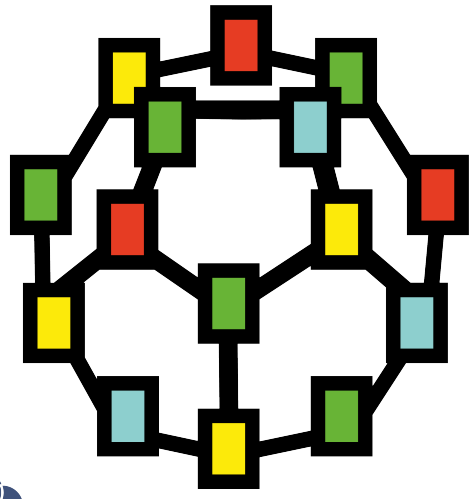
Архивирование и сохранение уставок



ПромАвтоматика

Руководство пользователя

06.2024
версия 1.1

Содержание

Используемые термины и сокращения

ОС – операционная система.

ПК – персональный компьютер.

ПЛК – программируемый логический контроллер.

FTP (File Transfer Protocol) — протокол прикладного уровня для передачи файлов по сети.

OPC UA (Open Platform Communications, Unified Architecture) – протокол для обмена данными с ПЛК и для управления ими.

SQL (**Structured Query Language**) – язык программирования для хранения и обработки информации в реляционной базе данных.

Введение

Настоящее руководство описывает настройку архивации и сохранения уставок для контроллеров ОВЕН, программируемых в среде **Полигон**. Подразумевается, что читатель обладает базовыми навыками работы с **Полигон**, поэтому общие вопросы (например, создание и загрузка проектов) в данном документе не рассматриваются – они подробно описаны в документах Руководство по программированию. Библиотека paCore и Быстрый старт.

Настройка архивации и сохранения уставок на диск контроллера или внешние накопители в среде **Полигон** осуществляется с помощью функциональных блоков из библиотеки ***paCore*** (раздел ***Сохранение данных***), а также с помощью библиотек ***profiLogger*** и ***profiLoggerLight***.

Примеры в документе актуальны для версии среды **Полигон** – **1917**, версий библиотек **paCore** – **979**, **profiLogger** – **62**, **profiLoggerLight** – **69** и выше.

1 Основные сведения по работе с файловой системой контроллера

1.1 Память контроллеров

Контроллеры ОВЕН имеют следующие независимые области памяти:

- энергонезависимая память (Flash);
- оперативная память (RAM);
- Retain-память (MRAM).

Программы, созданные в среде **Полигон**, задействуют две из них – Flash и RAM-память, а также позволяют работать с внешними накопителями USB-Flash/MicroSD.

Работа с файлами в большинстве случаев подразумевает работу с Flash-памятью. Flash-память имеет значительный, но ограниченный ресурс перезаписи – поэтому для архивации данных в большинстве случаев рекомендуется использовать внешние накопители (USB-Flash/MicroSD). Ресурс перезаписи внешних накопителей также ограничен, но их выход из строя не повлияет на работоспособность контроллера и накопители можно оперативно заменить. Информация об общем доступном объеме памяти приведена в руководстве по эксплуатации на соответствующий контроллер.

Информация о количестве свободной/занятой памяти доступна в web-конфигураторе контроллера в разделе ***Система/Точки монтирования***.

1.2 Требования к подключаемым накопителям (USB-Flash/MicroSD)

Поддерживаемый тип разделов – MBR (GPT не поддерживается). Методика определения стиля разделов доступна по ссылке.

Рекомендуется использовать накопители с одним разделом – тогда гарантируется монтирование по путям, указанным в разделе 1.3.

Поддерживаемые файловые системы накопителей – FAT16/FAT32 и ext4.

Перед началом работы рекомендуется отформатировать накопитель с помощью **HP USB Disk Storage Format Tool** (для ОС Windows) или любой другой утилитой для форматирования накопителей.

1.3 Пути к файлам и накопителям, работа с накопителями (210-SD-USB)

Пути в файловой системе контроллера к рабочей директории **Полигон** и пути монтирования внешних накопителей выглядят следующим образом:

Таблица 1.1 – Пути к директориям ПЛК210

Директория	Путь
Рабочая директория	/home/root
USB-Flash-накопитель	/mnt/ufs/media/sda1 (ссылка /sda1)
MicroSD-накопитель	/mnt/ufs/media/mmcblk1p1 (ссылка /mmcblk1p1)
Директория FTP-сервера	/mnt/ufs/home/ftp/in

При работе с виртуальным контроллером рабочая директория находится в папке на ПК, где расположен проект, имя папки – ***build_имя модуля_ОС***.

Для работы с внешними накопителями в среде **Полигон** предназначен блок ***210-SD-USB*** из библиотеки ***paOwenIO***.

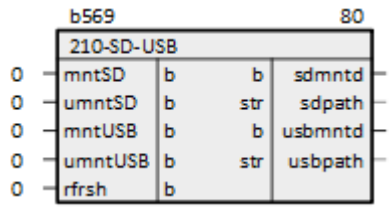


Рисунок 1.1 – Внешние накопители (210-SD-USB)

Блок **210-SD-USB** обеспечивает монтирование внешних накопителей (подключение/получение возможности работы с файлами) и их демонтаж (отключение/извлечение без потери данных).

Данный блок можно разместить только в ***Фоне***.

Логические входы блока реагируют при изменении значения с **0** на **1**.

Таблица 1.2 – Назначение входов и выходов 210-SD-USB

Элемент	Описание
Входы	
mntSD	Монтировать MicroSD-карту
umntSD	Размонтировать MicroSD-карту
mntUSB	Монтировать USB-накопитель
umntUSB	Размонтировать USB-накопитель
rfrsh	Обновление информации о статусах накопителей
Выходы	
sdmntd	Статус MicroSD-карты: **0** – карта отключена; **1** – карта подключена
sdpath	Путь к файлам MicroSD-карты: **Пустая строка** – накопитель отключен; **/mmcblk1p1** – ссылка на директорию монтирования накопителя
usbmntd	Статус USB-накопителя: **0** – накопитель отключен; **1** – накопитель подключен
usbpath	Путь к файлам USB-накопителя: **Пустая строка** – накопитель отключен; **/sda1** – ссылка на директорию монтирования накопителя

К входам блока ***210-SD-USB*** можно подключить сигналы от внешних кнопок или от панели оператора. Выходы статуса накопителя можно подключать к входам сброса ошибок блоков сохранения данных.

ПРИМЕЧАНИЕ
Информацию о размере свободной/занятой памяти на накопителе можно получить в программе пользователя с помощью блока ***DriveInfo*** из библиотеки ***paCore***.

1.4 Ограничения на имена файлов и каталогов в ОС Linux

Максимальная длина – **255** символов.

Символы кириллицы и символ **/** не поддерживаются.

Не рекомендуется использовать в названиях следующие символы: **! @ # $ & ~ % * ( ) [ ] { } ' " \ : ; > < ` пробел**

Регистр букв имеет значение: **Test.txt** и **test.txt** – это два разных файла.

1.5 Подключение к файловой системе контроллера

Для упрощения отладки программ, работающих с файлами, рекомендуется организовать подключение к файловой системе контроллера, чтобы иметь возможность просматривать и загружать файлы.

При работе на ПК с ОС Windows для этих целей рекомендуется использовать утилиту WinSCP. Утилита распространяется бесплатно и может быть загружена с сайта https://winscp.net/eng/download.php.

После запуска утилиты следует настроить соединение по протоколу **SFTP**, указав IP-адрес контроллера, имя пользователя – **root** и пароль (по умолчанию – **owen**, можно изменить в web-конфигураторе). Чтобы подключиться к контроллеру, следует нажать ***Войти***.

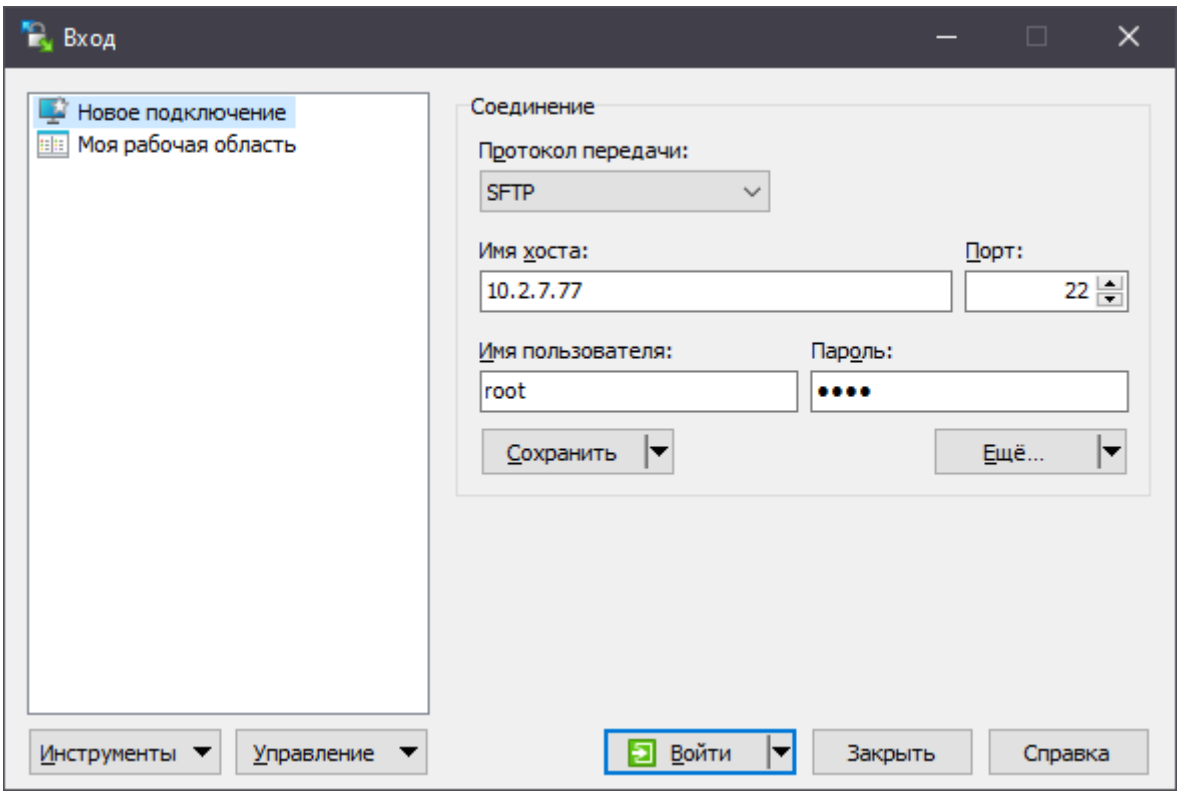


Рисунок 1.2 – Настройки подключения в WinSCP

При работе на ПК с ОС Linux можно воспользоваться утилитой Double Commander или любым другим файловым менеджером.

1.6 FTP-сервер контроллера

Контроллер может использоваться в режиме FTP-сервера. По умолчанию FTP-сервер контроллера запущен. Логин для доступа: **ftp**, пароль по умолчанию: **ftp** (может быть изменен в web-конфигураторе).

См. более подробную информацию в руководстве Краткое описание основных функций Web-интерфейса управления контроллеров.

Рабочая директория FTP-сервера по умолчанию (можно изменить в web-конфигураторе контроллера): **/mnt/ufs/home/ftp/in**.

2 Раздел Сохранение данных библиотеки paCore

Сохранение параметров на диске контроллера организуется с помощью блоков раздела ***Сохранение данных*** библиотеки ***paCore***.

Для добавления библиотеки в проект следует:

1. Перейти в меню ***Окна/Проекты***. В появившемся окне отобразится текущий проект и добавленные библиотеки.

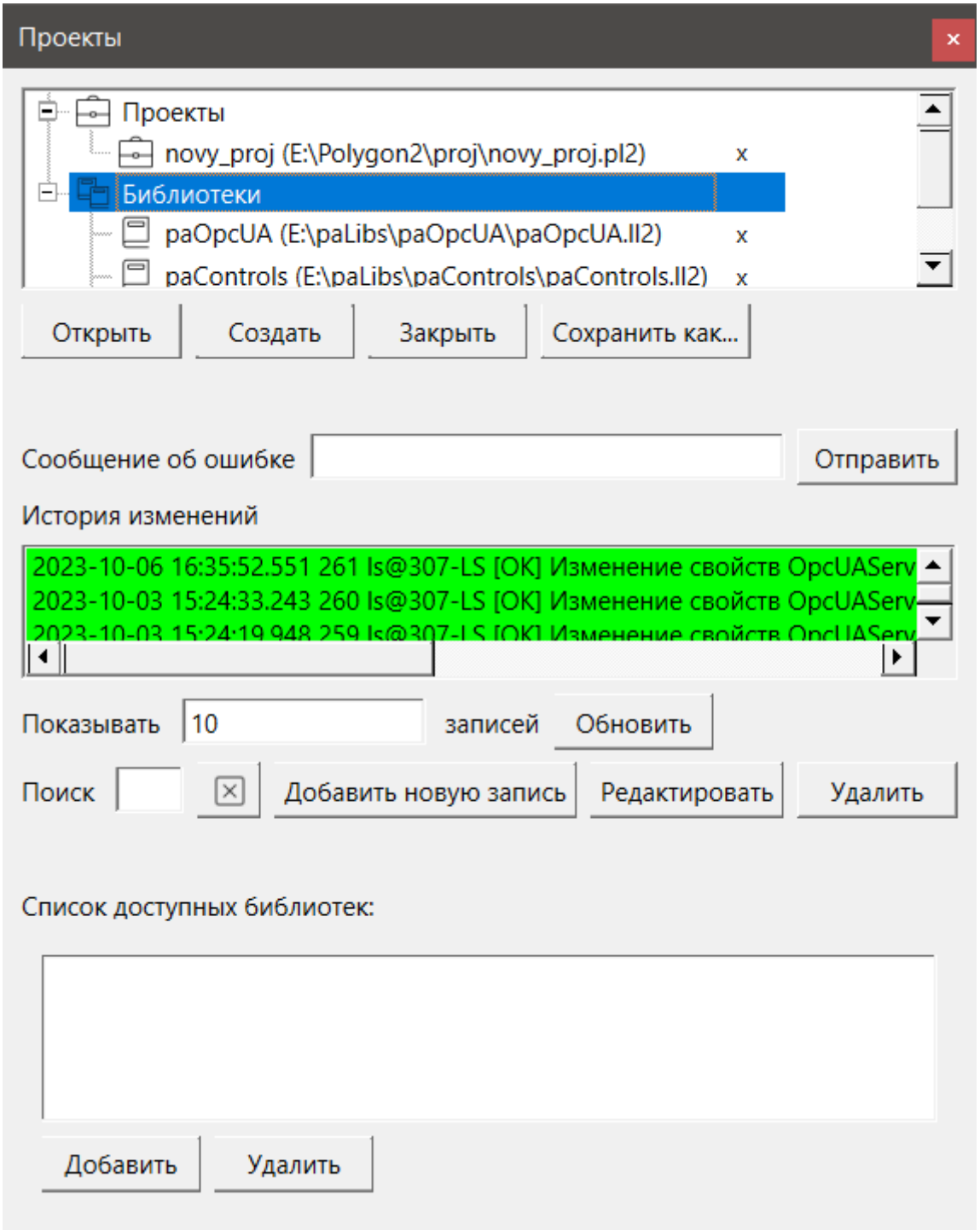


Рисунок 2.1 – Добавление библиотеки в проект

2. Для добавления библиотеки следует нажать кнопку ***Открыть*** и перейти в папку с файлами библиотеки, которую необходимо добавить. Затем в выпадающем списке выбрать тип файла ***Библиотека Полигон 2 (*.lI2)***.

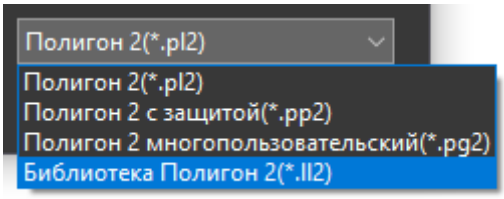


Рисунок 2.2 – Добавление библиотеки в проект

3. В окне появится файл библиотеки с расширением **.ll2**. Следует выбрать его и нажать открыть.

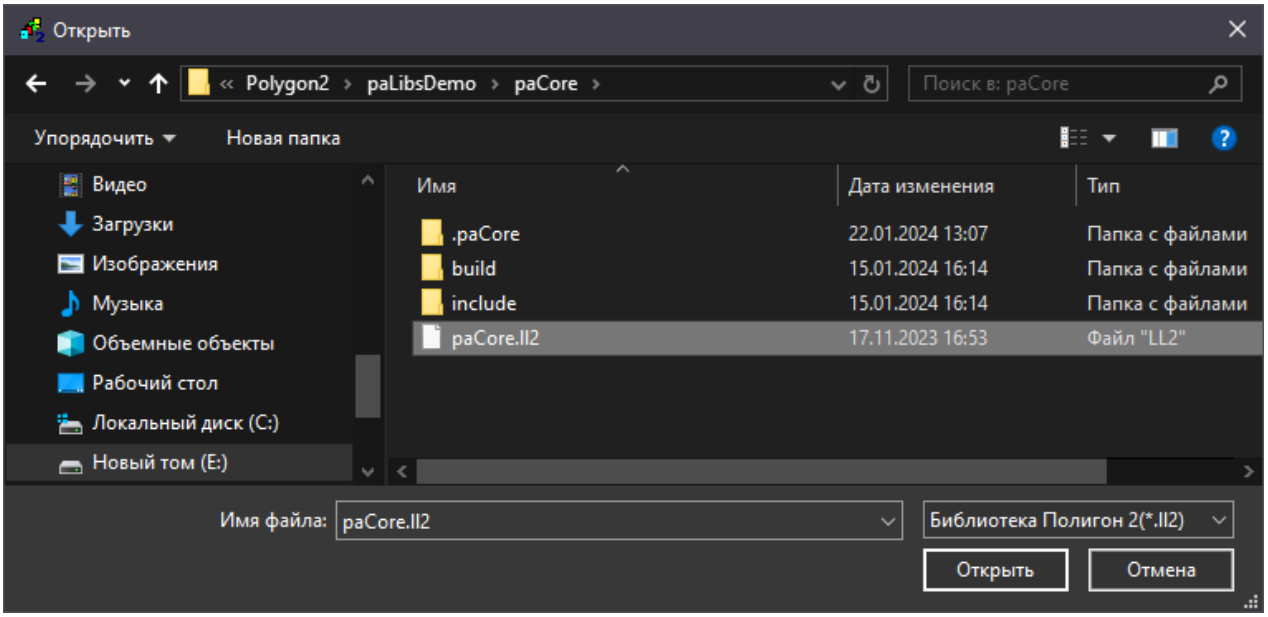


Рисунок 2.3 – Добавление библиотеки в проект

Добавленная библиотека отобразится в окне ***Проекты***.

2.1 Хранение параметров на диске (SaverEx)

Блок ***SaverEx*** обеспечивает хранение данных в архиве на диске контроллера в виде бинарного файла.

Поскольку операции файлового ввода/вывода занимают значительное время, данный блок следует размещать только в ***Фоне***.

Таблица 2.1 – Назначение входов и выходов SaverEx

Элемент	Описание
Входы	
rst	Сброс ошибок записи
fnm	Абсолютный путь и имя файла на диске (может быть пустым – задается автоматически), расширение игнорируется. При сохранении данных на внешнем накопителе следует использовать путь, указанный на выходе блока 210-SD-USB (константный)
wr	Запись на диск
in	Значение параметра (циклический)
typ	Тип параметра (циклический, константный): **DI** – 8-ми битный регистр; **AI** – вещественное значение; **II** – 16-ти битный регистр
ini	Значение для инициализации (циклический, константный)
Выходы	
next	Имя следующего файла
enb	Запись разрешена

Продолжение таблицы 2.1

Элемент	Описание
sts	Статус: **0** – после сброса; **1** – записан; **2** – прочитан; **<0** – ошибка
good	Количество удачных записей
bad	Количество ошибок записи
rej	Количество отклоненных записей
o	Текущее значение параметров (циклический)

Имя файла и путь к нему задается на входе **fnm**. Поле может быть пустым, тогда имя файла будет выбрано автоматически по индексу блока, а файл сохранится в рабочую директорию контроллера.

Данные организуются в виде переменных **in** с жестко заданным типом **typ**.

Поскольку входы **in** имеют тип **any**, следует строго соблюдать правила преобразования типов при проведении связей.

Если файла не существует на диске – входы инициализируются с помощью значений **ini**, происходит запись в файл.

Если файл существует на диске, выходы инициализируются сохраненными значениями. Запись в файл осуществляется только при изменении значений на входах **in**.

Запись на диск можно осуществить принудительно, подав команду **wr**.

Для надежной сохранности данных одновременно на диске находятся два файла, соответствующие одному архиву. Поэтому если контроллер будет перезагружен в момент записи на диск, данные не пропадут, а будут доступны предыдущие значения переменных, записанные в другом файле.

При чтении содержимое файла контролируется с помощью контрольной суммы и выдается на выходы только при ее корректности. Поэтому если, например, добавить в файл новую переменную, то записанные значения для выходов сбросятся на инициализирующие **ini**.

Если при записи файла на диск происходит однократная ошибка, блок пытается переименовать текущий файл и снова произвести запись. Если повторная запись оказывается удачной, то продолжается работа в обычном режиме, а выход **bad** инкрементируется. Следует принять меры по диагностике или замене носителя, поскольку сбои при записи могут быть следствием скорого выхода его из строя. Файл, на котором произошел сбой, остается на диске под тем же именем с добавленным к нему суффиксом равным метке времени сбоя (в мс от 1 января 1970 г). Не рекомендуется его удалять, чтобы повторно не использовать потенциально сбойный сектор.

Если происходит повторный сбой записи, то блок блокируется (выход **enb = 0**) и больше не производит попыток переименований файлов и записи до тех пор, пока ошибки не будут сброшены фронтом на входе **rst**.

ВНИМАНИЕ
При изменении числа входов блока ***SaverEx*** файлы на диске перезаписываются.

Пример работы с блоком приведен в разделе 5.1.

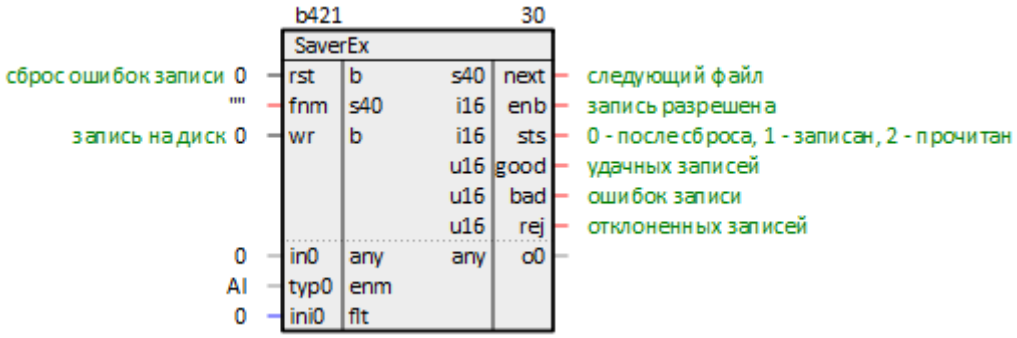


Рисунок 2.4 – Хранение параметров на диске (SaverEx)

2.2 Буфер чтения/записи уставок (BufSupEx)

Блок ***BufSupEx*** представляет собой двунаправленный буфер данных интерфейса и данные сохраняются в бинарном файле на диске контроллера. Блок сохраняет все значения на диске контроллера аналогично блоку SaverEx.

Поскольку операции файлового ввода/вывода занимают значительное время, данный блок следует размещать только в ***Фоне***.

Таблица 2.2 – Назначение входов и выходов BufSupEx

Элемент	Описание
Входы	
inter	Связь от интерфейса, к которому принадлежит данный буфер
group	Номер группы (константный)
fnm	Абсолютный путь и имя файла (может быть пустым – задается автоматически), расширение игнорируется. При сохранении данных на внешнем накопителе следует использовать путь, указанный на выходе блока 210-SD-USB (константный)
mask	Не используется
rst	Сброс ошибок записи
wr	Запись на диск
dan	Значение, которое записывается в буфер при **czap = 1** (циклический)
czap	Запись значения **dan** (циклический)
typ	Тип параметра (циклический, константный): **DI**, **DO** – 8-ми битный регистр; **AI**, **AO** – вещественное значение; **II**, **IO** – 16-ти битный регистр
adr	Адрес параметра (циклический, константный)
ini	Значение для инициализации (циклический, константный)
min	Минимум. Если принятое значение меньше **min**, то оно игнорируется (циклический)
max	Максимум. Если принятое значение больше **max**, то оно игнорируется (циклический)
Выходы	
pkt	Подключение к блокам ***OpcUAClient***, ***UABufSups*** из библиотеки ***paOpcUA***
next	Имя следующего файла
enb	Запись разрешена
sts	Статус: **0** – после сброса; **1** – записан; **2** – прочитан; **<0** – ошибка
good	Количество удачных записей
bad	Количество ошибок записи
rej	Количество отклоненных записей
dan	Значение параметра, полученное по интерфейсу или на вход **dan** (после проверки на **min** и **max**)
chn	Признак изменения, выставляется в **1** на один цикл выполнения программы, если значение **dan** изменилось
zap	Признак записи, выставляется в **1** на один цикл выполнения программы, если значение с входа **dan** было записано

Номер группы **group** используется в качестве Slave ID при подключении к блоку интерфейса Modbus Slave.

Имя файла и путь к нему задается на входе **fnm**. Поле может быть пустым, тогда имя файла будет выбрано автоматически по индексу блока, а файл сохранится в рабочую директорию контроллера.

Адрес переменной **adr** зависит от интерфейса, к которому подключен буфер, например, адрес регистра Modbus.

Поскольку входы **dan** имеют тип ***any***, следует строго соблюдать правила преобразования типов при проведении связей.

Если файл существует на диске, выходы инициализируются сохраненными значениями. Если файла не существует – выходы инициализируются значениями инициализации **ini**.

Запись в файл осуществляется при изменении значений на входах **dan** или по интерфейсу. Если файла на диске не существует и выходы **dan** приняли значения **ini**, то можно записать их на диск принудительно, подав команду **wr**.

Для надежной сохранности данных одновременно на диске находятся два файла, соответствующие одному архиву. Поэтому если контроллер будет перезагружен в момент записи на диск, данные не пропадут, а будут доступны предыдущие значения переменных, записанные в другом файле.

При чтении содержимое файла контролируется с помощью контрольной суммы и выдается на выходы только при ее корректности. Поэтому если, например, добавить в файл новую переменную, то записанные значения для выходов сбросятся на инициализирующие **ini**.

Если при записи файла на диск происходит однократная ошибка, блок пытается переименовать текущий файл и снова произвести запись. Если повторная запись оказывается удачной, то продолжается работа в обычном режиме, а выход **bad** инкрементируется. Следует принять меры по диагностике или замене носителя, поскольку сбои при записи могут быть следствием скорого выхода его из строя. Файл, на котором произошел сбой, остается на диске под тем же именем с добавленным к нему суффиксом равным метке времени сбоя (в мс от 1 января 1970 г). Не рекомендуется его удалять, чтобы повторно не использовать потенциально сбойный сектор.

Если происходит повторный сбой записи, то блок блокируется (выход **enb = 0**) и больше не производит попыток переименований файлов и записи до тех пор, пока ошибки не будут сброшены фронтом на входе **rst**.

ВНИМАНИЕ
При изменении числа входов блока ***BufSupEx*** файлы на диске перезаписываются.

Примеры работы с блоком приведены в разделе 5.2.

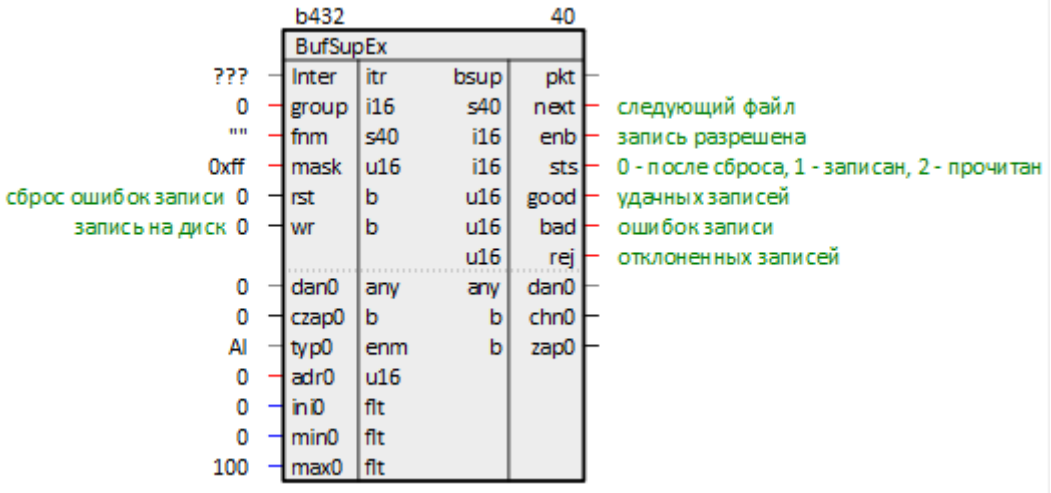


Рисунок 2.5 – Буфер чтения/записи уставок (BufSupEx)

2.3 Счетчик времени наработки (CounterMEx)

Блок ***CounterMEx*** предназначен для сохранения в бинарный файл наработки устройств.

Поскольку операции файлового ввода/вывода занимают значительное время, данный блок следует размещать только в ***Фоне***.

Таблица 2.3 – Назначение входов и выходов CounterMEx

Элемент	Описание
Входы	
rst	Сброс ошибок записи
fn	Абсолютный путь и имя файла (может быть пустым – задается автоматически), расширение игнорируется. При сохранении данных на внешнем накопителе следует использовать путь, указанный на выходе блока 210-SD-USB (константный)
ask	Запись на диск

Продолжение таблицы 2.3

Элемент	Описание
slv	Используется для изменения значений счетчиков, если **slv = 1**, то **cnt = mas**, **cfrn = mcfrn**
enbl	Работа устройства (циклический): пока **enbl = 1**, увеличивается время наработки **cfrn,** при изменении **enbl** с **0** на **1** число включений **cnt** увеличивается на **1**
rst	Сброс времени наработки **cnt** (циклический)
frm	Формат отображения для времени наработки. Не используется
rfr	Сброс числа включений **cfrn** (циклический)
mas	Балансировка времени наработки **cnt** (циклический), если **slv** = **1**, то **cnt** = **mas**
mcfrn	Балансировка числа включений **cfrn** (циклический), если **slv** = **1**, то **cfrn** = **mcfrn**
Выходы	
next	Имя следующего файла
enb	Запись разрешена
good	Количество удачных записей
bad	Количество ошибок записи
cnt	Время наработки в секундах (циклический)
hour	Время наработки в формате часы (циклический)
min	Время наработки в формате минуты (циклический)
sec	Время наработки в формате секунды (циклический)
day	Текущий день (циклический)
mnth	Текущий месяц (циклический)
year	Текущий год (циклический)
cfrn	Число включений (циклический)

Блок анализирует входы **enbl**.

На выходе **cfrn** отображается число включений устройства (количество изменений **enbl** с **0** на **1**), на выходе **cnt** отображается время наработки устройства (сколько секунд **enbl** был равен **1**).

Блок может сохранять число включений и время наработки в файл по фронту на входе **ask**. Сохраненные значения считываются из файла при инициализации.

Имя файла и путь к нему задается на входе **fnm**. Поле может быть пустым, тогда имя файла будет выбрано автоматически по индексу блока, а файл сохранится в рабочую директорию контроллера.

Для изменения числа включений и времени наработки следует подать **1** на вход **slv**, тогда **cnt = mas**, **cfrn = mcfrn**. Это может быть полезно для синхронизации в дублированных системах.

Для надежной сохранности данных одновременно на диске находятся два файла, соответствующие одному архиву. Поэтому если контроллер будет перезагружен в момент записи на диск, данные не пропадут, а будут доступны предыдущие значения наработки, записанные в другом файле.

При чтении содержимое файла контролируется с помощью контрольной суммы и выдается на выходы только при ее корректности. Поэтому если, например, добавить в файл новую переменную, то записанные значения для выходов сбросятся на инициализирующие **ini**.

Если при записи файла на диск происходит однократная ошибка, блок пытается переименовать текущий файл и снова произвести запись. Если повторная запись оказывается удачной, то продолжается работа в обычном режиме, а выход **bad** инкрементируется. Следует принять меры по диагностике или замене носителя, поскольку сбои при записи могут быть следствием скорого выхода его из строя. Файл, на котором произошел сбой, остается на диске под тем же именем с добавленным к нему суффиксом равным метке времени сбоя (в мс от 1 января 1970 г). Не рекомендуется его удалять, чтобы повторно не использовать потенциально сбойный сектор.

Если происходит повторный сбой записи, то блок блокируется до тех пор, пока ошибки не будут сброшены фронтом на входе **rst**.

ВНИМАНИЕ
При изменении числа входов блока ***CounterMEx*** файлы на диске перезаписываются.

Пример работы с блоком приведен в разделе 6.

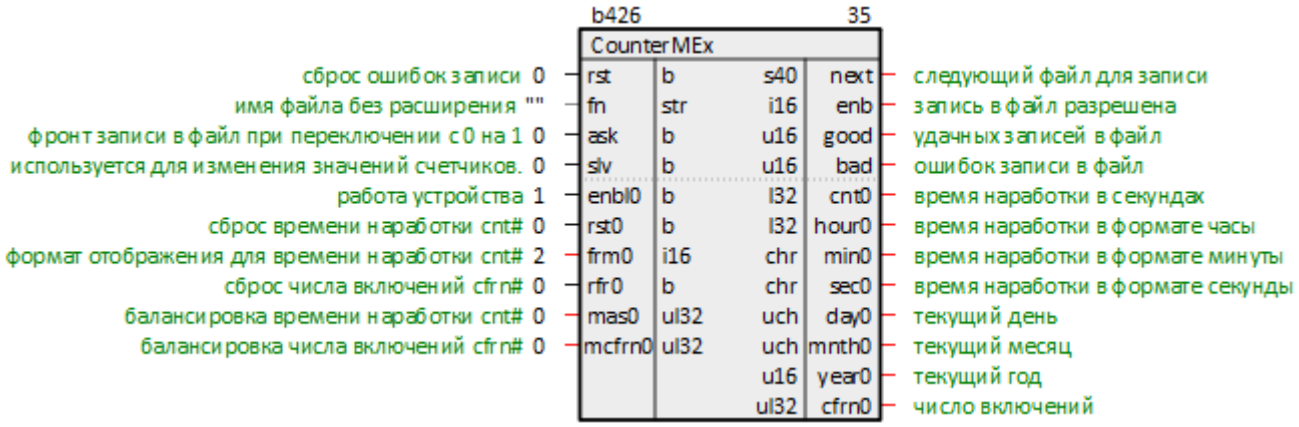


Рисунок 2.6 – Счетчик времени наработки (CounterMEx)

3 Библиотека profiLogger

Запись архивов на диск контроллера организуется с помощью блоков из библиотеки ***profiLogger***.

Добавление библиотеки в проект описано в разделе 2.

3.1 Массив значений параметров по событию (RamFRLogger)

Блок ***RamFRLogger*** совместно с блоком ComtradeLogger организует сохранение значений выбранных параметров в файл в течение заданного времени «до» события и в течение заданного времени «после» события.

Блок ***RamFRLogger*** формирует массив значений параметров для последующей записи на диск.

Раздел библиотеки: ***Архиваторы***.

Блок ***RamFRLogger*** следует размещать только в ***Таймере***.

Таблица 3.1 – Назначение входов и выходов RamFRLogger

Элемент	Описание
Входы	
trigger	Фронт срабатывания. При подаче **1** на данный вход блок начинает работать
befor	Время в секундах «до» события
after	Время в секундах «после» события
every	Кратность квантования. Например, если таймерный промежуток установлен **20 мс** и **every = 5**, запись будет производиться каждые **100 мс**
Выходы	
me	Не используется
array	Массив накопленных значений – для соединения с блоком ***ComtradeLogger***
ttime	Время срабатывания – для соединения с блоком ***ComtradeLogger***
cnt	Счетчик точек
pnum	Количество точек для записи
sts	Статус подготовки лога: **0** – нет ошибок; **1** – идет подготовка; **-1** – ошибка

Входы/выходы проекта, которые необходимо архивировать по событию, добавляются в раздел ***Данные*** внутри блока.

Для начала формирования массива значений следует установить **1** на входе **trigger**.

Работа блоков ***RamFRLogger*** и ***ComtradeLogger*** объединена в блоке «черный ящик» BlackBox. Пример работы с блоком ***BlackBox*** в разделе 8.

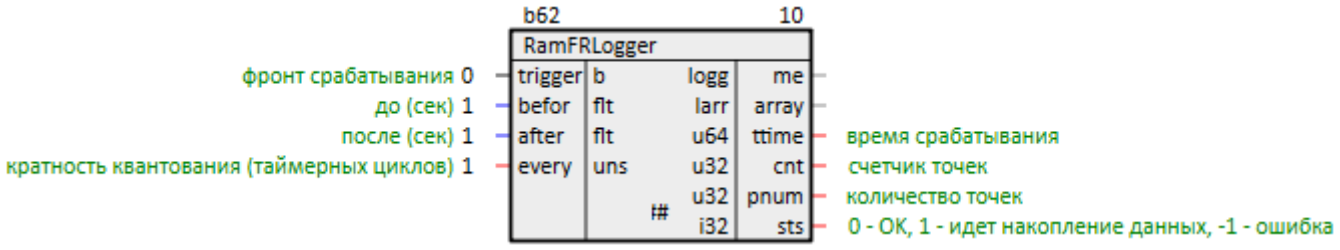


Рисунок 3.1 – Массив значений параметров по событию (RamFRLogger)

3.2 Сохранение значений параметров на диск по событию (ComtradeLogger)

Блок ***ComtradeLogger*** совместно с блоком RamFRLogger организует сохранение значений выбранных параметров в файл в течение заданного времени «до» события и в течение заданного времени «после» события.

Блок ***ComtradeLogger*** сохраняет значения параметров в файл на диск.

Раздел библиотеки: ***Архиваторы***.

Поскольку операции файлового ввода/вывода занимают значительное время, данный блок следует размещать только в ***Фоне***.

Таблица 3.2 – Назначение входов и выходов ComtradeLogger

Элемент	Описание
Входы	
enb	Разрешение работы блока: **1** – включен; **0** – выключен
name	Путь и начало имени файла. При сохранении данных на внешнем накопителе следует использовать путь, указанный на выходе блока 210-SD-USB (константный)
array	Массив накопленных значений – для соединения с блоком ***RamFRLogger***
ttime	Время срабатывания – для соединения с блоком ***RamFRLogger***
Выходы	
me	Не используется
fname	Абсолютный путь и начало имени результирующего файла ***.dat**
sts	Статус подготовки файла: **0** – блок не работает над файлом или данные для файла еще накапливаются; **1** – файл успешно создан (взводится на один цикл работы блока); **-1** – внутренняя ошибка при работе с массивом данных (массив не содержит данных или меток времени); **-2** – проблема создания/открытия файла; **-3** – неподдерживаемый формат входных данных

Для начала записи на диск следует установить **1** на входе **enb**.

По окончании записи формируются два файла: файл с расширением **.dat** – набор значений параметров, накопленных за временной отрезок, файл с расширением **.cfg** – служебная информация о событии.

Блок ***ComtradeLogger*** задает имена файлов так: "значение **fname**" плюс "метка времени начала записи параметров на диск в формате ***ГГГГ_ММ_ДД_ЧЧ_ММ_СС*** (без учета часового пояса)".

ВНИМАНИЕ
Рекомендуется записывать данные на внешний накопитель, чтобы избежать изнашивания внутреннего диска контроллера.

Работа блоков ***RamFRLogger*** и ***ComtradeLogger*** объединена в блоке «черный ящик» BlackBox. Пример работы с блоком ***BlackBox*** приведен в разделе 8.

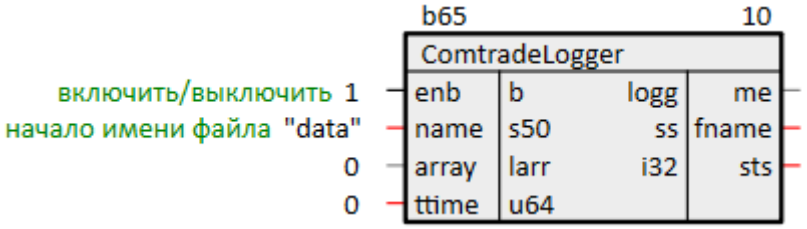


Рисунок 3.2 – Сохранение значений параметров на диск по событию (ComtradeLogger)

4 Библиотека profiLoggerLight

Запись архивов на диск контроллера организуется с помощью блоков из библиотеки ***profiLoggerLight***.

Добавление библиотеки в проект описано в разделе 2.

4.1 Файловый архив данных (FileDpLogger)

Блок ***FileDpLogger*** предназначен для архивирования значений выбранных входов/выходов в текстовых файлах.

Раздел библиотеки: ***Архивы***.

Таблица 4.1 – Назначение входов и выходов FileDpLogger

Элемент	Описание
Входы	
file_path	Абсолютный путь к директории, где будут храниться архивные файлы. При сохранении данных на внешнем накопителе следует использовать путь, указанный на выходе **usbpath** блока 210-SD-USB (константный)
enb	Разрешена работа блока: **0** – блок не работает; **1** – блок пишет данные в буфер
wr	Разрешена запись на диск: **0** – нет; **1** – да
loc	Использовать локальное время для смены файлов (константный): **0** – нет (использовать мировое время); **1** – да
grp	Группировка данных по файлам (константный): **NO_GROUP** – каждое данное в отдельном файле; **GROUP_ALLINONE** – все данные в одном файле
frmt	Формат записей в файлах (константный): **TAB_and_COMM** – формат записи с разделителем табуляция «→», в числах с плавающей точкой используется запятая «,»; **COMM_and_DOT** – формат записи с разделителем запятая «,», в числах с плавающей точкой используется точка «.». Использование запятой в комментарии приведет к некорректному формату файла; **TO_MSD200** – формат записи с разделителем точка с запятой «;», в числах с плавающей точкой используется запятая «,»
meth	Метод записи данных в файл (константный): **TIMESTAMP** – запись с временными метками с заполнением пустых; **INTERVAL** – запись с заданным интервалом (период архивирования по умолчанию **1 с**)
intr	Интервал записи, мс. Устанавливает период архивирования. В файл записывается последнее значение входа/выхода, полученное в этом периоде. Используется только в методе **INTERVAL** (константный)
qqlen	Размер буфера на каждое данное: минимальное значение **10**, максимальное зависит от характеристик контроллера и объема оперативной памяти, которую при работе проекта допускается выделить для хранения буфера данных. Для расчета максимального размера очереди нужно учитывать количество данных, которое планируется записывать (см. пример ниже) (константный)
prt	Приоритет потока записи из буфера на диск, по умолчанию **25** (константный)
rst	Сброс ошибок записи: изменение с **0** на **1** сбрасывает бит **iserr**, обнуляет **err** и **lost**
Выходы	
lg	Указатель на блок базового типа CBaseLogger, инициализируется указателем на базовый класс блока. Не используется в программе контроллера
ok	Количество удачно записанных строк, значение инициализации **0**
lost	Количество потерянных данных из-за заполненности очередей, значение инициализации **0**

Продолжение таблицы 4.1

Элемент	Описание
iserr	Была ошибка записи, значение инициализации **0**
err	Количество ошибок записи, значение инициализации **0**

Данные в архив записываются по изменению с учетом свойства ***Зона нечувствительности***. Формат, группировка и метод записи в файл выбираются на входах блока.

Входы/выходы в проекте, которые необходимо архивировать, добавляются в раздел ***Данные*** внутри блока.

Для того, чтобы блок начал запись на диск, следует установить значение **1** на входах **enb** и **wr**.

ВНИМАНИЕ
Рекомендуется записывать данные на внешний накопитель, чтобы избежать изнашивания внутреннего диска контроллера.

Каждый день запись производится в новый файл. На входе **loc** блока можно выбрать, какое время использовать для смены файлов (мировое или локальное). Дата записи данных отображается в имени файла.

Имена файлов зависят от заданных на входе блока значений:

- для формата **TO_MSD200**;
- ***Logs_ALLINONE_ГГГГ_ММ_ДД.txt*** – для форматов **TAB_and_COMM** и **COMM_and_DOT** с группировкой всех данных в одном файле **GROUP_ALLINONE**;
- ***Logs_<алиас данного>_ГГГГ_ММ_ДД.txt*** – для форматов **TAB_and_COMM** и **COMM_and_DOT** без группировки **NO_GROUP** (каждое данное в отдельном файле).

В заголовок таблицы по умолчанию попадают свойства входов/выходов ***Полный алиас*** и ***Комментарий***.

ВНИМАНИЕ
Комментарий отображается при условии, что у модуля добавлено свойство ***Трансляция: включить свойства входов/выходов***, равное ***Только из разделов***.

Если необходимо, чтобы комментарий не отображался, следует удалить свойство ***Трансляция: комментарии отдельно*** у раздела ***Данные***.

Если во время начала записи файл на диске с таким именем существует, то к имени существующего файла добавляется метка времени и создается новый файл. Название нового файла определяется по правилам выше.

ВНИМАНИЕ
При изменении числа параметров в разделе ***Данные*** создается новый файл архива.

ВНИМАНИЕ
Использование запятой в комментарии приведет к некорректному чтению файла трендом, в случае использования формата **COMM_and_DOT**.

ВНИМАНИЕ
Для корректного отображения сохраненных данных в программах просмотра графиков необходимо, чтобы у входов/выходов, добавленных в раздел, было добавлено свойство ***Полный алиас***. Алиасы должны быть уникальными в пределах раздела ***Данные***, чтобы обеспечить уникальность имен файлов. Уникальность алиаса должна обеспечиваться без учета регистра. Если алиас отсутствует, или не уникален, то данное не будет архивироваться.

ВНИМАНИЕ
Если блок устанавливается в таймере, то порядок выполнения всех блоков, входы/выходы которых добавлены в архив, имеет смысл выставлять либо выше, либо ниже, чем у блока архива. Так как метка времени блоков, порядок которых ниже, запишется с меткой времени текущего таймерного цикла, а те, которые выше – с меткой времени следующего цикла.

Каждый объект данных хранится в своем буфере, максимальная величина которого задается на входе блока **qqlen**. Максимальный размер очереди используется для ограничения количества оперативной памяти, которую может выделить блок для хранения буфера данных, т. к. на хранение каждого элемента очереди требуется **32 байта** оперативной памяти.

Пример

Например, при **qqlen = 1000**, для хранения **64** различных показателей, потребуется **2,048 мегабайт** оперативной памяти. Если нет возможности записать данные в файл, то эта очередь заполнится через **1000 секунд** (при записи методом **INTERVAL** и показателе **int = 1000 мс**).

При возникновении ошибок записи (недостаточно памяти, память испорчена, извлечена Flash-карта и т. п.) блок установит значение **1** на выходе **iserr** (если была ошибка записи), и увеличит счетчик ошибок **err**. После двух ошибок блок прекратит запись на диск, пока ошибки не будут сброшены сигналом на входе **rst**.

Если запись остановлена (**wr = 0** или **err = 2**), но при этом разрешена работа блока (**enb = 1**), то данные копятся в буфере, пока он не достигнет максимальной величины, заданной на входе **qqlen**.

Если за время отключения записи буфер заполнится, часть архивируемых данных будет утеряна. На выходе **lost** отобразится общее количество утерянных данных по всем входам/выходам. Количество потерянных данных сбрасывается при сбросе ошибок записи сигналом на входе **rst**.

Работа блока ***FileDpLogger*** осуществляется в отдельном потоке, приоритет которого задается на входе **prt**.

ПРИМЕЧАНИЕ
Формат **TO_MSD200** разрабатывался для визуализации в программе График ОВЕН МСД200.

В связи с особенностями работы программы *График ОВЕН МСД200* существуют следующие ограничения для обеспечения совместимости:

1. Ограничение группировки – только **GROUP_ALLINONE**, т. к. имя файла должно быть в формате ***ГГГГ_ММ_ДД.csv***.
2. Ограничение времени – только **INTERVAL** с интервалом не менее секунды (**intr = 1000**). Желательно, чтобы интервал был кратен секунде, т.к. программа использует время в формате ***ЧЧ:ММ:СС***. Миллисекунды считаются за пределами диапазона: блок округляет точность времени до секунды в большую сторону. Если программа обнаружит несколько идентичных значений времени, то файл не откроется.
3. Ограничение количества данных – максимум **64 канала**. При использовании большего количества точек архивации программа не откроет файл.

Пример работы с блоком приведен в разделе 7.

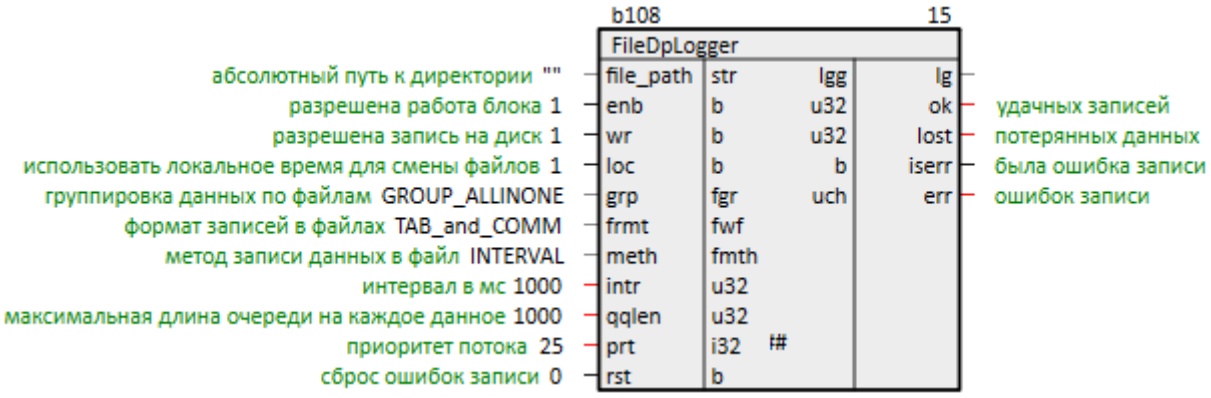


Рисунок 4.1 – Файловый архив данных (FileDpLogger)

4.2 Черный ящик (BlackBox)

Блок BlackBox совмещает в себе работу блоков RamFRLogger и ComtradeLogger из библиотеки ***profiLogger***. Этот блок сохраняет значения выбранных параметров в файл в течение заданного времени «до» события и в течение заданного времени «после» события.

Раздел библиотеки: ***Архиваторы***.

Блок ***BlackBox*** следует размещать только в ***Таймере***.

Таблица 4.2 – Назначение входов и выходов BlackBox

Элемент	Описание
Входы	
trigger	Фронт срабатывания. При подаче **1** на данный вход блок начинает работать
before	Время, в течение которого производится запись значений до срабатывания триггера
after	Время, в течение которого производится запись значений после срабатывания триггера
every	Кратность квантования. Например, если таймерный промежуток (свойство модуля ***Таймерный промежуток***) установлен **20 мс** и **every = 5**, запись будет производиться каждые **100 мс**. Должно быть отлично от **0**
enb	Разрешение записи на диск: **0** – выключена; **1** – включена

Продолжение таблицы 4.2

Элемент	Описание
name	Путь и начало имени файла. При сохранении данных на внешнем накопителе следует использовать путь, указанный на выходе блока 210-SD-USB (константный)
priority	Приоритет потока ожидания выполнения команды
Выходы	
logger_sts	Статус подготовки лога в ОЗУ: **0** – нет ошибок; **1** – идет подготовка лога; **-1** – неверное значение параметра **every** или отсутствует раздел ***Данные*** в блоке
me	Указатель на блок базового типа CBaseLogger, инициализируется указателем на базовый класс блока. Не используется в программе контроллера
fname	Абсолютный путь и начало имени результирующего файла ***.dat**, инициализируется **0**
comtrade_sts	Статус подготовки файла, инициализируется **0**: **0** – блок не работает над файлом или данные для файла еще накапливаются; **1** – файл успешно создан (взводится на один цикл работы блока); **-1** – внутренняя ошибка при работе с массивом данных (массив не содержит данных или меток времени); **-2** – проблема создания/открытия файла; **-3** – неподдерживаемый формат входных данных
pnum	Число накопленных значений в массиве для записи в ПЗУ, инициализируется **0**

Входы/выходы в проекте, которые необходимо архивировать по событию, добавляются в раздел ***Данные*** внутри блока.

Блок сохраняет в ОЗУ текущие значения с фиксированной частотой дискретизации в течение установленного в **before** времени.

Если на вход **trigger** подано значение больше **0**, отмечается время наступления события и пишутся значения в течение установленного в **after** времени.

Когда запись окончена, сформированный массив передается в фоновый поток записи массива данных в ПЗУ. Если в **enb** установлено значение **1**, формируется следующие два файла: файл с расширением **.dat** – набор значений параметров, накопленных за временной отрезок, файл с расширением **. cfg** – служебная информация о событии.

Имена файлов определяются по входу **name** плюс метка времени (без учета часового пояса) начала записи параметров на диск в формате ***ГГГГ_ММ_ДД_ЧЧ_ММ_СС***.

ПРИМЕЧАНИЕ
Запись рекомендуется производить на внешний накопитель, чтобы избежать изнашивания внутреннего диска контроллера.

Пример работы с блоком приведен в разделе 8.

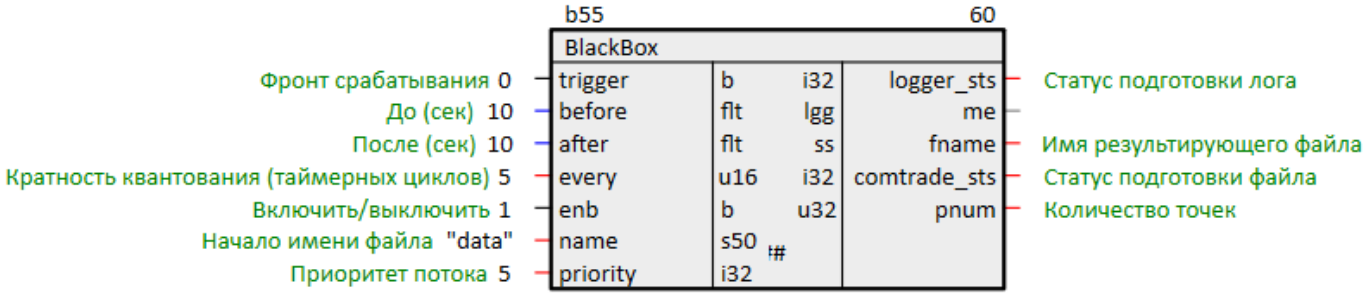


Рисунок 4.2 – Черный ящик (BlackBox)

5 Сохранение уставок

Сохранение уставок на диск контроллера или внешний накопитель организуется с помощью блоков SaverEx и BufSupEx из библиотеки ***paCore***, а также с помощью блока BufSupFltEx из библиотеки ***paModbus***.

5.1 Сохранение уставок из программы контроллера (SaverEx)

Сохранение уставок из программы контроллера организуется с помощью блока SaverEx.

Для надежной сохранности данных одновременно на диске находятся два файла, соответствующие одному архиву. Файлы для сохранения чередуются.

На вход блока **fnm** подается абсолютный путь и имя файла, куда будет осуществляться запись. Расширение файла игнорируется – на диск записываются бинарные файлы с расширениями **.da1** и **.da2**. Если оставить данное поле пустым – имя файла будет определяться автоматически по индексу блока, файл будет сохраняться в рабочую директорию контроллера.

На входах блока **typ** задаются типы данных сохраняемых параметров:

- **DI** – 8-ми битный регистр;
- **AI** – вещественное значение;
- **II** – 16-ти битный регистр.

На входах **ini** задаются значения инициализации параметров.

Рисунок 5.1 – Конфигурация SaverEx для сохранения уставок на MicroSD-накопитель в файлах saver.da1 и saver.da2

Если файла на диске не существует, при запуске программы входы **in** и выходы **o** примут значения, заданные на входах **ini**. На диске сохранится файл с расширением **.da1**. Выход **good** блока инкрементируется.

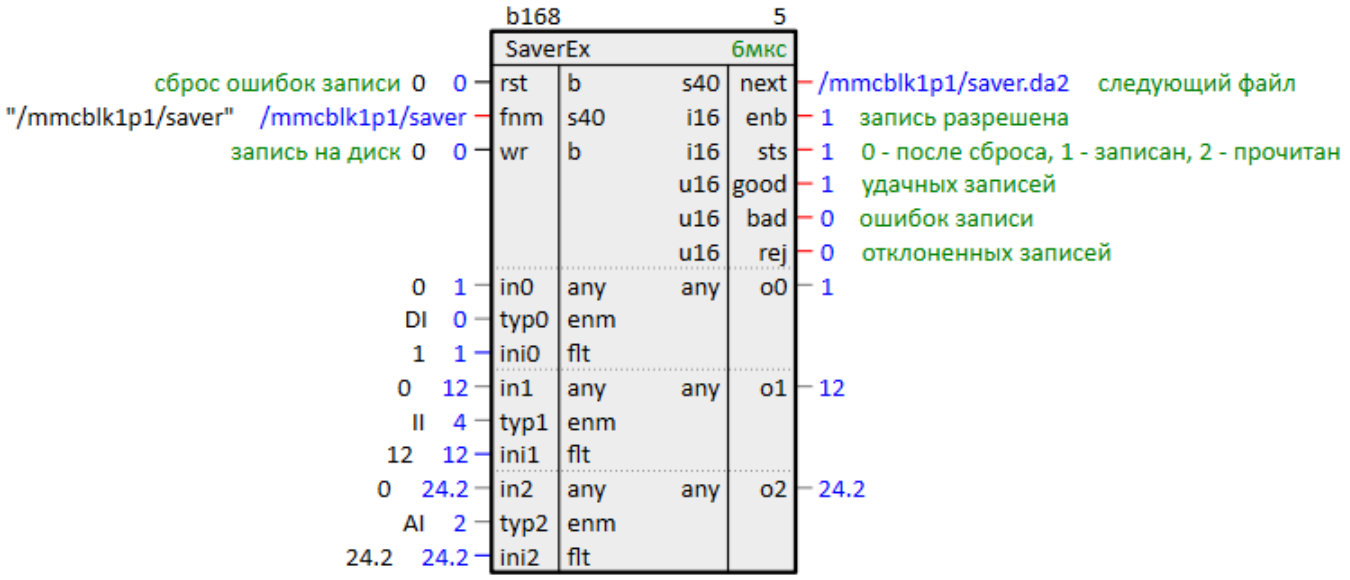


Рисунок 5.2 – Сохранение уставок на MicroSD-накопитель. Первый запуск программы

Далее запись на диск будет производится только по изменению значений на входах **in**.

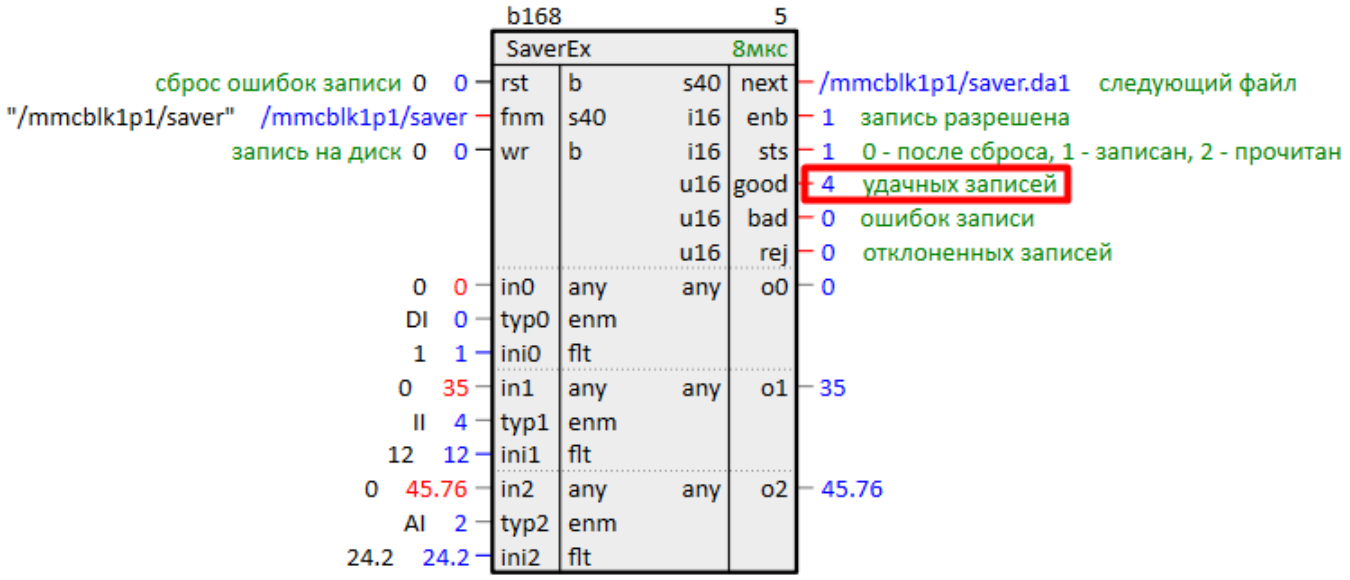


Рисунок 5.3 – Сохранение уставок на MicroSD-накопитель. Изменение уставок из программы

Если файл на диске существует, при перезапуске программы входы **in** и выходы **o** примут значения, сохраненные на диске.

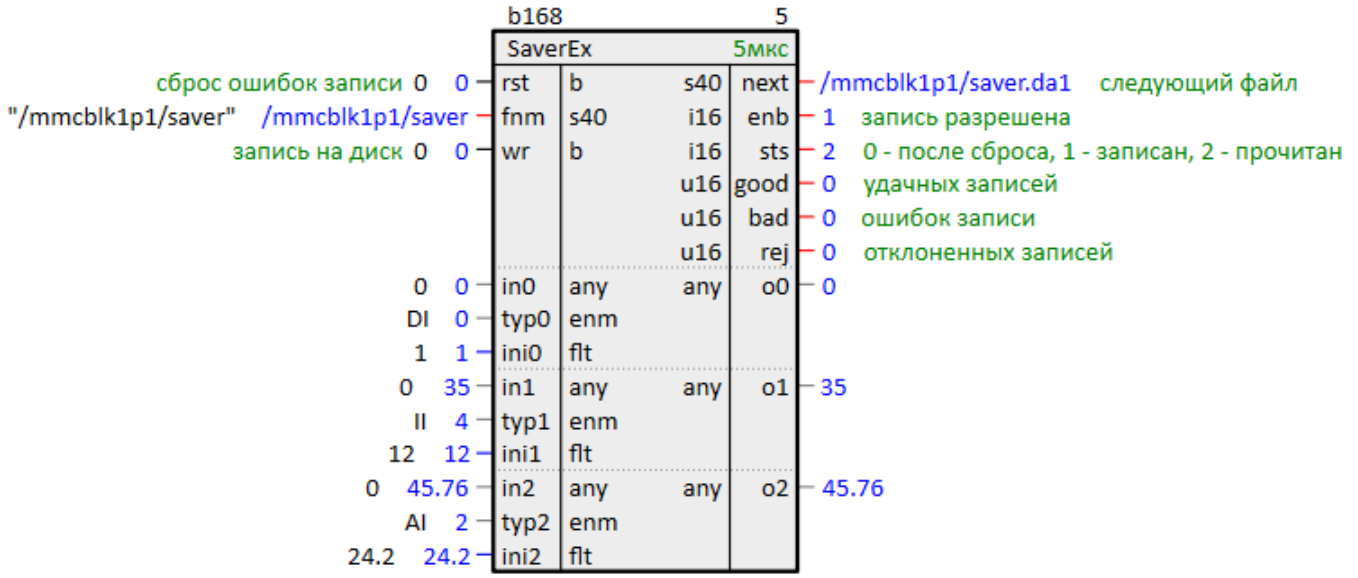


Рисунок 5.4 – Сохранение уставок на MicroSD-накопитель. Перезапуск программы

Файлы, в которые производится запись уставок – бинарные, просмотреть их содержимое нельзя.

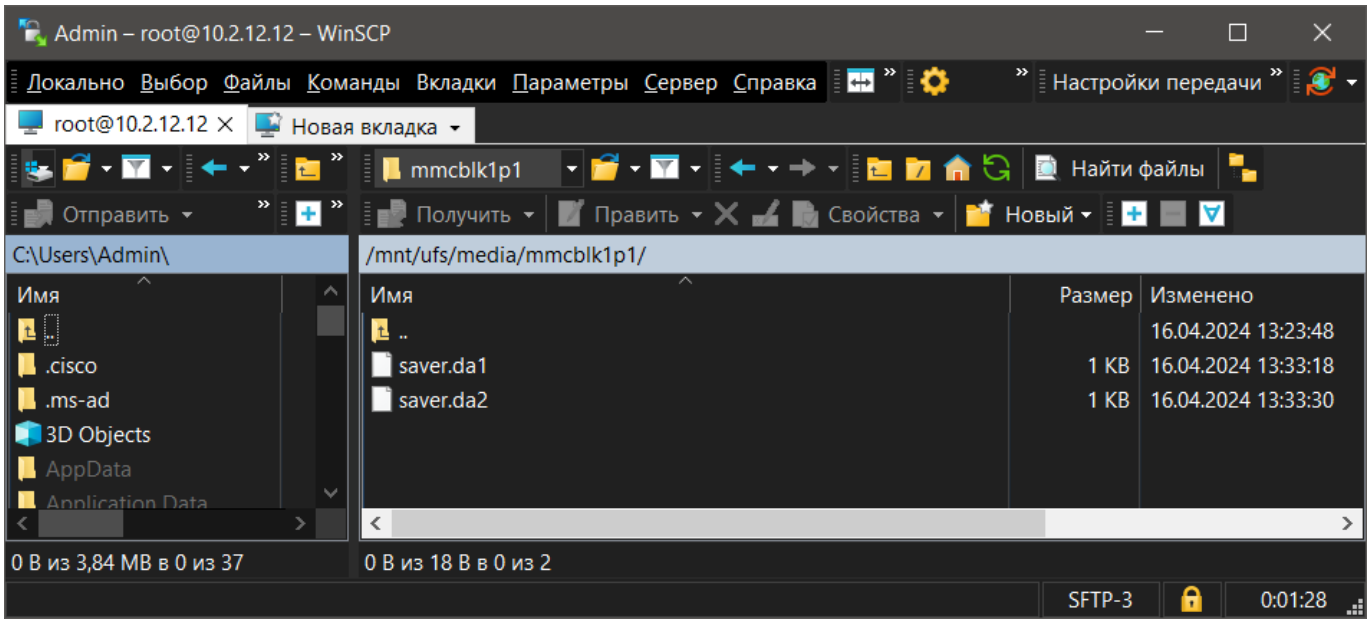


Рисунок 5.5 – Сохранение уставок на MicroSD-накопитель. Файлы на диске

Поведение блока при возникновении ошибки записи описано в разделе 2.1.

При сохранении данных на внешний накопитель на вход **rst** можно завести выход флага монтирования накопителя блока 210-SD-USB. Сброс ошибок **rst** выполняется по фронту перехода из **0** в **1**.

ПРИМЕЧАНИЕ
При изменении числа входов блока ***SaverEx*** файлы на диске перезаписываются.

5.2 Сохранение уставок с использованием OPC UA/Modbus (BufSupEx)

Для записи уставок с удаленного OPC UA-клиента и/или по протоколу Modbus используются блоки ***BufSupEx***.

Работа блока ***BufSupEx*** аналогична работе блока SaverEx и описана в разделе 2.2.

5.2.1 Запись уставок с удаленного OPC UA-клиента

Для записи уставок в блоки BufSupEx с удаленного OPC UA-клиента используется блок ***UABufSups*** из библиотеки ***paOpcUA***.

Настройка OPC UA-сервера подробно описана в документе Обмен с верхним уровнем. Библиотека paOpcUA.

Блоки ***BufSupEx*** в проекте контроллера-сервера подключаются выходами **pkt** к входам **buf** блока ***UABufSups***.

Вход **inter** блока ***BufSupEx*** можно при необходимости подключить к другому блоку протокола (например, к блокам ***Modbus Slave*** из библиотеки ***paModbus***, см. пример в разделе 5.2.3).

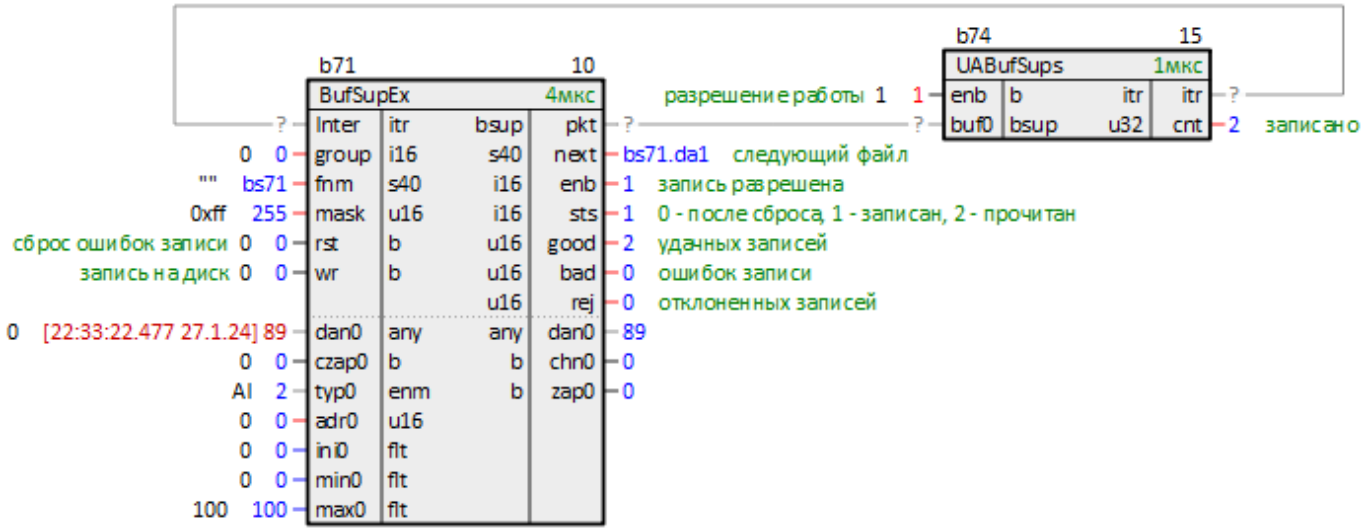


Рисунок 5.6 – Запись уставок с OPC UA-клиента

При отключении блока ***UABufSups*** (или при его отсутствии) для записи уставки с OPC UA-клиента потребуется подать импульс на входы **czap**.

Параметры на диске сохраняются в бинарных файлах **.da1** и **.da2**.

ПРИМЕЧАНИЕ
При изменении числа входов блока ***BufSupEx*** файлы на диске перезаписываются.

Поведение блока ***BufSupEx*** при ошибках записи описано в разделе 2.2.

5.2.2 Синхронизация записи уставок между контроллерами по OPC UA

Для синхронизации записи уставок блоков BufSupEx между двумя контроллерами по протоколу OPC UA требуется дублировать во второй контроллер программу (или целиком место работы), в которой добавлен блок ***BufSupEx***.

Для этого следует перетащить на модуль второго контроллера (оба модуля должны быть в одном проекте) требуемую программу, в выпадающем меню выбрать ***Добавить***.

Обе программы подсветятся желтым. Теперь все изменения на страницах данной программы будут одинаково применены в обоих модулях.

Настройка работы по протоколу OPC UA подробно описана в документе Обмен с верхним уровнем. Библиотека paOpcUA.

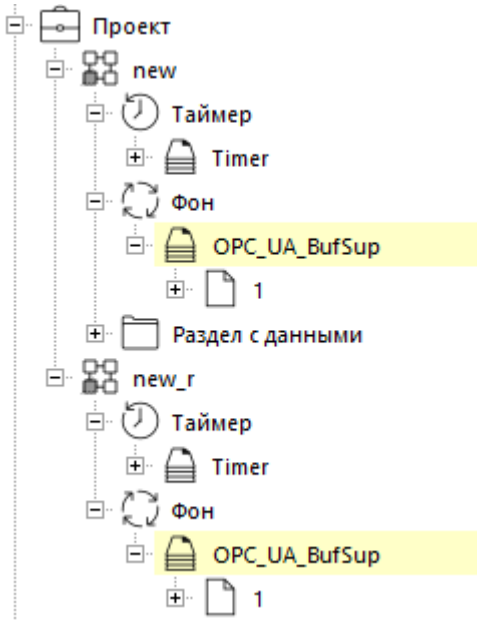


Рисунок 5.7 – Дублированные программы

Блоки ***BufSupEx*** в данной программе следует соединить выходами **pkt** с входами **buf** блока ***OpcUAClient*** из библиотеки ***paOpcUA***.

Входы ***OpcUAClient***, которые отвечают за настройку обмена (IP-адреса и порты), следует задавать с помощью SQL-запросов к соответствующим свойствам модулей.

Запрос IP адреса (prop_ip):

```
"<sql>SELECT value FROM blocks_prop WHERE indx=:module AND type="prop_ip"</sql>"
```

Запрос номера порта (prop_debug_port):

```
<sql>SELECT value FROM blocks_prop WHERE indx=:module AND type="prop_debug_port"</sql>
```

Запрос пользовательского свойства ***Пользовательское свойство 00*** (prop_0):

```
<sql> SELECT value FROM blocks_prop WHERE indx=:module AND type="prop_0"</sql>
```

(обычно используется на входах **rip** и **rprt**).

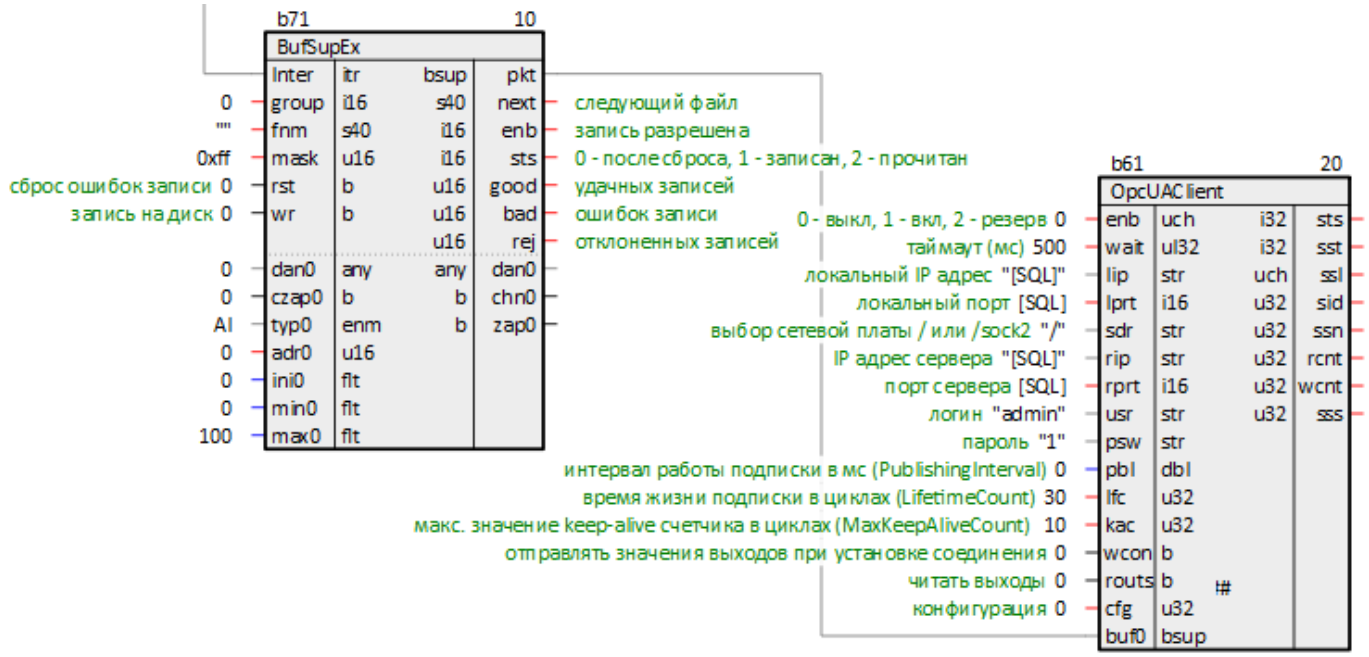


Рисунок 5.8 – Подключение BufSupEx к OpcUAClient

После запуска программ на обоих контроллерах, один из них следует назначить «ведущим» – отключить ***OpcUAClient*** (**enb = 0**), а второй «ведомым» – включить ***OpcUAClient*** (**enb = 1**).

Теперь при изменении уставок ведущего контроллера изменения будут дублироваться в ведомый с помощью OPC UA-клиента.

При попытке изменить уставки ведомого контроллера изменения в ведущем контроллере не применятся.

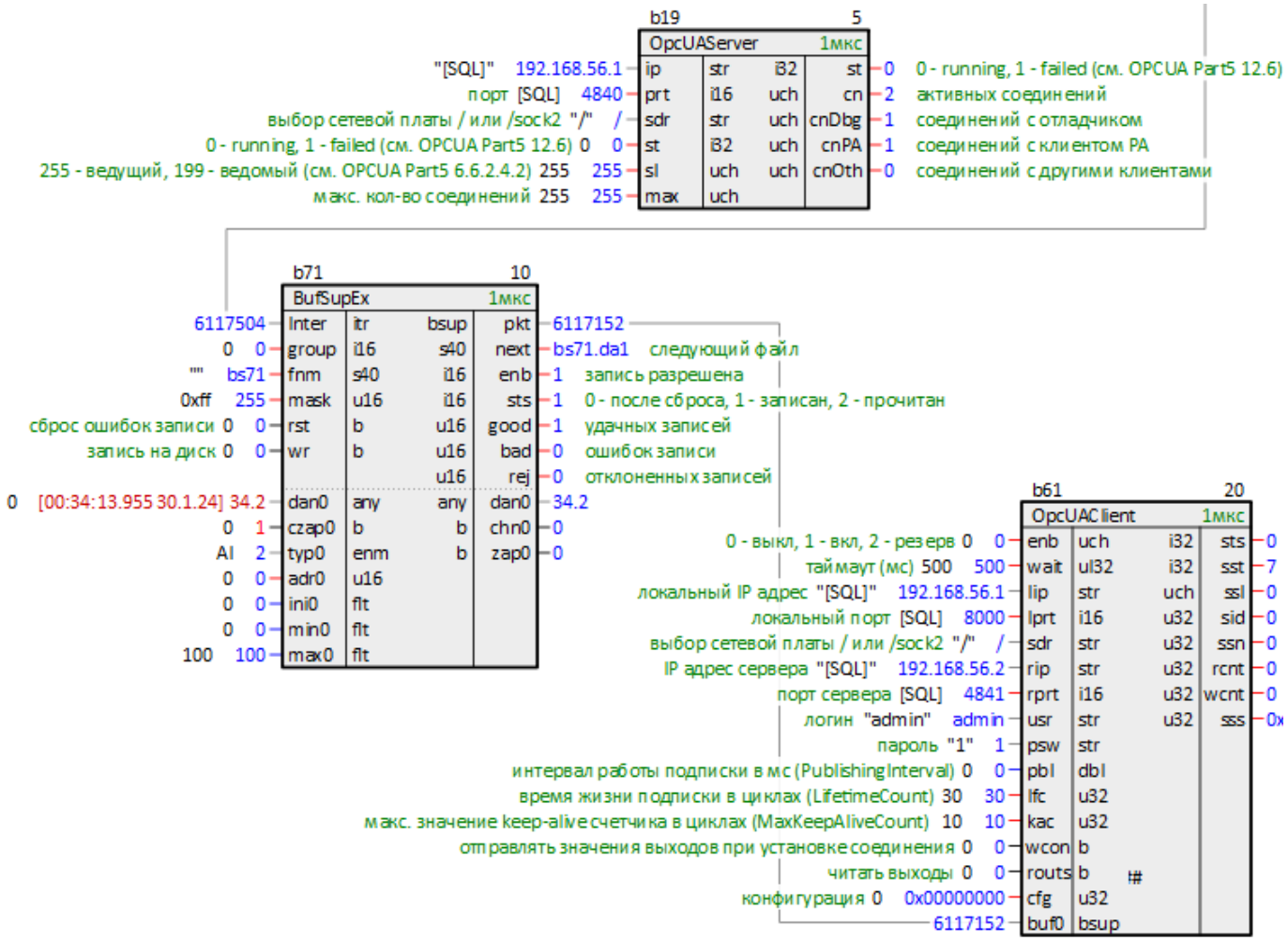


Рисунок 5.9 – Изменение уставки с ведущего контроллера

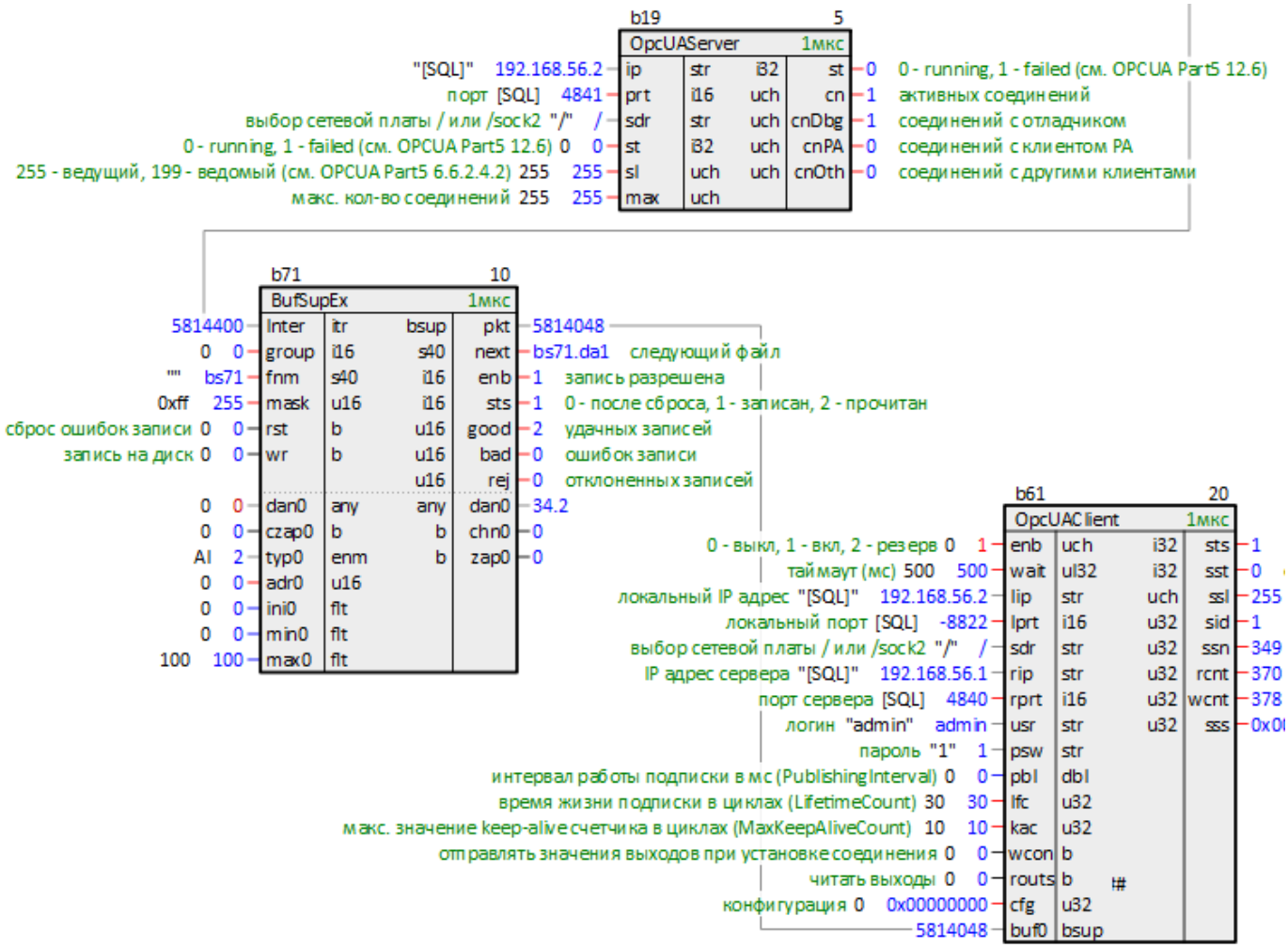


Рисунок 5.10 – Ведомый контроллер

Параметры на диске сохраняются в бинарных файлах с расширениями **.da1** и **.da2**.

ПРИМЕЧАНИЕ
При изменении числа входов блока ***BufSupEx*** файлы на диске перезаписываются.

Поведение блока ***BufSupEx*** при ошибках записи описано в разделе 2.2.

5.2.3 Запись целочисленных уставок по протоколу Modbus

Для записи уставок по протоколу **Modbus** вход **inter** блока BufSupEx подключается к выходу блока протокола ***Modbus TCP Slave*** или ***Modbus RTU Slave*** из библиотеки ***paModbus***.

Мастер в сети Modbus может читать и записывать данные на диск. Для чтения блок ***BufSupEx*** реализует функцию **0x03**, для записи – **0x06** и **0x10**.

Вход **group** определяет Slave ID устройства (ID = **1** соответствует значению входа **0x100**).

Входы **dan** используются для записи уставок из программы контроллера. Для того, чтобы значение записалось на диск из программы, и его прочитал мастер сети Modbus, следует также подать импульс на соответствующий вход **czap**.

Входы **typ** определяют тип данных **dan**, при работе по Modbus могут принимать только значения **II**, **IO** (16-ти битный регистр), так как Modbus работает с целочисленными регистрами.

Сохранение по протоколу Modbus уставок с плавающей точкой рассмотрено в разделе 5.2.4.

Входы **adr** определяют адреса выделяемых регистров Modbus.

Входы **min** и **max** задают минимальное и максимальное возможное значение **dan**, при изменении значения из программы или мастером сети оно проверяется на условие соответствия этому диапазону. Значение, выходящее за этот диапазон, записано не будет.

Выходы **dan** отображают текущие значения уставок, сохраненных на диск.

Параметры на диске сохраняются в бинарных файлах с расширениями **.da1** и **.da2**.

ПРИМЕЧАНИЕ
При изменении числа входов блока ***BufSupEx*** файлы на диске перезаписываются.

Поведение блока ***BufSupEx*** при ошибках записи описано в разделе 2.2.

Подробно настройка обмена по протоколу Modbus описана в документе Обмен по протоколу Modbus. Библиотека paModbus.

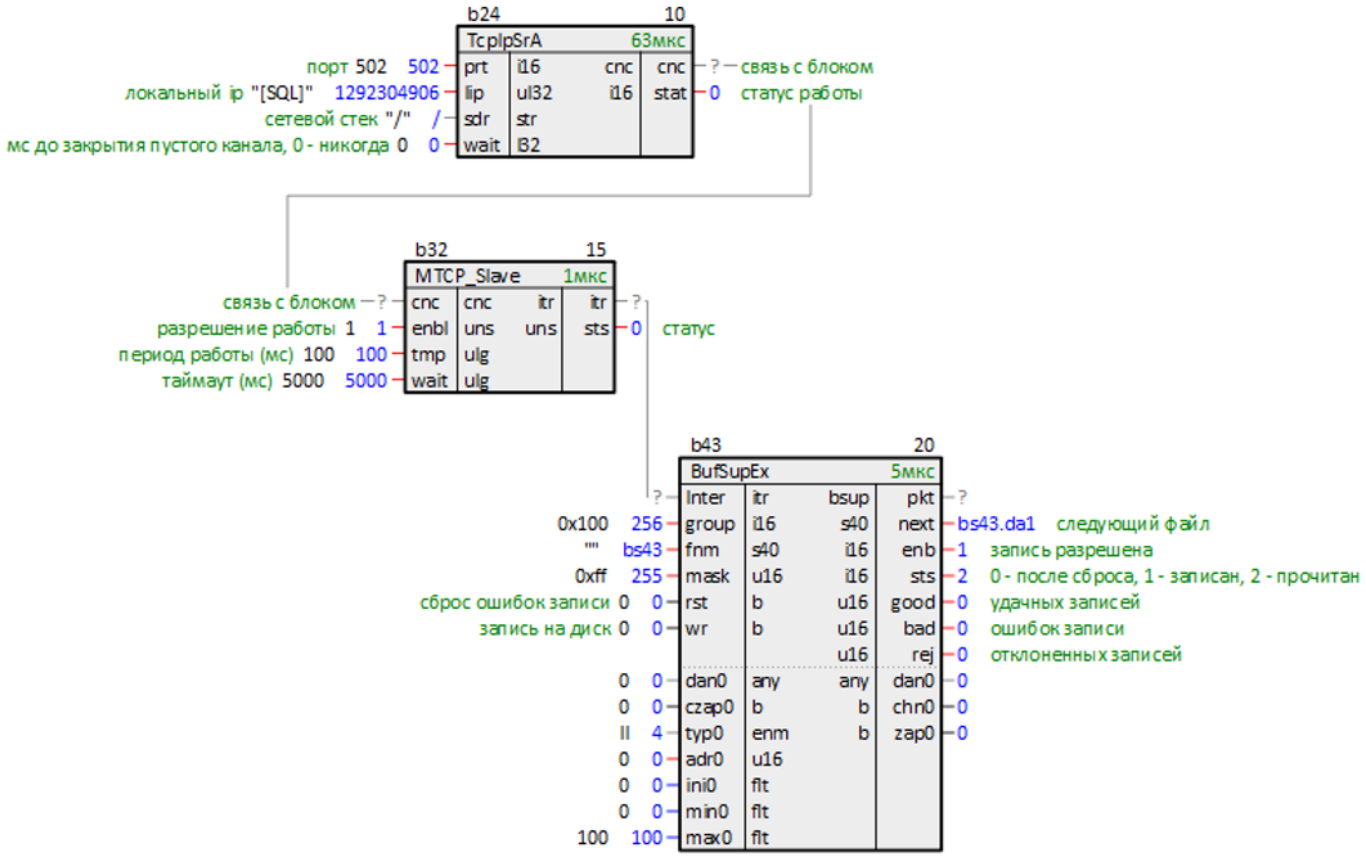


Рисунок 5.11 – Подключение BufSupEx к Modbus TCP Slave

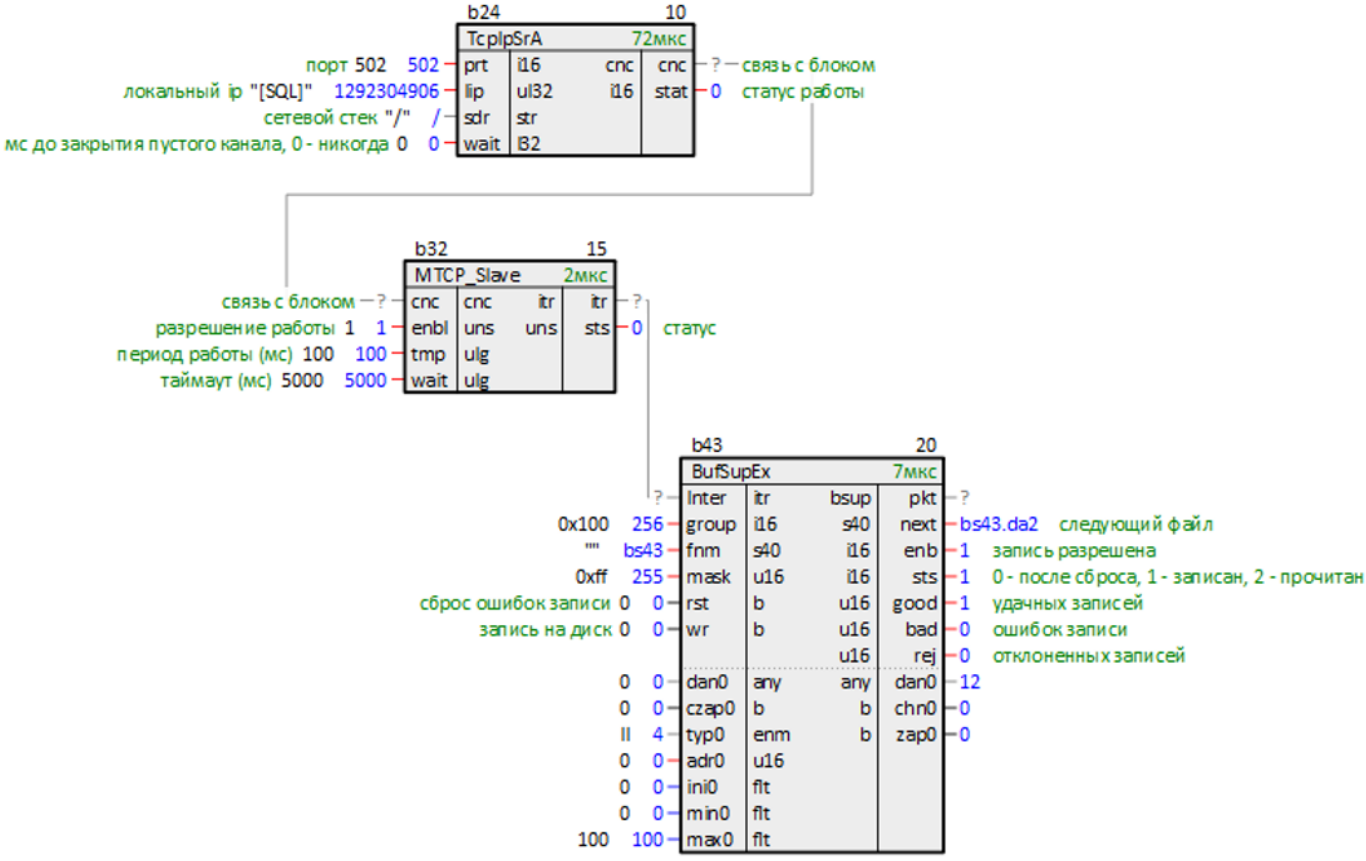


Рисунок 5.12 – Запись уставки мастером сети Modbus

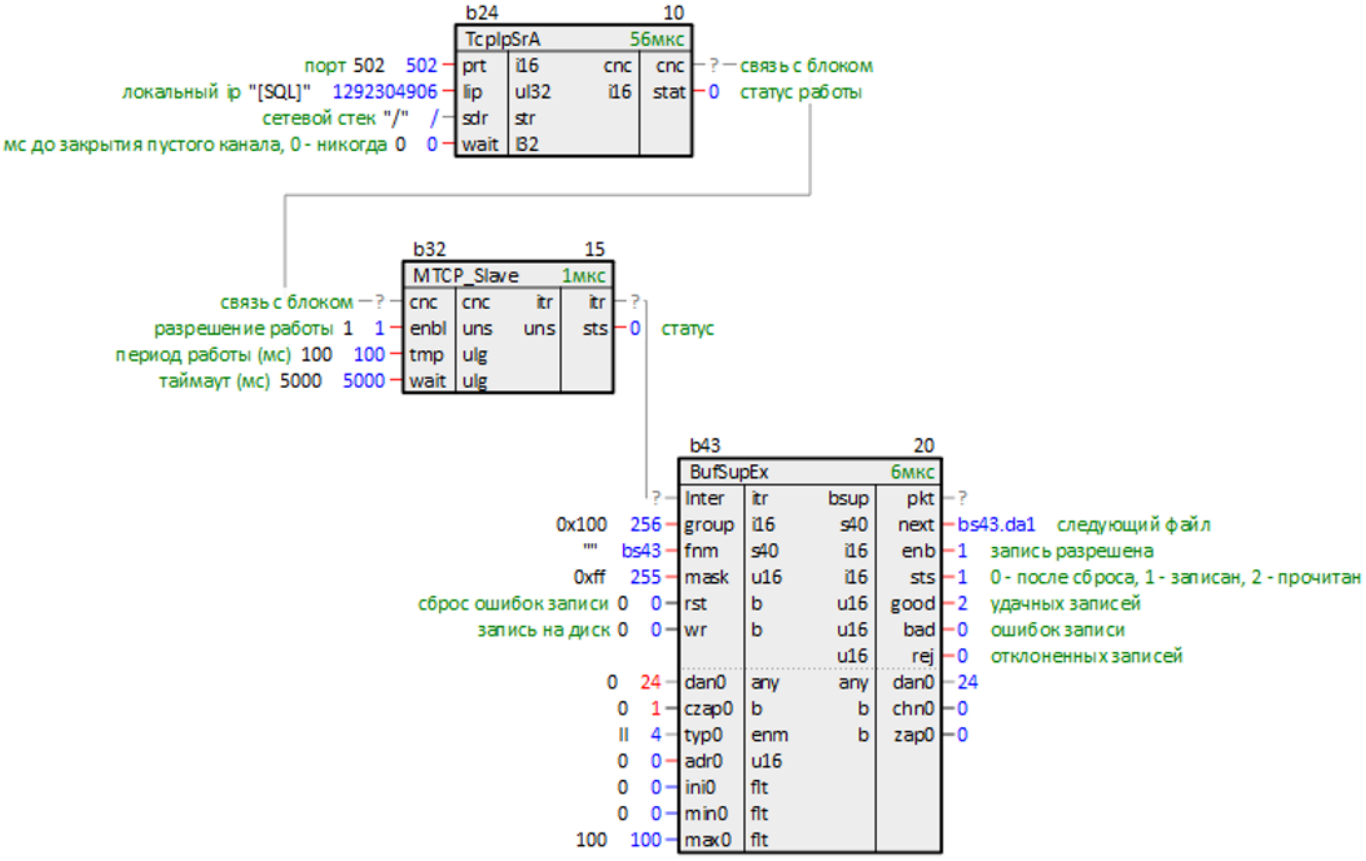


Рисунок 5.13 – Запись уставки из программы контроллера

5.2.4 Запись уставок с плавающей точкой по протоколу Modbus (BufSupFltEx)

Для записи уставок с плавающей точкой по протоколу **Modbus** используются блоки ***BufSupFltEx*** из библиотеки ***paModbus***.

Вход **inter** блока ***BufSupFltEx*** подключается к выходу блока протокола ***Modbus TCP Slave*** или ***Modbus RTU Slave*** из библиотеки ***paModbus***.

Мастер в сети Modbus может читать и записывать данные на диск. Для чтения блок ***BufSupFltEx*** реализует функцию **0x03**, для записи – **0x10**.

Вход **group** определяет Slave ID устройства (ID = **1** соответствует значение входа **0x100**).

Входы **dan** используются для записи уставок из программы контроллера. Для того, чтобы значение записалось на диск из программы, и его прочитал мастер сети Modbus, следует также подать импульс на соответствующий вход **czap**.

Входы **typ** определяют тип данных **dan**, могут принимать только значения **AI**, **AO** (вещественное число).

Входы **adr** определяют адреса выделяемых регистров Modbus. Для опроса каждого значения **dan** выделяется 2 регистра Modbus.

Входы **min** и **max** задают минимальное и максимальное возможное значение **dan**, при изменении значения из программы или мастером сети оно проверяется на условие соответствия этому диапазону. Значение, выходящее за этот диапазон, записано не будет.

Выходы **dan** отображают текущие значения уставок, сохраненные на диск.

Параметры на диске сохраняются в бинарных файлах с расширениями **.da1** и **.da2**.

ВНИМАНИЕ
При изменении числа входов блока ***BufSupFltEx*** файлы на диске перезаписываются.

Поведение блока ***BufSupFltEx*** аналогично блоку ***BufSupEx*** и описано в разделе 2.2.

Подробно настройка обмена по протоколу Modbus описана в документе Обмен по протоколу Modbus. Библиотека paModbus.

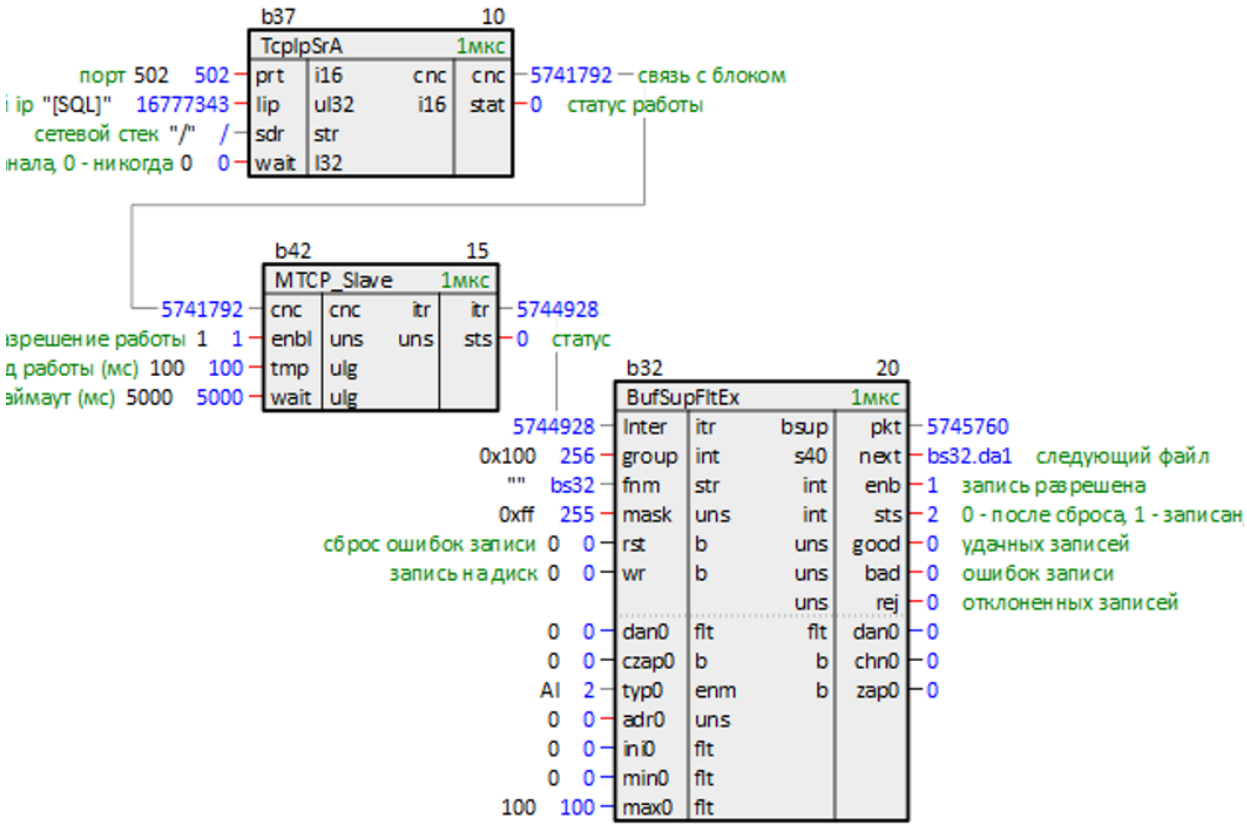


Рисунок 5.14 – Подключение *BufSupFltEx* к Modbus TCP Slave

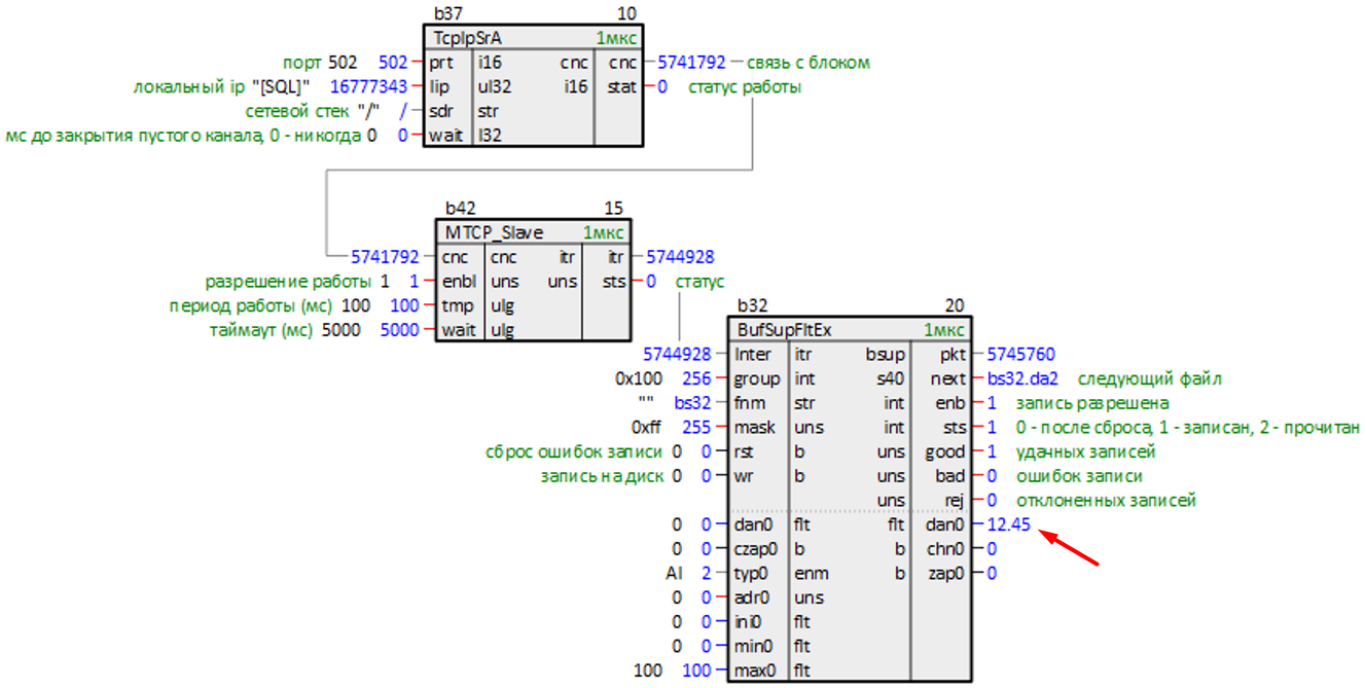


Рисунок 5.15 – Запись уставки мастером сети Modbus

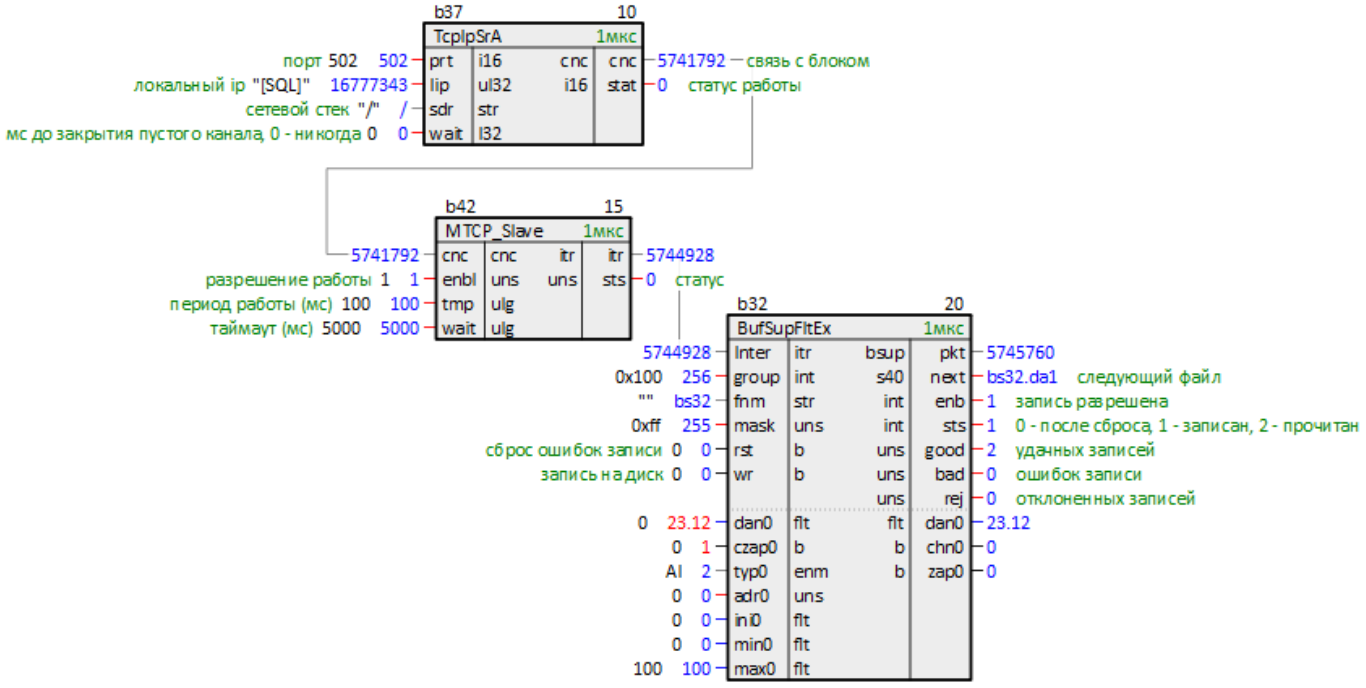


Рисунок 5.16 – Запись уставки из программы контроллера

6 Работа с счетчиком времени наработки (CounterMEx)

Для сохранения времени наработки устройств используется блок CounterMEx.

Для надежной сохранности данных одновременно на диске находятся два файла, соответствующие одному архиву. Файлы для сохранения чередуются.

На вход блока **fn** подается абсолютный путь и имя файла, куда будет осуществляться запись. Расширение файла игнорируется – на диск записываются бинарные файлы с расширениями **.da1** и **.da2**. Если оставить данное поле пустым – имя файла будет определяться автоматически по индексу блока, файл будет сохраняться в рабочую директорию контроллера.

Для начала отсчета времени наработки механизма следует подать **1** на вход **enbl**. Время наработки выводится на выход **cnt** в секундах. Число включений выводится на выход **cfrn**.

Чтобы записать текущее значение счетчика в файл, следует подать **1** на вход **ask**.

Если файл на диске существует, при перезапуске программы выходы **cnt** и **cfrn** инициализируются сохраненными значениями.

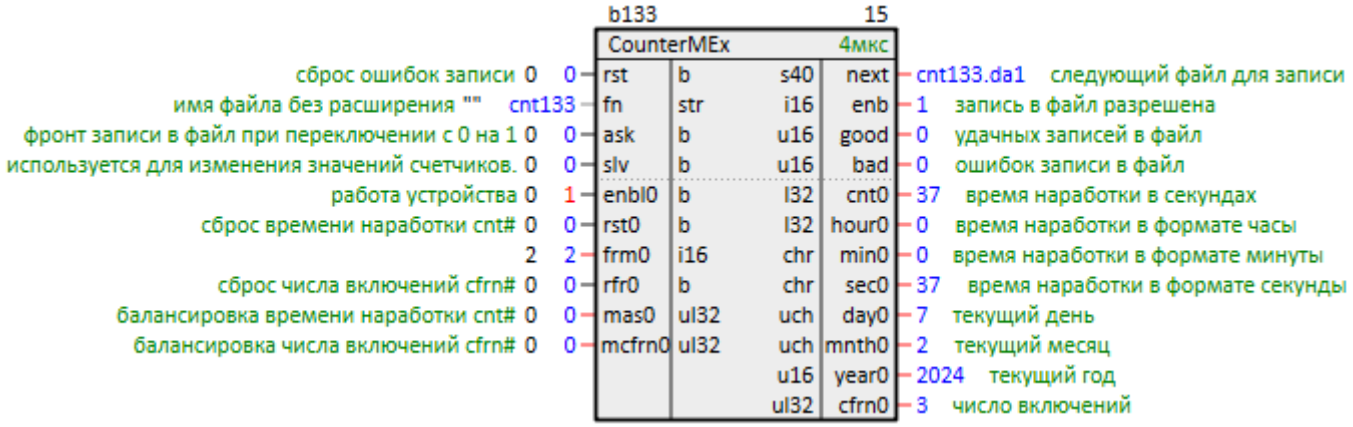


Рисунок 6.1 – Счет времени наработки устройства

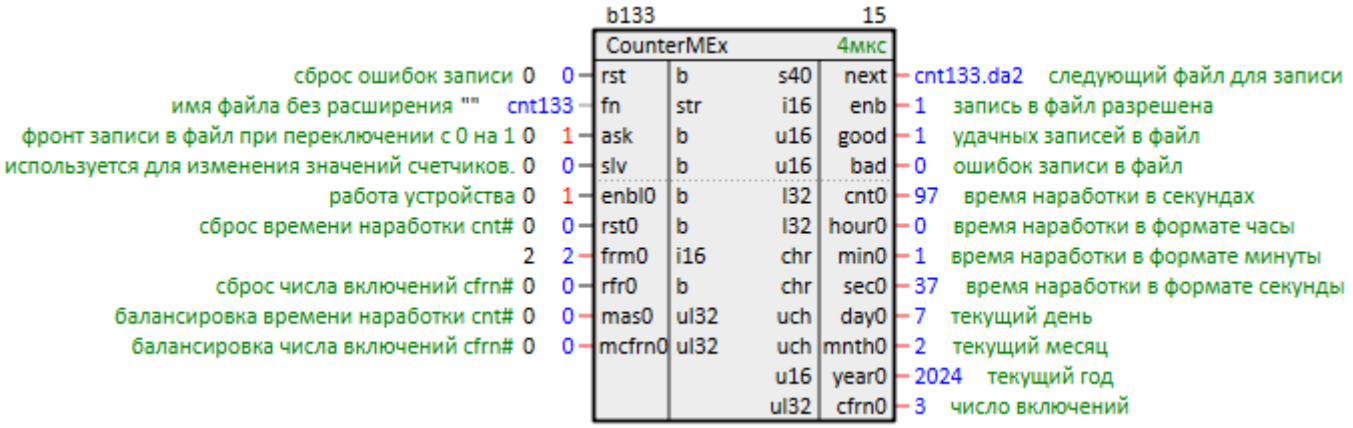


Рисунок 6.2 – Запись времени наработки устройства на диск

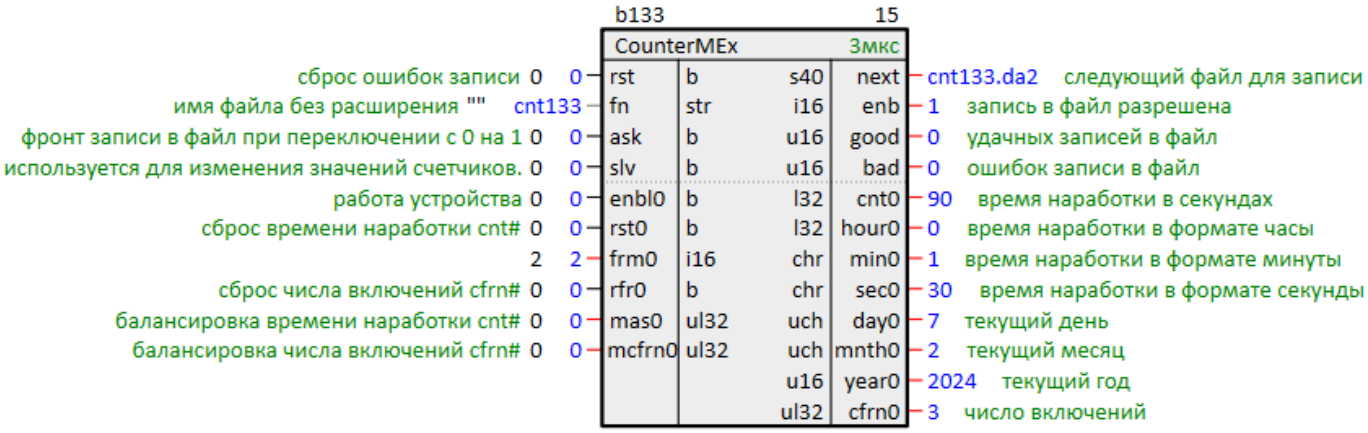


Рисунок 6.3 – Восстановление значения времени наработки после перезапуска программы

Поведение блока при возникновении ошибки записи описано в разделе 2.3.

При сохранении данных на внешний накопитель на вход **rst** можно завести выход флага монтирования накопителя блока 210-SD-USB. Сброс ошибок **rst** выполняется по фронту перехода из **0** в **1**.

ВНИМАНИЕ
При изменении числа входов блока ***CounterMEx*** файлы на диске перезаписываются.

7 Архивирование параметров на диск (FileDpLogger)

Для архивации параметров на диск контроллера используется блок FileDpLogger.

На вход блока **file_path** подается абсолютный путь, куда будет осуществляться сохранение файла.

Формат сохранения файла определяется входами блока **grp**, **frmt**, **meth**. Подробнее см. в разделе 4.1.

Для сохранения параметров на диск периодически в формате **.csv** следует установить следующие значения входов **grp = GROUP_ALLINONE**, **frmt = TO_MSD200**, **meth = INTERVAL**.

Период архивации задается на входе **intr** в миллисекундах.

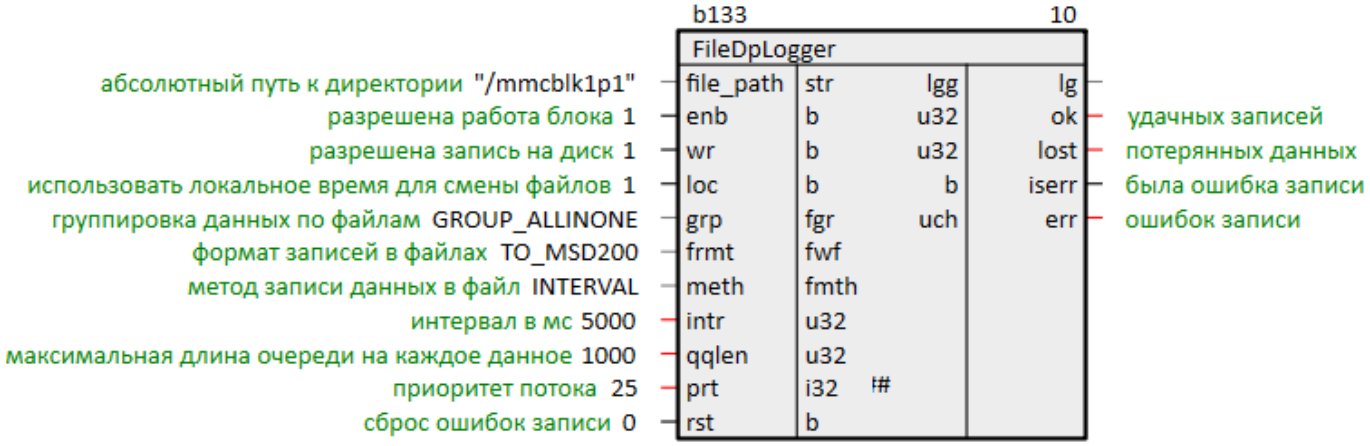


Рисунок 7.1 – Сохранение архива параметров на MicroSD-накопитель

Параметры, которые необходимо архивировать, добавляются в раздел ***Данные*** внутри блока.

Добавить параметр в раздел можно одним из следующих способов:

1. Открыть на одной странице блок ***FileDpLogger***, на другой странице блок с входом/выходом, который необходимо добавить. Выделить вход/выход и с нажатым **Ctrl** перетащить его на блок ***FileDpLogger***. Отпустить, выбрать команду ***Добавить***.
2. Открыть блок ***FileDpLogger*** в дереве (со страницы это проще всего сделать командой ***Показать в дереве***), раскрыть его. Вход/выход перетащить в раздел ***Данные***, выбрать команду ***Добавить***.

Входам/выходам в проекте, которые необходимо архивировать, следует задать свойство ***Полный алиас***. Алиас должен быть уникальным внутри раздела ***Данные***. Алиасы отображаются в заголовке таблицы.

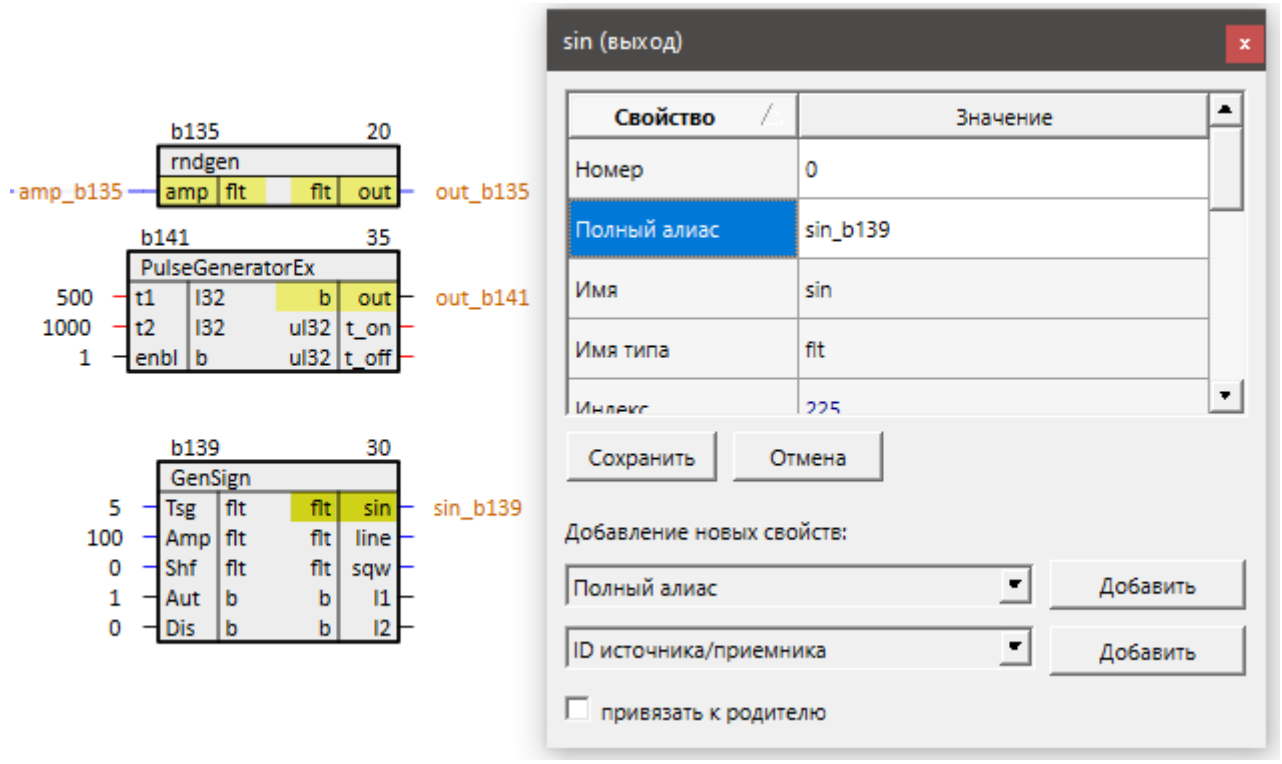


Рисунок 7.2 – Свойство Полный алиас параметров для архивации

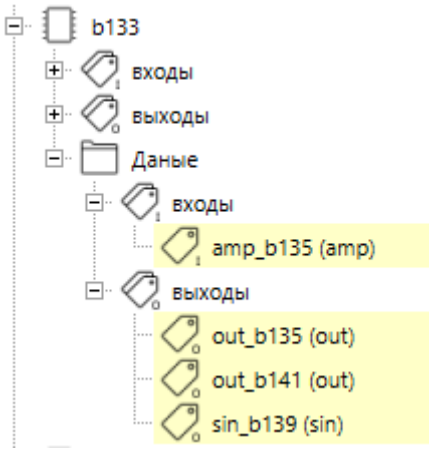


Рисунок 7.3 – FileDpLogger в дереве проекта

Для запуска архивации следует подать **1** на входы **enb** и **wr** блока **FileDpLogger**. Количество удачных записей выводится на выходе блока **ok**.

Каждый день создается новый файл архива.

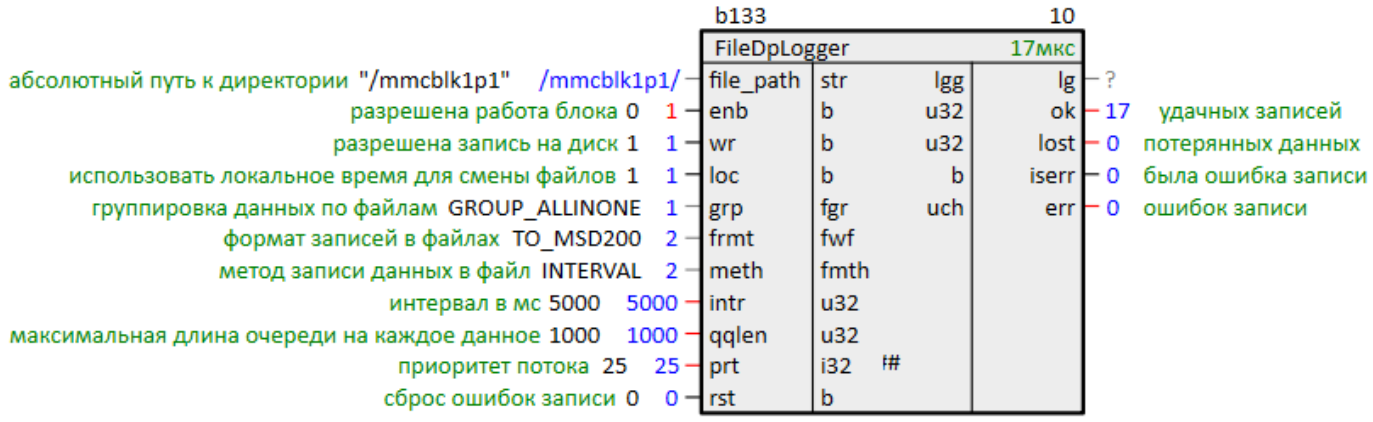


Рисунок 7.4 – Сохранение архива параметров на MicroSD-накопитель

Просмотреть полученный архив параметров в формате **.csv** можно через программу MS Excel. В первом столбце находится метка времени сохранения параметров, в следующих – сохраненные значения параметров.

	A	B	C	D	E
1	Метка времени	amp_b135	out_b135	out_b141	sin_b139
2	5:38:48	10	9,50894	0	-30,901819
3	5:38:53	10	7,51912	1	-44,838436
4	5:38:58	10	5,54866	1	-59,790604
5	5:39:03	10	0,91865	0	-71,153664
6	5:39:08	10	9,90446	0	-83,752853
7	5:39:13	10	6,57403	1	-89,940559
8	5:39:18	10	0,6504	0	-95,486473
9	5:39:23	10	1,88465	0	-99,211472
10	5:39:28	10	3,61597	0	-100
11	5:39:33	10	4,45936	1	-98,865143
12	5:39:38	10	7,66364	0	-95,486397
13	5:39:43	10	0,89453	0	-89,940453
14	5:39:48	10	8,47347	1	-80,901619
15	5:39:53	10	2,78563	0	-71,153488
16	5:39:58	10	0,73156	0	-57,757179
17	5:40:03	10	6,73827	0	-44,838219
18	5:40:08	10	2,40944	1	-30,901585
19	5:40:13	10	2,38349	0	-11,285517
20	5:40:18	10	4,79851	0	1,256726
21	5:40:23	10	3,07189	1	18,738253
22	5:40:28	10	7,12611	0	33,28207
23	5:40:33	10	2,33815	0	49,272846

2024_04_18

Рисунок 7.5 – Просмотр архива параметров в MS Excel

При описанной настройке блока **FileDpLogger** архивы параметров можно отобразить и сохранить в виде графиков в программе График ОВЕН МСД200.

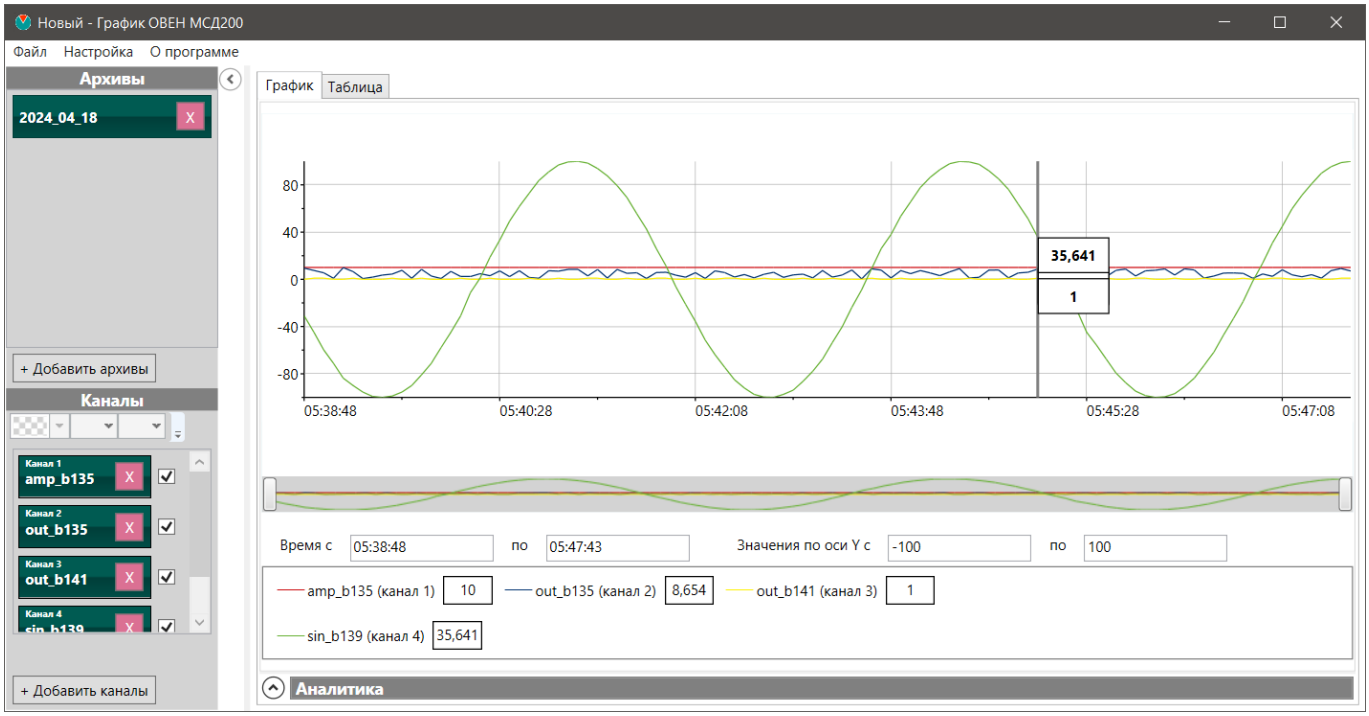


Рисунок 7.6 – Отображение архива параметров в программе График ОВЕН МСД200

Поведение блока при возникновении ошибки записи описано в разделе 4.1.

При сохранении данных на внешний накопитель на вход **rst** можно завести выход флага монтирования накопителя блока 210-SD-USB. Сброс ошибок **rst** выполняется по фронту перехода из **0** в **1**.

8 Работа с «черным ящиком» (BlackBox)

Блок «черный ящик» BlackBox сохраняет значения выбранных параметров в файл в течение заданного времени «до» события и в течение заданного времени «после» события. Этот блок совмещает работу блоков блоков RamFRLogger и ComtradeLogger.

На вход блока **name** подается абсолютный путь, куда будет осуществляться сохранение файла и начало имени файла. Если путь не указан – файл сохраняется в рабочую директорию контроллера.

Входы блока **before** и **after** определяют отрезок времени, в течение которого блок сохраняет значения параметров соответственно «до» и «после» события.

Вход **every** определяет периодичность записи параметров на диск и задается как количество таймерных промежутков (свойство модуля ***Таймерный промежуток***). Например, если таймерный промежуток установлен **20 мс** и **every = 5**, запись будет производиться каждые **100 мс**.

Параметры, которые необходимо сохранять по событию, добавляются в раздел ***Данные*** внутри блока.

Добавить параметр в раздел можно одним из следующих способов:

1. Открыть на одной странице блок ***BlackBox***, на другой странице блок с входом/выходом, который необходимо добавить. Выделить вход/выход и с нажатым **Ctrl** перетащить его на блок ***BlackBox***. Отпустить, выбрать команду ***Добавить***.
2. Открыть блок ***BlackBox*** в дереве (со страницы это проще всего сделать командой ***Показать в дереве***), раскрыть его. Вход/выход перетащить в раздел ***Данные***, выбрать команду ***Добавить***.

Количество точек для сохранения рассчитывается на выходе блока **pnum**.

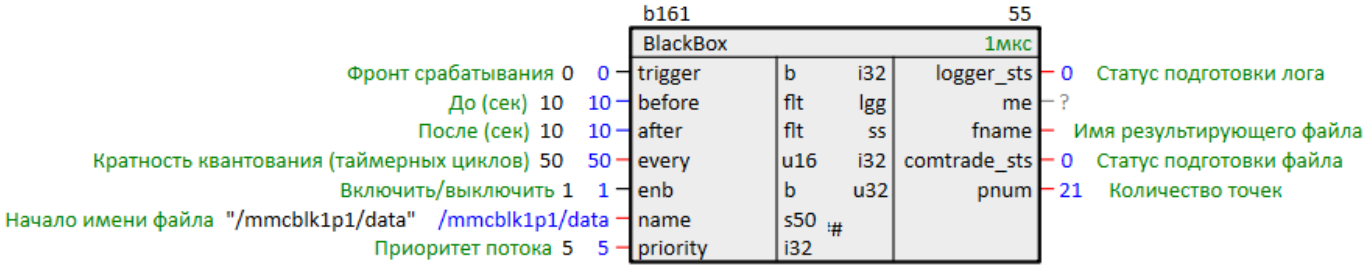


Рисунок 8.1 – Сохранение параметров на MicroSD-накопитель по событию

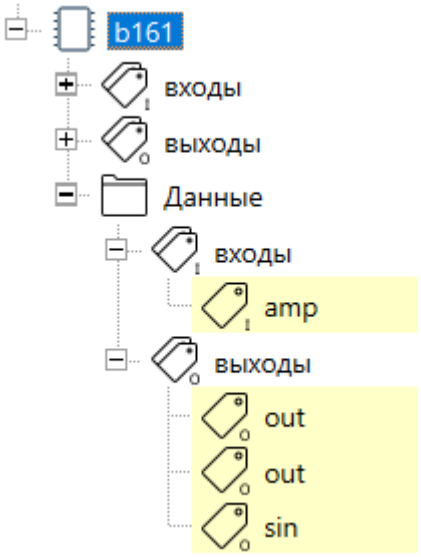


Рисунок 8.2 – BlackBox в дереве проекта

По сигналу **1** на входе **trigger** происходит сохранение массива значений параметров в ОЗУ.

Начало записи параметров на диск происходит при условии, что **enb = 1**.

На время подготовки лога к записи на диск на выходе **logger_sts** должно быть значение **1**.

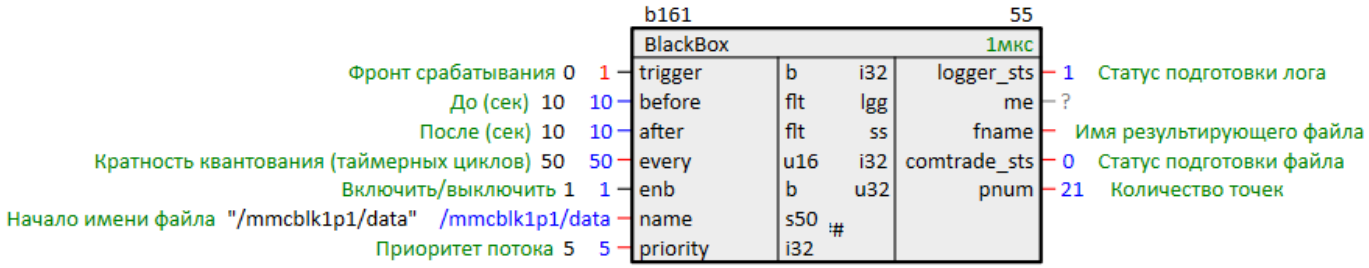


Рисунок 8.3 – Сохранение параметров на MicroSD-накопитель по событию. Подготовка лога

По окончании записи блок задает на выходе **logger_sts** значение **0**. На выходе **fname** отображается имя файла, сохраненного на диск, на выходе **comtrade_sts** на один цикл работы блока задается **1**.

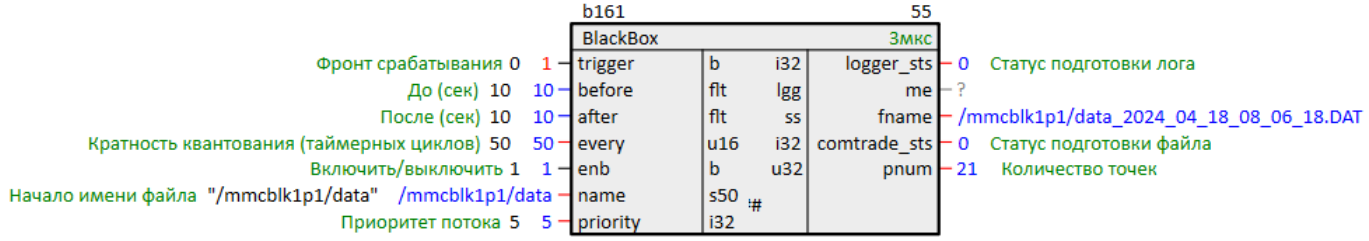


Рисунок 8.4 – Сохранение параметров на MicroSD-накопитель по событию. Запись параметров на диск

По окончании записи на диске контроллера формируются два файла:

- ***.dat** – набор значений параметров, накопленных за временной отрезок;
- ***.cfg** – служебная информация о событии.

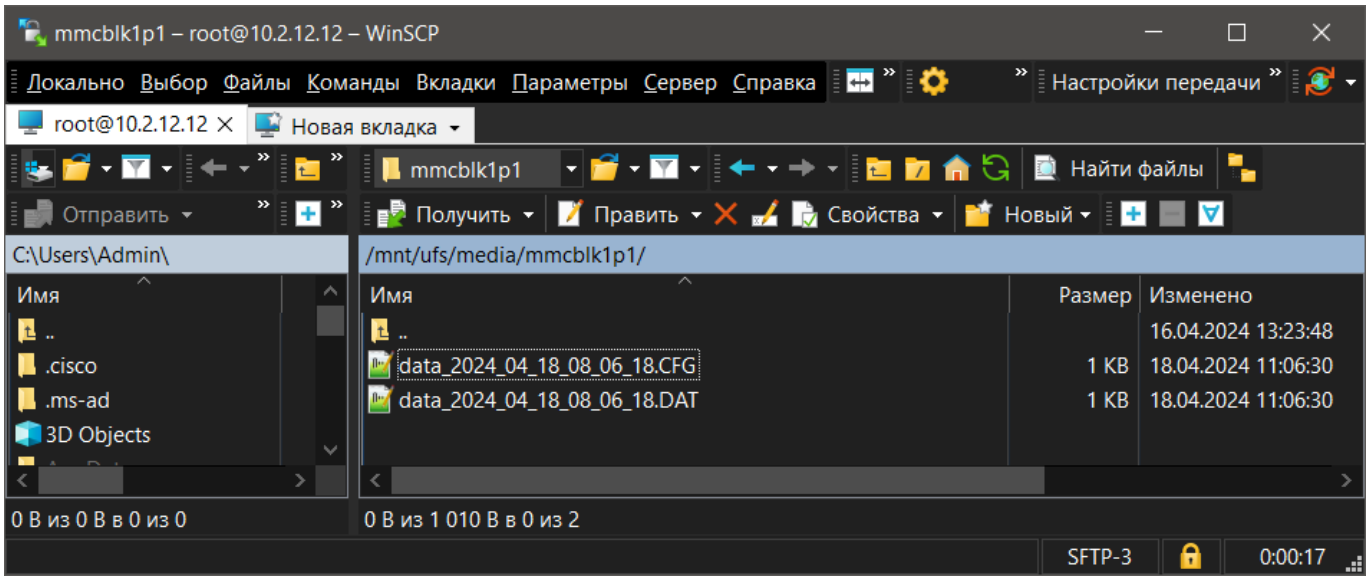


Рисунок 8.5 – Файлы на диске контроллера

Просмотреть файлы можно в текстовом редакторе или с помощью программы Notepad++.

В файле ***. dat** сохраняются: номер точки, метка времени в микросекундах от момента сохранения параметров «до» события, значения параметров.

В файле ***.cfg** можно посмотреть точное время возникновения сигнала на входе **trigger** (без учета часового пояса).

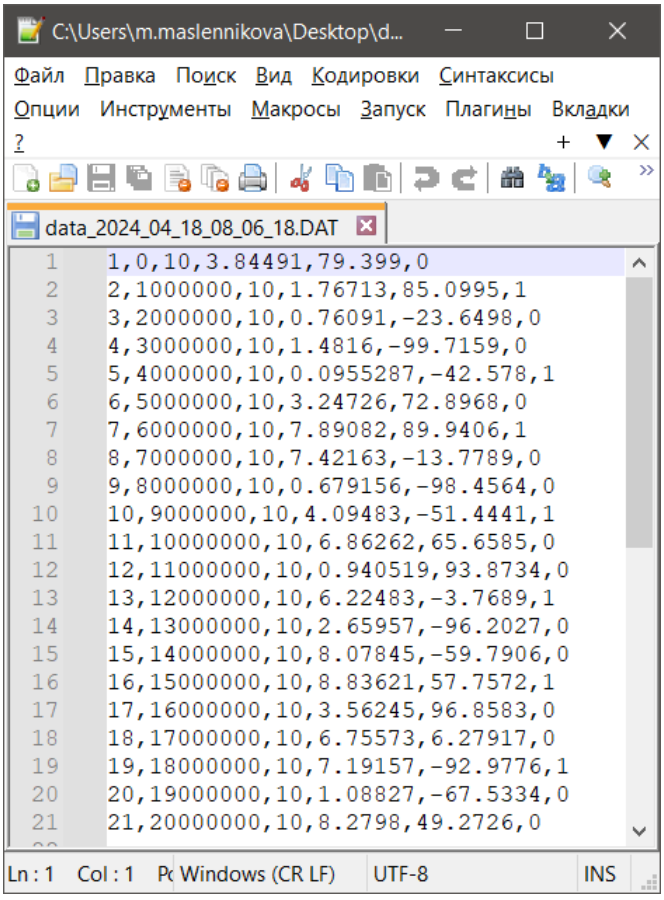


Рисунок 8.6 – Файл с данными *.dat

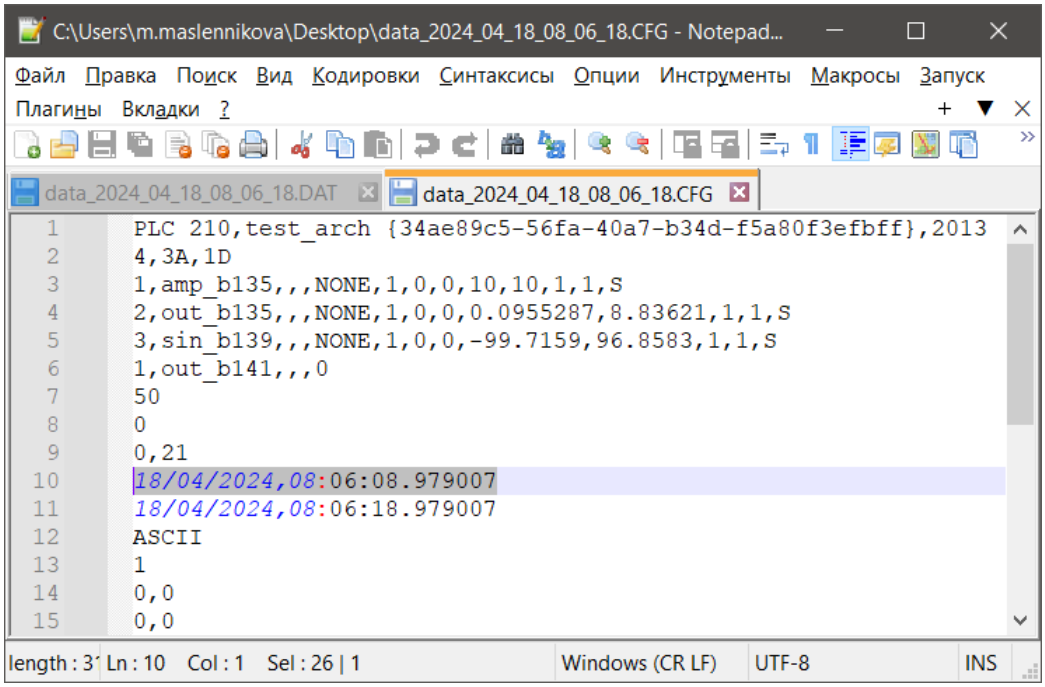


Рисунок 8.7 – Файл со служебной информацией *.cfg

Россия, 111024, Москва, 2-я ул. Энтузиастов, д. 5, корп. 5
тел.: +7 (495) 641-11-56, факс: (495) 728-41-45
тех. поддержка 24/7: 8-800-775-63-83, support@owen.ru
отдел продаж: sales@owen.ru
Веб-сайт ООО "ПромАвтоматика-Софт": www.pa.ru
рег.:1-RU-134367-1.1