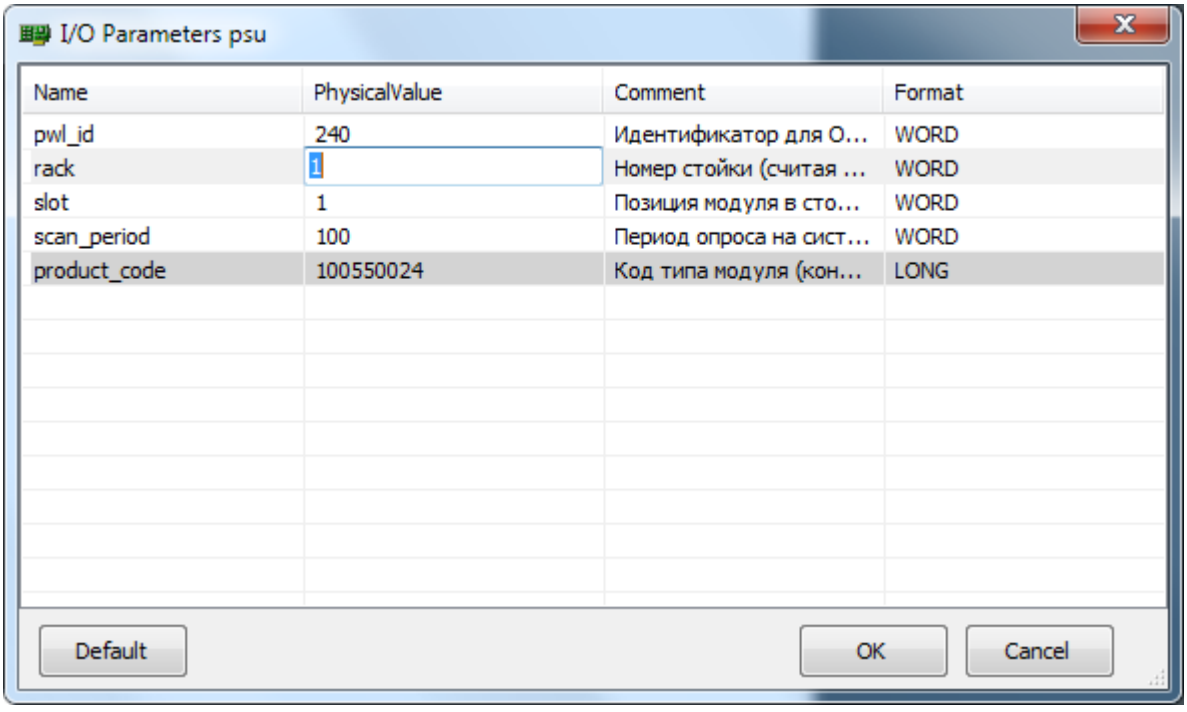


Рис. 49 – Окно редактирования параметров модуля изделия

Для настройки параметров модуля изделия следует после добавления устройства в «I/O Device» дважды кликнуть левой кнопкой мыши на любом из параметров модуля. В открывшемся окне редактирования (Рис. 49) следует настроить параметры модуля изделия, затем нажать кнопку «OK».

3.1.4 Привязка каналов модулей изделия

После добавление модуля изделия в проект и настройки его параметров следует связать его каналы с пользовательскими переменными проекта.

⚠ ВНИМАНИЕ

У абсолютно всех модулей изделия проекта (включая дополнительные устройства ввода-вывода) следует обязательно связать с пользовательскими переменными все их каналы, особенно в случае проекта с поддержкой Failover. Пренебрежение этим правилом приводит к постоянному выполнению циклов синхронизации и, как следствие, к завышенному и негарантированному времени выполнения программы пользователя и к повышенной на 30-40% нагрузке на процессоры модулей CPU.

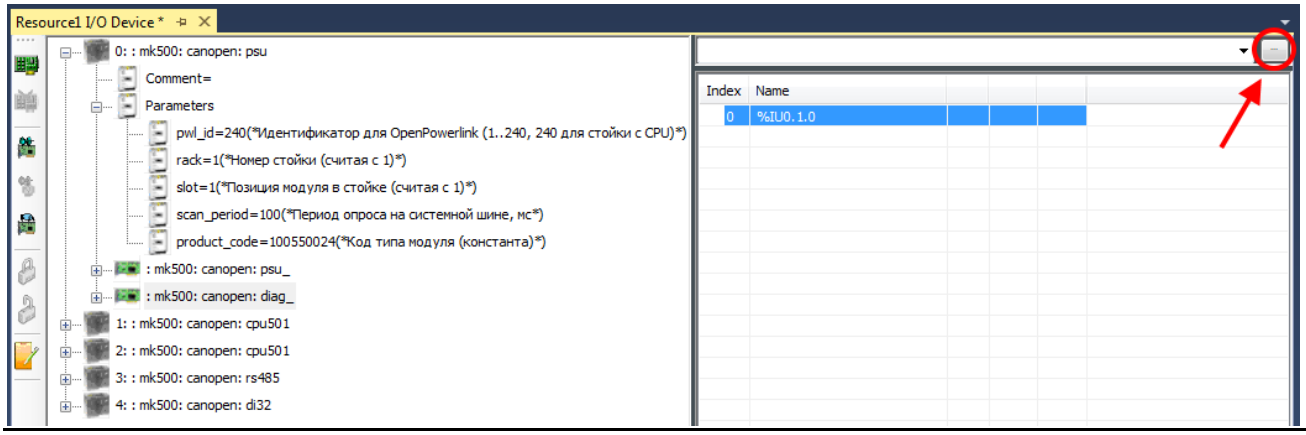


Рис. 50 – Кнопка «Variable Selector» (правый верхний угол)

Для привязки канала ввода-вывода модуля изделия следует выбрать в окне устройств интересующее нас простое устройство и в окне каналов дважды кликнуть левой кнопкой мыши на интересующем нас канале либо нажать кнопку «Variable Selector» (Рис. 50).

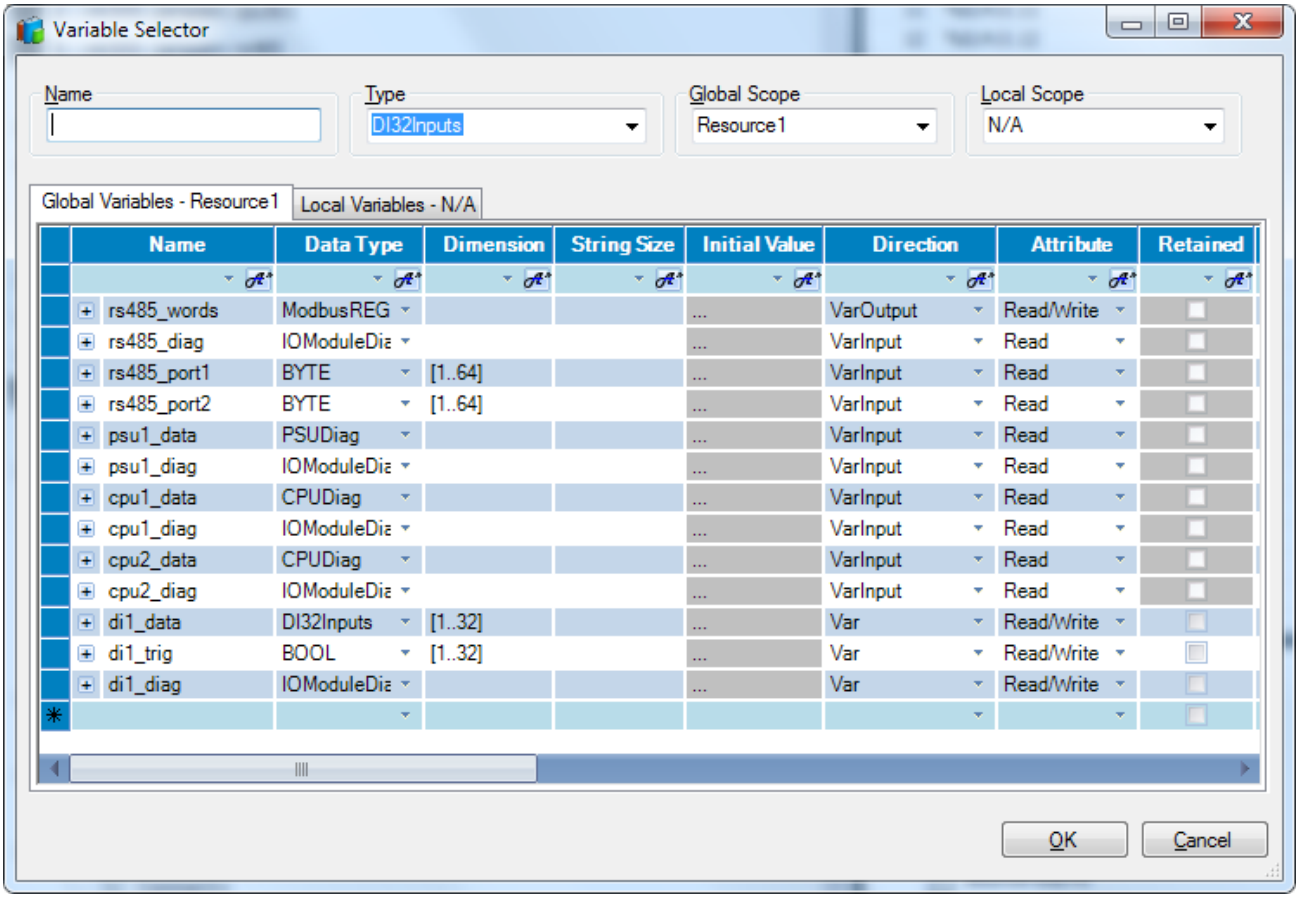


Рис. 51 – Окно «Variable Selector»

В открывшемся окне выбора переменной (Рис. 51) следует либо выбрать уже созданную переменную из списка, либо создать свою переменную, после чего нажать кнопку «OK». Допускается выбор массива переменных, в этом случае массив будет спроецирован на последовательность каналов, начиная с выбранного.

В случае несоответствия типа и/или направления (VarInput/VarOutput) переменной типу и направлению переменной канала, будет выдано окно с предложением скорректировать тип и/или направление выбранной переменной (Рис. 52).

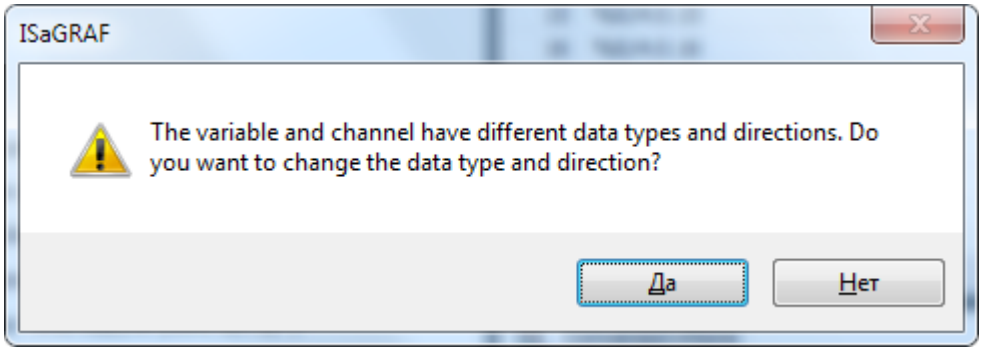


Рис. 52 – Предложение скорректировать тип и направление выбранной в «Variable Selector» переменной

Надо отметить, что выходные переменные каналов с направлением VarOutput создаются с атрибутом доступа Write Only. Однако для них в окне «Variable Selector» можно переключить атрибут доступа из Write only в Read/Write (Рис. 53), и в дальнейшем не только записывать в канал выходные данные, но и читать из него возвращаемые значения. Тем самым реализуется отсутствующее в ISaGRAF направление VarInputOutput. Подробнее см. в пп. 3.7.1-3.7.3 и п. 3.12.

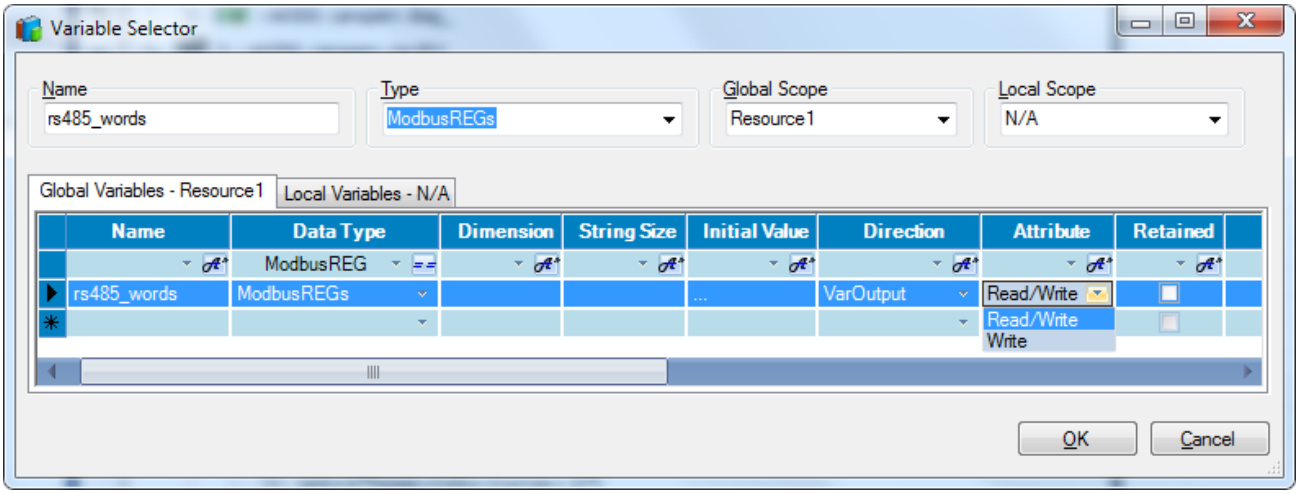


Рис. 53 – Смена атрибута доступа для переменной с направлением `VarOutput`

При необходимости отвязать переменную от канала следует выбрать канал и нажать кнопку «Free channel» на панели инструментов (Рис. 54).

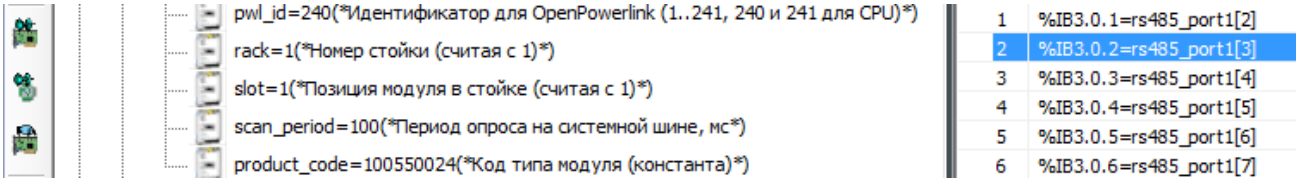


Рис. 54 – Кнопки «Free all channels of selected device» (слева сверху) и «Free channel» (слева посередине) панели инструментов окна `I/O Device`

При необходимости отвязать все переменные от всех каналов модуля следует выбрать канал и нажать кнопку «Free all channels of selected device» на панели инструментов (Рис. 54).

3.1.5 Настройки параметров каналов модулей изделия

Среда разработки АСР позволяет задать для каждого канала модуля изделия (в зависимости от типа данных канала) функцию преобразования значения, коэффициенты линейного масштабирования, режим инверсии, а также параметры, если они предусмотрены для этого модуля изделия. Текущие значения параметров выбранного канала отображаются в окне параметров канала (Рис. 55).

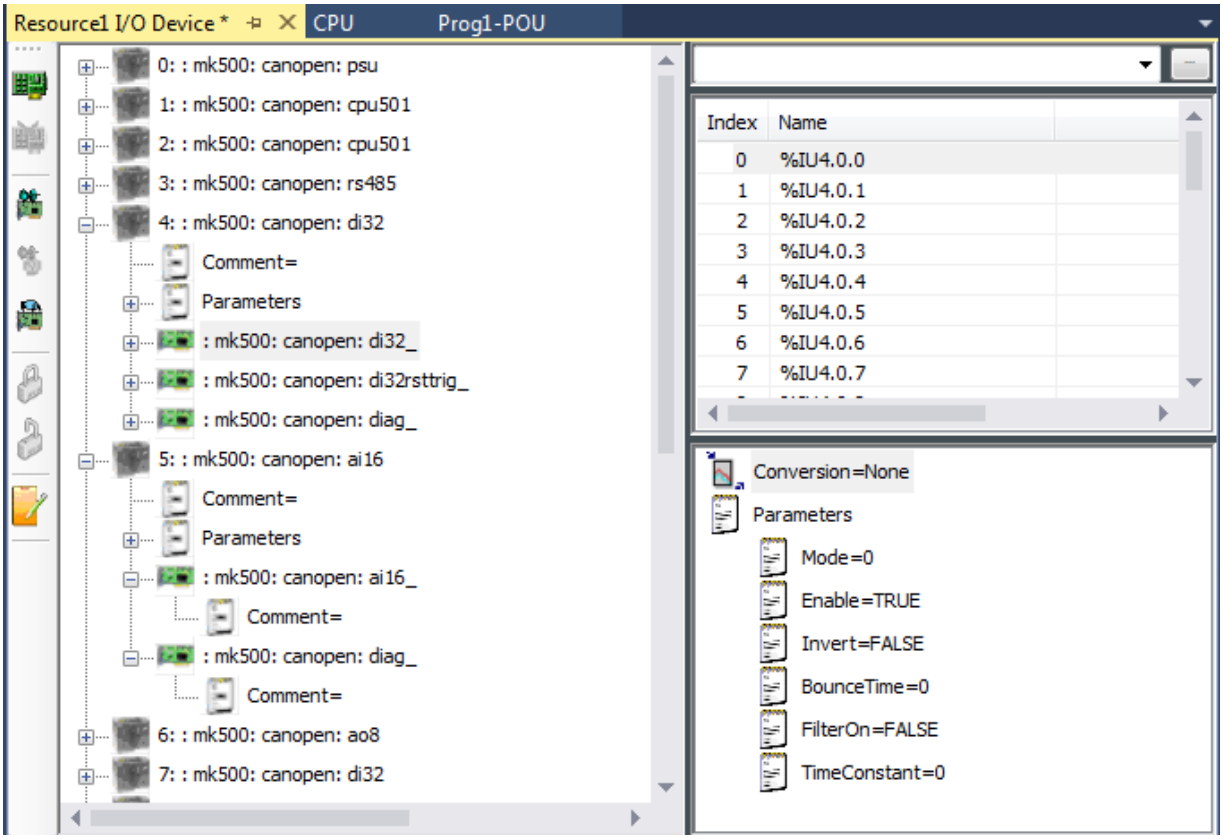


Рис. 55 – Пример содержимого окна параметров канала модуля дискретных входов

Функция преобразования (Conversion) присутствует для всех типов каналов, но ни для одного из модулей изделия она не определена (всегда None), поэтому в дальнейшем не упоминается.

Коэффициенты линейного масштабирования Gain и Offset могут быть заданы для каналов с целыми типами данных (BYTE, WORD). Для редактирования коэффициентов следует дважды кликнуть левой кнопкой мыши на поле Gain или Offset в окне параметров канала, затем в окне «I/O Filter» задать необходимые значения в полях Gain и Offset и нажать кнопку «OK» (Рис. 56).

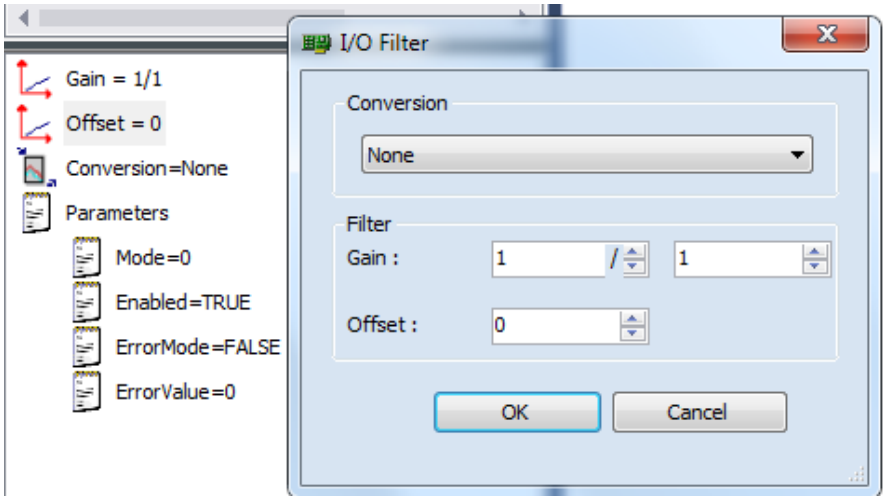


Рис. 56 – Окно изменения коэффициентов линейного масштабирования значения канала модуля аналоговых выходов

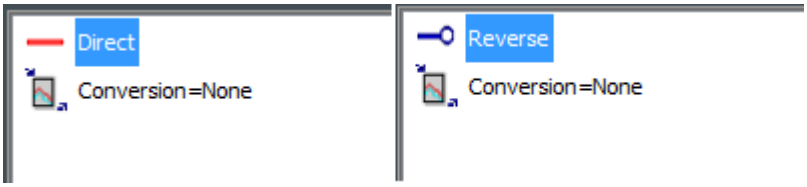


Рис. 57 – Применение режима инверсии к каналу модуля дискретных входов

Режим инверсии может быть применён для каналов с булевым типом данных (BOOL). Для применения режима инверсии следует в окне параметров канала дважды кликнуть левой кнопкой мыши на поле Direct, которое должно поменяться на Invert (Рис. 57). Повторным двойным кликом канал переводится в режим работы без инверсии.

Редактирование параметров каналов рассматривается ниже, в разделах, посвящённых работе с конкретными модулями ввода-вывода изделия.

3.1.6 Диагностический канал модулей изделия

Все модули изделия имеют в своём составе диагностический канал, реализованный с помощью простого устройства `diag_` (Рис. 58). Диагностический канал предназначен для получения идентификационной информации о модуле изделия, состоянии его CAN-шин и о том, совместим ли реально подключенный модуль с модулем изделия, добавленным в проект.

Диагностический канал имеет одну переменную-структуру типа `IOModuleDiag`. Поля структуры приведены на Рис. 59.

Пояснения к полям структуры `IOModuleDiag`:

- поля с `vendorID` по `hwVersion` служат для идентификации модуля;
- поле `currentCANbus` принимает значение 0, если рабочая CAN-шина не определена, 1 для текущей шины CAN1 и 2 для текущей шины CAN2;
- поля `CAN1heartbeat` и `CAN2heartbeat` принимают значения 127, если модуль изделия не в рабочем состоянии по данной шине и 5, если модуль изделия инициализован и в рабочем состоянии;
- поле `isCompatible` принимает значение TRUE, если тип реально установленного в данной стойке и позиции модуля изделия совпадает с заданным в конфигурации;
- поле `state` содержит код, соответствующий состоянию светодиодной индикации модуля;
- поле `crc32` содержит контрольную сумму метрологически значимой части ПО модуля (для модулей аналогового ввода-вывода).

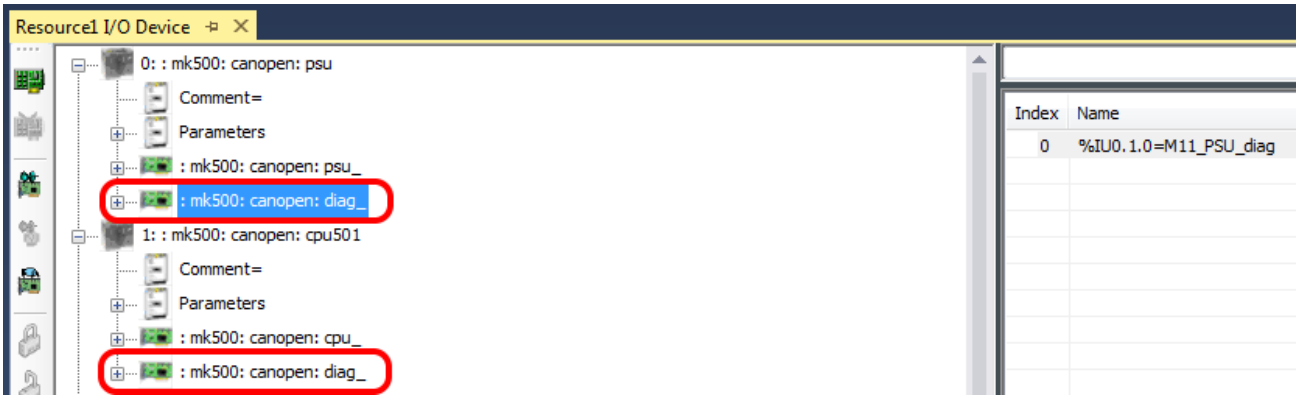


Рис. 58 – Диагностические каналы в составе модулей изделия

Resource1-VAR

Name	Logical Value	Physical Value	Comment
+ PSU_Diag_4_1	...	...	
+ PSU_Diag_5_1	...	...	
- PSU_IOdiag_1_1	...	...	
PSU_IOdiag_1_1.vendorID	0	0	Идентификатор организации-производителя модуля
PSU_IOdiag_1_1.productCode	0	0	Код модуля
PSU_IOdiag_1_1.revisionNumber	0	0	Номер ревизии модуля
PSU_IOdiag_1_1.serialNumber	0	0	Серийный номер модуля
PSU_IOdiag_1_1.deviceType	0	0	Код типа модуля
PSU_IOdiag_1_1.manufStatus	0	0	Регистр статуса (версия производителя)
PSU_IOdiag_1_1.deviceName	---	---	Наименование модуля
PSU_IOdiag_1_1.swVersion	---	---	Версия программного обеспечения модуля
PSU_IOdiag_1_1.hwVersion	---	---	Версия аппаратного обеспечения модуля
PSU_IOdiag_1_1.currentCANbus	0	0	Номер текущей рабочей CAN-шины
PSU_IOdiag_1_1.CAN1heartbeat	0	0	Heartbeat модуля на шине CAN1
PSU_IOdiag_1_1.CAN2heartbeat	0	0	Heartbeat модуля на шине CAN2
PSU_IOdiag_1_1.isCompatible			TRUE - если модуль на шине совместим с требуемым
PSU_IOdiag_1_1.state	0	0	Код состояние модуля
PSU_IOdiag_1_1.crc32	0	0	Контрольная сумма метрологически значимой части ПО модуля, если она есть

Рис. 59 – Описание структуры диагностики `IOModuleDiag` модуля изделия

3.2 Общие принципы работы с дополнительными модулями ввода-вывода

К дополнительным модулям ввода вывода в данном документе относятся устройства ввода-вывода ISaGRAF, доступные к добавлению в окно «I/O Device» и не относящиеся к модулям изделия (Табл. 2). Их общая черта – отсутствие привязки к физическому устройству в составе стойки.

Дополнительные устройства ввода-вывода можно поделить на несколько групп:

- устройства, реализующие поддержку протокола Modbus RTU/TCP в модулях CPU;
- устройства, реализующие поддержку протокола IEC 60870-5-104 в модулях CPU;
- устройства, реализующие поддержку протокола OPC-UA;
- устройства, реализующие поддержку протокола Powerlink;
- устройства, реализующие вспомогательные функции в рамках ресурса.

Табл. 2 – Обозначения дополнительных модулей ввода-вывода в окне «Device Selector» и их функциональное назначение

Обозначения дополнительных модулей ввода в окне «Device Selector»	Функциональное назначение модуля ввода-вывода
modbusRTU_mst*	Поддержка протокола Modbus RTU (ведущий) в модуле CPU (п. 3.7.1)
modbusRTU_sl*	Поддержка протокола Modbus RTU (ведомый) в модуле CPU (п. 3.7.2)
modbusTCP_mst*	Поддержка протокола Modbus TCP (ведущий) в модуле CPU (п. 3.7.1)
modbusTCP_sl*	Поддержка протокола Modbus TCP (ведомый) в модуле CPU (п. 3.7.2)
SerialPort_	Используется в составе модулей modbusRTU_mst* и modbusRTU_sl*, не рекомендуется к использованию отдельно
EthernetPort_	Используется в составе модулей modbusTCP_mst*, modbusTCP_sl* и opcua_server. Отдельно - поддержка протокола IEC 60870-5-104 (п. 3.7.3) в модуле CPU
iec104*	Поддержка протокола IEC 60870-5-104 (сервер) в модуле CPU (п. 3.7.3)
opcua_server	Поддержка протокола OPC-UA (сервер) в модуле CPU (п. 3.7.4)
mk546	Коммуникационный модуль-расширение для МК-503-120 Powerlink MN с 2 портами
powerlinkMN	OpenPowerlink, программный модуль функций MN для МК-502-142
current_time	Текущее время активного CPU
system_info	Системные параметры активного CPU (п. 3.7.4)

Все дополнительные модули ввода-вывода необходимо добавлять в проект после модулей изделия, так как дополнительные модули ввода-вывода могут в своей работе ссылаться на модули CPU (п. 3.7).

Дополнительные модули ввода-вывода могут быть как комплексными, так и простыми.

В Табл. 2 приведён перечень доступных дополнительных модулей ввода-вывода и их функциональное назначение.

3.3 Общие принципы работы с модулями центрального процессора

Наличие модулей CPU (`cpu501`, `cpu502` или `cpu503`, см. Табл. 1) в конфигурации является обязательным. Для проектов с поддержкой Failover необходимым является наличие в конфигурации проекта двух модулей CPU.

⚠ ВНИМАНИЕ

Запрещено наличие более одного модуля CPU в конфигурации без поддержки резервирования и более двух модулей CPU в конфигурации с поддержкой Failover. Нарушение этого правила приведёт к невозможности работы всех модулей CPU проекта.

Модули CPU обеспечивают в проекте индикацию текущего состояния программы пользователя на лицевой панели модуля CPU изделия и предоставляют диагностическую информацию о состоянии физического модуля CPU (нагрузка на процессор, состояние портов Ethernet и т.п., подробно см. п. 3.7).

Также модули CPU являются контейнерами последовательных портов и портов Ethernet, на них неявно ссылаются простые устройства `SerialPort_` и `EthernetPort_`.

3.4 Общие принципы работы с модулями ввода-вывода

Модули ввода-вывода изделия (`ai16`, `ai8`, `ai8a`, `ai8ahart`, `ao8`, `ao8a`, `ao8ahart`, `di32`, `di32history`, `do32`, см. Табл. 1) обеспечивают взаимодействие программы пользователя с аналоговыми и дискретными входами и выходами физических модулей ввода-вывода. Модуль `psu` не обеспечивает взаимодействия с входами и выходами контроллера, но позволяет получить информацию о текущем состоянии модуля блока питания.

Скорость обновления данных в программе пользователя определяется двумя факторами: заданным временем цикла программы `Cycle Time` (п. 2.5.2) и периодом опроса данных модуля в его параметрах `scan_period` (п. 3.1.3). Для обновления данных модуля ввода-вывода в программе пользователя в каждом цикле программы период опроса модуля должен быть меньше времени цикла в 2-3 раза. При этом не рекомендуется делать период опроса модулей ввода-вывода слишком малым (меньше 20..30 мс) во избежание резкого роста нагрузки на процессор модуля CPU.

3.5 Общие принципы работы с коммуникационными модулями

Коммуникационные модули изделия (`rs485`, `rs485new`, `cn545`, см. Табл. 1) обеспечивают обмен с периферийными устройствами по протоколу Modbus RTU и Powerlink.

При выборе периода опроса данных коммуникационных модулей следует руководствоваться теми же соображениями, что и для модулей ввода-вывода (п. 3.4).

Коммуникационные модули изделия `rs485` и `rs485new` позволяют выполнять на каждый порт RS485 до 64 независимых команд, и имеют 960 регистров Modbus адресного пространства. Адресное пространство `rs485` и `rs485new` является общим для всех типов команд, и доступно через канал ввода-вывода `rs485data_` (п. 3.12).

⚠ ВНИМАНИЕ

В связи с ограниченной пропускной способностью шины CAN, допускается использование не более 8 коммуникационных модулей rs485 и rs485new в одном проекте, с суммарным числом не более 2000 используемых регистров Modbus адресного пространства. Нарушение этой рекомендации приведёт к нестабильному обмену на шине CAN.

В отличие от остальных модулей ввода-вывода, инициализация коммуникационного модуля изделия `rs485` и переход его в рабочее состояние происходит строго после вызова функции `InitRS485ModuleModbus` (п. 3.12). Также вызовы этой функции используются для передачи команд портам RS485 коммуникационного модуля изделия.

У модуля изделия `rs485new` для передачи команд портам RS485 коммуникационного модуля в составе есть устройства `rs485request1_` и `rs485request2_`. Вызов функции инициализации `InitRS485ModuleModbus` не требуется.

При работе переменной, связанной с каналом ввода-вывода `rs485data_`, следует соблюдать следующие рекомендации:

- в программе следует работать с отдельной копией данных канала ввода-вывода, не привязанной к каналу; копирование данных в рабочую переменную следует выполнять в начале цикла обработки, копировать изменённые данные рабочей переменной в связанную переменную следует в конце цикла обработки данных коммуникационного модуля.
- не следует объединять в одной переменной массива данных коммуникационного модуля выходное значение (например, код команды) и возвращаемое значение (например, результат выполнения команды). Значение этой переменной будет слишком зависеть от соотношения периодов опроса коммуникационного модуля, времени цикла программы и времени обработки этой переменной в коммуникационном модуле. В случае неверного выбора этих времён (например, время цикла программы слишком мало) значение переменной может не изменяться вообще по причине постоянного обновления его в программе ещё не обновившимся значением со стороны коммуникационного модуля.

Модули `cn545` позволяют установить время цикла обмена по Powerlink (параметр `cycleTime_ms`) и предоставляют диагностическую информацию о состоянии физического модуля МК-545 (версию прошивки, состояние портов Ethernet и количество ошибок диагностического канала, подробно см. п. 3.7.6).

⚠ ВНИМАНИЕ

Для коммуникационного модуля `cn545` приоритетным является время цикла обмена по Powerlink, указанное в конфигурационном файле (см. п. 3.7.5). Если в конфигурационном файле время цикла обмена по Powerlink не задано, используется время, заданное параметром `cycleTime_ms` коммуникационного модуля `cn545`.

3.6 Работа с модулями питания МК-550-024

Согласно Табл. 1, модулю питания МК-550-024 в среде разработки ACP Workbench ISaGRAF 6.50 соответствует модуль изделия `psu`.

Кроме диагностического канала, модуль изделия `psu` имеет в своём составе канал данных модуля питания МК-550-024, реализованный в простом входном устройстве `psu_`. Данный канал имеет одну переменную-структуру типа `PSUDiag` (Рис. 60), предназначенную для диагностики состояния модуля питания.

Name	Data Type	String Size	Comment
PSUDiag			
PSUDiag	PSUDiag		
voltage	REAL		Выходное напряжение, В
temperature	INT		Температура на плате, в градусах Цельсия
errorCode	WORD		Код ошибки работы модуля
id	WORD		ID модуля
canBusSpeed	WORD		Скорость CAN-шины, кБит/с
*			

Рис. 60 – Описание структуры данных `PSUDiag` модуля питания изделия

В поле `errorCode` модуль возвращает битную маску ошибок в своей работе. Расшифровка кодов ошибок работы модуля приведена в Табл. 3.

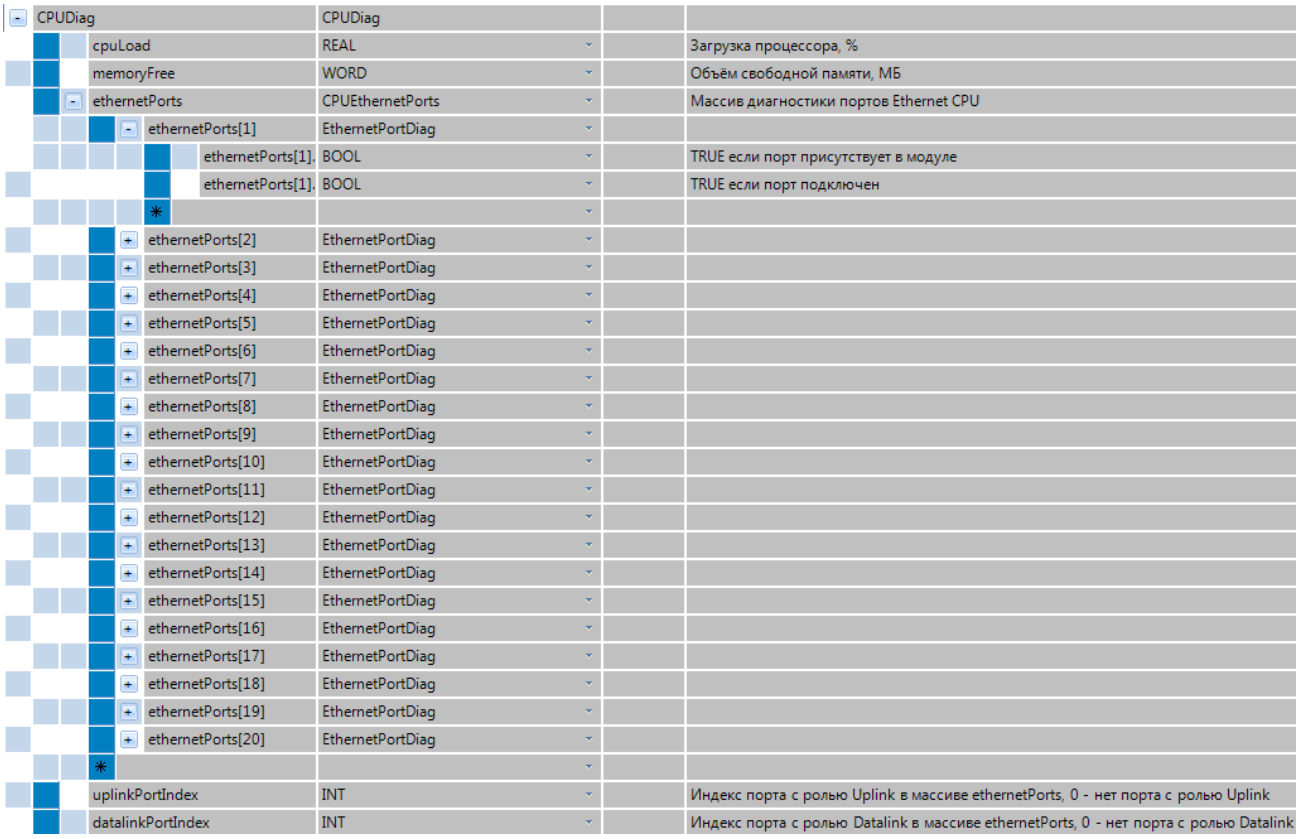
Табл. 3 – Расшифровка кодов ошибок `errorCode` структуры `PSUDiag`

Номер бита errorCode	Расшифровка
0	1 – пониженное напряжение 5В
1	1 – переключатель адреса CAN (ADDRESS) в запрещённом положении
2	1 – переключатель скорости CAN (BITRATE) в запрещённом положении
3..15	Не используются

3.7 Работа с модулями центрального процессора МК-501-022, МК-502-142 и МК-503-120

Согласно Табл. 1, модулям центрального процессора МК-501-022, МК-502-142 и МК-503-120 в среде разработки ACP Workbench ISaGRAF 6.50 соответствуют модуль изделия `cpu501`, `cpu502` и `cpu503` соответственно.

Кроме диагностического канала, модули изделия `cpu501`, `cpu502` и `cpu503` имеют в своём составе канал данных модуля CPU, реализованный в простом устройстве `cpu_`. Данный канал имеет одну переменную-структуру типа `CPUDiag` (Рис. 61).



CPUDiag	CPUDiag		
cpuLoad	REAL		Загрузка процессора, %
memoryFree	WORD		Объём свободной памяти, МБ
ethernetPorts	CPUEthernetPorts		Массив диагностики портов Ethernet CPU
ethernetPorts[1]	EthernetPortDiag		
ethernetPorts[1].	BOOL		TRUE если порт присутствует в модуле
ethernetPorts[1].	BOOL		TRUE если порт подключен
ethernetPorts[2]	EthernetPortDiag		
ethernetPorts[3]	EthernetPortDiag		
ethernetPorts[4]	EthernetPortDiag		
ethernetPorts[5]	EthernetPortDiag		
ethernetPorts[6]	EthernetPortDiag		
ethernetPorts[7]	EthernetPortDiag		
ethernetPorts[8]	EthernetPortDiag		
ethernetPorts[9]	EthernetPortDiag		
ethernetPorts[10]	EthernetPortDiag		
ethernetPorts[11]	EthernetPortDiag		
ethernetPorts[12]	EthernetPortDiag		
ethernetPorts[13]	EthernetPortDiag		
ethernetPorts[14]	EthernetPortDiag		
ethernetPorts[15]	EthernetPortDiag		
ethernetPorts[16]	EthernetPortDiag		
ethernetPorts[17]	EthernetPortDiag		
ethernetPorts[18]	EthernetPortDiag		
ethernetPorts[19]	EthernetPortDiag		
ethernetPorts[20]	EthernetPortDiag		
uplinkPortIndex	INT		Индекс порта с ролью Uplink в массиве ethernetPorts, 0 - нет порта с ролью Uplink
datalinkPortIndex	INT		Индекс порта с ролью Datalink в массиве ethernetPorts, 0 - нет порта с ролью Datalink

Рис. 61 – Описание структуры данных `CPUDiag` модуля CPU изделия

3.7.1 Реализация поддержки протоколов `Modbus RTU` и `Modbus/TCP` (ведущий) в модулях CPU

Для работы со встроенным в модуль CPU протоколом Modbus RTU/TCP в режиме ведущий предназначены специальные составные устройства (Табл. 4).

Табл. 4 – Типы `Modbus` устройств с режимом ведущий

Тип Modbus-устройства	Имя составного устройства	Имена простых устройств	Число регистров WORD	Число регистров BOOL	Число команд

Малое	`modbusRTU_mst_4k`
	`modbusTCP_mst_4k`
		`EthernetPort_/`
		`SerialPort_`
		`mb_ctrl8_`
		`mb_ctrl_diag8_`
		`mb_regs4_`
		`mb_bits1_`
		`mb_control_`
			4096	1024	128

Среднее	`modbusRTU_mst_16k`
	`modbusTCP_mst_16k`
		`EthernetPort_/`
		`SerialPort_`
		`mb_ctrl16_`
		`mb_ctrl_diag16_`
		`mb_regs16_`
		`mb_bits4_`
		`mb_control_`
			16384	4096	256

Тип Modbus-устройства	Имя составного устройства	Имена простых устройств	Число регистров WORD	Число регистров BOOL	Число команд
Большое	modbusRTU_mst_64k modbusTCP_mst_64k	EthernetPort_/ SerialPort_ mb_ctrl32_ mb_ctrl_diag32_ mb_regs64_ mb_bits16_ mb_control_	65536	16384	512

Составные Modbus-устройства с поддержкой режима «ведущий» включают в себя простые устройства, с помощью которых реализуется привязка к аппаратному порту, адресное пространство, управление и диагностика устройства. Составные Modbus-устройства созданы трёх типов: малое, среднее и большое, отличаясь доступным размером пространства регистров и числом команд.

Для ускорения синхронизации данных устройств Modbus RTU/TCP при работе с включённым режимом Failover, рекомендуется выбирать составные Modbus-устройства с количеством регистров, максимально соответствующим задаче (то есть для задачи с потребностью в 1000 регистров WORD выбирать малое Modbus-устройство, а не среднее или большое).

В состав Modbus-устройства с поддержкой режима ведущий входят следующие простые устройства:

1. Входное устройство порт последовательный (`SerialPort_`) или Ethernet (`EthernetPort_`), предназначенное для привязки Modbus-устройства к аппаратному порту модуля CPU. Данный канал имеет одну переменную-структуру типа `CPUPortStatus` (Рис. 62), предназначенную для диагностики работы порта (Табл. 5). Через параметры этого простого устройства (Рис. 63 и Рис. 64) выполняется настройка физического порта модуля CPU.

Name	Data Type	String Size	Comment
CPUPortStatus			
CPUPortStatus	CPUPortStatus		
mode	INT		Режим работы порта (0 - свободен, 1 - ModbusSlave, 2 - ModbusMaster)
status	INT		Текущее состояние порта
parameters	STRING	80	Строка параметров порта

Рис. 62 – Описание структуры `CPUPortStatus`

Табл. 5 – Расшифровка значений поля `status` структуры `CPUPortStatus`

Значение status	Расшифровка
1	Порт инициализован и готов к работе
0	Порт инициализован, но не в работе
-1	Ошибка инициализации библиотеки Modbus
-2	Ошибка подключения к порту (в режиме ведомого)

Значение status	Расшифровка
-3	Ошибка выделения памяти
-4	Ошибка работы с портом
-5	Системная ошибка
-6	Ошибка выделения памяти в режиме ведущего
-7	Не инициализован массив команд в режиме ведущего
-8	Порт уже занят

Для привязки к аппаратному порту модуля CPU следует в параметрах устройства `SerialPort_` или `EthernetPort_` настроить поле `port_id` согласно Табл. 6.

Так как поле `port_id` ссылается на аппаратный порт в составе модуля CPU, следует в ходе ранжирования модулей (п. 3.1.2) помещать Modbus-устройство ниже модулей CPU, в противном случае Modbus-устройство работать не будет.

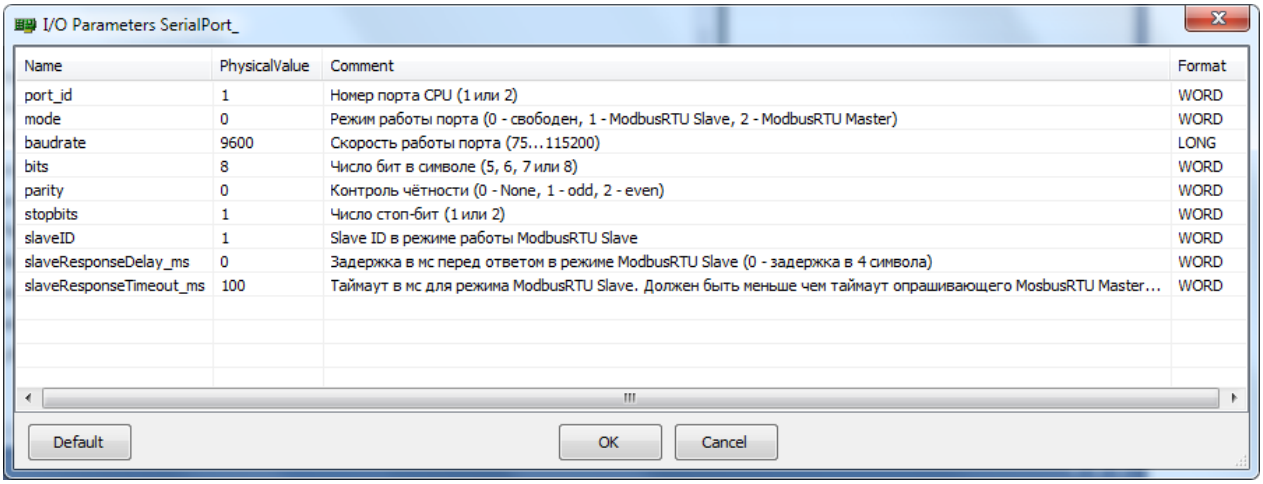


Рис. 63 – Описание параметров устройства `SerialPort_` модуля CPU изделия

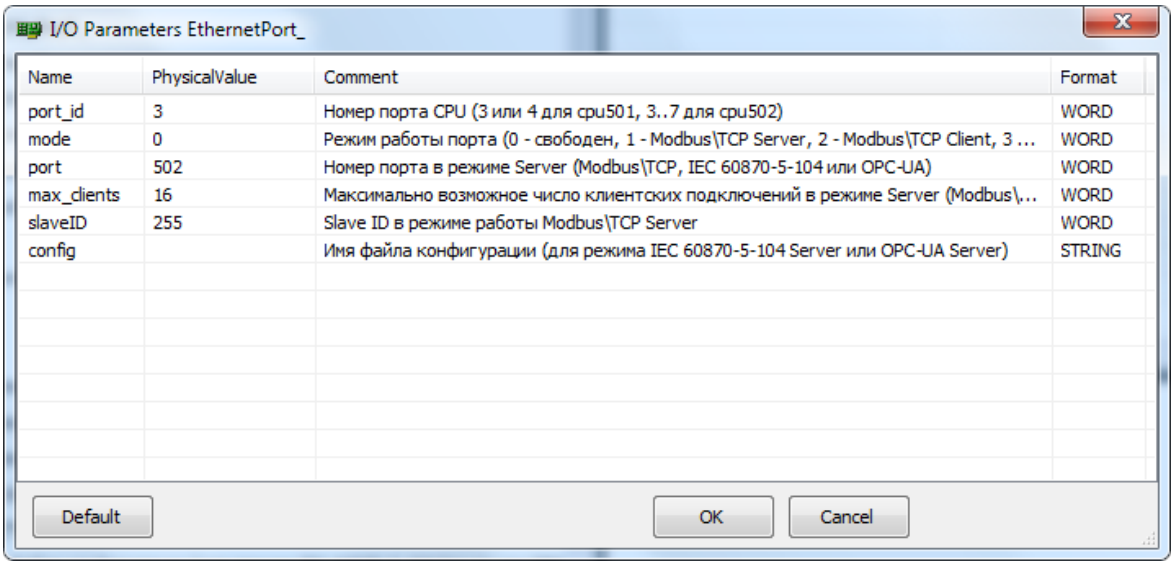


Рис. 64 – Описание параметров устройства `EthernetPort_` модуля CPU изделия

Для работы порта в режиме Modbus в режиме ведущего следует (Табл. 7) в параметрах устройства `SerialPort_` или `EthernetPort_` настроить поле `mode` равным 2.

При работе с устройством `SerialPort_` следует обратить внимание на поля `baudrate`, `bits`, `parity` и `stopbits` параметров устройства и настроить их сообразно настройками ведомых устройств ModbusRTU.

Поля `slaveID`, `slaveResponceDelay_ms` и `slaveResponceTimeout_ms` в параметрах устройства `SerialPort_` для устройства ModbusRTU в режиме ведущего значения не имеют.

Поля `port`, `max_clients`, `slaveID` и `config` в параметрах устройства `EthernetPort_` для устройства Modbus\TCP в режиме ведущего значения не имеют.

Табл. 6 – Расшифровка значений поля `port_id` параметров устройств `SerialPort_` и `EthernetPort_`

Значение port_ID	Тип модуля CPU	Аппаратный порт
1	cpu501 cpu502	Порт 1 интерфейса RS-485
2	cpu501 cpu502	Порт 2 интерфейса RS-485
3	cpu501	Порт ETH1 интерфейса Ethernet
	cpu502	Порт Fiber Optic
4	cpu501	Порт ETH2 интерфейса Ethernet
	cpu502	Порт ETH1 интерфейса Ethernet
5	cpu502	Порт ETH2 интерфейса Ethernet
6	cpu502	Порт ETH3 интерфейса Ethernet
7	cpu502	Порт ETH4 интерфейса Ethernet

Табл. 7 – Расшифровка значений поля `mode` параметров устройств `SerialPort_` и `EthernetPort_`

Значение mode	Тип порта	Режим работы порта
0	SerialPort_ EthernetPort_	Порт свободен
1	SerialPort_	ModbusRTU в режиме ведомый
	EthernetPort_	Modbus\TCP в режиме ведомый
2	SerialPort_	ModbusRTU в режиме ведущий
	EthernetPort_	Modbus\TCP в режиме ведущий
3	SerialPort_	Не поддерживается, порт свободен
	EthernetPort_	IEC 60870-5-104 в режиме сервера

Значение mode	Тип порта	Режим работы порта
4	SerialPort_	Не поддерживается, порт свободен
	EthernetPort_	OPC-UA в режиме сервера

2. Выходное устройство управления mb_ctrl* (Табл. 4), предназначенное для передачи команд в драйвер Modbus. Данный канал имеет 8, 16 или 32 переменных-структуры типа ModbusMasterRequests (массив из 16 переменных типа ModbusMasterRequest, Рис. 65). Через параметры этого простого устройства (Рис. 68) настраивается время цикла опроса драйвером Modbus ведомых устройств.

Name	Data Type	String Size
ModbusMasterRequests		
ModbusMasterRequests	ModbusMasterRequests	
Requests	ModbusMasterReqArr	
Requests[1]	ModbusMasterRequest	
Requests[2]	ModbusMasterRequest	
Requests[15]	ModbusMasterRequest	
Requests[16]	ModbusMasterRequest	

Рис. 65 – Описание структуры ModbusMasterRequests

Name	Data Type	String Size	Comment
ModbusMasterRequest			
ModbusMasterRequest	ModbusMasterRequest		
enable	BOOL		1 - запрос выполняется, 0 - пропускается
single_request	BOOL		1 - запрос выполняется единожды, 0 - запрос выполняется непрерывно
on_modify_request	BOOL		1 - запрос (на запись) выполняется автоматически при изменении данны
do_single_req	BOOL		1 - выполняет одиночный запрос, сбрасывается в 0 по выполнении
command	ModbusMasterCommand		Собственно выполняемая команда Modbus Master

Рис. 66 – Описание структуры ModbusMasterRequest

Структура команды ModbusMasterRequest (Рис. 66) состоит из полей управления enable, single_request, on_modify_request и do_single_req, и собственно команды Modbus, поля command типа ModbusMasterCommand (Рис. 67).

Поле enable служит для включения и отключения запросов в ходе выполнения программы пользователя. Для разрешения работы запроса должно быть установлено в 1. Рекомендуется явно сбрасывать его в 0 для всех неиспользуемых запросов.

Поле single_request определяет режим отправки запроса: 0 – непрерывная отправка каждый цикл, 1 – отправка по установке поля do_single_req в 1. Сброс значения поля do_single_req в 0 должен выполнятся в программе пользователя.

Поле on_modify_request не поддерживается в драйвере Modbus для модуля CPU, но поддерживается модулем RS485 (п. 3.12).

Name	Data Type	String Size	Comment
ModbusMasterCommand			
ModbusMasterCommand	ModbusMasterCommand		
slave_tcp_address	STRING	20	IPv4 адрес запрашиваемого устройства (для Modbus\TCP)
slave_tcp_port	WORD		tcp-порт запрашиваемого устройства (для Modbus\TCP)
slave_id	BYTE		Slave ID запрашиваемого устройства
func_code	BYTE		Код modbus-функции. Поддерживаются функции 1, 2, 3, 4, 5, 6, 15 и 16.
slave_data_addr	WORD		Адрес в запрашиваемом устройстве, с которым будет осуществляться взаимодействие
slave_data_length	WORD		Размер данных в запрашиваем устройстве
offset	WORD		Смещение от начала массива данных, связанных с Modbus Master.
timeout_ms	WORD		Таймаут одного запроса, в мс
delay_before_ms	WORD		Задержка перед выполнением запроса (0 - задержка в 4 символа), в мс
repeats	WORD		Число повторов запроса в случае неудачи
skip_repeats_when_bad	WORD		Число запросов, которое пропускается перед опросом не отвечающего устройства
repeat_over_scan	BOOL		1 - повтор неудачного запроса выполнять через скан, 0 - повтор выполнять сразу

Рис. 67 – Описание структуры `ModbusMasterCommand`

Структура `ModbusMasterCommand` рассчитана на работу как с Modbus RTU и с Modbus\TCP, ненужные в конкретном случае поля игнорируются драйвером Modbus.

Параметр `cycle_time_ms` в параметрах устройства управления служит для настройки полного времени цикла опроса ведомых устройств драйвером Modbus. В случае, если все команды успевают выполниться за указанное время, драйвер будет ждать истечения указанного времени до повторной отправки первого запроса. В случае превышения указанного времени либо в случае установки указанного параметра в 0 первый запрос оправляется повторно сразу после завершения последнего.

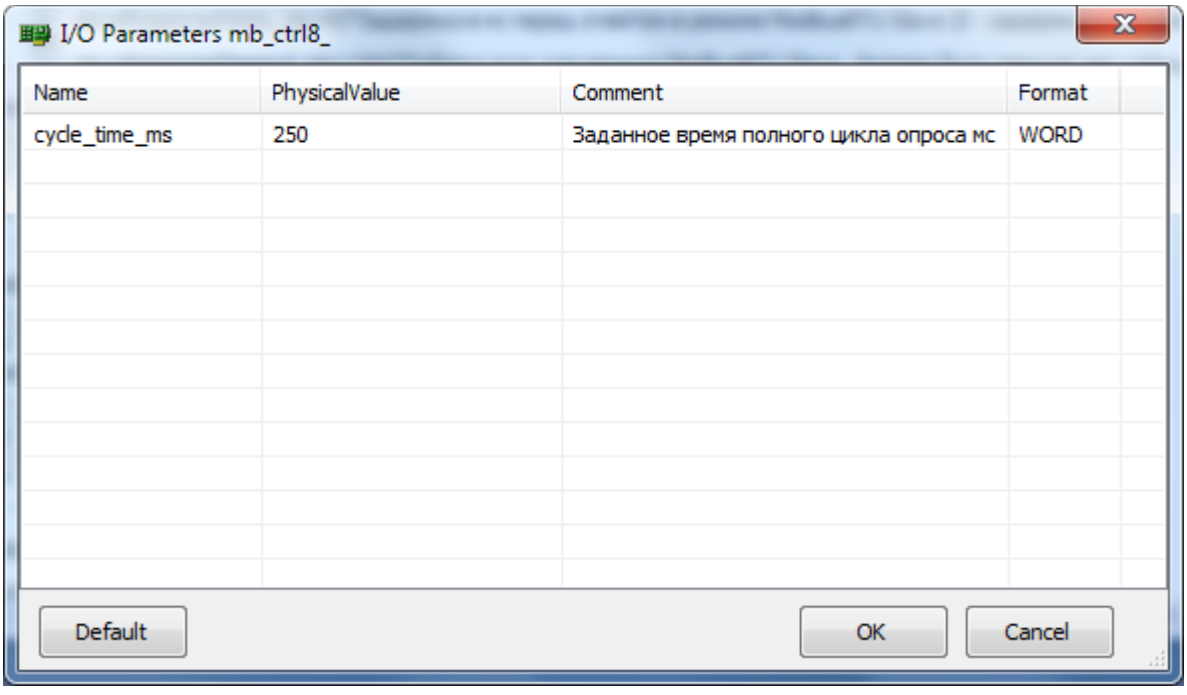


Рис. 68 – Описание параметров устройства `mb_ctrl*` модуля CPU изделия

3. Входное устройство диагностики команд `mb_ctrl_diag*` (Табл. 4), предназначенное для получения статуса выполнения команд из драйвера Modbus. Данный канал имеет 8, 16 или 32 переменных-структуры типа `modbusMasterRequestsAndStats` (массив из 16 переменных типа `ModbusMasterReqStatus`, Рис. 69).

Name	Data Type	String Size
ModbusMasterRequestsAndStats		
modbusMasterRequestsAndStats	modbusMasterRequestsAndStats	
RequestsAndStats	ModbusMasterReqStatArr	
RequestsAndStats[1]	ModbusMasterReqStatus	
RequestsAndStats[2]	ModbusMasterReqStatus	
RequestsAndStats[15]	ModbusMasterReqStatus	
RequestsAndStats[16]	ModbusMasterReqStatus	

Рис. 69 – Описание структуры `modbusMasterRequestsAndStats`

Name	Data Type	String Size	Comment
ModbusMasterReqStatus			
ModbusMasterReqStatus	ModbusMasterReqStatus		
status_request	INT		Обновляющийся код состояния запроса
status_exectime_ms	INT		Обновляющееся последнее время выполнения запроса, в мс
status_repeats	UINT		Обновляющееся число выполненных повторных запросов (при неудаче)
status_skips	UINT		Обновляющееся число пропусков в опросе для не отвечающего устройства
requests_total	DWORD		Общее число выполненых запросов
requests_good	DWORD		Число запросов, выполненых без ошибок

Рис. 70 – Описание структуры `ModbusMasterReqStatus`

Табл. 8 – Расшифровка значений поля `status_request` структуры `MosbusMasterReqStatus`

Значение status_request	Расшифровка
0	Запрос выполнен успешно
1	Неверная функция
2	Неверный адрес данных
3	Неверное значение данных
4	Общий сбой устройства сервера
5	Ведомое устройство приняло запрос и обрабатывает его, но это требует много времени
6	Устройство сервера занято
7	Ведомое устройство не может выполнить программную функцию, заданную в запросе
8	Данный код не поддерживается
9	Не определено стандартом

Значение status_request	Расшифровка
10	Данный код не поддерживается
11	Нет ответа от целевого устройства, возможно оно отсутствует в сети (для Modbus\TCP)
12	Контрольная сумма не совпала
13	Данные ответа не соответствуют запросу
14	Неизвестный код ошибки
15	Код ошибки не соответствует запросу
16	При корректной CRC данных в ответе больше, чем 125 для регистров и 2000 для битов
17	Ответ от ведомого устройства с другим Slave ID
20	Нет ответа по истечении заданного времени (таймаут)
21	Неправильно заполнены поля структуры запроса
22	Ошибка инициализации библиотеки Modbus
23	Не удалось соединиться с Modbus\TCP-сервером
24	Передано/принято данных меньше, чем следовало
25	Адрес/длина принимаемых/передаваемых данных выходят за пределы связанных с устройством массивов регистров/битов
30	Запрос обновляется (передаётся по CAN из CPU в модуль RS485). **Только для устройства rs485new.**
31	Ошибка обновления запроса модуля RS485 (при этой ошибка следует повторить попытку обновления). **Только для устройства rs485new.**
254	Запрос обрабатывается
255	Запрос ещё ни разу не обрабатывался

Структура `ModbusMasterReqStatus` (Рис. 70) содержит в себе диагностические и статистические данные по Modbus-запросу:

- поле `status_request` (Табл. 8) содержит информацию о ходе выполнения запроса;
- поле `status_exectime_ms` содержит информацию о времени выполнения последнего запроса, обновляется после завершения выполнения запроса;
- поле `status_repeats` содержит число повторов запроса при ошибке в ходе выполнения, сбрасывается в 0 при успешном выполнении запроса;
- поле `status_skips` содержит число пропусков запроса при достижении полем `status_repeats` значения, определённого параметром `repeats` в настройках запроса (Рис. 67) , сбрасывается в 0 при успешном выполнении запроса;

– поле `requests_total` содержит общее число запросов, отправленных за всё время работы программы пользователя, сбрасывается в 0 только при рестарте программы пользователя;
– поле `requests_good` содержит число запросов, отправленных и успешно обработанных за всё время работы программы пользователя, сбрасывается в 0 только при рестарте программы пользователя.

4. Выходные устройства регистровых и битных переменных (`mb_regs*` и `mb_bits*`, Табл. 4), предназначенные для хранения отправляемых и получаемых командами Modbus данных. В начале цикла выполнения программы каналы этих устройств содержат в себе обновившиеся в ходе обработки команд Modbus данные. Поэтому для связанных с этими устройствами переменными следует устанавливать атрибут Read/Write (п. 3.1.4), а в начале каждого цикла следует копировать значения переменных в отдельные, не связанные с каналами рабочие переменные. Устройство mb_regs* имеет 4, 16 или 64 переменных-структуры типа `ModbusREGs` (массив из 1024 переменных типа `WORD`, Рис. 71). Устройство mb_bits* имеет 1, 4 или 16 переменных-структуры типа `ModbusDiscrets` (массив из 1024 переменных типа `BOOL`, Рис. 72).

Через параметр `offset` устройства (Рис. 73) передаётся настройка смещения адресного пространства, которая учитывается при формировании команд.

Name	Data Type	String Size	Comment
ModbusREGs			
ModbusREGs	ModbusREGs		
regs	Words1k		Массив регистров Modbus
regs[1]	WORD		
regs[2]	WORD		
regs[3]	WORD		
regs[4]	WORD		
regs[5]	WORD		
regs[6]	WORD		
regs[7]	WORD		
regs[8]	WORD		
regs[9]	WORD		
regs[10]	WORD		
regs[1020]	WORD		
regs[1021]	WORD		
regs[1022]	WORD		
regs[1023]	WORD		
regs[1024]	WORD		
*			

Рис. 71 – Описание структуры `ModbusREGs`

Name	Data Type	String Size	Comment
ModbusDiscrets			
ModbusDiscrets	ModbusDiscrets		
discrets	Bools1k		Массив дискретов modbus
discrets[1]	BOOL		
discrets[2]	BOOL		
discrets[3]	BOOL		
discrets[4]	BOOL		
discrets[5]	BOOL		
discrets[6]	BOOL		
discrets[7]	BOOL		
discrets[8]	BOOL		
discrets[9]	BOOL		
discrets[10]	BOOL		
discrets[1020]	BOOL		
discrets[1021]	BOOL		
discrets[1022]	BOOL		
discrets[1023]	BOOL		
discrets[1024]	BOOL		

Рис. 72 – Описание структуры `ModbusDiscrets`

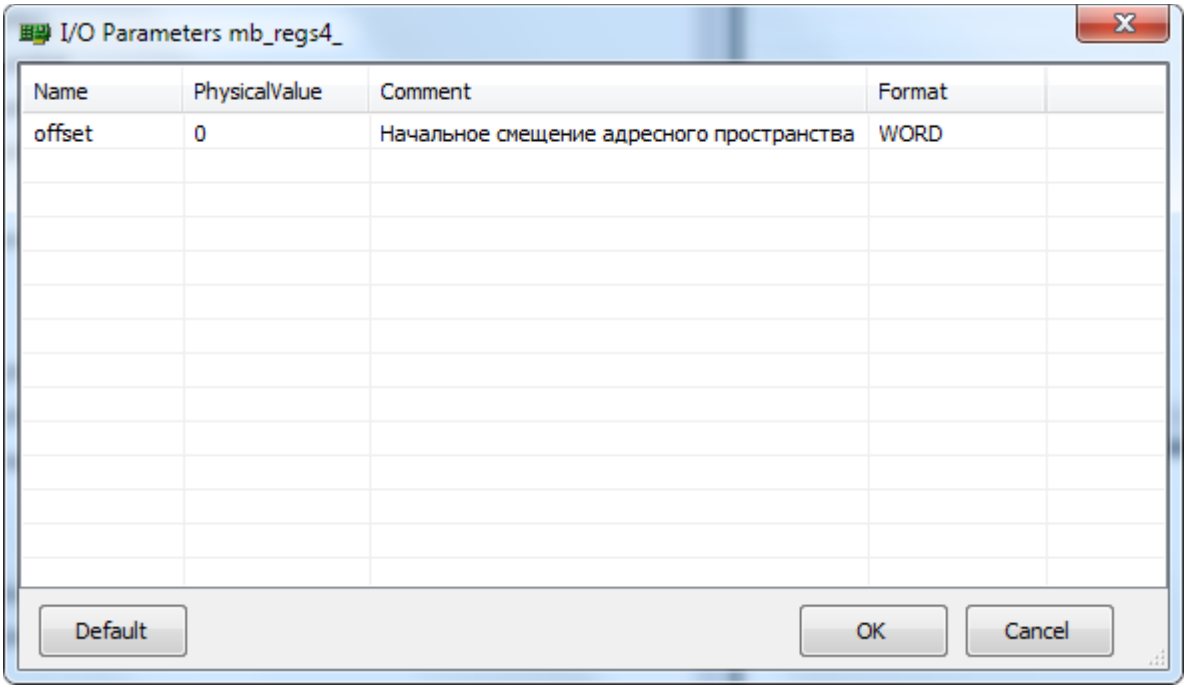


Рис. 73 – Описание параметров устройств `mb_regs*` и `mb_bits*` модуля CPU изделия

Выходное устройство управления `mb_ctrl*` (Табл. 4), предназначенное для передачи команд в драйвер Modbus. Данный канал имеет 8, 16 или 32 переменных-структуры типа `ModbusMasterRequests` (массив из 16 переменных типа `ModbusMasterRequest`, Рис. 65). Через параметры этого простого устройства (Рис. 68) настраивается время цикла опроса драйвером Modbus ведомых устройств.

5. Выходное устройство управления работой драйвера Modbus `mb_control_`. Данный канал имеет одну переменную типа `BYTE`, в которую следует записывать команду управления работой драйвера Modbus. Перечень допустимых значений команд приведён в Табл. 9.

Табл. 9 – Расшифровка значений выхода устройства mb_control_

Значение выхода mb_control_	Имя Defined Word	Расшифровка команды
0	CPUPORT_STOP	Остановить работу драйвера Modbus
1	CPUPORT_START	Запустить драйвер Modbus в работу
2	CPUPORT_RESTART	Перезапустить драйвер Modbus

3.7.2 Реализация поддержки протоколов Modbus RTU и Modbus/TCP (ведомый) в модулях CPU

Для работы со встроенным в модуль CPU протоколом Modbus RTU/TCP в режиме ведомый предназначены специальные составные устройства (Табл. 10).

Составные Modbus-устройства с поддержкой режима «ведомый» включают в себя простые устройства, с помощью которых реализуется привязка к аппаратному порту, адресное пространство, управление и диагностика устройства. Составные Modbus-устройства созданы трёх типов: малое, среднее и большое, отличаясь доступным размером пространства регистров.

Для ускорения синхронизации данных устройств Modbus RTU/TCP при работе с включённым режимом Failover, рекомендуется выбирать составные Modbus-устройства с количеством регистров, максимально соответствующим задаче (то есть для задачи с потребностью в 1000 Holding Register выбирать малое Modbus-устройство, а не среднее или большое).

Табл. 10 – Типы Modbus устройств с режимом ведомый

Тип Modbus-устр.	Имя составного устройства	Имена простых устройств	Число Holding Registers	Число Input Registers	Число Coils	Число Discrete Inputs

Малое	`modbusRTU_sl_4k`
	`modbusTCP_sl_4k`
		`EthernetPort_/`
		`SerialPort_`
		`mb_HR4_`
		`mb_DI1_`
		`mb_IR1_`
		`mb_Coils1_`
		`mb_control_`
			4096	1024	1024	1024

Среднее	`modbusRTU_sl_16k`
	`modbusTCP_sl_16k`
		`EthernetPort_/`
		`SerialPort_`
		`mb_HR16_`
		`mb_DI4_`
		`mb_IR4_`
		`mb_Coils4_`
		`mb_control_`
			16384	4096	4096	4096

Тип Modbus-устр.	Имя составного устройства	Имена простых устройств	Число Holding Registers	Число Input Registers	Число Coils	Число Discrete Inputs
Большое	modbusRTU_sl_64k modbusTCP_sl_64k	EthernetPort_/ SerialPort_ mb_HR64_ mb_DI16_ mb_IR16_ mb_Coils16_ mb_control_	65536	16384	16384	16384

В состав Modbus-устройства с поддержкой режима ведомый входят следующие простые устройства:

1. Входное устройство порт последовательный (SerialPort_) или Ethernet (EthernetPort_), предназначенное для привязки Modbus-устройства к аппаратному порту модуля CPU. Его назначение и параметры полностью аналогичны описанным в п. 3.7.1 для Modbus-устройств с поддержкой режима ведущий.

Для привязки к аппаратному порту модуля CPU следует в параметрах устройства SerialPort_ или EthernetPort_ настроить поле port_id согласно Табл. 6.

Так как поле port_id ссылается на аппаратный порт в составе модуля CPU, следует в ходе ранжирования модулей (п. 3.1.2) помещать Modbus-устройство ниже модулей CPU, в противном случае Modbus-устройство работать не будет.

Для работы порта в режиме Modbus в режиме ведомого следует (Табл. 7) в параметрах устройства SerialPort_ или EthernetPort_ настроить поле mode равным 1.

Значение поля slaveID следует настроить согласно требованиям к реализуемому Modbus-устройству (по умолчанию 255, что неприемлемо для устройств ModbusRTU).

При работе с устройством SerialPort_ следует обратить внимание на поля baudrate, bits, parity и stopbits параметров устройства и настроить их сообразно настройками ведущего устройства ModbusRTU.

Поле slaveResponceDelay_ms в параметрах устройства SerialPort_ для устройства ModbusRTU в режиме работы «ведомый» задаёт длительность паузы в миллисекундах перед ответом на запрос со стороны ведущего (0 – автоматически вычисляемая пауза длительностью в 4 символа).

Поле slaveResponceTimeout_ms в параметрах устройства SerialPort_ для устройства ModbusRTU в режиме работы «ведомый» задаёт величину таймаута на запрос со стороны ведущего. Его величина должна быть меньше значения таймаута на запрос в устройстве ведущего, в противном случае есть вероятность несогласованной работы драйвера Modbus.

Поле port в параметрах устройства EthernetPort_ следует установить согласно настройкам ведущего устройства Modbus\TCP (по умолчанию 502).

Поле max_clients в параметрах устройства EthernetPort_ следует установить равным предполагаемому числу подключаемых устройств Modbus\TCP в режиме ведущий (не более 255, значение 0 трактуется как 1).

Поле config в параметрах устройства EthernetPort_ для устройства Modbus в режиме ведомого значения не имеет.

2. Выходные устройства регистровых и битных переменных для каждого типа данных (`mb_HR*`, `mb_DI*`, `mb_IR*` и `mb_Coils*`, Табл. 10), предназначенных для формирования адресного пространства Modbus. В начале цикла выполнения программы каналы этих устройств содержат в себе обновившиеся в ходе обработки команд Modbus данные. Поэтому для связанных с этими устройствами переменными следует устанавливать атрибут Read/Write (п. 3.1.4), а в начале каждого цикла следует копировать значения переменных в отдельные, не связанные с каналами рабочие переменные.

Устройство `mb_HR*` имеет 4, 16 или 64 переменных-структуры типа `ModbusREGs` (массив из 1024 переменных типа `WORD`, Рис. 71).

Устройство `mb_IR*` имеет 1, 4 или 16 переменных-структуры типа `ModbusREGs` (массив из 1024 переменных типа `WORD`, Рис. 71).

Устройства `mb_DI*` и `mb_Coils*` имеют 1, 4 или 16 переменных-структуры типа `ModbusDiscrets` (массив из 1024 переменных типа `BOOL`, Рис. 72).

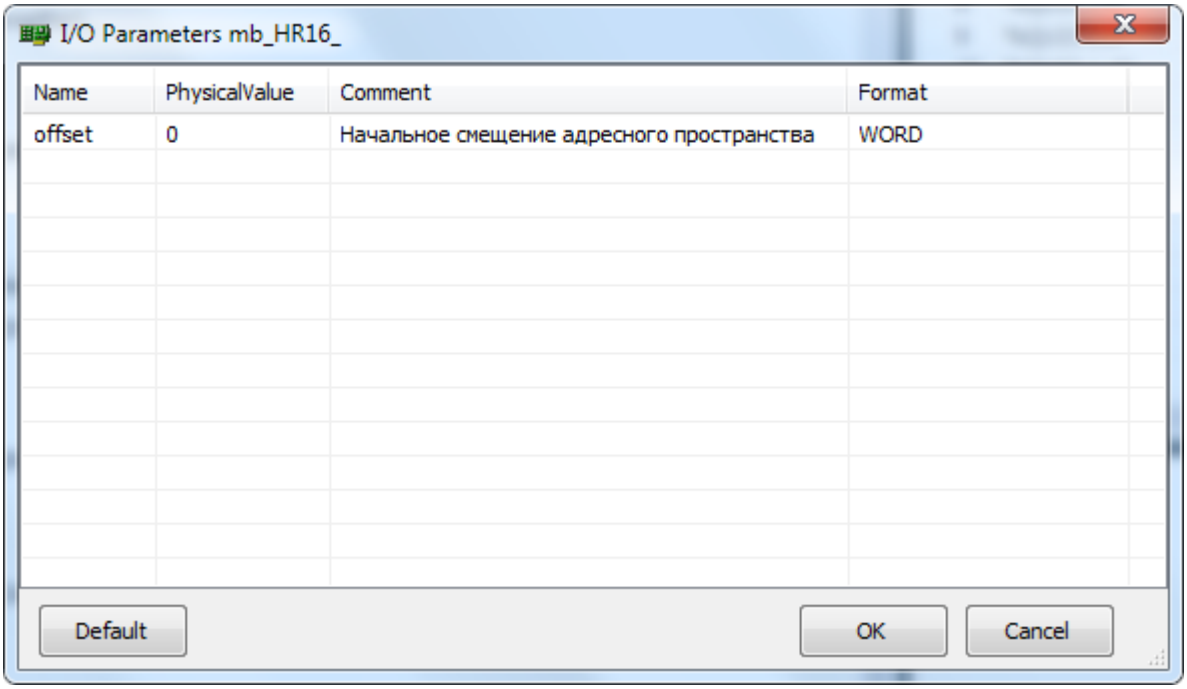


Рис. 74 – Описание параметров устройств mb_HR*, mb_DI*, mb_IR* и mb_Coils* модуля CPU изделия

Через параметр `offset` устройства (Рис. 74) передаётся настройка смещения адресного пространства (отдельная для каждого типа данных), которая учитывается при получении запросов от ведущих устройств.

3. Выходное устройство управления работой драйвера Modbus `mb_control_`. Данный канал имеет одну переменную типа `BYTE`, в которую следует записывать команду управления работой драйвера Modbus. Перечень допустимых значений команд приведён в Табл. 9.

3.7.3 Реализация поддержки протокола IEC 60870-5-104 (сервер) в модулях CPU

Для работы со встроенным в модуль CPU протоколом IEC 60870-5-104 в режиме сервера предназначены следующие простые устройства ввода-вывода:

- простое устройство `EthernetPort_`, с помощью которого реализуется привязка сервера IEC 60870-5-104 к аппаратному порту;
- специальные простые устройства (Табл. 11 и Табл. 12), с помощью которых реализуется адресное пространство IEC 60870-5-104;
- простое устройство `iec104_server_diag_`, с помощью которого реализуется диагностика работы сервера IEC 60870-5-104.

Табл. 11 – iec104*-устройства типа output (выходные данные)

Имя устройства	Типы IEC 60870-5-104		
	Имя	Номер	Назначение
`iec104_SP_out_`	M_SP_NA_1	1	Одноэлементная информация
	M_SP_TB_1	30	Одноэлементная информация с меткой времени
`iec104_DP_out_`	M_DP_NA_1	3	Двухэлементная информация
	M_DP_TB_1	31	Двухэлементная информация с меткой времени
`iec104_MEa_out_`	M_ME_NA_1	9	Значение измеряемой величины, нормализованное значение
	M_ME_TD_1	34	Значение измеряемой величины, нормализованное значение с меткой времени
`iec104_MEb_out_`	M_ME_NB_1	11	Значение измеряемой величины, масштабированное значение
	M_ME_TE_1	35	Значение измеряемой величины, масштабированное значение с меткой времени
`iec104_MEc_out_`	M_ME_NC_1	13	Значение измеряемой величины, короткий формат с плавающей запятой
	M_ME_TF_1	36	Значение измеряемой величины, короткий формат с плавающей запятой с меткой времени
`iec104_IT_out_`	M_IT_TB_1	37	Интегральные суммы с меткой времени

Табл. 12 – iec104*-устройств типа input (команды)

Имя устройства	Типы IEC 60870-5-104		
	Имя	Номер	Назначение
iec104_SP_in_	C_SC_NA_1	45	Одноэлементная команда
iec104_DP_in_	C_DC_NA_1	46	Двухэлементная команда
iec104_MEa_in_	C_SE_NA_1	48	Команда уставки, нормализованное значение
iec104_MEb_in_	C_SE_NB_1	49	Команда уставки, масштабированное значение
iec104_MEc_in_	C_SE_NC_1	50	Команда уставки, короткий формат с плавающей запятой

Для ускорения синхронизации данных устройств IEC 60870-5-104 при работе с включённым режимом Failover, рекомендуется при добавлении устройств `iec104*` указывать реально используемое число каналов вместо предлагаемого по умолчанию значения 512 (Рис. 75).

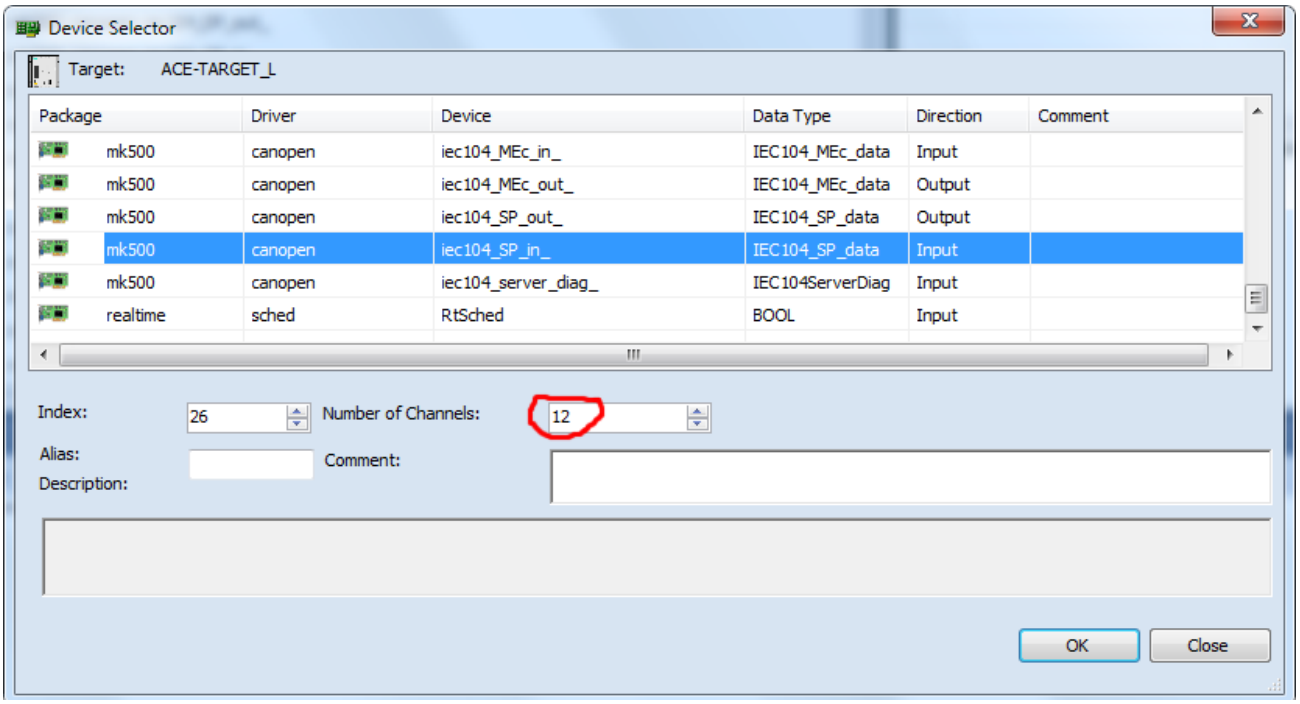


Рис. 75 – Выбор необходимого числа каналов при добавлении устройства iec104*

Для поддержки сервера IEC 60870-5-104 в проект следует добавить следующие простые устройства (желательно в порядке перечисления):

1. Входное устройство порта Ethernet (`EthernetPort_`), предназначенное для привязки сервера IEC 60870-5-104 к аппаратному порту модуля CPU. Его назначение и параметры полностью аналогичны описанным в п. 3.7.1.

Для привязки к аппаратному порту модуля CPU следует в параметрах устройства `EthernetPort_` настроить поле `port_id` согласно Табл. 6.

Так как поле `port_id` ссылается на аппаратный порт в составе модуля CPU, следует в ходе ранжирования модулей (п. 3.1.2) помещать устройство `EthernetPort_` ниже модулей CPU, в противном случае сервер IEC 60870-5-104 работать не будет.

Для работы порта в режиме сервера IEC 60870-5-104 следует (согласно Табл. 7) в параметрах устройства `EthernetPort_` настроить поле `mode` равным 3.

Поле `port` в параметрах устройства `EthernetPort_` следует установить согласно настройкам клиента IEC 60870-5-104. Стандартным значением является 2404 (по умолчанию установлено 502).

Поле `max_clients` в параметрах устройства `EthernetPort_` следует установить равным предполагаемому числу подключаемых клиентов IEC 60870-5-104 (не более 255, значение 0 трактуется как 1).

В поле `config` в параметрах устройства `EthernetPort_` в режиме привязки к серверу IEC 60870-5-104 следует ввести имя файла (с расширением) конфигурации (см. ниже), загруженного в модули CPU проекта.

2. Выходные устройства для реализации пространства iec104*-данных (Табл. 11).

В начале цикла выполнения программы каналы этих устройств содержат в себе обновившиеся в ходе работы сервера IEC 60870-5-104 данные или команды. Поэтому для связанных с этими устройствами переменными следует устанавливать атрибут Read/Write (п. 3.1.4), а в начале каждого цикла следует копировать значения переменных в отдельные, не связанные с каналами рабочие переменные.

Все доступные устройства позволяют работать как с IEC104-типами с меткой времени, так и без неё.

Устройство `iec104_SP_out_` имеет от 1 до 512 переменных-структур типа `IEC104_SP_data` (Рис. 76).

Устройство `iec104_DP_out_` имеет от 1 до 512 переменных-структур типа `IEC104_DP_data` (Рис. 77).

Устройство `iec104_MEa_out_` имеет от 1 до 512 переменных-структур типа `IEC104_MEab_data` (Рис. 78).

Устройство `iec104_MEb_out_` имеет от 1 до 512 переменных-структур типа `IEC104_MEab_data` (Рис. 78).

Устройство `iec104_MEc_out_` имеет от 1 до 512 переменных-структур типа `IEC104_MEc_data` (Рис. 79).

Устройство `iec104_IT_out_` имеет от 1 до 512 переменных-структур типа `IEC104_IT_data` (Рис. 80).

Параметры всех устройств идентичны (Рис. 81).

Параметр `port_id` каждого устройства должен совпадать с параметром `port_id` устройства `EthernetPort_`, к экземпляру сервера IEC 60870-5-104 которого относится данный диапазон данных.

В параметре `ioa` передаётся настройка смещения адресного пространства (отдельная для каждого устройства), которая учитывается при обмене данными с сервером IEC 60870-5-104. Адресные пространства не должны пересекаться, значения данных в каналах с совпадающими адресами в ходе работы не определены.

Name	Data Type	String Size	Comment
IEC104_SP_Data			
IEC104_SP_data	IEC104_SP_data		
value	BOOL		Значение
BL	BOOL		Флаг блокировки
SB	BOOL		Флаг замещения
NT	BOOL		Флаг актуальности значения
IV	BOOL		Флаг действительности значения

Рис. 76 – Описание структуры IEC104_SP_Data

Name	Data Type	String Size	Comment
IEC104_DP_Data			
IEC104_DP_data	IEC104_DP_data		
value	SINT		Значение
BL	BOOL		Флаг блокировки
SB	BOOL		Флаг замещения
NT	BOOL		Флаг актуальности значения
IV	BOOL		Флаг действительности значения

Рис. 77 – Описание структуры IEC104_DP_Data

Name	Data Type	String Size	Comment
IEC104_MEab_Data			
IEC104_MEab_data	IEC104_MEab_data		
value	INT		Значение
OV	BOOL		Флаг переполнения
BL	BOOL		Флаг блокировки
SB	BOOL		Флаг замещения
NT	BOOL		Флаг актуальности значения
IV	BOOL		Флаг действительности значения

Рис. 78 – Описание структуры IEC104_MEab_Data

Name	Data Type	String Size	Comment
IEC104_MEc_Data			
IEC104_MEc_data	IEC104_MEc_data		
value	REAL		Значение
OV	BOOL		Флаг переполнения
BL	BOOL		Флаг блокировки
SB	BOOL		Флаг замещения
NT	BOOL		Флаг актуальности значения
IV	BOOL		Флаг действительности значения

Рис. 79 – Описание структуры IEC104_MEc_Data

Name	Data Type	String Size	Comment
IEC104_IT_Data			
IEC104_IT_data	IEC104_IT_data		
value	DINT		Значение
SQ	BYTE		Порядковый номер
CY	BOOL		Флаг переполнения
CA	BOOL		Флаг инициализации
IV	BOOL		Флаг действительности значения

Рис. 80 – Описание структуры IEC104_IT_Data

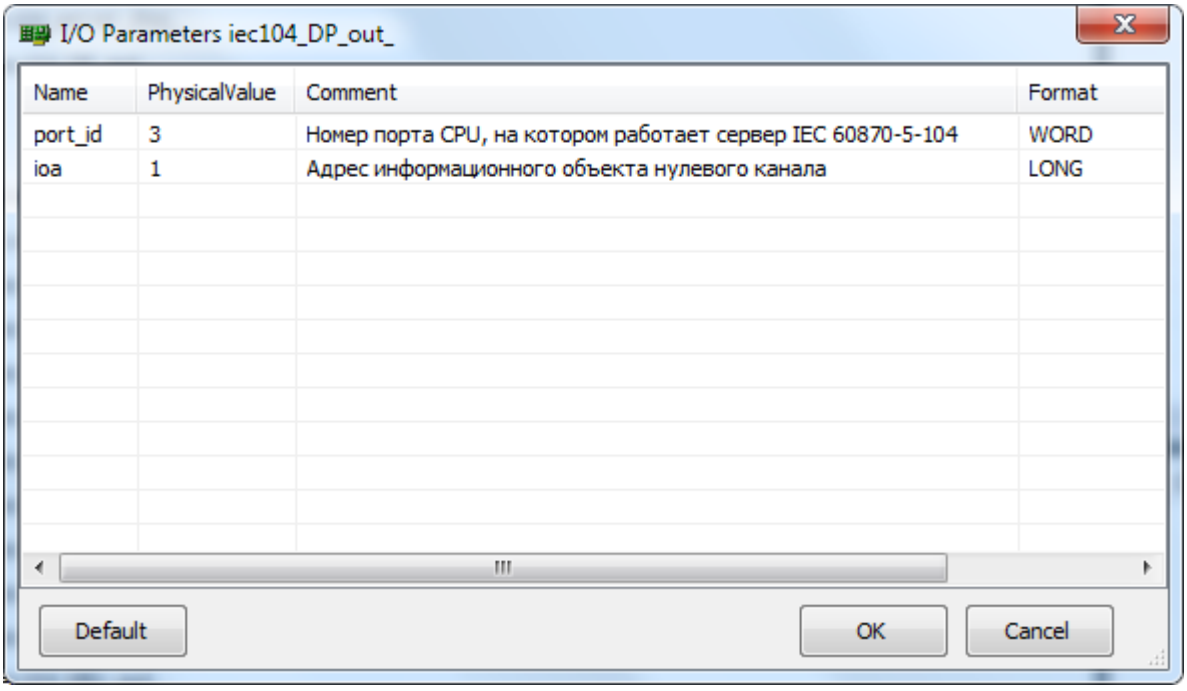


Рис. 81 – Описание параметров устройств iec104_* на примере устройства iec104_DP_out_

3. Входные устройства для реализации пространства iec104*-команд (Табл. 12).

Устройство `iec104_SP_in_` имеет от 1 до 512 переменных-структур типа `IEC104_SP_data` (Рис. 76).

Устройство `iec104_DP_in_` имеет от 1 до 512 переменных-структур типа `IEC104_DP_data` (Рис. 77).

Устройство `iec104_MEa_in_` имеет от 1 до 512 переменных-структур типа `IEC104_MEab_data` (Рис. 78).

Устройство `iec104_MEb_in_` имеет от 1 до 512 переменных-структур типа `IEC104_MEab_data` (Рис. 78).

Устройство `iec104_MEc_in_` имеет от 1 до 512 переменных-структур типа `IEC104_MEc_data` (Рис. 79).

Параметры всех устройств идентичны (Рис. 81).

Параметр `port_id` каждого устройства должен совпадать с параметром `port_id` устройства `EthernetPort_`, к экземпляру сервера IEC 60870-5-104 которого относится данный диапазон данных.

В параметре `ioa` передаётся настройка смещения адресного пространства (отдельная для каждого устройства), которая учитывается при обмене данными с сервером IEC 60870-5-104. Адресные пространства не должны пересекаться, значения данных в каналах с совпадающими адресами в ходе работы не определены.

4. Входное устройство диагностики работы сервера IEC 60870-5-104, `iec104_server_diag_`, Данный канал имеет одну переменную типа `IEC104ServerDiag`.

Структура `IEC104ServerDiag` (Рис. 82) содержит в себе диагностические данные сервера IEC 60870-5-104:

– в поле `clients` передаётся число подключенных клиентов IEC 60870-5-104;

– поле `server_running` показывает, запущен сервер IEC 60870-5-104 или нет;

– поле `server_error_code` содержит код ошибки работы сервера IEC 60870-5-104 (Табл. 13).

Name	Data Type	String Size	Comment
IEC104ServerDiag			
IEC104ServerDiag	IEC104ServerDiag		
clients	INT		Число подключившихся клиентов
server_running	BOOL		False если сервер не запущен, True если сервер запущен
server_error_code	INT		0 - нет ошибок, иначе код ошибки

Рис. 82 – Описание структуры IEC104ServerDiag

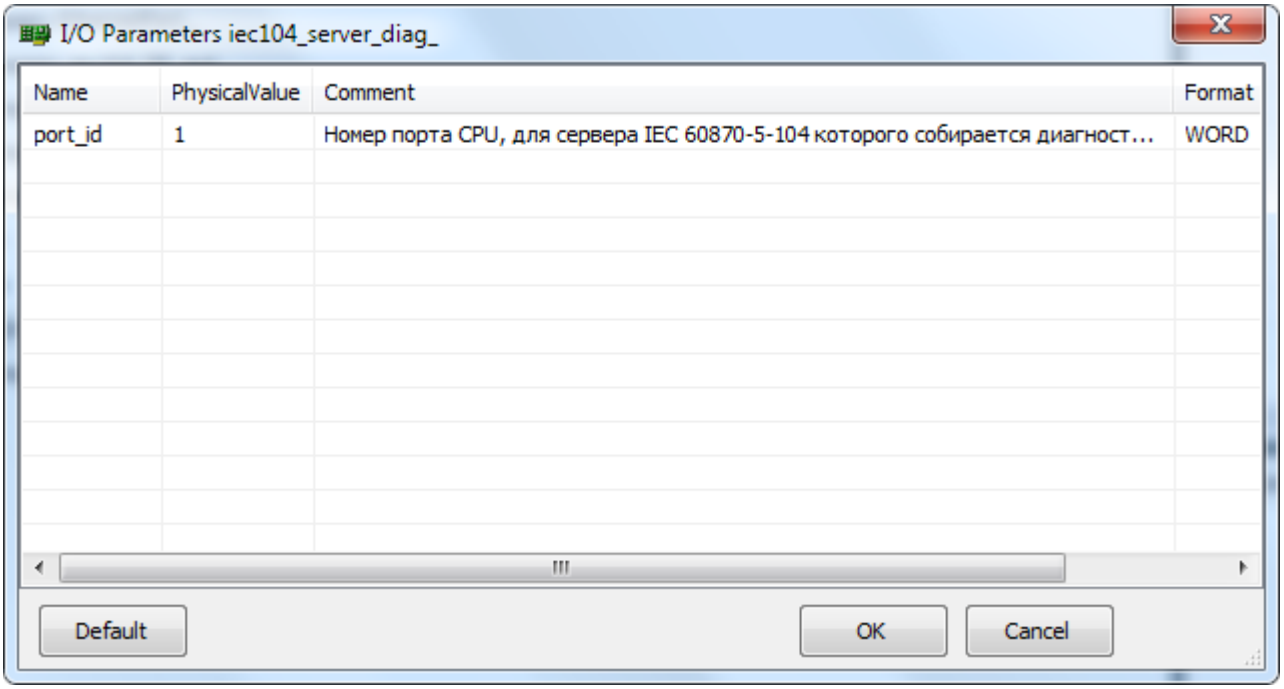


Рис. 83 – Описание параметров устройства iec104_server_diag_ модуля CPU

Табл. 13 – Расшифровка кодов ошибки server_error_code устройства iec104_server_diag_

Значение server_error_code	Расшифровка
0	Сервер работает без ошибок
1	Ошибка инициализации сервера
2	Неизвестный тип IEC 60870-5-104
3	Недопустимая комбинация тип IEC 60870-5-104 – IOA
4	Выполняется подключение клиента
5	Сервер остановлен
6	Не найден файл конфигурации
7	Неверный файл конфигурации
8	Определено более одного сервера

Параметр `port_id` устройства (Рис. 83) должен совпадать с параметром `port_id` устройства `EthernetPort_`, с экземпляра сервера IEC 60870-5-104 которого собирается диагностическая информация.

Также для работы сервера IEC 60870-5-104 требуется сформировать и передать во все модули CPU проекта по протоколу ftp (п. 2.3.1) конфигурационный файл. Отсутствие конфигурационного файла, ошибка в указании его имени в параметрах устройства `EthernetPort_`, либо ошибки в формате конфигурационного файла приводят к невозможности старта сервера IEC 60870-5-104.

Конфигурационный файл быть составлен в следующем формате:

1. Секция конфигурации сервера открывается заголовком [IEC104]. Далее следуют пары ключ=значение. Порядок следования ключей не критичен, между знаком равенства и ключом/значением допускается наличие пробела. Имена ключей и заголовков критичны к регистру. Параметры ASDUADR, W, K, T0-T3 должны совпадать с параметрами мастера.
   Список ключей:
   1.1 TSporadic - временной период посылки спорадических переменных мастеру, в миллисекундах;
   1.2 ASDUADR - общий адрес ASDU для устройства;
   1.3 W - Последнее подтверждение после приема W APDU;
   1.4 K - максимальная разность между переменной состояния передачи и номером последнего подтвержденного APDU;
   1.5 T0 - Тайм-аут при установлении соединения, в секундах;
   1.6 T1 - Тайм-аут при отправке или тестировании APDU, в секундах;
   1.7 T2 - Тайм-аут для подтверждения в случае отсутствия сообщения с данными (меньше T1) , в секундах;
   1.8 T3 - Тайм-аут для отправки блоков тестирования в случае долгого простоя, в секундах (0 – не посылать блоки тестирования);
   1.9 buffsize - Размер буферов для каждого соединения в МБ (от 1 до 8, по умолчанию 1 МБ);
   1.10 collect - Сохранять ли данные в буфере в случае потери соединения (1 - сохранять, 0 - не сохранять, по умолчанию 0);
   1.11 ip - сетевой адрес, с которого принимаются подключения к серверу, может быть несколько строк. Пример:
   ```
   ip=10.155.26.220
   ip=10.155.26.138
   ```
   По умолчанию, архивирование данных в буфер в оффлайн-режиме отключено. Для управления архивацией следует в конце IP адреса дописать (true) - включения для архивирования, (false) - для отключения архивирования. По умолчанию применяется (false). Пример:
   ```
   ip=10.155.26.220(true)
   ip=10.155.26.138(false)
   ```
   Добавление строки ip=0.0.0.0 позволяет принимать указанное в параметре Connections (см. ниже) число подключений с прочих IP-адресов.
   1.12 Connections - Максимально доступное количество подключений с адресов, не перечисленных в ключах ip=xxx.xxx.xxx.xxx, 0 – не ограниченное число подключений. По умолчанию принимает значение 0.

2. Секция конфигурации переменных сервера открывается заголовком [variables]. Переменные одного типа заключаются во вложенный тег, внутри которого перечисляются отдельные переменные данного типа. Имя тега соответствует типу переменной согласно стандарту IEC 60870-5-104.

Пример:

```
<M_SP_NA_1>
...
Переменные
...
</M_SP_NA_1>
```

Каждая переменная начинается ключевым словом point, после которого через пробел следует указать следующие параметры:

2.1 `ioa` – уникальный адрес объекта информации, обязательно должен присутствовать;

2.2 `sporadic` - является ли объект информации спорадически передаваемым (1 – является, 0 – не является, по умолчанию 0);

2.3 `cycle` - передавать ли данные объекта информации с каждым циклом (1 – передавать, 0 – не передавать, по умолчанию 0).

Пример:

```
point ioa=1,sporadic=0,cycle=1
point ioa=1000,sporadic=1,cycle=0
```

3. Допускается наличие однострочных комментариев. Строка, начинающаяся с символа «#», игнорируется при разборе конфигурационного файла.

Пример:

```
# Параметры насоса
```

Пример содержимого конфигурационного файла приведён в Листинге 1.

```
[IEC104]
# секция настроек
ASDUADR=1000
W=8
K=10
T0=0
T1=10
T2=20
T3=30
Connections = 2
buffsize = 4
ip=0.0.0.0
ip=10.155.26.220
ip=10.155.26.138(true)
[variables]
# начало секции переменных
<M_ME_TF_1>
point ioa=1,sporadic=0,cycle=1
point ioa=2,sporadic=0,cycle=0
</M_ME_TF_1>
<C_SC_NA_1>
point ioa=3
point ioa=4
</C_SC_NA_1>
```

Листинг 1 – Пример файла конфигурации сервера IEC 60870-5-104

3.7.4 Реализация поддержки протокола OPC-UA (сервер) в модулях CPU

⚠ ВНИМАНИЕ

На 01.06.2018 описание раздела не закончено, содержание носит ознакомительный характер.

Для работы со встроенным в модуль CPU протоколом OPC-UA в режиме сервера создано специальное составное устройство `opcua_server`.

В состав устройства `opcua_server` входят следующие простые устройства:

1. Входное устройство порт Ethernet (`EthernetPort_`), предназначенное для привязки OPC-UA сервера к аппаратному порту модуля CPU. Его назначение и параметры полностью аналогичны описанным в п. 3.7.1 для Modbus-устройств с поддержкой режима ведущий.

Для привязки к аппаратному порту модуля CPU следует в параметрах устройства `EthernetPort_` настроить поле `port_id` согласно Табл. 6.

Так как поле `port_id` ссылается на аппаратный порт в составе модуля CPU, следует в ходе ранжирования модулей (п. 3.1.2) помещать OPC-UA-устройство ниже модулей CPU, в противном случае OPC_UA-устройство работать не будет.

Для работы порта в режиме сервера OPC-UA следует (согласно Табл. 7) в параметрах устройства `EthernetPort_` настроить поле `mode` равным 4.

Поле `port` в параметрах устройства `EthernetPort_` следует установить согласно настройкам клиента OPC-UA. Стандартным значением является 4840 (по умолчанию установлено 502)

Поле `max_clients` в параметрах устройства `EthernetPort_` следует установить равным предполагаемому числу подключаемых клиентов OPC-UA (не более 255, значение 0 трактуется как 1).

Поле `slaveID` в параметрах устройства `EthernetPort_` для устройства OPC-UA значения не имеет.

В поле `config` в параметрах устройства `EthernetPort_` в режиме привязки к серверу OPC-UA следует ввести имя файла (с расширением) конфигурации (см. ниже), загруженного в модули CPU проекта.

2. Входное устройство диагностики работы сервера OPC-UA, `opcua_server_diag_`, Данный канал имеет одну переменную типа `OPCUAServerDiag`.

Структура `OPCUAServerDiag` (Рис. 84) содержит в себе диагностические данные сервера OPC-UA:

- в поле `clients` передаётся число подключенных клиентов OPC-UA;
- поле `server_running` показывает, запущен сервер OPC-UA или нет;
- поле `server_error_code` содержит код ошибки работы сервера OPC-UA (Табл. 14).

Рис. 84 – Описание структуры `OPCUAServerDiag`

Табл. 14 – Расшифровка кодов ошибки server_error_code устройства opcua_server_diag_

Значение server_error_code	Расшифровка
0	Сервер работает без ошибок
1	Ошибка инициализации сервера

3. Выходное устройство управления работой драйвера OPC-UA `opcua_sever_ctrl_`. Данный канал имеет одну переменную типа `BYTE`, в которую следует записывать команду управления работой драйвера OPC-UA. Перечень допустимых значений команд приведён в Табл. 15.

Табл. 15 – Расшифровка значений выхода устройства opcua_sever_ctrl_

Значение выхода opcua_sever_ctrl_	Имя Defined Word	Расшифровка команды
0	CPUPORT_STOP	Остановить работу драйвера OPC-UA
1	CPUPORT_START	Запустить драйвер OPC-UA в работу
2	CPUPORT_RESTART	Перезапустить драйвер OPC-UA

Также для работы сервера OPC-UA требуется сформировать и передать во все модули CPU проекта по протоколу ftp (п. 2.3.1) конфигурационный файл. Отсутствие конфигурационного файла, ошибка в указании его имени в параметрах устройства `EthernetPort_`, либо ошибки в формате конфигурационного файла приводят к невозможности старта сервера OPC-UA.

Конфигурационный файл быть составлен в следующем формате:

1. Секция конфигурации сервера открывается заголовком [`OpcUaServer`]. Далее следуют пары `ключ=значение`. Порядок следования ключей не критичен, между знаком равенства и ключом/значением допускается наличие пробела. Имена ключей и заголовков критичны к регистру.
   Список ключей:
   1.1 `EndpointUrl` - IP адрес и порт для подключения клиента OPC в формате …
   Пример: `opc.tcp://10.155.26.180:4842`;
   1.2 `EndpointUrl` - IP;

2. Секция конфигурации переменных открывается заголовком `[Variables]`. Далее следует перечисление имён переменных в формате …
3. Допускается наличие однострочных комментариев. Строка, начинающаяся с символа «#», игнорируется при разборе конфигурационного файла.
   Пример:
   `# Параметры сервера`
   Пример содержимого конфигурационного файла приведён в Листинге 2.

```
[OpcUaServer]
# секция настроек
EndpointUrl=opc.tcp://10.155.26.180:4842

[variables]
# начало секции переменных
AI_number
AI_data1
```

Листинг 2 – Пример файла конфигурации сервера OPC-UA

3.7.5 Реализация поддержки протокола Powerlink (ведущий) в модулях МК-502-142 и МК-503-120

Для работы со встроенным в модуль CPU протоколом Powerlink в режиме ведущий предназначено специальное составное устройство `PowerlinkMN`.

Составное устройство с поддержкой режима «ведущий» включает в себя простое устройство `EthernetPort_`, с помощью которого реализуется привязка к аппаратному порту и его диагностика (Рис. 85).

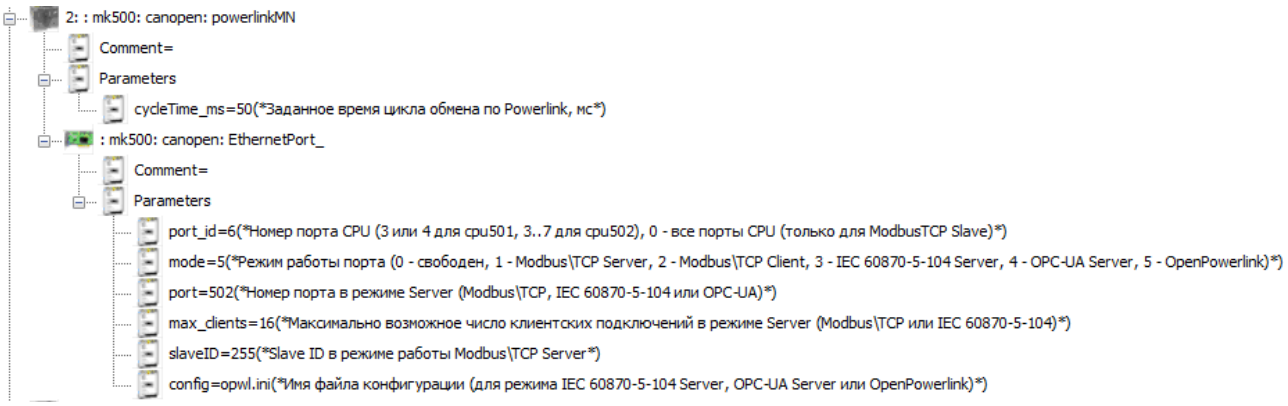


Рис. 85 – Описание структуры PowerlinkMN

Входное устройство порт Ethernet (`EthernetPort_`), предназначенное для привязки Powerlink к аппаратному порту модуля CPU. Его назначение и параметры полностью аналогичны описанным в п. 3.7.1 для Modbus-устройств с поддержкой режима ведущий.

Для привязки к аппаратному порту модуля CPU следует в параметрах устройства `EthernetPort_` настроить поле `port_id` согласно Табл. 6.

Так как поле `port_id` ссылается на аппаратный порт в составе модуля CPU, следует в ходе ранжирования модулей (п. 3.1.2) помещать Powerlink-устройство ниже модулей CPU, в противном случае Powerlink-устройство работать не будет.

Для работы порта в режиме Powerlink следует (согласно Табл. 7) в параметрах устройства `EthernetPort_` настроить поле `mode` равным 5.

Поля `port`, `max_clients` и `slaveID` в параметрах устройства `EthernetPort_` для устройства Powerlink значения не имеет.

В поле config в параметрах устройства `EthernetPort_` в режиме привязки к серверу Powerlink следует ввести имя файла (с расширением) конфигурации, загруженного в модули CPU проекта.

Для работы Powerlink требуется сформировать и передать во все модули CPU проекта по протоколу ftp (п. 2.3.1) конфигурационный файл. Отсутствие конфигурационного файла, ошибка в указании его имени в параметрах устройства `EthernetPort_`, либо ошибки в формате конфигурационного файла приводят к невозможности старта обмена по сети Powerlink.

Конфигурационный файл быть составлен в следующем формате:

1. Секция конфигурации сервера открывается заголовком `[Powerlink]`. Далее следуют пары `ключ=значение`, где ключ - адрес COD (Can Object Dictionary - словарь объектов CAN) в формате `<index>_<subindex>`, значение - текстовое представление значения элемента словаря. `<index>` и `<subindex>` ожидаются в шестнадцатеричном формате (как в документации). Значение можно передавать в десятичном или в шестнадцатеричном (с префиксом 0x) представлении.

В случае ошибки (отсутствует элемент словаря, нет прав на запись, значение вне допустимых пределов) параметр игнорируется.

Все значения секции относятся к MN и применяются в PLCPowerlinkMN. Порядок следования ключей не критичен, между знаком равенства и ключом/значением допускается наличие пробела. Имена ключей и заголовков критичны к регистру.

Поддерживаемые COD представлены ниже (Табл. 16).

Табл. 16 – COD

Индекс (hex)	Субиндекс (hex)	Назначение	Минимальное	Индекс (hex)	Субиндекс (hex)
1006	00	Время цикла Powerlink, в мкс	10000	4294967295	50000
1C00	03	Число ошибок CRC в пакетах MN, приводящих к ошибке «CRC Error» (1 ошибка +8 к счётчику, 1 цикл без ошибки -1 к счётчику)	8	1000	15
1C02	03	Число выходов за пределы времени цикла MN, приводящих к ошибке «Cycle time exceeded» (1 ошибка +8 к счётчику, 1 цикл без ошибки -1 к счётчику)	8	1000	15
1C09	01..28	Число потерь CN-ом пакетов Pres, приводящих к ошибке «Loss of PollResponse» (1 ошибка +8 к счётчику, 1 цикл без ошибки -1 к счётчику) **Пока не реализовано**	8	1000	15
1C0B	03	Число потерь CN-ом пакетов SoC, приводящих к ошибке «Loss of Soc» (1 ошибка +8 к счётчику, 1 цикл без ошибки -1 к счётчику)	8	1000	15
1C0D	03	Число потерей CN-ом пакетов Preq, приводящих к ошибке «Loss of Preq» (1 ошибка +8 к счётчику, 1 цикл без ошибки -1 к счётчику)	8	1000	15

Индекс (hex)	Субиндекс (hex)	Назначение	Минимальное	Индекс (hex)	Субиндекс (hex)
1C17	01..28	Число потерь MN-ом пакетов StatusResponse для соответствующего CN, приводящих к ошибке «Lost StatusResponse» (1 ошибка +8 к счётчику, 1 цикл без ошибки -1 к счётчику) **Пока не реализовано**	8	1000	15
1C14	00	Гистерезис для 0x1C0B_03, в нс	1000	4294967295	100000
1F89	01	Время, в течение которого MN остаётся в NMT_MS_NOT_ACTIVE и ждёт кадры POWERLINK до того, как перейдёт в NMT_MS_PRE_OPERATIONAL_1, в мкс	250	10000000	1000000
1F89	02	Таймаут, в течение которого MN ждёт в состоянии NMT_MS_PRE_OPERATIONAL_1 прохождения IdentRequest/IdentResponse всеми обязательными CN, прежде чем перейдёт в ошибку, в мкс 0 – нет таймаута	0	5000000	500000
1F89	03	Время, в течение которого MN ждёт в состоянии NMT_MS_PRE_OPERATIONAL_1, в мкс 0 – MN выходит из NMT_MS_PRE_OPERATIONAL_1 как только все обязательные CN будут идентифицированы	0	5000000	500000

Индекс (hex)	Субиндекс (hex)	Назначение	Минимальное	Индекс (hex)	Субиндекс (hex)
1F89	04	Сколько мкс MN ждёт в MS_PRE_OPERATIONAL_2 пока все обязательные CN не перейдут в CS_READY_TO_OPERATE. Если все CN опциональные - сколько мкс MN ждёт после отправки в CN команды NMTEnableReadyToOperate до перехода CN в BOOT_STEP2	0	4294967295	500000
1F89	05	Сколько мкс MN ждёт в MS_READY_TO_OPERATE пока все обязательные CN не перейдут в CS_OPERATIONAL. 0 - MN будет ждать неопределённо долго	0	5000000	500000
1F89	0A	Приоритет RMN, согласно которого выполняется захват шины при отказе AMN (1 - высший приоритет, 10 - низший)	1	10	1 для 241, 2 для 242
1F89	0B	Коэффициент для голосования RMN	0	4294967295	10
1F89	0C	Коэффициент для голосования RMN	0	4294967295	10

2. Секция кратности открывается заголовком `[TPDO_Mul]` либо `[RPDO_Mul]`. Секция `TPDO_Mul` содержит список модулей, для которых данные, получаемые ими из MN, следует передавать в несколько приёмов.

Секция `RPDO_Mul` содержит список модулей, для которых данные, передаваемые ими в MN, следует передавать в несколько приёмов.

Каждая из секций содержит в себе пары вида "`адрес=кратность`", где адрес - полный индекс модуля ввода-вывода за CN в формате `<opwlIndex_canIndex>`, кратность - значение кратности передачи данных модуля (допустимые значения 2, 4, 8, 16) в соответствующем направлении. Модули с кратностью 1 описывать не нужно, это значение кратности по умолчанию.

Значения `opwlIndex` и `canIndex` приводятся в десятичном формате.

В ходе работы сети Powerlink между MN и каждой CN происходит обмен одной PDO за скан: на 1 TDPO от MN следует ответ 1 RPDO от CN.

Максимальный размер TPDO/RPDO составляет 1490 байт. В эти данные должна поместиться оперативная информация о модулях и данные.

Оперативная информация занимает n+1 байт, где n - число модулей ввода-вывода (включая МК-545-010) на CAN-шине удалённой стойки. То есть для стойки с 20 модулями остаётся свободным под данные 1469 байт в TPDO и 1469 байт в RPDO.

В таблице ниже приводится информация о числе байт в TPDO и RPDO для всех типов модулей ввода-вывода (Табл. 17).

Табл. 17 – Число байт в TPDO и RPDO для всех типов модулей ввода-вывода

Модель	Официальное наименование	Описание	Байт в TPDO	Байт в RPDO
МК-550-024	Модуль питания МК-550-024	Модуль питания напряжения постоянного тока с интерфейсом CAN	0	8
МК-545-010	Коммуникационный модуль МК-545-010	Коммуникационный модуль Powerlink CN с 2 портами	0	8
МК-521-032	Модуль дискретного ввода МК-521-032	Модуль дискретного ввода напряжения постоянного тока с 32 дискретными входами	8	32
МК-531-032	Модуль дискретного вывода МК-531-032	Модуль дискретного вывода напряжения постоянного тока с 32 дискретными выходами	16	0
МК-513-016	Модуль аналогового ввода МК-513-016	Модуль аналогового ввода с 16 аналоговыми входами 0 -20 (4 - 20) мА	0	32
МК-516-008 А	Модуль аналогового ввода МК-516-008 А	Модуль аналогового ввода, 8 аналоговых входов 0 -20 (4 - 20) мА, изолированных друг от друга; исполнение с быстросъёмным разъёмом 40 контактов с пружинными клеммами типа PUSH-IN	0	16
МК-514-008	Модули аналогового вывода МК-514-008	Модуль аналогового вывода с 8 аналоговыми выходами 0 -20 (4 - 20) мА, с 20-контактным разъемом PUSH-IN	16	8
МК-514-008 А	Модули аналогового вывода МК-514-008 А	Модуль аналогового вывода с 8 аналоговыми выходами 0 -20 (4 - 20) мА, исполнение с быстросъёмным разъёмом 40 контактов с пружинными клеммами типа PUSH-IN	16	8
МК-576-008 А	Модуль аналоговых входов МК-576-008 А	Модуль аналогового ввода с 8 изолированными аналоговыми входами 0(4)-20 мА с поддержкой HART; исполнение с быстросъёмным разъёмом 40 контактов с пружинными клеммами типа PUSH-IN	0	176

Модель	Официальное наименование	Описание	Байт в TPDO	Байт в RPDO
МК-574-008 А	Модуль аналоговых выходов МК-574-008 А	Модуль аналогового вывода с 8 аналоговыми выходами 0(4) - 20 мА с поддержкой HART; исполнение с быстросъёмным разъёмом 40 контактов с пружинными клеммами типа PUSH-IN	16	168
МК-541-002	Коммуникационный модуль МК-541-002	Коммуникационный модуль с 2 интерфейсами RS-485	2048	2048

Как видно, модуль МК-541-002 не помещается в кадр PDO, поэтому вводится понятие кратности: для каждого модуля можно выбрать кратность, с которой передаются его данные в определённом направлении. Например, для модуля МК-541-002 кадрирование 4 на TPDO и 8 на RPDO означает, что данные TPDO будут передаваться в модуль в течение 4 циклов обмена Powerlink, занимая при этом 512 байт, а данные RPDO будут приниматься из модуля в течение 8 циклов обмена Powerlink, занимая при этом 256 байт.

Рекомендуется указывать равные размеры кратности для одного модуля в обоих направлениях.

Пример содержимого конфигурационного файла приведён в Листинге 3.

```
[Powerlink]
1006_00=25000

[RPDO_Mul]
1_4=2
2_5=4

[TPDO_Mul]
2_4=2
2_5=4
```

Листинг 3 – Пример файла конфигурации Powerlink

3.7.6 Реализация поддержки протокола Powerlink (ведомый) в модулях cn545

Для работы с протоколом Powerlink в режиме ведомый предназначено специальное составное устройство `cn545` (Рис. 86). Только по наличию устройства `cn545` в устройствах проекта определяются корректные pwl_id.

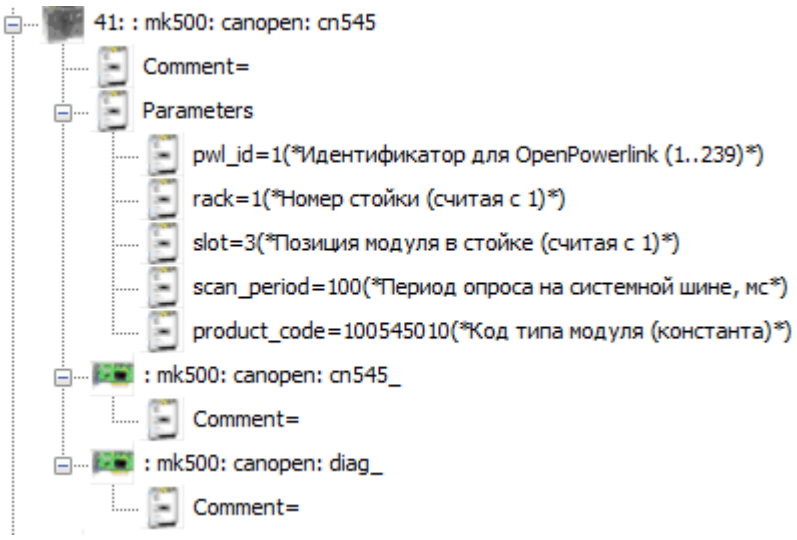


Рис. 86 – Описание структуры cn545

Устройство `cn545` имеет в своем составе простое входное устройство `cn545_` предназначенное для диагностики работы портов. Данное простое входное устройство имеет два канала с переменной-структурой типа PowerlinkDiag (Рис. 87).

PowerlinkDiag	PowerlinkDiag		
Linkup	BOOL		Наличие соединения на порту (TRUE - есть соединение)
FullDuplex	BOOL		Режим работы порта (TRUE - полный дуплекс, FALSE - полудуплекс)
Speed	BYTE		Скорость порта (0 - 10 МБит/с, 1 - 100 МБит/с, 2 - 1000 МБит/с)
PortBlocking	BOOL		TRUE - порт блокирован
RemoteRXFailure	BOOL		TRUE - отсутствует приём на удалённом порту
RXFailure	BOOL		TRUE - отсутствует приёма на порту
Loopback	BOOL		TRUE - наличие на порту петли
MultipleErrors	BOOL		TRUE - множественные ошибки на порту
DiagErrorsCounter	BYTE		Счётчик ошибок диагностического канала

Рис. 87 – Описание структуры PowerlinkDiag

Протокол позволяет добавлять в проект модули ввода-вывода и коммуникационные модули удаленной стойки изолированной CAN-шины. Модуль МК-545-010 в удаленной стойке заменяет собой модуль CPU.

⚠ ВНИМАНИЕ

Параметр `rack` устройств удалённых стоек не связан с параметром rack стоек процессорных модулей, и назначается независимо, как правило с 1

Для работы с модулями ввода-вывода и коммуникационными модулями необходимо в параметрах устройства `cn545` указать идентификатор OpenPowerlink от 1 до 239 (параметр `pwl_id`). Этот идентификатор является одинаковым для всех модулей стойки и его так же нужно указать для всех модулей. Идентификатор OpenPowerlink для каждой стойки должен быть уникальным.

На 01.06.2020 в одном проекте должно быть не более 40 устройств `cn545` (номера идентификаторов OpenPowerlink должны быть с 1 по 40).

3.7.7 Реализация сбора диагностической информации модуля CPU

В силу особенностей работы модулей CPU в режиме Failover использование в программе пользователя диагностических переменных времени исполнения (доступны в секции Global Variables, имена имеют вид `__SYSVA_xxx`) не допускается. Поэтому все их значения реализованы в качестве входных переменных в дополнительном устройстве `system_info`. Также устройство `system_info` возвращает состояние служб модуля CPU.

Устройство `system_info` имеет один входной канал типа `SysInfo`. Расшифровка его полей приводится в Табл. 18.

Табл. 18 – Расшифровка значений полей структуры `SysInfo`

Поле структуры SysInfo	Описание
`Cycle.CycleCounter`	Число выполненных циклов программы пользователя
`Cycle.CycleDateTime`	Время с момента начала работы программы пользователя
`Cycle.ScanCounter`	Число выполненных чтений входных переменных
`Cycle.CycleTime`	Заданное время цикла, мс
`Cycle.CurrentCycleTime`	Текущее время выполнения программы пользователя, мс
`Cycle.MaxCycleTime`	Максимальное время выполнения программы пользователя, мс
`Cycle.CycleOverflows`	Число превышений заданного времени цикла
`ResInfo.Name`	Полное имя ресурса проекта
`ResInfo.Mode`	Режим выполнения ресурса
`ResInfo.CRC32`	CRC32 файлов ресурса в CPU
`ResInfo.ModifyDate`	Дата последнего изменения ресурса
`ResInfo.ModifyTime`	Время последнего изменения ресурса
`Failover.IsEnable`	TRUE если разрешён режим работы Failover
`Failover.isPrimary`	TRUE если CPU стартовал как первичный, FALSE если CPU стартовал как вторичный
`Failover.IsActive`	TRUE если CPU является активным
`Failover.ErrCode`	Код ошибки Failover, согласно документации
`Failover.DataSyncTime`	Время синхронизации данных между CPU, мс
`Failover.DataSyncCount`	Число полных синхронизаций с момента запуска программы пользователя
`Failover.HeartbeatSyncTime`	Время между тактовыми импульсами Failover, мс
`Failover.ActiveCPURack`	Номер стойки с активным CPU (1..8)
`Failover.ActiveCPUSlot`	Номер слота с активным CPU (3 или 4)
`Services.FTPStatus`	**0** - служба FTP остановлена, **1** - запущена, **-1** - запускается, **-2** - останавливается

3.8 Работа с модулями аналогового ввода МК-513-016

Согласно Табл. 1, модулю аналогового ввода МК-513-016 в среде разработки АСР Workbench ISaGRAF 6.50 соответствует модуль изделия `ai16`.

Кроме диагностического канала, модуль изделия `ai16` имеет в своём составе простое устройство `ai16_` с 16 входными каналами данных типа `WORD`.

Значение каждого канала устройства `ai16_` соответствует коду АЦП соответствующего входа модуля. Код АЦП канала изменяется линейно, изменение на 1 мА соответствует изменению кода АЦП на 660 единиц. Расшифровка кодов АЦП модуля аналогового ввода приводится в Табл. 19.

Табл. 19 – Расшифровка значений входов устройства `ai16_`

Значения кода АЦП канала	Величина входного тока канала, мА
0	0,00 (минимальное значение)
2640	4,00
13200	20,00
16383	24,82 (максимальное значение)

Также каждый канал устройства `ai16_` имеет свои независимые параметры (Рис. 88).

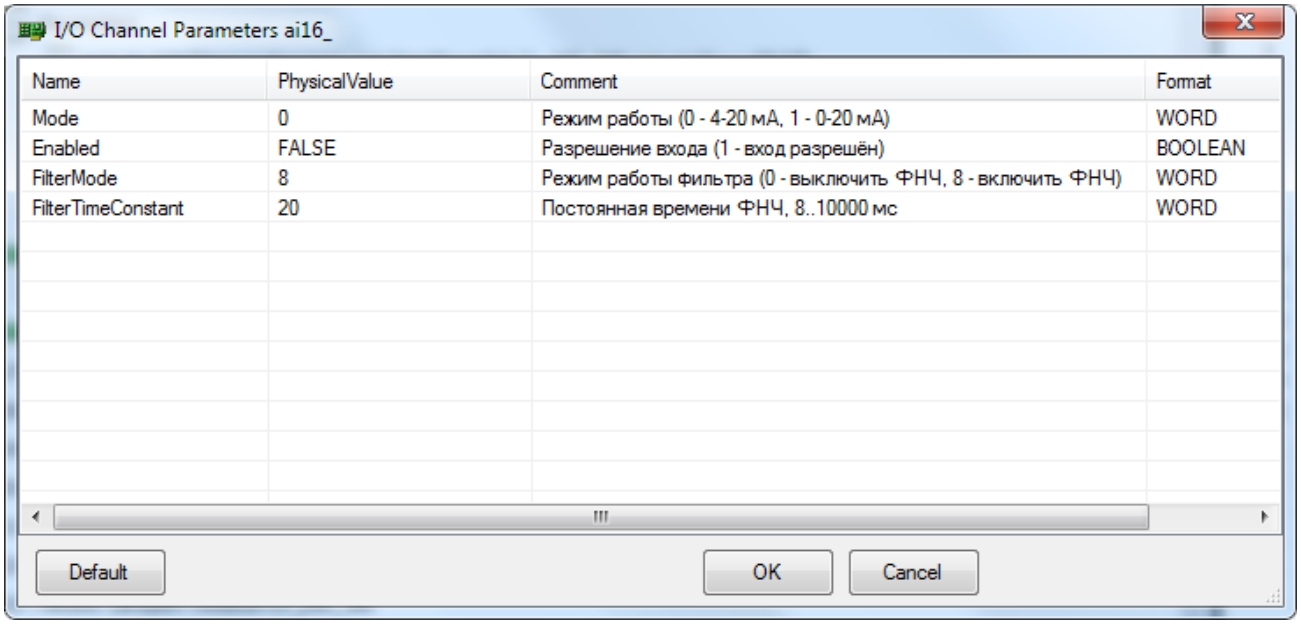


Рис. 88 – Описание параметров каналов устройства `ai16_`

Параметры каналов устройства `ai16_`:

– `Mode` – режим работы подключаемого к аналоговому входу датчику. 0 – 4-20 мА, 1 – 0-20 мА. Влияет только на порог срабатывания индикации обрыва на передней панели модуля аналогового ввода МК-513-016, измерение текущего значения тока при обоих режимах работы производится от 0 мА. По умолчанию все каналы работают в режиме 4-20 мА.

– `Enabled` – разрешение работы канала. FALSE – запрещён, TRUE – разрешён. Запрещённый канал постоянно возвращает код АЦП равный 0. По умолчанию все каналы разрешены.

– `FilterMode` – режим работы фильтров канала (Табл. 20). Работа фильтров канала при иных значениях параметра `FilterMode` не определена. По умолчанию фильтрация на всех каналах отключена.

– FilterLowPassFreq – постоянная времени фильтра низких частот, от 3 до 10000 мс. Имеет значение только при FilterMode=8. По умолчанию постоянная времени на всех каналах равна 20 мс.

Табл. 20 – Расшифровка значений параметра FilterMode канала устройства ai16_

Значение параметра FilterMode	Режим работы фильтров канала устройства ai16_
0	Фильтр низких частот выключен
8	Фильтр низких частот включен

3.9 Работа с модулями аналогового вывода МК-514-008, МК-514-008А

Согласно Табл. 1, модулям аналогового вывода МК-514-008 и МК-514-008А в среде разработки АСР Workbench ISaGRAF 6.50 соответствует модули изделия ao8 и ao8a соответственно.

Кроме диагностического канала, модуль изделия ao8 имеет в своём составе простое устройство ao8_ с 8 выходными каналами данных типа WORD, и простое устройство ao8status_ с 8 входными каналами данных типа BYTE.

Значение каждого канала устройства ao8_ соответствует коду ЦАП соответствующего выхода модуля. Код ЦАП канала изменяется линейно, изменение на 1 мА соответствует изменению кода ЦАП на 2730,625 единиц. Расшифровка кодов ЦАП модуля аналогового вывода приводится в Табл. 21.

Табл. 21 – Расшифровка значений выходов устройства ao8_

Значения кода ЦАП канала	Величина выходного тока канала, мА
0	0,00 (минимальное значение)
10922	4,00
54613	20,00
65535	24,00 (максимальное значение)

Также каждый канал устройства ao8_ имеет свои независимые параметры (Рис. 89).

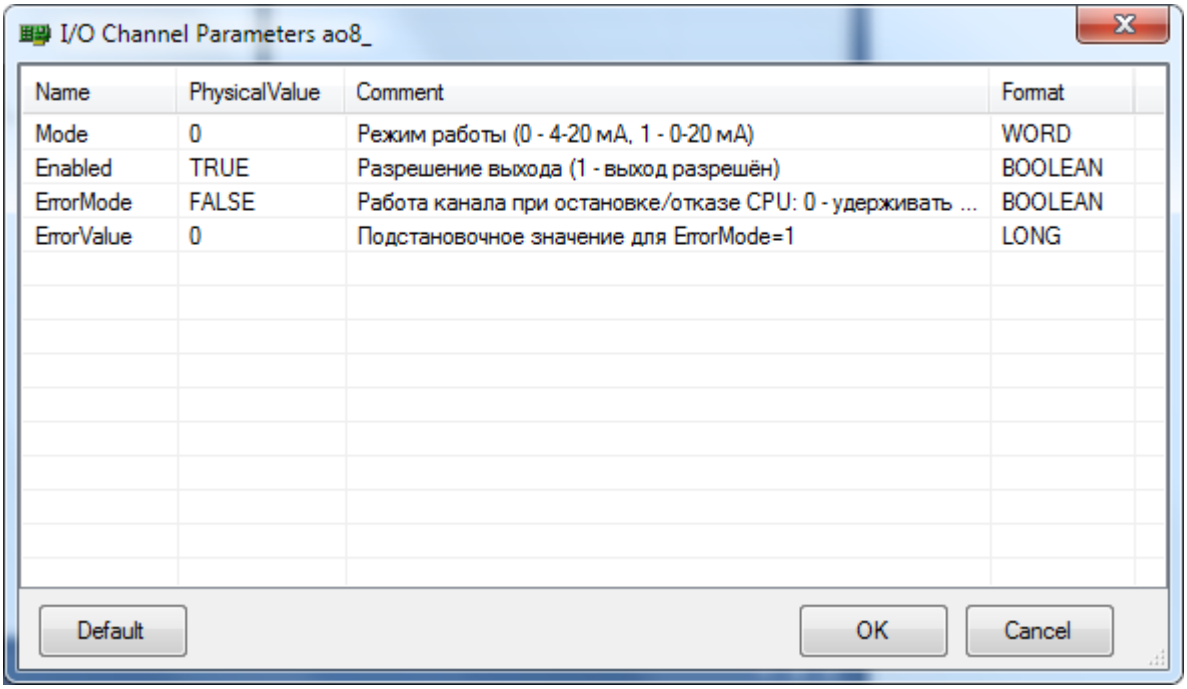


Рис. 89 – Описание параметров каналов устройства ao8_

Параметры каналов устройства ao8_:

– Mode – режим работы аналогового выхода. 0 – 4-20 мА, 1 – 0-20 мА. В режиме 4-20 мА при задании кода ЦАП ниже значения 10922 (4 мА) выходной ток канала всегда составляет 4 мА, в режиме 0-20 мА выходной ток канала строго соответствует заданному значению. По умолчанию все каналы работают в режиме 4-20 мА.

– Enabled – разрешение работы канала. FALSE – запрещён, TRUE – разрешён. Выходной ток запрещённого канала равен последнему заданному значению для уже работающего и 0 мА для не инициализованного модуля аналогового вывода. По умолчанию все каналы разрешены.

– ErrorMode – режим работы канала при потере модулем аналогового вывода связи с модулем CPU. FALSE – при потере связи с модулем CPU фиксировать значение тока канала, TRUE – присваивать коду ЦАП модуля значение параметра ErrorValue. По умолчанию все каналы имеют ErrorMode=FALSE.

– ErrorValue – значение кода ЦАП канала в режиме работы ErrorMode=TRUE при потере модулем аналогового вывода связи с CPU. По умолчанию ErrorValue всех каналов равны 0.

Значение каждого канала устройства ao8status_ соответствует состоянию электрических цепей канала модуля аналогового вывода. Расшифровка кодов каналов устройства ao8status_ модуля аналогового вывода приводится в Табл. 22.

Табл. 22 – Расшифровка значений входов устройства `ao8status_`

Значения кода канала	Состояние аналогового входа
0	Нет ошибок
1	Разрыв цепи
2	Отсутствует внешнее питание модуля

3.10 Работа с модулями аналогового вывода МК-576

Согласно Табл. 1, модулям аналогового вывода МК-514-008 и МК-514-008А в среде разработки ACP Workbench ISaGRAF 6.50 соответствует модули изделия `ao8` и `ao8a` соответственно.

3.11 Работа с модулями аналогового вывода МК-516-008, МК-516-008А

Согласно Табл. 1, модулям аналогового ввода МК-516-008 и МК-516-008А в среде разработки ACP Workbench ISaGRAF 6.50 соответствует модули изделия `ai8` и `ai8a` соответственно.

Кроме диагностического канала, модуль изделия `ai8` имеет в своём составе простое устройство `ai8_` с 8 входными каналами данных типа `WORD`.

Значение каждого канала устройства `ai8_` соответствует коду АЦП соответствующего входа модуля. Код АЦП канала изменяется линейно, изменение на 1 мА соответствует изменению кода АЦП на 2621,4 единиц. Расшифровка кодов АЦП модуля аналогового ввода приводится в Табл. 23.

Табл. 23 – Расшифровка значений входов устройства `ai8_`

Значения кода АЦП канала	Величина входного тока канала, мА
0	0,00 (минимальное значение)
10486	4,00
52428	20,00
65535	25,00 (максимальное значение)

Также каждый канал устройства `ai8_` имеет свои независимые параметры (Рис. 90).

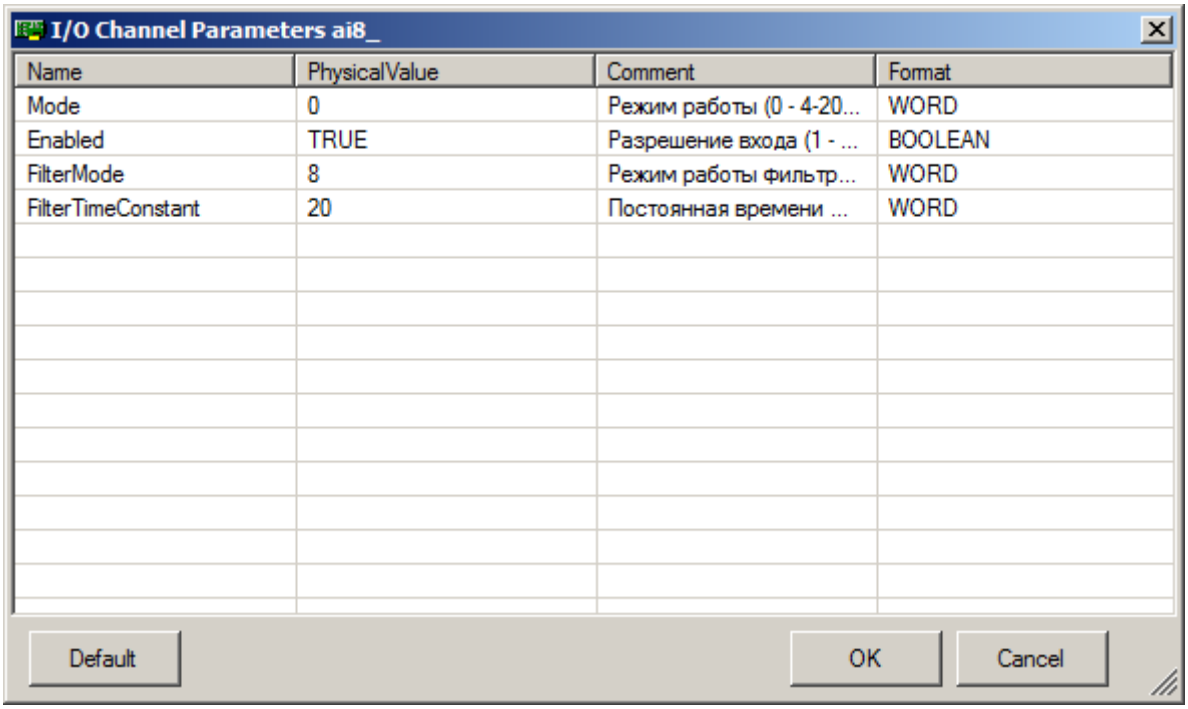


Рис. 90 – Описание параметров каналов устройства `ai8_`

Параметры каналов устройства `ai8_`:

– `Mode` – режим работы подключаемого к аналоговому входу датчику. 0 – 4-20 мА, 1 – 0-20 мА. Влияет только на порог срабатывания индикации обрыва на передней панели модуля аналогового ввода МК-516-08, измерение текущего значения тока при обоих режимах работы производится от 0 мА. По умолчанию все каналы работают в режиме 4-20 мА.

– `Enabled` – разрешение работы канала. FALSE – запрещён, TRUE – разрешён. Запрещённый канал постоянно возвращает код АЦП равный 0. По умолчанию все каналы разрешены.

– `FilterMode` – режим работы фильтров канала (Табл. 24). Работа фильтров канала при иных значениях параметра `FilterMode` не определена. По умолчанию фильтрация на всех каналах отключена.

– `FilterLowPassFreq` – постоянная времени фильтра низких частот, от 3 до 10000 мс. Имеет значение только при `FilterMode`=8. По умолчанию постоянная времени на всех каналах равна 5 мс.

Табл. 24 – Расшифровка значений параметра `FilterMode` канала устройства `ai8_`

Значение параметра FilterMode	Режим работы фильтров канала устройства ai8_
0	Фильтр низких частот выключен
8	Фильтр низких частот включен

3.12 Работа с модулями дискретного ввода МК-521-032

Согласно Табл. 1, модулю дискретного ввода МК-521-032 в среде разработки ACP Workbench ISaGRAF 6.50 соответствуют модули изделия `di32` и `di32history`.

Кроме диагностического канала, модуль изделия `di32` имеет в своём составе простое устройство `di32_` с 32 входными каналами данных типа `DI32Inputs` (Рис. 91), и простое устройство `di32rsttrig_` с 32 выходными каналами данных типа `BOOL`.

Name	Data Type	String Size	Comment
DI32Inputs			
DI32Inputs	DI32Inputs		
input	BOOL		Дискретный вход
trigger	BOOL		Триггер, срабатывающий по срезу входа

Рис. 91 – Описание структуры данных `DI32Inputs` устройства `di32_` модуля дискретного ввода изделия

Входные каналы устройства `di32_` в структуре `DI32Inputs` возвращают текущее значение дискретного входа в поле `input` и флаг срабатывания триггера в поле `trigger`, переходящего в TRUE по срезу дискретного входа канала.

Также каждый канал устройства `di32_` имеет свои независимые параметры (Рис. 92).

Параметры каналов устройства `di32_`:

- `Enabled` – разрешение работы канала. FALSE – запрещён, TRUE – разрешён. Для запрещённого канала поля `input` и `trigger` структуры `DI32Inputs` всегда равны FALSE. По умолчанию все каналы разрешены.
- `Invert` – режим инверсии поля `input` структуры `DI32Inputs` канала. FALSE – инверсия выключена, TRUE – инверсия включена. По умолчанию инверсия всех каналов выключена.
- `BounceTime` – минимально допустимая длина входного сигнала на канале, от 0 до 100000 мкс. Входной сигнал длительностью меньше значения `BounceTime` игнорируется. Данный параметр используется для защиты входов от дребезга. По умолчанию значение `BounceTime` всех каналов равно 0.

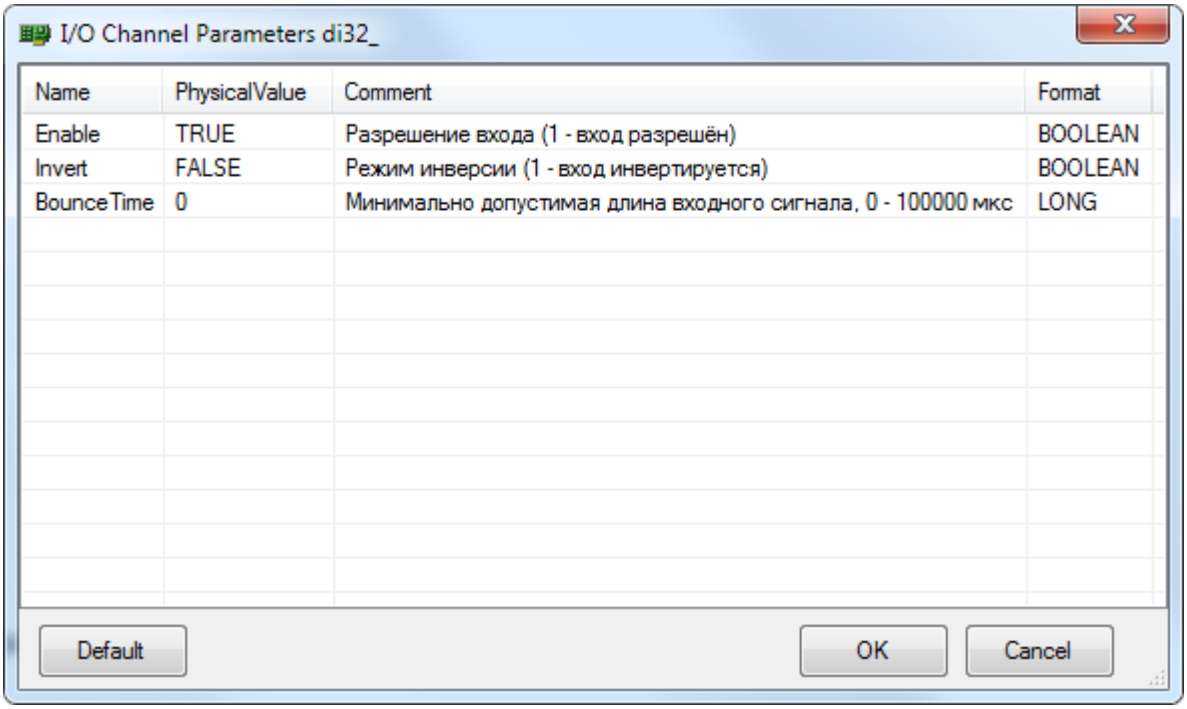


Рис. 92 – Описание параметров каналов устройства `di32_`

Выходные каналы простого устройства `di32rsttrig_` предназначены для сброса полей `trigger` соответствующих входных каналов устройства `di32_`. Сброс поля `trigger` производится записью TRUE в выходной канал.

Выходные каналы простого устройства `di32rsttrig_` предназначены для сброса полей `trigger` соответствующих входных каналов устройства `di32_`. Сброс поля `trigger` производится записью TRUE в выходной канал.

В отличие от `di32`, `di32history` имеет в своем составе простое устройство `di32history_`с 32 входными каналами данных типа `DI32WithTimeStamp` (Рис. 93). Простое устройство `di32rsttrig_` в `di32history` отсутствует.

Name	Data Type	String Size	Comment
DI32WithTimeStamp	DI32WithTimeStamp		
TimeStamp	NowTime		Метка времени для значений входов
TimeStamp.sec_unix	DINT		Число секунд с 1970/01/01
TimeStamp.sec_2000	DINT		Число секунд с 2000/01/01
TimeStamp.msec	DINT		Число миллисекунд в текущей секунде
TimeStamp.usec	DINT		Число микросекунд в текущей секунде
TimeStamp.lastScan_msec	DINT		Время выполнения последнего скана, в миллисекундах
TimeStamp.current_date	DATE		Текущая дата в формате DATE
TimeStamp.current_time	TIME		Текущее время в формате TIME
*			
Offset	DINT		Смещение времени (относительно текущего) для значений входов, мс
Values	DI32HistoryArray		Значения всех входов в указанный момент времени
Values[1]	BOOL		
Values[2]	BOOL		

Рис. 93 – Описание структуры данных `DI32WithTimeStamp` устройства `di32history` модуля дискретного ввода изделия

Простое устройство `di32history_` предназначено для получения изменявшихся значений дискретных входов за время последнего цикла программы пользователя.

В поле `Values` структуры `DI32WithTimeStamp` хранятся значения всех 32 дискретных входов (начиная с самого «старого»). Метки времени изменения входов фиксируется в поле `TimeStamp`. В следующей после завершающей записи в устройстве `di32history_` обнуляется поле `TimeStamp`.

⚠ ВНИМАНИЕ

Сохранение изменённых значений дискретных входов происходит не чаще, чем раз в 10 мс.

3.13 Работа с модулями дискретного вывода МК-531-032

Согласно Табл. 1, модулю дискретного вывода МК-531-032 в среде разработки АСР Workbench ISaGRAF 6.50 соответствует модуль изделия `do32`.

Кроме диагностического канала, модуль изделия `do32` имеет в своём составе простое устройство `do32_` с 32 выходными каналами данных типа `DO32Outputs` (Рис. 94). Часть каналов устройства `do32_` может работать в режиме широтно-импульсной модуляции (далее ШИМ).

⚠ ВНИМАНИЕ

На 01.06.2018 в описании параметров `PWM1_Freq`, `PWM2_Freq` и `PWM3_Freq` допущена ошибка. При конфигурировании параметров устройства `do32_` следует руководствоваться настоящим руководством, а не комментариями в окне настройки параметров устройства `do32_`.

Выходные каналы устройства `do32_` в структуре `DO32Outputs` передают в модуль дискретного вывода изделия значение дискретного выхода в поле `output` либо значение скважности выходного сигнала в поле `pwm` (при работе канала в режиме ШИМ).

Name	Data Type	String Size	Comment
DO32Outputs			
DO32Outputs	DO32Outputs		
output	BOOL		Дискретный выход
pwm	USINT		Выход управления ШИМ

Рис. 94 – Описание структуры данных `DO32Outputs` устройства `do32_` модуля дискретного вывода изделия

В режиме ШИМ может работать до 8 каналов устройства `do32_`, при этом несущую частоту можно задавать только для групп каналов (Рис. 95 и Рис. 96). Допустимые значения частоты модуляции ШИМ – от 2 до 250 Гц. По умолчанию значение частоты ШИМ равно 0 Гц.

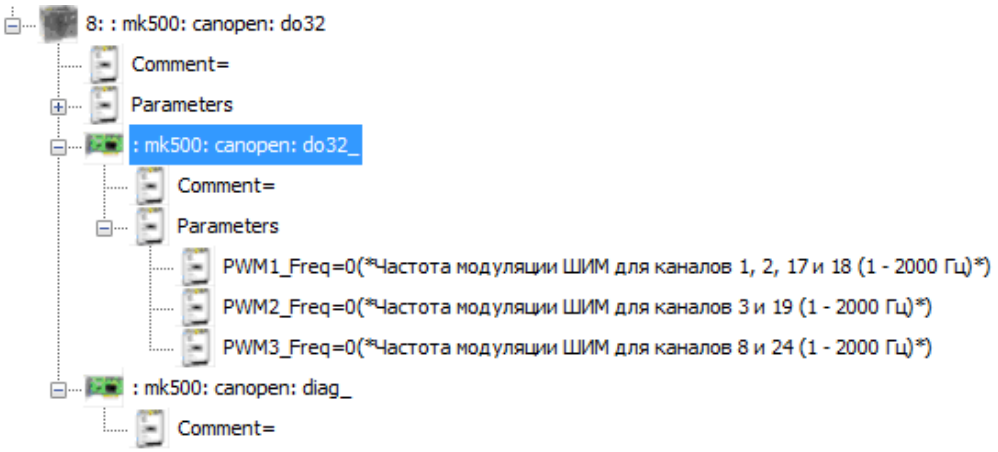


Рис. 95 – Параметры устройства `do32_`

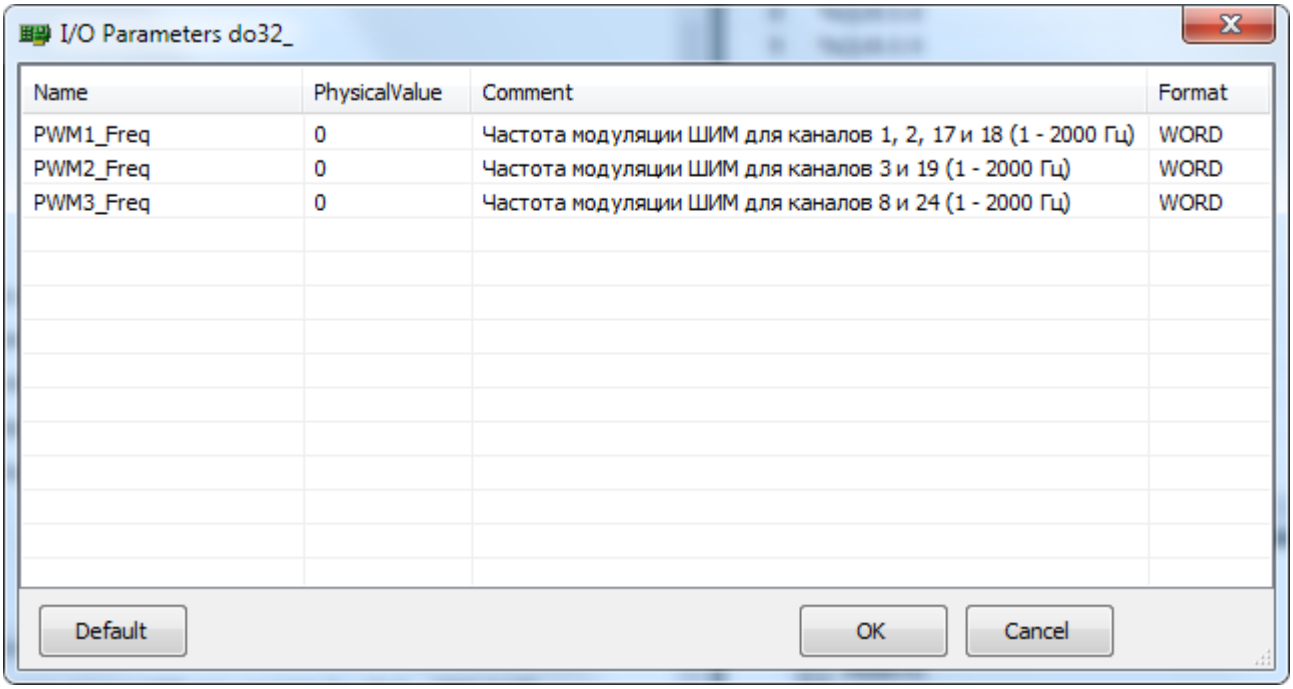


Рис. 96 – Окно настройки параметров ШИМ устройства `do32_`

Также каждый канал устройства `do32_` имеет свои независимые параметры (Рис. 97).

- `Mode` – режим работы дискретного выхода. 0 – дискретный выход, 1 – ШИМ. Режим ШИМ можно включить только для каналов с номерами 1, 2, 3, 8, 17, 18, 19 и 24. По умолчанию все каналы работают в режиме дискретного выхода.
- `Enabled` – разрешение работы канала. FALSE – запрещён, TRUE – разрешён. Выходной сигнал запрещённого канала равен последнему заданному значению для уже работающего и FALSE для не инициализованного модуля дискретного вывода. По умолчанию все каналы разрешены.
- `Invert` – режим инверсии поля `output` структуры `DO32Outputs` канала. FALSE – инверсия выключена, TRUE – инверсия включена. По умолчанию инверсия всех каналов выключена.
- `ErrorMode` – режим работы канала при потере модулем дискретного вывода связи с модулем CPU. FALSE – при потере связи с модулем CPU фиксировать значение канала, TRUE – присваивать каналу модуля значение параметра `ErrorValue`. Режим `ErrorMode` работает только для каналов в режиме дискретного выхода. По умолчанию все каналы имеют `ErrorMode=FALSE`.
- `ErrorValue` – значение канала в режиме работы `ErrorMode=TRUE` при потере модулем дискретного вывода связи с CPU. По умолчанию `ErrorValue` всех каналов равен `False`.

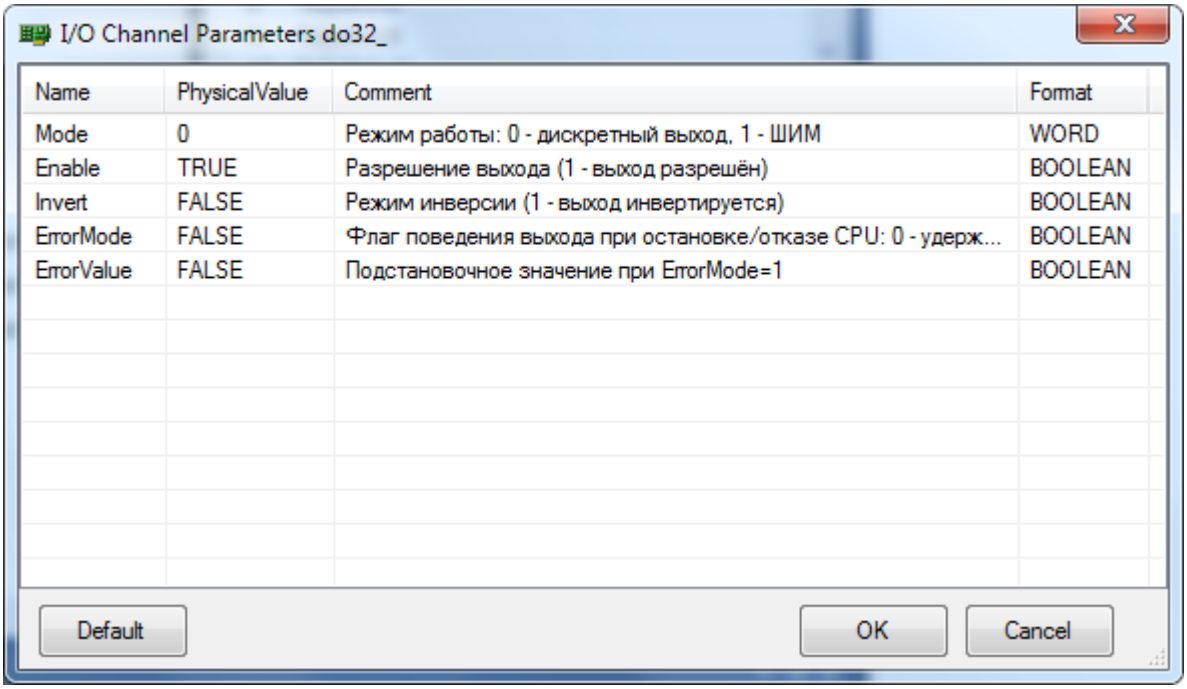


Рис. 97 – Описание параметров каналов устройства do32_

3.14 Работа с коммуникационными модулями МК-541-002

Согласно Табл. 1, коммуникационному модулю МК-541-002 в среде разработки АСР Workbench ISaGRAF 6.50 соответствуют модули `rs485` и `rs485new`.

Кроме диагностического канала, модули `rs485` и `rs485new` имеют в своём составе два простых устройства `rs485port1_` и `rs485port2_` с 64 входными каналами данных типа `BYTE`, и простое устройство `rs485data_` с 1 выходным каналом данных типа `ModbusREGs`.

В режиме работы Modbus «ведущий» порт модуля `rs485` циклически выполняет команды (до 64 команд на порт), переданные в него вызовом функции инициализации `InitRS485ModuleModbus` (см. ниже), в режиме работы Modbus «ведомый» работает ведомым устройством с фиксированным диапазоном данных с 0 по 959 Holding Registers.

Модуль `rs485new` имеет в своем составе два выходных устройства управления `rs485request1_` и `rs485request2_` предназначенных для передачи команд в драйвер Modbus. Поэтому модуль `rs485new` не требует инициализации вызовом функции `InitRS485ModuleModbus`. Устройства `rs485request1_` и `rs485request2_` имеют по 64 переменных-структуры типа `ModbusMasterRequest` (п. 3.7.1).

Простые устройства `rs485port1_` и `rs485port2_` служат для конфигурирования портов 1 и 2 коммуникационного модуля изделия, а также для получения диагностической информации о выполняемых каждым из этих портов командах. Оба простых устройства возвращают во входных значениях статус соответствующей команды для режима работы Modbus «ведущий» (расшифровка значений дана в Табл. 8), и не имеют смысла для режима работы Modbus «ведомый».

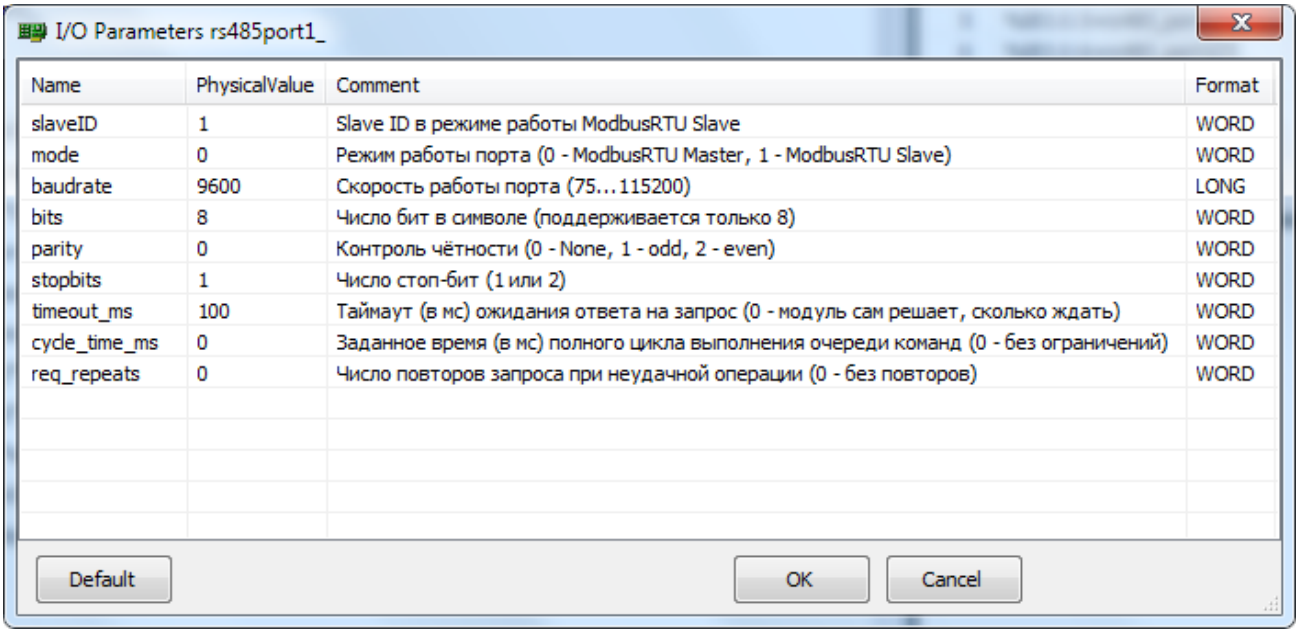


Рис. 98 – Окно настройки параметров устройства `rs485port1_` и `rs485port2_`

Конфигурирования портов 1 и 2 коммуникационного модуля изделия выполняется с помощью параметров устройств `rs485port1_` и `rs485port2_` соответственно (Рис. 98). Параметры каналов устройств `rs485port1_` и `rs485port2_`:

- `slaveID` – идентификатор порта на шине Modbus для режима работы «ведомый», по умолчанию установлен 1.
- `mode` – режим работы порта: 0 – ModbusRTU в режиме «ведущий», 1 - ModbusRTU в режиме «ведомый». По умолчанию включен режим 0.
- `baudrate` – скорость работы порта, по умолчанию 9600.
- `bits` – число бит в символе. Поддерживается только режим 8 бит.
- `parity` – контроль чётности:0 – нет, 1 – even, 2 – odd. По умолчанию 0 – без контроля чётности.
- `stopbits` – число стоп-бит: 1 – 1 стоп-бит, 2 – 2 стоп-бита. По умолчанию 1.
- `timeout_ms` – таймаут ожидания ответа на запрос (в мс) для режима работы Modbus «ведущий», 0 – автоматическое определение времени ожидания ответа. По умолчанию 100 мс.
- `cycle_time_ms` – заданное время полного выполнения очереди команд (в мс) для режима работы Modbus «ведущий», 0 – нет ограничений на полное время выполнения. По умолчанию 0 – без ограничений.
- `req_repeats` – число повторов неудачно выполненной команды для режима работы Modbus «ведущий», 0 – нет повторов. По умолчанию 0, без повторов.

Простое устройство `rs485data_` предназначено для хранения отправляемых и получаемых командами Modbus данных (только типа `WORD`, то есть регистровых данных Modbus) для режима работы Modbus «ведущий», и для формирования адресного пространства устройства на шине Modbus для режима работы Modbus «ведомый».

Устройство `rs485data_` имеет 1 переменную-структуру типа `ModbusREGs` (массив из 1024 переменных типа `WORD`, Рис. 71). В начале цикла выполнения программы канал этого устройства содержит в себе обновившиеся в ходе обработки команд Modbus данные. Поэтому для связанной с этим устройством переменной следует устанавливать атрибут Read/Write (п. 3.1.4), а в начале каждого цикла следует копировать значение переменной в отдельную, не связанную с каналами рабочую переменную.

⚠ ВНИМАНИЕ

В обоих режимах работы порта поддерживается только 960 регистров из 1024 доступных в переменной типа `ModbusREGs`.

⚠ ВНИМАНИЕ

Регистровое пространство устройства `rs485data_` используется обоими портами независимо друг от друга. Задача разделения данных между командами обоих портов возлагается на разработчика программы пользователя.

3.14.1 Инициализация и передача команды в модуль rs485

Для инициализации модуля `rs485` следует выполнить в программе пользователя функцию `InitRS485ModuleModbus`. Пример вызова функции приведён в Листинге 4, параметры функции и её возвращаемое значение приведено в Табл. 25. Расшифровка возвращаемого значения функции приведена в Табл. 26.

```
rack := 1;
slot := 5;
FOR i := 0 TO 63 DO
      request_port1[i].command.slave_id := ANY_TO_BYTE(i+1);
      request_port1[i].enable := true;
      request_port1[i].single_request := false;
      request_port1[i].on_modify_request := false;
      request_port1[i].command.func_code := 16;
      request_port1[i].command.slave_data_addr := ANY_TO_WORD(0+(i*13));
      request_port1[i].command.slave_data_length := 13;
      request_port1[i].command.offset := ANY_TO_WORD(0+(i*13));

      request_port2[i].command.slave_id := ANY_TO_BYTE(64+i+1);
      request_port2[i].enable := true;
      request_port2[i].single_request := false;
      request_port2[i].on_modify_request := false;
      request_port2[i].command.func_code := 16;
      request_port2[i].command.slave_data_addr := ANY_TO_WORD(0+(i*13));
      request_port2[i].command.slave_data_length := 13;
      request_port2[i].command.offset := ANY_TO_WORD(0+(i*13));
END_FOR;
res    :=    InitRS485ModuleModbus(opwl_id,    rack,    slot,    request_port1,
request_port1);
```

Листинг 4 – Формат вызова функции `InitRS485ModuleModbus` для режима работы порта «ведущий»

Размерность массивов параметров `request_port1` и `request_port2` может быть как больше, так и меньше 64. В этом случае будут обработаны первые 64 команды либо все команды массива соответственно.

Табл. 25 – Параметры и возвращаемые значения функции `InitRS485ModuleModbus`

`res := InitRS485ModuleModbus(rack, slot, req_port1, req_port2);`		
Имя параметра	Тип параметра	Описание
`res`	`INT`	Возвращаемое значение – результат выполнения функции
`opwl_id`	`WORD`	Идентификатор OpenPowerlink
`rack`	`WORD`	Номер стойки, в которой расположен модуль RS485
`slot`	`WORD`	Номер позиции в стойке, в которой расположен модуль RS485
`req_port1`	`ModbusMasterRequest[1..64]`	Массив команд для порта 1 модуля RS485
`req_port2`	`ModbusMasterRequest[1..64]`	Массив команд для порта 2 модуля RS485

Табл. 26 – Расшифровка возвращаемого значения функции `InitRS485ModuleModbus`

Значение res	Расшифровка
0	Функция выполнена без ошибок
-1	Параметры rack или slot выходят за пределы допустимых значений
-2	Ошибка инициализации среды исполнения IsaVM
-3	Не найден модуль RS485
-4	Ошибка инициализации модуля RS485

Описание полей `ModbusMasterRequest` приводилось выше в п. 3.7.1. Следует отметить, что для модулей `rs485` и `rs485new` кроме режимов непрерывного опроса и одиночного запроса (п. 3.7.1, описание поля `do_single_req`) поддерживается режим записи по изменениям (`do_single_req=false, on_modify_request=true`).

В режиме записи по изменениям проверяется диапазон адресов, на которые ссылается команда (`command.slave_data_addr` и `command.slave_data_length`), и в случае их изменения выполняется команда. Данный режим имеет смысл только для функций записи (5, 6, 15 и 16 команды Modbus).

В случае если и `do_single_req` и `on_modify_request` равны `true`, команда исполняется в режиме непрерывного опроса.

Для режима работы какого-либо порта Modbus «ведомый» можно вообще не передавать соответствующий массив. См. Листинг 5.

```
opwl_id := 240;
rack := 1;
slot := 5;
res := InitRS485ModuleModbus(opwl_id, rack, slot, , );
```

Листинг 5 – Формат вызова функции `InitRS485ModuleModbus` для режима работы порта «ведомый»

Функцию `InitRS485ModuleModbus` следует вызывать каждый скан, а не только при начале работы программы пользователя – по двум причинам:

1. В проектах с режимом Failover включение второго модуля CPU в работу может произойти после старта. В этом случае флаги начала работы уже будут сброшены и типовая конструкция разового вызова (Листинг 6) не сработает.

```
if init = false then
    init := true;
    opwl_id := 240;
    rack := 1;
    slot := 5;
    res := InitRS485ModuleModbus(opwl_id, rack, slot, requests1,
requests2);
end_if;
```

Листинг 6 – Типовая конструкция разового вызова функции

2. Через вызовы функции `InitRS485ModuleModbus` в модуль `rs485` передаются команды управления.

3.14.2 Инициализация и передача команды в модуль `rs485new`

При работе с модулем устройства `rs485new` функцию `InitRS485ModuleModbus` вызывать не следует совсем.

Отличительной особенностью устройства `rs485new` является возможность обновления тела команд в ходе выполнения программы пользователя. При обновлении команд следует учитывать следующие особенности:

1. Команды обновляются только блоками по 16 команд (1..16, 16..32 и так далее).
2. Блоки команд обновляются по одному. Блоки команд для разных портов обновляются последовательно: сначала для первого порта, потом для второго.
3. При обнаружении обновлений в первом встреченном блоке, обновления в последующих блоках игнорируются до завершения обновления первого встреченного блока.
4. Обновление блоков команд не происходит мгновенно, и может завершиться неудачей. Для контроля хода обновления данных следует анализировать поле `status_request` (Табл. 8) устройства `rs485port1_` или `rs485port2_` в любом канале, лежащем внутри изменяемого блока. Интерес представляют возвращаемые коды 30 и 31.

Глава 4 Использование функций расширения МК500

К функциям расширения МК500 относятся функции и функциональные блоки, добавленные в стандартную библиотеку ISaGRAF с целью расширения возможностей среды исполнения ISaGRAF. Подробнее о стандартной библиотеке ISaGRAF сказано в документах «Функциональные блоки» (Irsb_ISa6_ru), «Функции» (Irsf_ISa6_ru) и «Стандартные операции» (Irso_ISa6_ru).

Для использования функций расширения, поддерживаемых процессорным модулем изделия, следует открыть окно BlockLibrary, выполнив «VIEW»-«Block Library» (Рис. 99), В окне BlockLibrary следует нажать правую кнопку мыши и в контекстном меню выбрать пункт «Category» (Рис. 100). Из списка доступных категорий следует выбрать вкладку «МК500» (Рис. 101) и добавить необходимые функции расширения в окно разработки прикладной программы перетаскиванием.

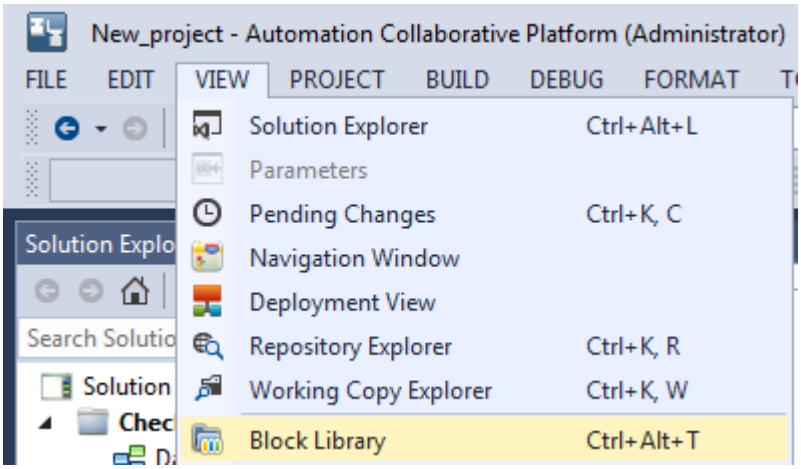


Рис. 99 – Открытие окна `BlockLibrary`

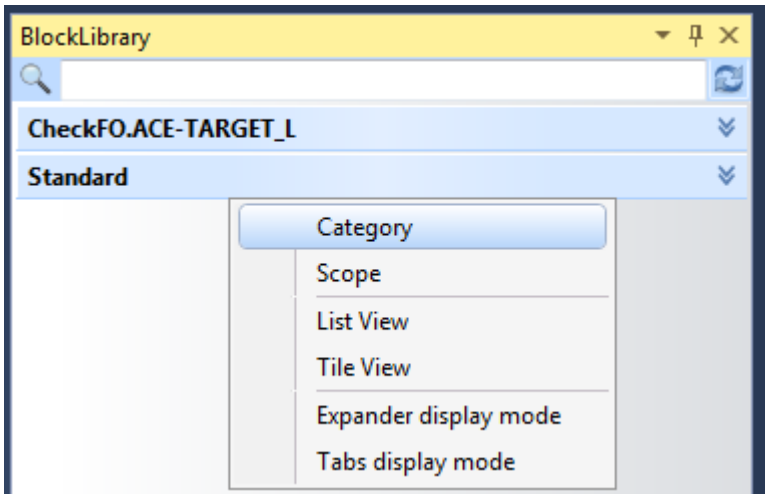


Рис. 100 – Переключение окна `BlockLibrary` в режим просмотра по категориям

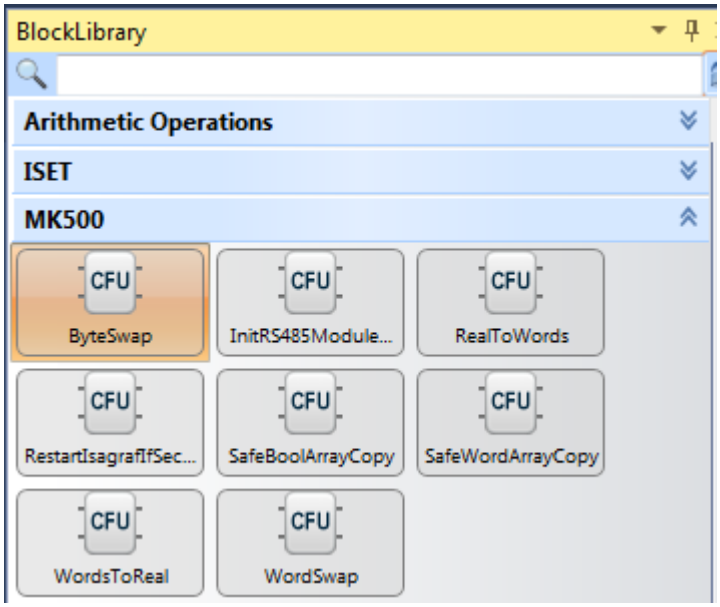


Рис. 101 – Окно `BlockLibrary` с функциями расширения МК500

4.1 Функции ByteSwap и WordSwap

Функции `ByteSwap` и `WordSwap` меняют местами в переменной-аргументе байты и слова соответственно, и возвращают их в возвращаемой переменной.

Данные функции могут применяться для согласования данных, принятых либо передаваемых на устройства с отличающимся порядком байт и/или слов.

Параметры и возвращаемые значения функций `ByteSwap` и `WordSwap` приведены в Табл. 27 и Табл. 28 соответственно, пример использования функций приведён в Листинге 7.

Табл. 27 – Параметры и возвращаемые значения функции `ByteSwap`

`res := ByteSwap(value);`		
Имя параметра	Тип параметра	Описание
`res`	`WORD`	Возвращаемое значение с переставленными байтами
`value`	`WORD`	Исходная переменная, в которой требуется поменять байты местами

Табл. 28 – Параметры и возвращаемые значения функции `WordSwap`

`res := WordSwap(value);`		
Имя параметра	Тип параметра	Описание
`res`	`DWORD`	Возвращаемое значение с переставленными словами
`value`	`DWORD`	Исходная переменная, в которой требуется поменять слова местами

```
wData := 16#1234;
dData := 16#12345678;

new_wData := ByteSwap(wData);
(*new_wData = 16#3412*)

new_dData := WordSwap(dData);
(*new_dData = 16#56781234*)
```

Листинг 7 – Использование функций `ByteSwap` и `WordSwap`

4.2 Функции WordsToReal и RealToWords

Функции `WordsToReal` и `RealToWords` выполняют побитное копирование указанного числа байт из массива типа `WORD` в массив типа `REAL` и из массива типа `REAL` в массив типа `WORD` соответственно.

Данные функции могут применяться для преобразования данных, принятых либо передаваемых по протоколу Modbus (который умеет оперировать только данными типа `BOOL` и `WORD`).

Параметры и возвращаемые значения функций `WordsToReal` и `RealToWords` приведены в Табл. 29 и Табл. 30 соответственно, пример использования функций приведён в Листинге 8.

Табл. 29 – Параметры и возвращаемые значения функции `WordsToReal`

`res := WordsToReal(wArray, wOffset, rArray, rOffset, size);`		
Имя параметра	Тип параметра	Описание
`res`	`INT`	Результат выполнения преобразования: 0 – успешно, -1 – неверные параметры
`wArray`	`WORD[a..b]`	Массив данных типа `WORD`, подлежащих преобразованию в данные типа `REAL` (**младшими словами вперёд**). Допускается произвольная допустимая размерность массива.
`wOffset`	`INT`	Смещение в массиве `wArray` (**в единицах индекса массива**), начиная с которого будет выполняться преобразование.
`rArray`	`REAL[c..d]`	Массив данных типа `REAL`, в который будут помещаться преобразованные значения. Допускается произвольная допустимая размерность массива.
`rOffset`	`INT`	Смещение в массиве `rArray` (**в единицах индекса массива**), начиная с которого будут помещаться преобразованные данные.
`size`	`INT`	Число преобразуемых **байт** данных массива `wArray`

Табл. 30 – Параметры и возвращаемые значения функции `RealToWords`

`res := RealToWords(rArray, rOffset, wArray, wOffset, size);`		
Имя параметра	**Тип параметра**	**Описание**
`res`	`INT`	Результат выполнения преобразования: 0 – успешно, -1 – неверные параметры
`rArray`	`REAL[a..b]`	Массив данных типа `REAL`, подлежащих преобразованию в данные типа `WORD`. Допускается произвольная допустимая размерность массива.
`rOffset`	`INT`	Смещение в массиве `rArray` (**в единицах индекса массива**), начиная с которого будет выполняться преобразование.
`wArray`	`WORD[c..d]`	Массив данных типа `WORD`, в который будут помещаться преобразованные значения (**младшими словами вперёд**). Допускается произвольная допустимая размерность массива.
`wOffset`	`INT`	Смещение в массиве `wArray` (**в единицах индекса массива**), начиная с которого будут помещаться преобразованные данные.
`size`	`INT`	Число преобразуемых **байт** данных массива `rArray`

Преобразование не будет выполнено (функции вернут -1) в случае, если после приведения переданных параметров будет получаться нулевой или отрицательный размер копируемых данных и/или отрицательный размер смещения внутри любого массива. Во всех прочих случаях будет выполнено преобразование исходя из минимально общего размера данных (из размеров областей данных внутри массивов и указанного числа байт)

```
(*wArray – WORD[1..4], rArray – REAL[1..2]*)
wArray[1] := 16#0000;
wArray[2] := 16#3F80;
wArray[3] := 16#0000;
wArray[4] := 16#4120;
res := WordsToReal(wArray, 0, rArray, 0, 8);
(*res = 0,  rArray[1] = 1.0, rArray[2] = 10.0 *)

rArray[2] := rArray[2] + 0.1;
res := RealToWords(rArray, 1, wArray, 2, 4);
(*res = 0, wArray[3] = 16#99A4, wArray[4] = 16#4121 *)
```

Листинг 8 – Использование функций `WordsToReal` и `RealToWords`

4.3 Функции DWordToReal и RealToDWord

Функции DWordToReal и RealToDWord выполняют побитное копирование из переменной типа DWORD в переменную типа REAL и, из переменной типа REAL в переменную типа DWORD соответственно.

Параметры и возвращаемые значения функций DWordToReal и RealToDWord приведены в Табл. 31 и Табл. 32 соответственно, пример использования функций приведён в Листинге 9.

Табл. 31 – Параметры и возвращаемые значения функции DWordToReal

VarReal := DWordsToReal(VarDWord);		
Имя параметра	Тип параметра	Описание
VarReal	REAL	Переменная типа REAL, в которую будет помещаться преобразованное значение
VarDWord	DWORD	Переменная типа DWORD, подлежащая преобразованию в переменную типа REAL.

Табл. 32 – Параметры и возвращаемые значения функции RealToDWord

VarDWord := RealToDWord(VarReal);		
Имя параметра	Тип параметра	Описание
VarDWord	DWORD	Переменная типа DWORD, в которую будет помещаться преобразованное значение
VarReal	REAL	Переменная типа REAL, подлежащая преобразованию в переменную типа DWORD.

```
(*VarDWord – DWORD, VarReal – REAL*)
VarDWord:= 1092616192;
VarReal := DWordsToReal(VarDWord)
(*VarReal = 10.0*)

VarReal := 0.004723788;
VarDWord := RealToDWord(VarDWord);
(*VarDWord = 1000000001*)
```

Листинг 9 – Использование функций DWordToReal и RealToDWord

4.4 Функция Safe2DimWordArrayCopy

Функция Safe2DimWordArrayCopy выполняет копирование данных из одного двумерного массива WORD в другой двумерный массив WORD, не совпадающий по размерностям (иначе можно копировать присваиванием).

Параметры и возвращаемые значения функции Safe2DimWordArrayCopy приведены в Табл. 33, пример использования функции приведён в Листинге 10.

Табл. 33 – Параметры и возвращаемые значения функции `Safe2DimWordArrayCopy`

`res := Safe2DimWordArrayCopy (wArray_in, wArray_out, row_in, column_in, row_out, column_out, count);`		
Имя параметра	**Тип параметра**	**Описание**
`res`	`DINT`	Результат выполнения преобразования: 0 – успешно, -1 – неверные параметры
`wArray_in`	`WORD`	Массив данных типа `WORD`, из которого будет выполняться копирование. Допускается произвольная размерность массива.
`wArray_out`	`WORD`	Массив данных типа `WORD`, в который будет выполняться копирование. Допускается произвольная размерность массива.
`row_in`	`DINT`	Первый индекс смещение в массиве `wArray_in`, начиная с которого будет выполняться копирование.
`column_in`	`DINT`	Второй индекс смещение в массиве `wArray_in`, начиная с которого будет выполняться копирование.
`row_out`	`DINT`	Первый индекс смещение в массиве `wArray_out`, начиная с которого будут помещаться скопированные данные.
`column_out`	`DINT`	Второй индекс смещение в массиве `wArray_out`, начиная с которого будут помещаться скопированные данные.
`count`	`DINT`	Число копируемых переменных типа `WORD`.

```
 (*wArray_in – WORD [1..3,1..10], wArray_out – WORD [1..10,1..10]*)
wArray_in[1,1]:=10;
wArray_in[2,5]:=15;
wArray_in[3,10]:=20;
res := Safe2DimWordArrayCopy (wArray_in, wArray_out,1,1,5,1,30);
(*res = 0, wArray_out[5,1] = 10, wArray_out[6,5] = 15 wArray_out[7,10] = 20*)
```

Листинг 10 – Использование функции `Safe2DimWordArrayCopy`

4.5 Функции SafeBoolArrayCopy и SafeWordArrayCopy

Функции `SafeBoolArrayCopy` и `SafeWordArrayCopy` выполняют безопасное копирование данных из одного массива в другой для данных типа `BOOL` и `WORD` соответственно. Под безопасностью копирования подразумевается проверка размерностей массивов и копирование минимально совпадающего количества данных.

Параметры и возвращаемые значения функций `SafeBoolArrayCopy` и `SafeWordArrayCopy` приведены в Табл. 34 и Табл. 35 соответственно, пример использования функций приведён в Листинге 11.

Табл. 34 – Параметры и возвращаемые значения функции `SafeBoolArrayCopy`

`res := SafeBoolArrayCopy(bArray_src, bArray_dst);`		
Имя параметра	Тип параметра	Описание
`res`	`INT`	Результат выполнения преобразования: =>0 – число скопированных переменных типа `BOOL`, -1 – попытка скопировать 0 или меньше переменных типа `BOOL`, -2 – одно из значений индекса массива находится за пределами границ массива.
`bArray_src`	`BOOL[a..b]`	Массив данных типа `BOOL`, из которого будет выполняться копирование. Допускается произвольная допустимая размерность массива.
`bArray_dst`	`BOOL[c..d]`	Массив данных типа `BOOL`, в который будет выполняться копирование. Допускается произвольная допустимая размерность массива.

Табл. 35 – Параметры и возвращаемые значения функции `SafeWordArrayCopy`

`res := SafeWordArrayCopy(wArray_src, wArray_dst);`		
Имя параметра	Тип параметра	Описание
`res`	`INT`	Результат выполнения преобразования: =>0 – число скопированных переменных типа `WORD`, -1 – попытка скопировать 0 или меньше переменных типа `WORD`, -2 – одно из значений индекса массива находится за пределами границ массива.
`wArray_src`	`WORD[a..b]`	Массив данных типа `WORD`, из которого будет выполняться копирование. Допускается произвольная допустимая размерность массива.
`wArray_dst`	`WORD[c..d]`	Массив данных типа `WORD`, в который будет выполняться копирование. Допускается произвольная допустимая размерность массива.

```
(*bArray1 – BOOL[1..100], bArray2 – BOOL[1..200]*)
res := SafeBoolArrayCopy(bArray1, bArray2);

(*wArray1 – WORD[1..20], wArray2 – WORD[1..5]*)
res := SafeWordArrayCopy(wArray1, wArray2);
```

Листинг 11 – Использование функций `SafeBoolArrayCopy` и `SafeWordArrayCopy`

4.6 Функции SafeCopyFromModbusREGsArray и SafeCopyToModbusREGsArray

Функциии `SafeCopyFromModbusREGsArray` и `SafeCopyToModbusREGsArray` выполняют копирование данных из двумерного массива ModbusREGs в двумерный массив для данных типа `WORD` и копирование данных из двумерного массива для данных типа `WORD` в двумерный массив ModbusREGs.

Параметры и возвращаемые значения функций `SafeCopyFromModbusREGsArray` и `SafeCopyToModbusREGsArray` приведены в Табл. 36 и Табл. 37 соответственно, пример использования функций приведён в Листинге 12.

Табл. 36 – Параметры и возвращаемые значения функции `SafeCopyFromModbusREGsArray`

`res := SafeCopyFromModbusREGsArray(data_in, data_out, row_in, column_in, row_out, column_out, count);`		
Имя параметра	Тип параметра	Описание
`res`	`DINT`	Результат выполнения преобразования: Если => 0 – число скопированных переменных типа `WORD` массива data_in, -1 – массив data_in не одномерный или data_out не двумерный, -2 – одно из значений номера строки/столбца находится за пределами границ массива.
`data_in`	`ModbusREGs`	Массив данных типа `ModbusREGs`, из которого будет выполняться копирование.
`data_out`	`WORD`	Двумерный массив данных типа `WORD`, в который будет выполняться копирование.
`row_in`	`DINT`	Первый индекс смещение в массиве `data_in`, начиная с которого будет выполняться копирование.
`column_in`	`DINT`	Второй индекс смещение в массиве `data_in`, начиная с которого будет выполняться копирование.
`row_out`	`DINT`	Первый индекс смещение в массиве `data_out`, начиная с которого будут помещаться скопированные данные.
`column_out`	`DINT`	Второй индекс смещение в массиве `data_out`, начиная с которого будут помещаться скопированные данные.
`count`	`DINT`	Число копируемых регистров `WORD`.

Табл. 37 – Параметры и возвращаемые значения функции `SafeCopyToModbusREGsArray`

`res:= SafeCopyToModbusREGsArray (data_in, data_out, row_in, column_in, row_out, column_out, count);`		
Имя параметра	Тип параметра	Описание
`res`	`DINT`	Результат выполнения преобразования: Если => 0 – число скопированных переменных типа `WORD` массива data_in, -1 – массив data_in не двумерный или data_out не одномерный, -2 – одно из значений номера строки/столбца находится за пределами границ массива.
`data_in`	`WORD`	Массив данных типа `WORD`, из которого будет выполняться копирование.
`data_out`	`ModbusREGs`	Массив данных типа `ModbusREGs`, в который будет выполняться копирование.
`row_in`	`DINT`	Первый индекс смещение в массиве `data_in` , начиная с которого будет выполняться копирование.
`column_in`	`DINT`	Второй индекс смещение в массиве `data_in`, начиная с которого будет выполняться копирование.
`row_out`	`DINT`	Первый индекс смещение в массиве `data_out`, начиная с которого будут помещаться скопированные данные.
`column_out`	`DINT`	Второй индекс смещение в массиве `data_out`, начиная с которого будут помещаться скопированные данные.
`count`	`DINT`	Число копируемых регистров `WORD`.

```
(*data_in_fmr – ModbusREGs [0..3], data_out_fmr – WORD [0..3,1..1024]*)
data_in_fmr[0].regs[1]:=1;
data_in_fmr[1].regs[1024]:=1025;
data_in_fmr[3].regs[1024]:=1027;
res_fmr:=SafeCopyFromModbusREGsArray(data_in_fmr,data_out_fmr,0,1,0,1,4096);
(*res_fmr = 4096, data_out_fmr[0,1] = 1, data_out_fmr[1,1024] = 1025,
data_out_fmr [3,1024] = 1027*)

(*data_in_tmr – WORD [0..3,1..1024], data_out_tmr – ModbusREGs [0..3]*)
data_in_tmr [0,0]:=5;
data_in_tmr [1,1024]:=6;
data_in_tmr [3,1024]:=7;
res := SafeCopyToModbusREGsArray (data_in_tmr,data_out_tmr,1,2,0,1,2039);
(*res_tmr = 4096, data_out_tmr[0,1] = 5, data_out_tmr[1,1024] = 6,
data_out_tmr [3,1024] = 7*)
```

Листинг 12 – Использование функций `SafeCopyFromModbusREGsArray` и `SafeCopyToModbusREGsArray`

4.7 Функции WriteReal2DimArray и ReadReal2DimArray

Функциии `WriteReal2DimArray` и `ReadReal2DimArray` выполняют копирование данных из двумерного массива `REAL` в файл данных и копирование из файла данных в двумерный массив `REAL`.

Параметры и возвращаемые значения функций `WriteReal2DimArray` и `ReadReal2DimArray` приведены в Табл. 38 и Табл. 39 соответственно, пример использования функций приведён в Листинге 13.

Табл. 38 – Параметры и возвращаемые значения функции `WriteReal2DimArray`

`res := WriteReal2DimArray(filename,rArray);`		
Имя параметра	Тип параметра	Описание
`res`	`DINT`	Результат выполнения записи: 0 – успешно, -1 – недопустимое имя файла, -2 – ошибка открытия файла, -3 – ошибка записи данных в файл.
`filename`	`STRING`	Переменная типа `STRING`. Название файла, в который будет скопирован двумерный массив данных типа `REAL`.
`rArray`	`REAL`	Двумерный массив данных типа `REAL`, который будет скопирован в файл.

Табл. 39 – Параметры и возвращаемые значения функции `ReadReal2DimArray`

`res := ReadReal2DimArray(RetainReal,rArray_out);`		
Имя параметра	Тип параметра	Описание
`res`	`DINT`	Результат выполнения чтения: 0 – успешно, -1 – файла с таким именем нет в папке, -2 – ошибка открытия файла, -4 – ошибка чтения данных из файла, -5 - размер файла не совпадает с размером массива данных типа `REAL`
`filename`	`STRING`	Переменная типа `STRING`. Название файла, из которого будет производиться копирование в двумерный массив данных типа `REAL`.
`rArray`	`REAL`	Двумерный массив данных типа `REAL`, в который будут скопированы данные из файла.

```
(*rArray_wr – REAL [0..9,0..9], rArray_rr – REAL [0..9,0..9]*)
rArray_wr[0,0]:=10.0;
rArray_wr[4,9]:=20.0;
rArray_wr[9,9]:=30.0;
res_wr:=WriteReal2DimArray('RETAIN',rArray_wr);
res_rr:=ReadReal2DimArray('RETAIN',rArray_rr);

(*res_wr = 0, res_rr = 0, rArray_rr [0,0] = 10.0, rArray_rr [4,9] = 20.0,
rArray_rr [9,9] = 30.0*)
```

Листинг 13 – Использование функций `WriteReal2DimArray` и `ReadReal2DimArray`

4.8 Функция CompareWordArrays

Фунция `CompareWordArrays` выполняет сравнивание значений двух массивов `WORD`.

Параметры и возвращаемые значения функции `CompareWordArrays` приведены в Табл. 40, пример использования функции приведён в Листинге 14.

Табл. 40 – Параметры и возвращаемые значения функции `CompareWordArrays`

`res := CompareWordArrays(wArray_1, wArray_2, result);`		
Имя параметра	Тип параметра	Описание
`res`	`DINT`	Число пар сравниваемых элементов массивов.
`wArray_1`	`WORD`	Первый сравниваемый массив данных типа `WORD`.
`wArray_2`	`WORD`	Второй сравниваемый массив данных типа `WORD`.
`result`	`BOOL`	Массив данных типа `BOOL`,в который записываются результаты сравнений двух массивов данных типа `WORD`.

Сравнение двух массивов данных начинается с первого элемента. Результат сравнения записывается в соответствующий элемент массива данных `result`, если элементы массивов данных типа `WORD` равны, в соответствующий элемент массива данных `result` запишется значение `TRUE`. Сравнивается минимально общее число элементов во всех трёх массивах.

```
(*wArray_1 – WORD [1..5], wArray_1 – WORD [20..40], result – BOOL [0..99]*)
wArray_1[1]:=1;
wArray_2[20]:=1;
wArray_1[2]:=2;
wArray_2[21]:=2;
wArray_1[3]:=3;
wArray_2[22]:=10;
wArray_1[4]:=4;
wArray_2[23]:=4;
wArray_1[5]:=5;
wArray_2[24]:=5;
res := CompareWordArrays(wArray_1, wArray_2, result);
(*res = 5, result[0] = TRUE, result[1] = TRUE, result[2] = FALSE, result[3] =
TRUE, result[4] = TRUE*)
```

Листинг 14 – Использование функции `CompareWordArrays`

4.9 Функция CRC16ForModbusRegs

Функция `CRC16ForModbusRegs` выполняет считывание контрольной суммы CRC16 для подмножества элементов типа `WORD` массива данных типа `ModbusREGs`.

Параметры и возвращаемые значения функции `CRC16ForModbusRegs` приведены в Табл. 41, пример использования функции приведён в Листинге 15.

Табл. 41 – Параметры и возвращаемые значения функции `CRC16ForModbusRegs`

`res := CRC16ForModbusRegs(MB_TCP_sl4k_HR, row, column, count);`		
Имя параметра	Тип параметра	Описание
`res`	`UINT`	Значение контрольной суммы CRC16. Если массив данных `MB_TCP_sl4k_HR` не одномерный или индексы row или column лежат за пределами массива данных функция вернет значение 0xffff.
`MB_TCP_sl4k_HR`	`ModbusREGs`	Массив данных типа `ModbusREGs` для подмножества элементов типа `WORD` которого будет расчет контрольная сумма CRC16.
`row`	`DINT`	Первый индекс смещение в массиве `MB_TCP_sl4k_HR`, начиная с которого будет выполняться расчет контрольной суммы CRC16.
`column`	`DINT`	Второй индекс смещение в массиве `MB_TCP_sl4k_HR`, начиная с которого будет выполняться расчет контрольной суммы CRC16.
`count`	`DINT`	Число элементов типа `WORD` массива данных типа `ModbusREGs` для которых будет рассчитана контрольная сумма CRC16.

```
(*MB_TCP_sl4k_HR – ModbusREGs [1..4]*)
res := CRC16ForModbusRegs (MB_TCP_sl4k_HR, 0, 0, 4096);
(*res = 65535*)
```

Листинг 15 – Использование функции `CRC16ForModbusRegs`

4.10 Функция CRC16ForWords

Функция `CRC16ForWords` выполняет расчет контрольной суммы CRC16 для подмножества элементов массива данных типа `WORD`.

Параметры и возвращаемые значения функции `CRC16ForWords` приведены в Табл. 42, пример использования функции приведён в Листинге 16.

Табл. 42 – Параметры и возвращаемые значения функции `CRC16ForWords`

`res := CRC16ForWords (wArray, count);`		
Имя параметра	Тип параметра	Описание
`res`	`UINT`	Значение контрольной суммы CRC16. Если массив данных `wArray` не одномерный или нулевой размерности функция вернет значение 0xffff.
`wArray`	`WORD`	Массив данных типа `WORD` для *подмножества* элементов которого будет рассчитываться контрольная сумма CRC16.
`count`	`DINT`	Число элементов массива данных типа `WORD` для которых будет рассчитана контрольная сумма CRC16.

```
(*wArray – WORD [1..10]*)
res := CRC16ForWords (wArray, 5);
(*res = 1904*)
```

Листинг 16 – Использование функции `CRC16ForWords`

4.11 Функция InitRS485ModuleModbus

Функция `InitRS485ModuleModbus` выполняет передачу в указанный коммуникационный модуль `rs485` команд для работы в режиме «ведущий», а также выполняет инициализацию коммуникационного модуля `rs485` при старте программы пользователя.

Параметры, возвращаемые значения функции `InitRS485ModuleModbus`, а также примеры работы с ней приведены в п. 3.13.

4.12 Функция SendMessage

Функция `SendMessage` устарела и не рекомендуется к использованию.

4.13 Функция SetDateTime

Функция `SetDateTime` выполняет установку даты и времени для модуля CPU. Параметры даты и времени указываются и применяются без учёта часового пояса. Перед применением функции необходимо отключить NTP-сервер с помощью плагина MK500 IODevice (Рис. 102), иначе значения даты и времени скорректируются в соответствии с NTP-сервером.

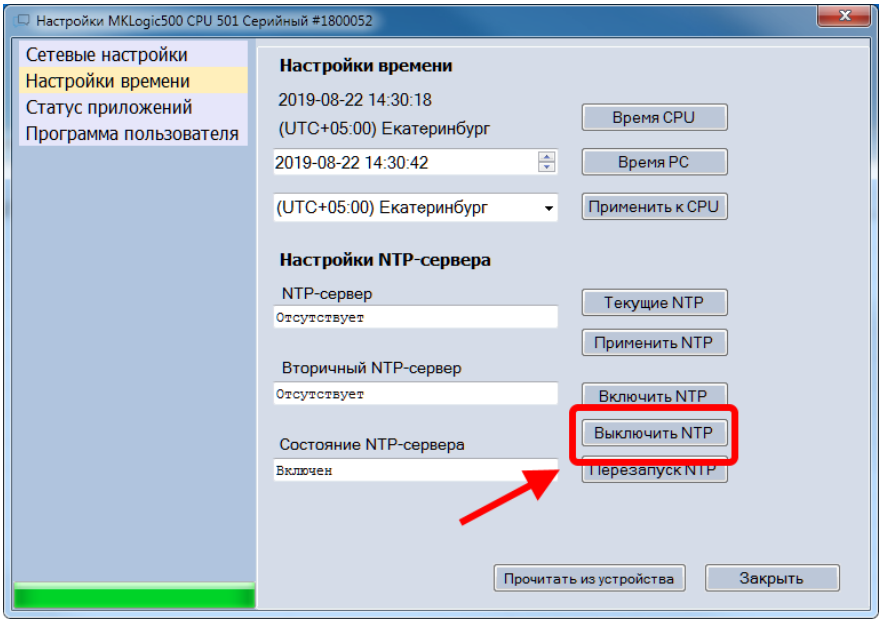


Рис. 102 – Окно настроек времени модуля CPU

Параметры и возвращаемые значения функции `SetDateTime` приведены в Табл. 43, пример использования функции приведён в Листинге 17.

Табл. 43 – Параметры и возвращаемые значения функции `SetDateTime`

`res := SetDateTime (year, month, day, hour, minute, second);`		
Имя параметра	Тип параметра	Описание
`res`	`BOOL`	Результат выполнения функции. TRUE - удалось применить время, FALSE - не удалось применить время, в том числе и по причине некорректно введённых входных значений.
`year`	`DINT`	Год
`month`	`DINT`	Месяц
`day`	`DINT`	День
`hour`	`DINT`	Часы
`minute`	`DINT`	Минуты
`second`	`DINT`	Секунды

```
res := SetDateTime(2019, 1, 1, 0, 0, 0);
(*res = TRUE*)
```

Листинг 17 – Использование функции `SetDateTime`

4.14 Функция UdpMessage

Функция `UdpMessage` выполняет отправку отладочных сообщений в специальную программу-приёмник сообщений udp_debug (на 20.08.2019 актуальная версия 0.4) из текста программы пользователя. Метка времени CPU автоматически добавляется к тексту сообщения.

Параметры и возвращаемые значения функции `UdpMessage` приведены в Табл. 44, пример использования функции приведён в Листинге 18.

Табл. 44 – Параметры и возвращаемые значения функции `UdpMessage`

`res := UdpMessage(udp_address, message);`		
Имя параметра	Тип параметра	Описание
`res`	`BOOL`	Результат выполнения функции. TRUE - удалось отправить сообщение, FALSE - не удалось отправить сообщение (некорректный UDP-адрес, либо ошибка функции отправки сообщения).
`udp_address`	`STRING`	UDP-адрес компьютера, на котором запущена программа udp_debug, принимающая сообщения.
`message`	`STRING`	Текст сообщения

```
res := UdpMessage('10.155.27.1','error');
(*res = TRUE*)
```

Листинг 18 – Использование функции `UdpMessage`

4.15 Функция SwapActiveCPU

Функция `SwapActiveCPU` выполняет рестарт среды исполнения ISaGRAF для проектов с поддержкой Failover на модуле CPU в режиме «Running» при наличии рабочего модуля CPU в режиме «Standby» (Рис. 33). В результате выполнения этой функции меняются ролями модули CPU: резервный модуль CPU переходит в режим «Running», а перезапущенный модуль после старта среды исполнения переходит в режим «Standby».

Ограничением функции `SwapActiveCPU` является то, что на время перезапуска модуля CPU проект работает без резерва, что может быть недопустимо по условиям эксплуатации.

Параметры и возвращаемые значения функции `SwapActiveCPU` приведены в Табл. 45, пример использования функции приведён в Листинге 19.

```
(*Все проверки на допустимость вызова выполняются внутри функции*)
res := SwapActiveCPU ();
```

Листинг 19 – Использование функций `SwapActiveCPU`

Табл. 45 – Параметры и возвращаемые значения функции `SwapActiveCPU`

`res := SwapActiveCPU ();`		
Имя параметра	Тип параметра	Описание
`res`	`INT`	Результат выполнения: 1 – выполняется рестарт, 0 – команда проигнорирована, -1 или -2 – системная ошибка, безопасное выполнение рестарта невозможно

4.16 Функции GetBitFromDInt, GetBitFromDWord и GetBitFromWord

Функции `GetBitFromDInt`, `GetBitFromDWord` и `GetBitFromWord` выполняют извлечение бита из переменной типа `DINT`, `DWORD` и `WORD`.

Параметры и возвращаемые значения функций `GetBitFromDInt`, `GetBitFromDWord` и `GetBitFromWord` приведены в Табл. 46, Табл. 47 и Табл. 48 соответственно, пример использования функций приведён в Листинге 20.

Табл. 46 – Параметры и возвращаемые значения функции `GetBitFromDInt`

`ResBitDINT := GetBitFromDInt(InBitDINT,NoBitDINT);`		
Имя параметра	Тип параметра	Описание
`ResBitDINT`	`BOOL`	Извлеченный бит
`InBitDINT`	`DINT`	Переменная, из которой извлекается бит
`NoBitDINT`	`DINT`	Номер бита (от 0 до 31)

Табл. 47 – Параметры и возвращаемые значения функции `GetBitFromDWord`

`ResBitDWORD := GetBitFromDWord(InBitDWORD,NoBitDWORD);`		
Имя параметра	Тип параметра	Описание
`ResBitDWORD`	`BOOL`	Извлечённый бит
`InBitDWORD`	`DWORD`	Переменная, из которой извлекается бит
`NoBitDWORD`	`DINT`	Номер бита (от 0 до 31)

Табл. 48 – Параметры и возвращаемые значения функции `GetBitFromWord`

`ResBitWORD := GetBitFromWord(InBitWORD,NoBitWORD);`		
Имя параметра	Тип параметра	Описание
`ResBitWORD`	`BOOL`	Извлечённый бит
`InBitWORD`	`WORD`	Переменная, из которой извлекается бит
`NoBitWORD`	`DINT`	Номер бита (от 0 до 15)

```
(*InBitDINT – DINT, NoBitDINT – DINT *)
InBitDINT:= 1081410080;
NoBitDINT:=5;
ResBitDINT := GetBitFromDInt(InBitDINT,NoBitDINT);
(*ResBitDINT = 1*)

(*InBitDWORD – DWORD, NoBitDWORD – DINT *)
InBitDWORD:= 1081410080;
NoBitDWORD:=5;
ResBitDWORD:=GetBitFromDWord(InBitDWORD,NoBitDWORD);
(*ResBitDWORD = 1*)

(*InBitWORD – WORD, NoBitWORD – DINT *)
InBitWORD:= 33312;
NoBitWORD:=5;
ResBitWORD:=GetBitFromWord(InBitWORD,NoBitWORD);
(*ResBitWORD = 1*)
```

Листинг 20 – Использование функций `GetBitFromDInt`, `GetBitFromDWord` и `GetBitFromWord`

4.17 Функции GetByteFromDWord и GetByteFromWord

Функции `GetByteFromDWord` и `GetByteFromWord` выполняют извлечение байта из переменной типа `DWORD` и `WORD`.

Параметры и возвращаемые значения функций `GetByteFromDWord` и `GetByteFromWord` приведены в Табл. 49 и Табл. 50 соответственно, пример использования функций приведён в Листинге 21.

Табл. 49 – Параметры и возвращаемые значения функции `GetByteFromDInt`

`ResByteDWORD:=GetByteFromDWord(InByteDWORD,NoByteDWORD);`		
Имя параметра	Тип параметра	Описание
`ResByteDWORD`	`BYTE`	Извлечённый байт
`InByteDWORD`	`DWORD`	Переменная, из которой извлекается байт
`NoByteDWORD`	`DINT`	Номер байта (от 0 до 3)

Табл. 50 – Параметры и возвращаемые значения функции `GetByteFromWord`

`ResByteWORD:=GetByteFromWord(InByteWORD,NoByteWORD);`		
Имя параметра	Тип параметра	Описание
`ResByteDWORD`	`BYTE`	Извлечённый байт
`InByteDWORD`	`WORD`	Переменная, из которой извлекается байт
`NoByteDWORD`	`DINT`	Номер байта (от 0 до 1)

```
(*InByteDWORD – DWORD, NoByteDWORD – DINT *)
InByteDWORD:= 11141120;
NoByteDWORD:=2;
ResByteDWORD:=GetByteFromDWord(InByteDWORD,NoByteDWORD);
(*ResByteDWORD = 170*)

(*InByteWORD – WORD, NoByteWORD – DINT *)
InByteWORD:= 43520;
NoByteWORD:=1;
ResByteWORD:=GetByteFromWord(InByteWORD,NoByteWORD);
(*ResByteWORD = 170*)
```

Листинг 21 – Использование функций GetByteFromDWord и GetByteFromWord

4.18 Функция GetWordFromDWord

Функция GetWordFromDWord выполняет извлечение слова из переменной типа DWORD.

Параметры и возвращаемые значения функции GetWordFromDWord приведены в Табл. 51, пример использования функций приведён в Листинге 22.

Табл. 51 – Параметры и возвращаемые значения функции GetWordFromDWord

ResWordDWORD:=GetWordFromDWord(InWordDWORD,NoWordDWORD);		
Имя параметра	Тип параметра	Описание
ResWordDWORD	WORD	Извлеченное слово
InWordDWORD	DWORD	Переменная, из которой извлекается слово
NoWordDWORD	DINT	Номер слова (от 0 до 1)

```
(*InWordDWORD – DWORD, NoWordDWORD – DINT *)
InWordDWORD:= 2863267840;
NoWordDWORD:=1;
ResWordDWORD:=GetWordFromDWord(InWordDWORD,NoWordDWORD);
(*ResWordDWORD = 43690*)
```

Листинг 22 – Использование функции GetWordFromDWord

4.19 Функции SetBitToDInt, SetBitToDWord и SetBitToWord

Функции SetBitToDInt, SetBitToDWord и SetBitToWord выполняют замену указанного бита в переменной типа DINT, DWORD и WORD.

Параметры и возвращаемые значения функций SetBitToDInt, SetBitToDWord и SetBitToWord приведены в Табл. 52, Табл. 53 и Табл. 54 соответственно, пример использования функций приведён в Листинге 23.

Табл. 52 – Параметры и возвращаемые значения функции `SetBitToDInt`

`SResBitDINT:=SetBitToDInt(InSBitDINT,NoSBitDINT,SrcSBitDINT);`		
Имя параметра	Тип параметра	Описание
`SResBitDINT`	`DINT`	Измененная переменная с заменой указанного бита, либо исходная переменная при некорректном выборе бита
`InSBitDINT`	`BOOL`	Значение бита
`NoSBitDINT`	`DINT`	Номер бита (от 0 до 31)
`SrcSBitDINT`	`DINT`	Исходная переменная

Табл. 53 – Параметры и возвращаемые значения функции `SetBitToDWord`

`SResBitDWORD:=SetBitToDWord(InSBitDWORD,NoSBitDWORD,SrcSBitDWORD);`		
Имя параметра	Тип параметра	Описание
`SResBitDWORD`	`DWORD`	Измененная переменная с заменой указанного бита, либо исходная переменная при некорректном выборе бита
`InSBitDWORD`	`BOOL`	Значение бита
`NoSBitDWORD`	`DINT`	Номер бита (от 0 до 31)
`SrcSBitDWORD`	`DWORD`	Исходная переменная

Табл. 54 – Параметры и возвращаемые значения функции `SetBitToWord`

`SResBitWORD:=SetBitToWord(InSBitWORD,NoSBitWORD,SrcSBitWORD);`		
Имя параметра	Тип параметра	Описание
`SResBitWORD`	`WORD`	Измененная переменная с заменой указанного бита, либо исходная переменная при некорректном выборе бита
`InSBitWORD`	`BOOL`	Значение бита
`NoSBitWORD`	`DINT`	Номер бита (от 0 до 15)
`SrcSBitWORD`	`WORD`	Исходная переменная

```
(*InSBitDINT – BOOL, NoSBitDINT – DINT, SrcSBitDINT - DINT*)
InSBitDINT:= TRUE;
NoSBitDINT:=8;
SrcSBitDINT:=65279;
SResBitDINT:=SetBitToDInt(InSBitDINT,NoSBitDINT,SrcSBitDINT);
(*SResBitDINT = 65535*)

(*InSBitDWORD – BOOL, NoSBitDWORD – DINT, SrcSBitDWORD - DWORD*)
InSBitDWORD:= TRUE;
NoSBitDWORD:=8;
SrcSBitDWORD:=65279;
SResBitDWORD:=SetBitToDWORD (InSBitDWORD,NoSBitDWORD,SrcSBitDWORD);
(*SResBitDWORD = 65535*)

(*InSBitWORD – BOOL, NoSBitWORD – DINT, SrcSBitWORD - WORD*)
```

```
InSBitWORD:= TRUE;
NoSBitWORD:=8;
SrcSBitWORD:=65279;
SResBitWORD:=SetBitToWORD (InSBitWORD,NoSBitWORD,SrcSBitWORD);
(*SResBitWORD = 65535*)
```

Листинг 23 – Использование функций `SetBitToDInt`, `SetBitToDWord` и `SetBitToWord`

4.20 Функции SetByteToDWord и SetByteToWord

Функции `SetByteToDWord` и `SetByteToWord` выполняют замену указанного байта в переменной типа `DWORD` и `WORD`.

Параметры и возвращаемые значения функций `SetByteToDWord` и `SetByteToWord` приведены в Табл. 55 и Табл. 56 соответственно, пример использования функций приведён в Листинге 24.

Табл. 55 – Параметры и возвращаемые значения функции `SetByteToDWord`

`SResByteDWORD:=SetByteToDWord(InSByteDWORD,NoSByteDWORD,SrcSByteDWORD);`		
Имя параметра	Тип параметра	Описание
`SResByteDWORD`	`DWORD`	Измененная переменная с заменой указанного байта, либо исходная переменная при некорректном выборе байта
`InSByteDWORD`	`BYTE`	Значение байта
`NoSByteDWORD`	`DINT`	Номер байта (от 0 до 3)
`SrcSByteDWORD`	`DWORD`	Исходная переменная

Табл. 56 – Параметры и возвращаемые значения функции `SetByteToWord`

`SResByteWORD:=SetByteToWord(InSByteWORD,NoSByteWORD,SrcSByteWORD);`		
Имя параметра	Тип параметра	Описание
`SResByteWORD`	`WORD`	Измененная переменная с заменой указанного байта, либо исходная переменная при некорректном выборе байта
`InSByteWORD`	`BYTE`	Значение байта
`NoSByteWORD`	`DINT`	Номер байта (от 0 до 1)
`SrcSByteWORD`	`WORD`	Исходная переменная

```
(*InSByteDWORD – BYTE, NoSByteDWORD – DINT, SrcSByteDWORD - DWORD*)
InSByteDWORD:= 225;
NoSByteDWORD:=1;
SrcSByteDWORD:=65535;
SResByteDWORD:=SetByteToDWord(InSByteDWORD,NoSByteDWORD,SrcSByteDWORD);
(*SResByteDWORD = 57855*)

(*InSByteWORD – BYTE, NoSByteWORD – DINT, SrcSByteWORD - WORD*)
InSByteWORD:= 225;
```

```
NoSByteWORD:=1;
SrcSByteWORD:=65535;
SResByteWORD:=SetByteToWord(InSByteWORD,NoSByteWORD,SrcSByteWORD);
(*SResByteWORD = 57855*)
```

Листинг 24 – Использование функций `SetByteToDWord` и `SetByteToWord`

4.21 Функция SetWordToDWord

Функция `SetWordToDWord` выполняют замену указанного слова в переменной типа `DWORD`.

Параметры и возвращаемые значения функции `SetWordToDWord` приведены в Табл. 57, пример использования функции приведён в Листинге 25.

Табл. 57 – Параметры и возвращаемые значения функции `SetWordToDWord`

`SResWordDWORD:=SetWordToDWord(InSWordDWORD,NoSWordDWORD,SrcSWordDWORD);`		
Имя параметра	Тип параметра	Описание
`SResWordDWORD`	`DWORD`	Изменённая переменная с заменой указанного слова, либо исходная переменная при некорректном выборе слова
`InSWordDWORD`	`WORD`	Значение слова
`NoSWordDWORD`	`DINT`	Номер слова (от 0 до 1)
`SrcSWordDWORD`	`DWORD`	Исходная переменная

```
(*InSWordDWORD – WORD, NoSWordDWORD – DINT, SrcSWordDWORD - DWORD*)
InSWordDWORD:= 43690;
NoSWordDWORD:=1;
SrcSWordDWORD:= 4294967295;
SResWordDWORD:=SetWordToDWord(InSWordDWORD,NoSWordDWORD,SrcSWordDWORD);
(*SResWordDWORD = 2863333375*)
```

Листинг 25 – Использование функции `SetWordToDWord`

4.22 Функция SynchronizeFtpFiles

Функция `SynchronizeFtpFiles` используется для синхронизации пользовательских данных в папке FTP. Функция `SynchronizeFtpFiles` выполняет копирование пользовательских данных из папки FTP на CPU в режиме «Running» и размещает их в папке FTP на CPU в режиме «Standby».

Параметры и возвращаемые значения функции `SynchronizeFtpFiles` приведены в Табл. 58, пример использования функции приведён в Листинге 26.

Табл. 58 – Параметры и возвращаемые значения функции `SynchronizeFtpFiles`

`res := SynchronizeFtpFiles();`		
Имя параметра	Тип параметра	Описание
`res`	`INT`	Результат выполнения: 0 – синхронизация не выполнялась, 1 – синхронизация выполнена

```
res := SynchronizeFtpFiles();
```

Листинг 26 – Использование функций `SwapActiveCPU`

4.23 Функция PredictByDerivative

Функция `PredictByDerivative` выполняет прогнозирование достижения определённого значения в течении заданного времени.

Параметры и возвращаемые значения функции `PredictByDerivative` приведены в Табл. 59, пример использования функции приведён в Листинге 27.

Табл. 59 – Параметры и возвращаемые значения функции `PredictByDerivative`

`res:=PredictByDerivative(curVal,prevVal,vSet,tSet,direct,deltaT,bCalc);`		
Имя параметра	Тип параметра	Описание
`res`	`INT`	Результат выполнения: 0 – значение не будет достигнуто, 1 – значение будет достигнуто, -1 – неверно указано одно из значений времени (значение меньше либо равно нулю), -2 – значение направления не совпадает со знаком отношения текущего и ожидаемого значения (например, curVal > vSet и direct = TRUE), -3 – параметр bCalc равен FALSE0
`curVal`	`REAL`	Текущее значение
`prevVal`	`REAL`	Предыдущее значение
`vSet`	`REAL`	Ожидаемое значение
`tSet`	`REAL`	Ожидаемое время для достижения ожидаемого значения
`direct`	`BOOL`	Направление ожидаемого значения, TRUE – значение вверх, FALSE – значение вниз
`deltaT`	`REAL`	Время, через которое будут происходить замеры текущего значения
`bCalc`	`BOOL`	Вычислять, будет ли достигнуто ожидаемое значение за определённое время (TRYE) или нет (FALSE)

```
(*curVal - REAL, prevVal - REAL, vSet - REAL, tSet - REAL, direct - BOOL,
deltaT - REAL, bCalc - BOOL*)
curVal := 10.0;
prevVal := 9.0;
vSet := 200.0;
tSet := 300.0;
direct := TRUE;
deltaT := 1;
bCalc := TRUE;
res := PredictByDerivative(curVal,prevVal,vSet,tSet,direct,deltaT,bCalc);
(*res = 1*)
```

Листинг 27 – Использование функций `PredictByDerivative`

4.24 Функция GetDevInfo

Функция `GetDevInfo` выполняет поиск устройство с указанным openPowerLink ID, номером стойки, позиции модуля и копирует диагностические данные этого устройства в передаваемую переменную типа `IOModuleDiag`.

Параметры и возвращаемые значения функции `GetDevInfo` приведены в Табл. 60, пример использования функции приведён в Листинге 28.

Табл. 60 – Параметры и возвращаемые значения функции `GetDevInfo`

`res:= GetDevInfo(opwl_id, rack, slot, IOModulDiagIn)`		
Имя параметра	Тип параметра	Описание
`res`	`INT`	Результат выполнения: 0 – не удалось прочитать данные, с указанными параметрами, 1 – удалось прочитать данные.
`opwl_id`	`WORD`	openPowerLink ID модуля
`rack`	`WORD`	Номер стойки
`slot`	`WORD`	Позиция модуля в стойке
`IOModulDiagIn`	`IOModuleDiag`	Структура куда будет копироваться результат (переменная-структура подробно описана в п. 3.1.6). Поля структуры будут иметь нулевое значение, если не удалось найти модуль с указанными параметрами

```
(*opwl_id - WORD, rack - WORD, slot - WORD, IOModulDiagIn - IOModuleDiag*)
opwl_id := 240;
rack := 1;
slot := 1;
res:= GetDevInfo(opwl_id, rack, slot, IOModulDiagIn)
(*res = 1*)
```

Листинг 28 – Использование функций `GetDevInfo`

4.25 Функция GetFastModuleState

Функция `GetFastModuleState` выполняет сбор диагностики по всем модулям в конфигурации проекта и копирует эти данные в передаваемый массив. Элементы передаваемого массива должны иметь тип FastModuleState.

Следует обратить внимание, что если входной массив имеет размерность меньше чем количество модулей в стойке, то диагностика будет собрана не полностью; если элементов в массиве больше, то оставшиеся значения массива будут нулевыми.

Параметры и возвращаемые значения функции `GetFastModuleState` приведены в Табл. 61, пример использования функции приведён в Листинге 29.

Табл. 61 – Параметры и возвращаемые значения функции `GetFastModuleState`

`res := GetFastModuleState (resArr)`		
Имя параметра	Тип параметра	Описание
`res`	`INT`	Количество модулей, для которых были почитаны диагностические данные
`resArr`	`FastModuleState[a..b]`	Массив структур, куда будет копироваться результат. Каждый элемент массива содержит краткую диагностику по модулю. Поля структуры: pwl_id – идентификатор для OpenPowerlink, rack – номер стойки, slot – позиция модуля в стойке, state – статус диагностики (-1 – не соответствует, 0 – отсутствует, 1 – соответствует, 2 – виртуальный).

```
(*resArr – FastModuleState[0..2];
res := GetFastModuleState (resArr)
(*res = 3*)
```

Листинг 29 – Использование функций `GetFastModuleState`

4.26 Функция FTPEnable

Функция `FTPEnable` выполняет запуск либо остановку встроенного в модуль CPU FTP-сервера.

Работа функции `FTPEnable` имеет две особенности, которые следует учитывать:

1. Остановка и запуск FTP-сервера выполняется в течение определённого времени, обычно 1-2 секунды. Текущее состояние FTP-сервера следует контролировать через переменную `Services.FTPStatus` устройства `system_info` (п. 3.7.7).

2. Остановка и запуск FTP-сервера выполняется на обоих CPU резервированной пары, поэтому функция `FTPEnable` всегда возвращает `TRUE`.

Параметры и возвращаемые значения функции `FTPEnable` приведены в Табл. 62, пример использования функции приведён в Листинге 30.

Табл. 62 – Параметры и возвращаемые значения функции `FTPEnable`

`res := FTPEnable (cmd)`		
Имя параметра	Тип параметра	Описание
`res`	`BOOL`	Результат выполнения функции: Всегда TRUE.
`cmd`	`BOOL`	TRUE – команда запустить FTP-сервер; FALSE – команда остановить FTP-сервер.

```
(* system_info.Services.FTPStatus = 1 *)
res := FTPEnable (FALSE);
(* res = TRUE, system_info.Services.FTPStatus = -2 *)

(* через 1-2 секунды: system_info.Services.FTPStatus = 0 *)

res := FTPEnable (FALSE);
(* res = TRUE, system_info.Services.FTPStatus = 0 *)
res := FTPEnable (TRUE);
(* res = TRUE, system_info.Services.FTPStatus = -1 *)

(* через 1-2 секунды: system_info.Services.FTPStatus = 1 *)
```

Листинг 30 – Использование функций `FTPEnable`

Глава 5 Рекомендации по написанию программ пользователя

5.1 Оптимизация времени выполнения программ пользователя

Приведённые в данном разделе рекомендации одинаково применимы в системах с поддержкой резервирования и в системах без резервирования.

5.1.1 Уменьшение числа параметров функций и функциональных блоков

Каждый входной параметр функции либо функционального блока требует дополнительного времени на обработку. Уменьшение количества входных переменных приводит к ускорению работы функции.

5.1.2 Влияние параметра ресурса «Function Internal State Enabled»

Если параметр ресурса «Function Internal State Enabled» (п. 2.5.2) установлен в FALSE, то при каждом вызове функции все внутренние переменные будут принудительно устанавливаться в начальные значения (в 0 или FALSE если начального значения нет). В этом случае уменьшение количества внутренних переменных ведёт к ускорению работы функции

Если параметр ресурса «Function Internal State Enabled» установлен в TRUE, то при каждом вызове функции все внутренние переменные будут сохранять своё значение с прошлого вызова. В этом случае число внутренних переменных не влияет на время выполнения функции.

5.1.3 Использование оператора WHILE вместо FOR

Внутренняя реализация цикла WHILE имеет в 2,5 раза меньше накладных расходов, чем реализация цикла FOR. Это ошибка в реализации среды исполнения ISaGRAF, и в последующих версиях она будет исправлена.

В среде исполнения ISaGRAF 6.5 рекомендуется использовать цикл WHILE вместо FOR, особенно для циклов с большой вложенностью либо в циклах с большим числом итераций (Листинг 31).

При использовании циклов WHILE следует соблюдать аккуратность, так как среда исполнения ISaGRAF не определяет зацикливание внутри WHILE при неправильной работе с условием и не прерывает его работу, что приводит к зависанию среды исполнения и перегрузке модуля CPU.

```
 (* 65 мс *)
FOR i := 1 TO 1000 DO
      FOR j := 1 TO 40 DO
            b := ANY_TO_REAL(i);
            c := ANY_TO_REAL(j);
            a := (1.1*b + c)*c/b*2.22;
      END_FOR;
END_FOR;

(* 45 мс *)
i := 0;
WHILE i < 1000 DO
      i := i+1;
      j := 0;
      WHILE j < 40 DO
            j := j+1;
            b := ANY_TO_REAL(i);
            c := ANY_TO_REAL(j);
            a := (1.1*b + c)*c/b*2.22;
      END_WHILE;
END_WHILE;
```

Листинг 31 – Использование оператора WHILE вместо FOR

5.1.4 Использование промежуточных переменных

Затраты времени на доступ к вложенным переменным растут пропорционально степени вложенности. Поэтому при циклической обработке вложенных массивов целесообразно использовать вспомогательные переменные вместо прямого обращения к переменной с использованием нескольких индексов (Листинг 32).

⚠ ВНИМАНИЕ

Если вложенным в массив элементом является массив либо структура большого размера, затраты на копирование в промежуточную переменную начинают превышать экономию от её использования. Данный совет следует применять осторожно.

```
(* regs_2d - массив переменных типа ModbusREGs
   regs – промежуточная переменная типа Words1k
   ModbusREGs - тип-структура из одной переменной regs типа Words1k
   Words1k - тип-массив из 1024 WORD *)

(* 145 мс *)
i := 1;
WHILE i <= 100 DO
     i := i+1;
     j := 1;
     WHILE j <= 1024 DO
          j := j+1;
          regs_2d[i].regs[j] := ANY_TO_WORD(i+j);
     END_WHILE;
```

```
        END_WHILE;

        (* 96 мс *)
        i := 1;
        WHILE i <= 100 DO
            i := i+1;
            j := 1;
            regs := regs_2d[i].regs;
            WHILE j <= 1024 DO
                j := j+1;
                regs[j] := ANY_TO_WORD(i+j);
            END_WHILE;
            regs_2d[i].regs := regs;
END_WHILE;
```

Листинг 32 – Использование промежуточных переменных

5.2 Правила написания программ для процессорных модулей с поддержкой режима Failover

Работа программы пользователя в системах с поддержкой режима Failover имеет особенности, описанные в документе «Механизм обеспечения отказоустойчивости» (fvr_ISa6_ru).

Из этого документа следует, что несовпадение контрольных сумм оперативной памяти программы пользователя после окончания цикла выполнения на основном и на резервном модулях CPU влечёт выполнение синхронизации всей оперативной памяти в направлении от основного к резервному. Данная операция может занимать от нескольких десятков до нескольких сотен миллисекунд, почти наверняка приводит к нарушению заданного времени цикла и всегда приводит к повышению нагрузки на процессор в модулях CPU.

Следовательно, программы пользователя для систем с поддержкой режима Failover должны быть написаны так, чтобы минимизировать (в идеале свести к единственной операции синхронизации сразу после запуска) число синхронизаций. Далее перечислены рекомендации, нарушение которых приводит к незапланированному выполнению синхронизаций:

1. Обязательно привязывать все каналы ввода-вывода всех устройств ввода-вывода к пользовательским переменным. Наличие даже одного непривязанного канала ввода-вывода может привести к постоянным вызовам синхронизации при работе программы в системе с поддержкой Failover.
2. Избегать использования в программе (присвоения локальным переменным, сравнение и т.п.) системных переменных, их значения могут отличаться на различных модулях CPU.
3. Избегать использования функций и функциональных блоков, получающих данные не через каналы ввода-вывода, так как высока вероятность получения разных данных разными модулями CPU.
4. Избегать использования текущего времени в программе, например, полученного из функции `getCurrentTime`.
5. Избегать использования функциональных блоков для SFC-программ.

5.2.1 Особенности работы процессорных модулей изделия в режиме Failover

При настройке сетевых параметров в проекте с поддержкой Failover (п. 2.4.2) следует всегда включать режим Failover в окне «Failover Configuration», если в программе пользователя будет использоваться ModbusRTU либо Modbus\TCP в режиме «ведомый».

Это требование вызвано тем, что драйвер Modbus на резервном модуле CPU должен возвращать на все запросы мастера код ошибки 6 (Device Busy). При выключенном режиме Failover драйвер Modbus считает, что он всегда запускается на резервном модуле CPU (даже если модуль CPU единственный и работает в активном режиме) и отвечает на запросы мастера кодом ошибки 6.

Также следует иметь в виду, что при старте процессорных модулей в режиме Failover происходит сравнение сетевых настроек, в котором «побеждает» модуль CPU, стартовавший первым. Стартовавший вторым модуль CPU принимает его сетевые настройки (но как secondary, п. 2.3.2), а также программу пользователя. Это позволяет выполнять горячую замену неисправного модуля CPU новым без предварительной настройки его сетевых параметров.

Стартовавший первым модуль CPU при запуске программы пользователя в течение одной секунды мигает всей индикацией, стартовавший вторым модуль CPU – не мигает.

При одновременной подаче питания на стойку с модулями CPU невозможно предсказать, какой из модулей CPU запустится первым и применит свои сетевые настройки на второй модуль CPU. Следует избегать такой ситуации при работе с модулями CPU с разными сетевыми настройками, подавая питание на модули CPU либо вставляя их в стойку в необходимом порядке вручную.

5.2.2 Диагностика проблем при работе процессорных модулей с поддержкой режима Failover

Диагностика программы пользователя в системах с поддержкой режима Failover описана в документе «Механизм обеспечения отказоустойчивости» (fvr_ISa6_ru).

Диагностировать наличие постоянной синхронизации пользовательских данных удобнее всего по значению системной переменной __SYSVA_FO_DATASYNCCNT (как это сделать, описано в п. 2.8.2). Ей значение должно быть равна 0 или 1 и не должна увеличиваться. Постоянный рост значения этой переменной говорит о наличие проблемы, описанной в п. 5.2.

Также следует обратить внимание на значение системной переменной __SYSVA_FO_DATASYNCTIME. Её значение в ходе работы зависит от числа используемых переменных ввода-вывода и может достигать десятков миллисекунд при работе с большими устройствами Modbus или IEC-104. Но её значение также резко возрастает при синхронизации пользовательских данных и может служить признаком проблем, описанных в п. 5.2.

5.3 Часто встречающиеся проблемы и меры по их устранению

5.3.1 Проблемы при загрузке программы пользователя

Самой частой ошибкой при попытке загрузить программу пользователя в модуль CPU является несовпадение сетевых параметров проекта и модуля CPU, проблема с

сетевым подключением компьютера с установленной средой разработки ACP, либо невозможность подключиться к модулю CPU.

Признаком такой ошибки является сообщение об ошибке, приведённое на Рис. 103. При её возникновении следует проверить сетевое подключение компьютера с установленной средой разработки ACP, корректность сетевых настроек компьютера и модуля CPU (п. 2.3), сетевые настройки проекта (п. 2.4).

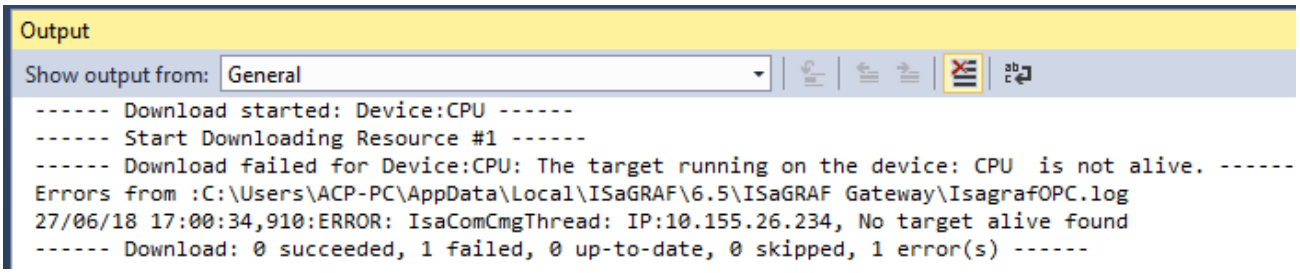


Рис. 103 – Пример сообщения об ошибке при проблемах загрузки программы пользователя

Загрузка программы пользователя, собранной с неподходящим для модуля CPU tdb-файлом (п. 2.5) заканчивается ошибкой, приведённой на Рис. 104. В случае её возникновения следует уточнить версии системного ПО используемого модуля CPU и его модификацию (с поддержкой режиме Failover или без, п. 2.3), и скорректировать проект.

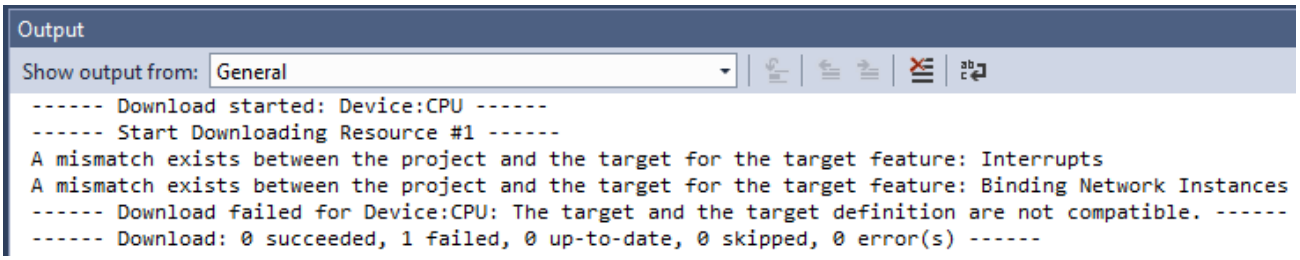


Рис. 104 – Пример сообщения об ошибке при несовпадении tdb-файла либо версии среды исполнения ISaGRAF

5.3.2 Зацикливание программы пользователя

Среда исполнения ISaGRAF не останавливает программу пользователя в случае её зацикливания (например, в случае ошибочно записанного вызова цикла типа WHILE). Зациклившаяся программа пользователя загружает процессор модуля CPU на 100% и делает невозможным заливку исправленной версии программы пользователя (команда записи возвращается с таймаутом).

Для решения этой проблемы следует выполнить следующие действия:

1. Подключиться к проблемному модулю CPU с помощью плагина MK500 IODevice (п. 2.3)
2. В окне плагина выбрать раздел «Программа пользователя» и нажать кнопку «Стереть программу пользователя».
3. Переключиться в раздел «Статус приложений» и через 15-20 секунд проверить состояние приложений, повторно нажав кнопку «Прочитать из CPU». Следует дождаться статуса приложений, показанного на Рис. 105. Также должны погаснуть светодиоды «Run» и «Act» на лицевой панели модуля CPU.

При зацикливании программы пользователя в проекте с поддержкой Failover следует выполнить эти операции над каждым из задействованных модулей CPU, предварительно разорвав между ними Datalink-соединение.

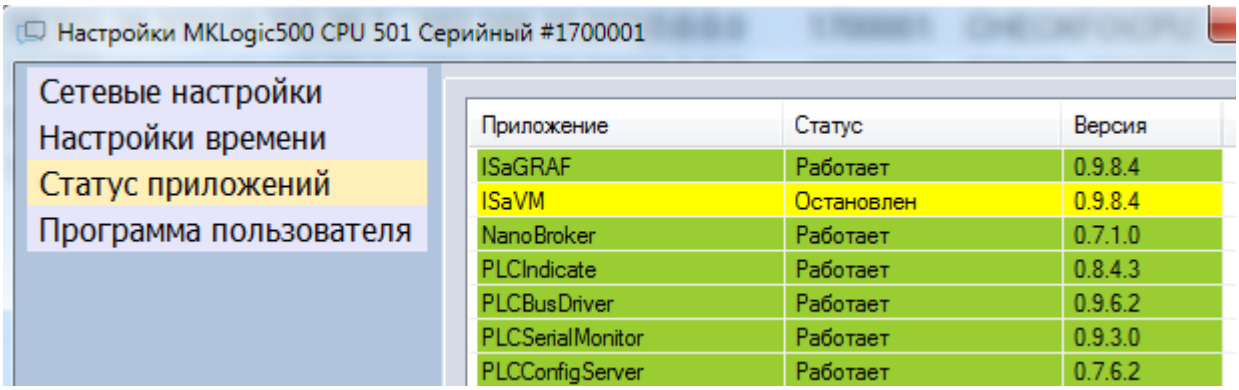


Рис. 105 – Состояние системного ПО модуля CPU со стёртой программой пользователя

5.3.3 Ошибки при сборке проекта

При сборке проекта часто возникает ошибка вида `<Имя_функции>: unexpected statement` при попытке использовать функцию, не присваивая переменной возвращаемое функцией значение. Пример возникновения подобной ошибки приведён в Листинге 33.

```
(*правильно res := SafeBoolArrayCopy(bArray1, bArray2);*)
SafeBoolArrayCopy(bArray1, bArray2);
(*При попытке скомпилировать будет ошибка:
SAFEBOOLARRAYCOPY: unexpected statement*)
```

Листинг 33 – Пример игнорирования возвращаемого значения фукнции

Также к интересным сообщениям об ошибках приводят пропуски замыкающих точек с запятой в строке, предшествующей строке с ключевым словом ST. В этом случае компилятор «склеивает» проблемную строку со следующей, и выдаёт ошибки, соответствующие отсутствию следующей строки, но не обязательно указывающие на пропуск точки с запятой. Пример подобного поведения приведён в Листинге 34.

```
(*в следующей строке пропущена точка с запятой*)
regs := regs_2d[i].regs
WHILE j <= 1024 DO
      j := j+1;
END_WHILE;
(*При попытке скомпилировать возникают следующие ошибки:
=:must precede an expression having the same type as the assignment variable
END_WHILE:expected
WHILE:unexpected statement*)
```

Листинг 34 – Пример пропуска точки с запятой перед ключевым словом ST

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Входящий № сопроводит. докум. и дата	Подп.	Дата
	изме-ненных	заменён-ных	новых	аннулиро-ванных					
1	-	все	-	-	-		-	Гайнцева	29.05.2020
2	-	все	-	-	-		-	Гайнцева	11.06.2020