Библиотека SimpleVisuSchedule

07.03.2025
Версия библиотеки: 3.5.17.1
Версия документа: 1.0

https://oscat.ru/

Оглавление

Введение

Библиотека **SimpleVisuSchedule** предназначена для управления оборудованием по расписанию и настройке расписания из визуализации. Актуальная версия библиотеки, примера её использования и данного документа доступны по ссылке: https://github.com/CodesysOneLove/CODESYS-V3.5---SimpleVisuSchedule

В основе библиотеки лежит две основные концепции:

- баланс между функциональностью и простотой реализации;
- открытость и возможность доработки библиотеки конечными пользователями (по этой же причине исходный код снабжен подробными комментариями).

Библиотека разработана в среде **CODESYS V3.5 SP17 Patch 3** и может использоваться в данной или более поздних версиях среды. Разработка и тестирование проводились в 32-битной версии IDE и рантайма, но предполагается, что она будет успешно работать и в 64-битном окружении.

Библиотека доступна в открытых исходных кодах (в виде файла с расширением .**library**) и распространяется по лицензии MIT. Общий смысл лицензии – пользователь может использовать библиотеку и её исходный код так, как посчитает нужным[1], но должен отдавать себе отчёт, что авторы библиотеки не могут ни при каких обстоятельствах нести ответственность за последствия использования.

Вместе с библиотекой распространяется пример её применения (в виде файлов с расширением .**project** и .**projectarchive**). Он описан в п. 1.9.

Библиотека имеет зависимость от следующих библиотек:

- Standard;
- Standard64;
- Util;
- CAA Memory;
- SysFile;
- библиотек визуализации CODESYS (в частности, Visu Elems);
- OwenStringUtils (версии **3.5.4.9**).

Все перечисленные библиотеки входят в дистрибутив CODESYS, за исключением **OwenStringUtils**: она приложена к файлам библиотеки, а также может быть загружена с сайта или ftp-сервера компании ОВЕН.

Разработчики библиотеки: Евгений Кислов, Михаил Троицкий

[1] Пожелание (но не требование) – сохранять при этом упоминание авторов библиотеки.

1. Руководство пользователя

1.1. Основная информация

Библиотека **SimpleVisuSchedule** реализует следующий функционал:

- настройку расписания из визуализации CODESYS;
- импорт/экспорт расписания в бинарном и текстовом (csv) формате с возможностью загрузки и выгрузки файлов расписания на устройствах, поддерживающих функционал **Visu File Transfer**;
- обработку расписания (формированию битовой маски с установкой бит, соответствующих активным в данный момент интервалам расписания).

1.2. Установка библиотеки и её добавление в проект CODESYS

Для установки библиотеки следует в среде CODESYS использовать команду **Инструменты – Репозиторий библиотек**, нажать кнопку **Установить** и выбрать данную библиотеку через проводник Windows (в правом нижнем углу окна выбора установите фильтр **Все файлы**).

Для добавления библиотеки в пользовательский проект следует открыть через дерево проекта компонент **Менеджер библиотек**, нажать кнопку **Добавить** и выбрать из списка данную библиотеку.

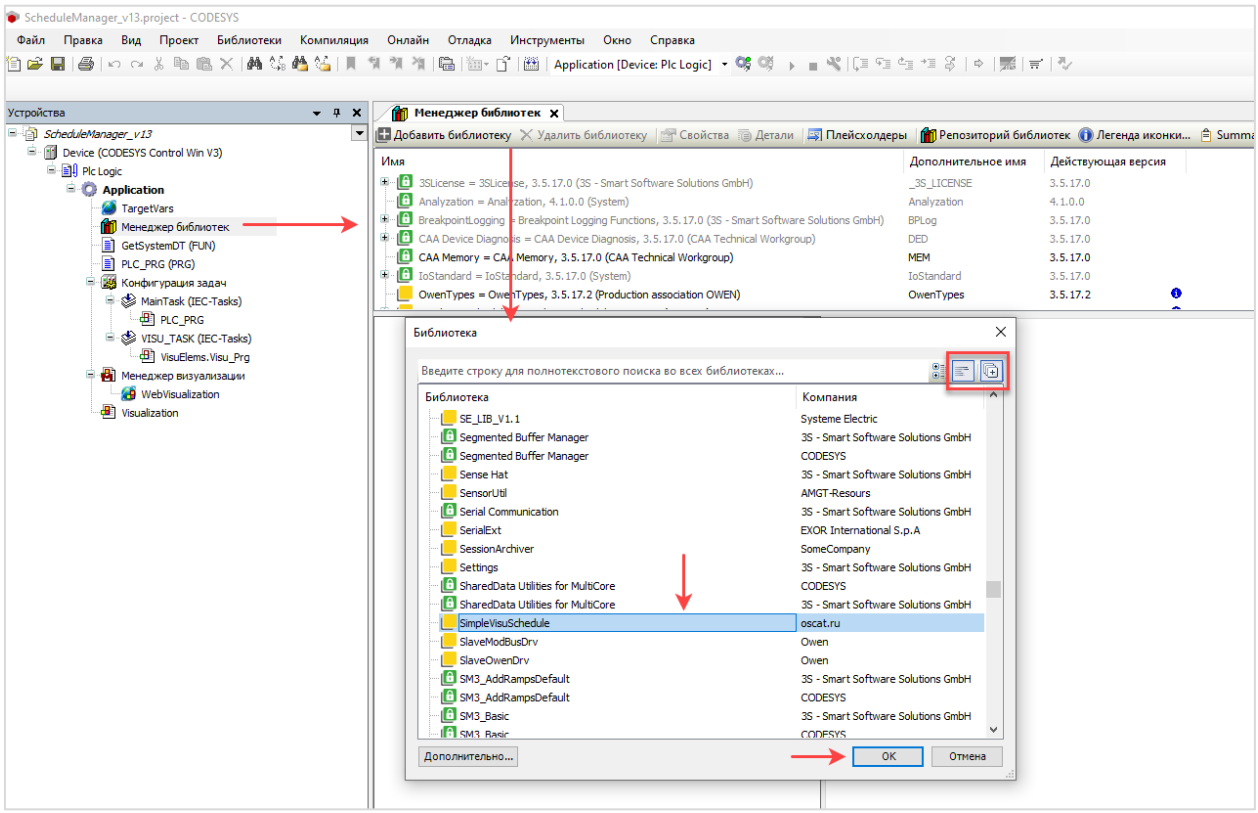

Рисунок 1.2.1 – Добавление библиотеки в проект CODESYS

1.3. Структура библиотеки и алгоритм использования

С точки зрения пользователя библиотека содержит три основных сущности:

- функциональный блок ScheduleManager, реализующий бизнес-логику (backend);
- фрейм frmScheduleManager, реализующий графический интерфейс (frontend);
- структуры и перечисления, используемые функциональным блоком.

Для использования библиотеки требуется:

- добавить в менеджер библиотек проекта библиотеку **SimpleVisuSchedule**;
- объявить экземпляр ФБ **ScheduleManager**;
- организовать в одной из программ проекта циклический вызов этого экземпляра с передачей ему на входы нужных значений;
- добавить в визуализацию проекта фрейм **frmScheduleManager**;
- в конфигурации фрейма привязать экземпляр ФБ **ScheduleManager**.

См. также пример использования библиотеки, который распространяется вместе с библиотекой.

1.4. Типы данных (структуры и перечисления)

Основным понятием библиотеки является **интервал** – объект расписания, который связывает дату, интервал времени и набор бит маски выходов, которые должны быть активны в этот момент.

Интервал описывается структурой **SCHED_INTERVAL**:

Таблица 1.4.1 – Описание полей структуры **SCHED_INTERVAL**

Поле	Тип данных	Описание
eState	SCHED_INTERVAL_STATE	Состояние интервала
eType	SCHED_INTERVAL_TYPE	Тип интервала
ausiInterval	ARRAY [0..3] OF USINT	Интервал времени
ausiDay	ARRAY [0..1] OF USINT	День (или дни), в которые применяется интервал
dwOutputsMask	DWORD	Битовая маска выходов, которые должны быть активны в течение интервала

Поле **eState** определяет состояние интервала и описывается перечислением **SCHED_INTERVAL_STATE**:

- **NOT_EXIST** – интервал не существует (например, ещё не создан или удалён);
- **ENABLED** – интервал существует и обрабатывается блоком;
- **DISABLED** – интервал существует, но не обрабатывается блоком.

Поле **eType** определяет тип интервала и описывается перечислением **SCHED_INTERVAL_TYPE**:

- **NOT_EXIST** – тип ещё не задан;
- **COMMON_DAYS** – недельное расписание. Интервал применяется в дни недели, определяемые битовой маской **ausiDay[0]**, в которой бит 0 – понедельник, бит 6 – воскресенье; бит 7 является незначимым;
- **SPECIAL_DATE** – особая дата. Интервал применяется в дату, определяемую месяцем **ausiDay[0]** и днём **ausiDay[1]**. При этом в данную дату не обрабатываются интервалы с типом **COMMON_DAYS**;
- **SPECIAL_DATE_PLUS** – аналогично **SPECIAL_DATE**, но в данную дату интервалы с типом **COMMON_DAYS** будут обрабатываться.

То есть назначение поля **ausiDay** определяется типом интервала **eType**. Присутствующие в библиотеке три типа позволяют достаточно гибко настраивать расписание:

- для «стандартного» расписания (например, соответствующего рабочей неделе или выходным) подходит тип **COMMON_DAYS**;
- для отступления от недельного расписания в конкретные дни (например, в праздничные дни) подходит тип **SPECIAL_DATE**;
- для дополнения недельного расписания в конкретные дни подходит тип **SPECIAL_DATE_PLUS**.

Поле **ausiInterval** определяет интервал времени:

- **ausiInterval[0]** – час начала интервала;
- **ausiInterval[1]** – минута начала интервала;
- **ausiInterval[2**] – час окончания интервала;
- **ausiInterval[3]** – минута окончания интервала.

Соответственно, минимальная длительность интервала времени – минута. Подразумевается, что момент начала интервала раньше его окончания и что максимально возможный час интервала – это 23; то есть интервал принадлежит одним суткам. Если оборудование должно включаться в 22:00 и выключаться в 04:00 – то нужно настроить два отдельных интервала (см. подробности в п. 1.11, вопрос 3).

Поле **dwOutputsMask** определяет битовую маску выходов, которые должны быть активны в течение интервала. В течение интервала в выходе **dwOutputsMask** ФБ **ScheduleManager** будут установлены соответствующие биты. В момент окончания интервала они будут сброшены.

Структура **SCHED_INTERVAL** оптимизирована с точки зрения занимаемого объёма памяти – это позволяет разместить массив этих структур в энергонезависимой (**RETAIN-** или **PERSISTENT**) памяти контроллера (при её наличии), объём которой обычно невелик.

В фрейме **frmScheduleManager** интервалы отображаются следующим образом:

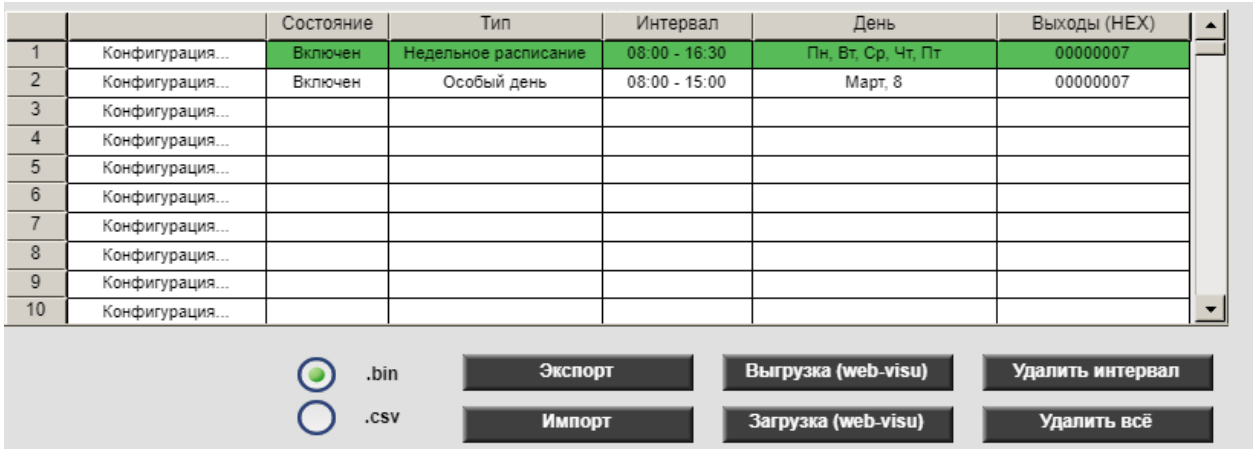

Рисунок 1.4.1 – Отображение интервалов в фрейме **frmScheduleManager**

К таблице привязан указатель на массив структур **SCHED_INTERVAL_VISU**. Каждый элемент массива соответствует строке таблицы:

Таблица 1.4.2 – Описание полей структуры **SCHED_INTERVAL_VISU**

Поле	Тип данных	Описание
usiConfigColumn	USINT	Индекс списка текстов для столбца «Конфигурация…». Нужен только для отображения этого столбца
wsState	WSTRING(10)	Состояние интервала
wsType	WSTRING(20)	Тип интервала
sInterval	STRING(15)	Интервал в формате «hh:mm – hh:mm»
wsDay	WSTRING(28)	Перечень дней (для типа **COMMON_DAYS**) или дата в формате «Название месяца, номер дня» (для типа **SPECIAL_DATE** и **SPECIAL_DATE_PLUS**)
sOutputsMask	STRING(11)	Битовая маска выходов, которые должны быть активны в течение интервала, в виде HEX-значения с ведущими нулями
xCurrentInterval	BOOL	Флаг активности данного интервала. Активные интервалы выделяются в таблице заданным цветом

Таким образом, **SCHED_INTERVAL_VISU** обеспечивает визуальное представление **SCHED_INTERVAL**. Строковые типы данных использованы по той причине, что они позволяют отобразить «нулевое» значение поля в виде пустой строки. Например, если использовать для отображения битовой маски тип **DWORD**, то у всех строк таблицы (включая те, которые соответствуют ещё не созданным или удалённым интервалам) в этом столбце отображалось бы значение «0».

Перечисление **SCHED_ERROR**, содержащее коды ошибок ФБ **ScheduleManager**, будет описано в следующем пункте.

1.5. Функциональный блок ScheduleManager

Функциональный блок **ScheduleManager** реализует бизнес-логику расписания и может обслуживать до **255** интервалов. Пользователь может использовать в своём проекте произвольное количество экземпляров ФБ. Экземпляр ФБ должен вызываться циклически.

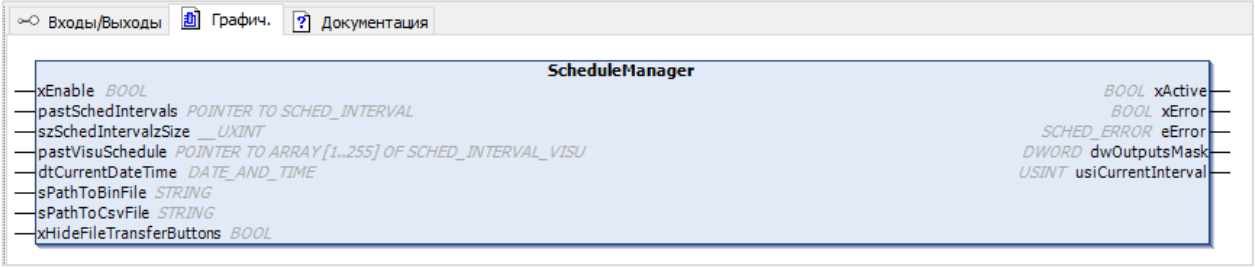

Рисунок 1.5.1 – Входы и выходы ФБ **ScheduleManager**

Таблица 1.5.1 – Описание входов и выходов ФБ **ScheduleManager**

Поле	Тип данных	Описание
Входы		
xEnable	BOOL	Вход управления блоком
pastSchedIntervals	POINTER TO SCHED_INTERVAL	Указатель на массив бинарных данных интервалов
szSchedIntervalsSize	__UXINT	Размер массива бинарных данных интервалов в байтах
pastVisuSchedule	POINTER TO ARRAY [1..255] OF SCHED_INTERVAL_VISU	Указатель на массив интервалов расписания, отображаемый в фрейме frmScheduleManager (см. рис. 1.6.1)
dtCurrentDateTime	DT	Текущие дата и время контроллера
sPathToBinFile	STRING	Путь к бинарному файлу расписания (используется для его импорта и экспорта)
sPathToCsvFile	STRING	Путь к csv файлу расписания (используется для его импорта и экспорта)
xHideFileTransferButtons	BOOL	Флаг скрытия кнопок загрузки и выгрузки файлов расписания через web-визуализацию (см. п. 1.8)
Выходы		
xActive	BOOL	**TRUE** – блок находится в работе
xError	BOOL	**TRUE** – в процессе запуска блока произошла ошибка
eError	SCHED_ERROR	Код ошибки
dwOutputsMask	DWORD	Битовая маска выходов, управляемая по расписанию
usiCurrentInterval	USINT	Индекс текущего активного интервала в массиве, размещённом по указателю **pastVisuSchedule**. Если одновременно активно несколько интервалов – то отображается индекс последнего из них. Если ни один интервал не является активным, то выход имеет значение **0**

По переднему фронту входа **xEnable** производится запуск блока. В процессе запуска производится валидация значений некоторых его входов. Если валидация прошла успешна – то выход **xActive** принимает значение **TRUE**, и блок начинает работу. Если при валидации обнаружены ошибки – то выход **xError** принимает значение **TRUE**, а на выходе **eError** отображается код ошибки из перечисления **SCHED_ERROR**:

- **INVALID_POINTER** – на вход **pastSchedIntervals** и/или **pastVisuSchedule** передан нулевой указатель;
- **INVALID_DATASIZE** – значение входа **szSchedIntervalzSize** является некорректным (равно **0**, или не является кратным **SIZEOF(SCHED_INTERVAL)**, или превышает максимально допустимое значение).

Блок использует значения своих входов «напрямую» – то есть они не должны меняться, пока он запущен.

По заднему фронту входа **xEnable** работа блока прекращается, и все его выходы принимают «нулевые» значения.

На вход **pastSchedIntervals** следует передать адрес массива структур SCHED_INTERVAL. Этот массив разумно объявить в области энергонезависимых переменных (**RETAIN-** или **PERSISTENT**). Число элементов не должно превышать **255**. В качестве начального индекса удобно использовать **1**.

На вход **szSchedIntervalsSize** следует передать размер массива, размещённого по указателю **pastSchedIntervals**, в байтах. Для определения размера удобно использовать оператор **SIZEOF**.

На вход **pastVisuSchedule** следует передать адрес массива структур SCHED_INTERVAL_VISU. Число элементов этого массива обязательно должно совпадать с числом элементов массива, размещённого по указателю **pastSchedIntervals** (и оно может быть меньше **255**, несмотря на тип входа). Этот массив нет смысла объявлять в области энергонезависимых переменных, потому что при запуске блока он формируется на основе массива, размещенного по указателю **pastVisuSchedule**. В качестве начального индекса удобно использовать **1**.

Массив, размещённый по указателю **pastVisuSchedule**, может быть заполнен тремя способами:

- из визуализации CODESYS (см. п. 1.6);
- путём импорта файла расписания (см. п. 1.7);
- из кода программы (см. п. 1.11, вопрос 6).

На вход **dtCurrentDateTime** следует передать текущее значение системного времени в виде переменной типа **DT** (и это значение должно обновляться. В зависимости от конкретного контроллера – способ получения системного времени может отличаться. В примере демонстрируется достаточно универсальный способ получения системного времени с помощью библиотеки SysTimeRtc из дистрибутива CODESYS.

На входы **sPathToBinFile** и **sPathToCsvFile** можно передать пути к файлам расписания в бинарном и текстовом форматах. Это позволяет реализовать экспорт и импорт расписания. См. подробности в п. 1.7.

Вход **xHideFileTransferButtons** позволяет скрыть кнопки загрузки и выгрузки файлов расписания через web-визуализацию. См. подробности в п. 1.8.

В процессе работы блока раз в секунду происходит проверка всех настроенных интервалов. Для каждого из активных интервалов в маске **dwOutputsMask** устанавливаются соответствующие биты. При этом значения остальных бит не меняются – то есть одновременно могут быть активны несколько интервалов, связанных с разными выходами. Если интервал перестаёт быть активным – то связанные с ним выходы маски отключатся; но при этом – если какие-то из этих выходов продолжают устанавливаться другим активным интервалом, то «прощёлкивания» не произойдёт.

1.6. Фрейм frmScheduleManager

Фрейм **frmScheduleManager** используется для работы с расписанием в визуализации.

Входом-выходом фрейма является экземпляр ФБ ScheduleManager.

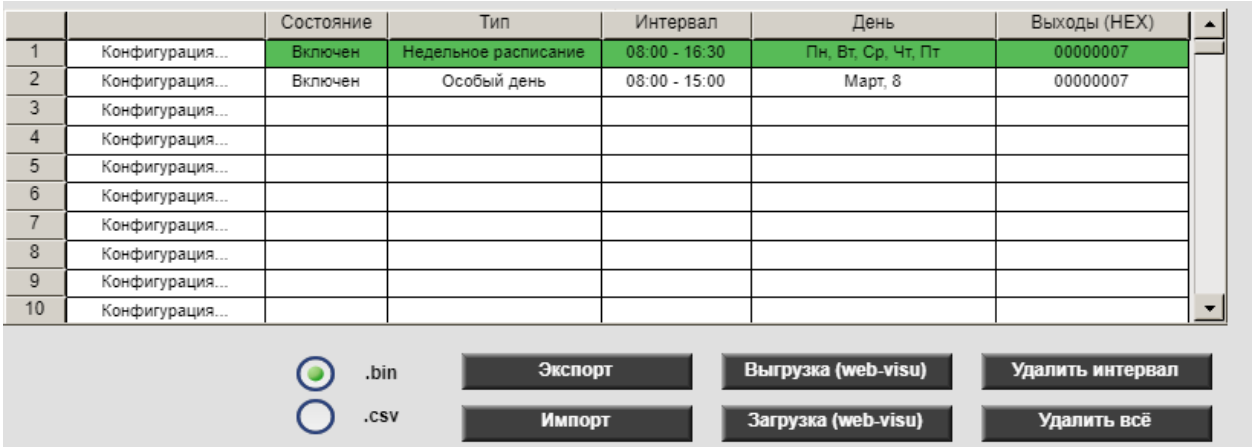

Рисунок 1.6.1 – Внешний вид фрейма **frmScheduleManager**

Фрейм включает в себя:

- таблицу для отображения, создания и редактирования интервалов расписания;
- кнопки удаления выбранного интервала и всего расписания;
- кнопки экспорта и импорта файлов расписания, а также чекбокс, определяющий формат файла;
- кнопки выгрузки и загрузки файлов расписания через веб-визуализацию.

Для создания нового или редактирования существующего интервала следует нажать на ячейку «Конфигурация…» нужной строки таблицы. Появится следующее диалоговое окно:

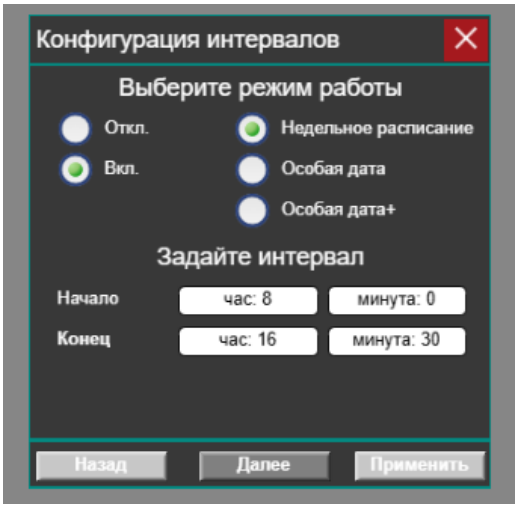

Рисунок 1.6.2 – Внешний вид диалога конфигурации интервала

Выберите состояние и тип интервала; задайте его начало и конец. Нажмите **Далее**.

Если интервал задан некорректно (конец интервала равен его началу или меньше него), то отобразится соответствующее сообщение, а кнопка **Далее** станет неактивной.

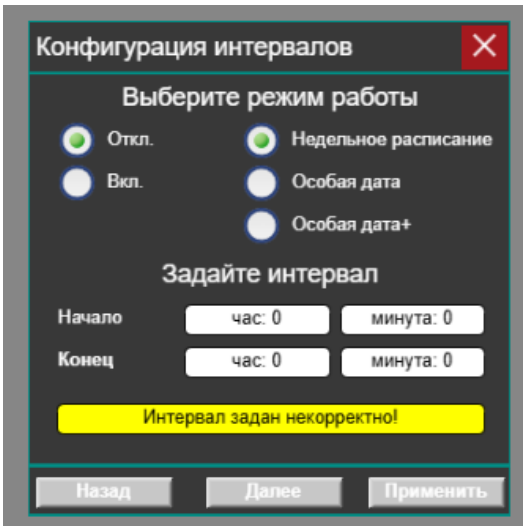

Рисунок 1.6.3 – Внешний вид окна конфигурации интервала с некорректно заданным интервалом

После нажатия на кнопку **Далее** появится окно, внешний вид которого зависит от выбранного типа интервала:

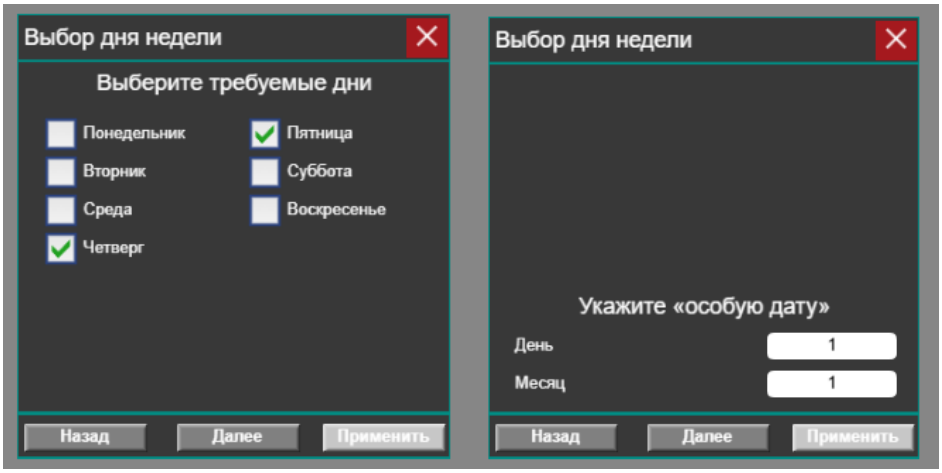

Рисунок 1.6.4 – Внешний вид окна выбора дня недели для режима недельного расписания (слева) и особой даты (справа)

Задайте нужные настройки и нажмите **Далее**. Для режима недельного написания требуется выбрать хотя бы один день – иначе кнопка **Далее** будет неактивна.

После нажатия на кнопку **Далее** появится окно выбора выходов, которые будут активны в течение активности интервала. Выберите нужные выходы и нажмите **Применить**.

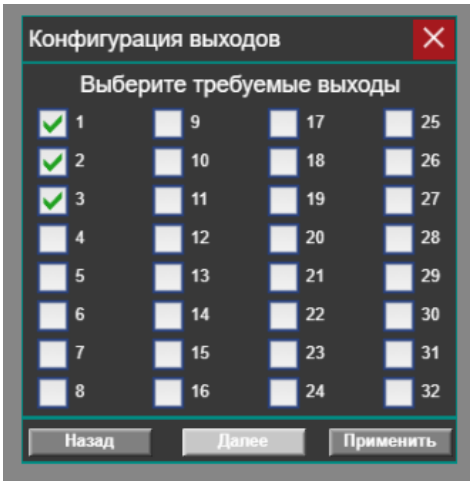

Рисунок 1.6.5 – Внешний вид окна конфигурации выходов интервала

Вне зависимости от номера строки, в которой была нажата ячейка «Конфигурация…», новый созданный интервал будет размещен следом за последним из существующих.

Для редактирования существующего интервала следует нажать на его ячейку «Конфигурация…».

Активные интервалы выделяются в таблице заданным цветом (по умолчанию – тёмно-зеленым).

Выделенная в данный момент строка таблицы также отображается заданным цветом (по умолчанию – серым). Для отмены выделения необходимо нажать на строку заголовка таблицы.

Нажатие на кнопку **Удалить интервал** приводит к удалению интервала выделенной строки. Если после данной строки были размещены другие интервалы – то таблица будет сдвинута на одну строку вверх.

Нажатие на кнопку **Удалить всё** приводит к удалению всех интервалов расписания.

Работа с файлами расписания и их загрузка/выгрузка через web-визуализацию описаны в следующих пунктах.

1.7. Экспорт и импорт файлов расписания

Функциональный блок **ScheduleManager** поддерживает экспорт и импорт расписания в виде файлов бинарного и текстового формата – например, для переноса на другие контроллеры. Пути к файлам должны быть заданы на входах **sPathToBinFile** и **sPathToCsvFile** ФБ ScheduleManager. Для экспорта и импорта используются соответствующие кнопки в фрейме frmScheduleManager.

Файловые плейсхолдеры в путях файлов

С определённого момента (по крайней мере – в версии **CODESYS V3.5 SP17** и выше) использование абсолютных путей для работы с файлами не рекомендуется из-за соображений безопасности.

В файловой системе контроллера есть рабочая директория рантайма CODESYS. Для разных контроллеров путь к ней может быть разным – уточните его у производителя вашего устройства. В рабочей директории находится папка **PlcLogic** – это директория проекта. В ней находятся вложенные папки, из которых отметим две – **PlcLogic/Application** (директория с загрузочным приложением) и **PlcLogic/visu** (директория файлов визуализации).

Эти директории описываются тремя известными файловыми плейсхолдерами:

- $$PlcLogic$$
- $$Application$$
- $$visu$$

Соответственно, если вы создадите папку **PlcLogic/my_schedule** – то можно присвоить входам **sPathToBinFile** и **sPathToCsvFile** ФБ ScheduleManager такие значения:

```
sPathToBinFile := '$$PlcLogic$$/my_schedule/sched.bin';
sPathToCsvFile := '$$PlcLogic$$/my_schedule/sched.txt[2]';
```

Производитель контроллера может поддержать свои файловые плейсхолдеры – например, для доступа к подключенным к контроллеру накопителям и т. д. Если для конкретного контроллера поддерживается редактирования конфиг-файла CODESYS – то пользователь может создать свои файловые плейсхолдеры (вы можете запросить информацию по этому вопросу у производителя вашего контроллера).

[2] См. вопрос 9

Описание форматов файлов

Файл бинарного формата представляет собой «слепок» блока памяти, размещенного по указателю **pastSchedIntervals**. Размер блока зависит от количества настроенных интервалов.

Файл формата csv состоит из строк одинаковой длины, выглядящих следующим образом (ниже приведены две строки из файла расписания):

```
1;0;08:00 - 16:30;1111100;00000000000000000000000000000001$R$N
1;1;00:00 - 15:00;sd08.03;00000000000000000000000000000001$R$N
```

Строка состоит из семи полей, разделённых точками с запятой:

Таблица 1.7.1 – Описание полей строки csv файла расписания

Номер поля	Описание	Значение в примере
1	Состояние интервала (**0** – не обрабатывается, **1** - обрабатывается)	1 1
2	Тип интервала (**0** – недельное расписание, **1** – особая дата, **2** – особая дата+)	0 1
3	Интервал в формате «hh:mm – hh:mm»	08:00 - 16:30 00:00 - 15:00
4	Для режима недельного расписания: битовая маска дней в двоичном формате (слева направо – от понедельника к воскресенью) Для режимов **Особая дата** и **Особая дата+**: дата в формате **dd.mm** с префиксом **sd**	1111100 sd08.03
5	Битовая маска выходов, которые должны быть активны в течение интервала, в двоичном формате, оканчивающаяся символами кодов переноса строки (в CODESYS они обозначаются как RN, но чаще известны как /r/n или CR LF. Это два байта со значениями 0x0D 0x0A, которые отображаются в текстовом редакторе как переход на новую строку)	00000000 00000000 00000000 00000001RN

При редактировании csv файла расписания на ПК – требуется обеспечить его соответствие формату (вплоть до отсутствия лишних пробелов). Для этого рекомендуется редактировать его с помощью Notepad++ или другого аналогичного редактора, не меняющего структуру файла (в частности, Microsoft Excel обычно меняет структуру файла). Именно поэтому в примере в качестве расширения файла используется **.txt**.

При импорте файла (как бинарного, так и csv) не производится никакой валидации – ответственность за корректность содержимого файлов возлагается на пользователя.

Влияние на производительность приложения контроллера

Операции экспорта и импорта файлов выполняется синхронно – в пределах одного цикла контроллера. Для файла формата csv, содержащего 255 строк, в течение цикла импорта время выполнения задачи визуализации **VISU_TASK** может вырасти на 70-100 мс (но это не повлияет на задачи реального времени). Это же справедливо и для экспорта расписания, состоящего из 255 строк, в файл формата csv.

Импорт/экспорт бинарного файла, состоящего из 255 интервалов, не оказывает значимого влияние на производительность приложения.

Обработка ошибок

Если операция экспорта/импорта завершается успешно – то в визуализации отображается соответствующее сообщение:

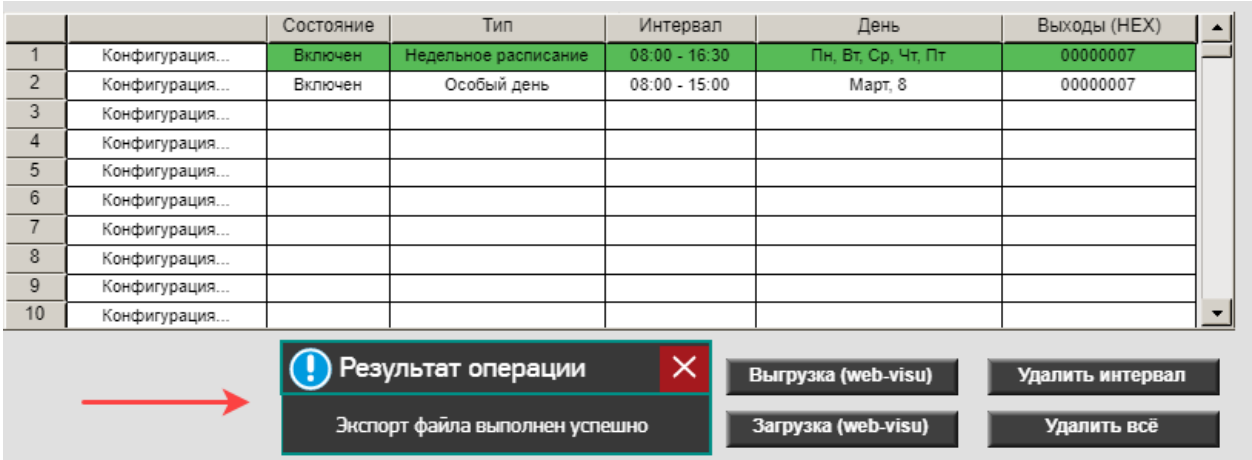

Рисунок 1.7.1 – Сообщение об успешном завершении операции с файлом

Если в процессе экспорта/импорта возникает ошибка – то в визуализации отображается соответствующее сообщение:

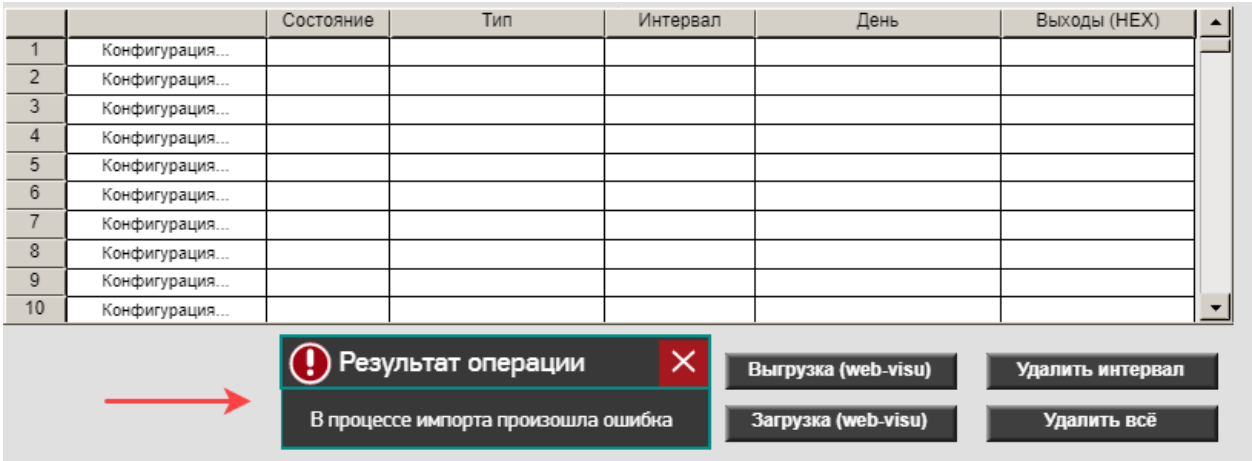

Рисунок 1.7.2 – Сообщение об ошибке в процессе операции с файлом

В логе контроллера (см. в среде CODESYS вкладку **Device – Журнал** при подключении к контроллеру) при этом будет опубликовано предупреждение от имени компонента **IECVisualization**, которое включает в себя:

- путь к экземпляру ФБ **ScheduleManager**, который сгенерировал предупреждение;
- название операции;
- название функции из библиотеки **SysFile**, при вызове которой произошла ошибка;
- код ошибки (см. список кодов ошибок в библиотеке CmpErrors).

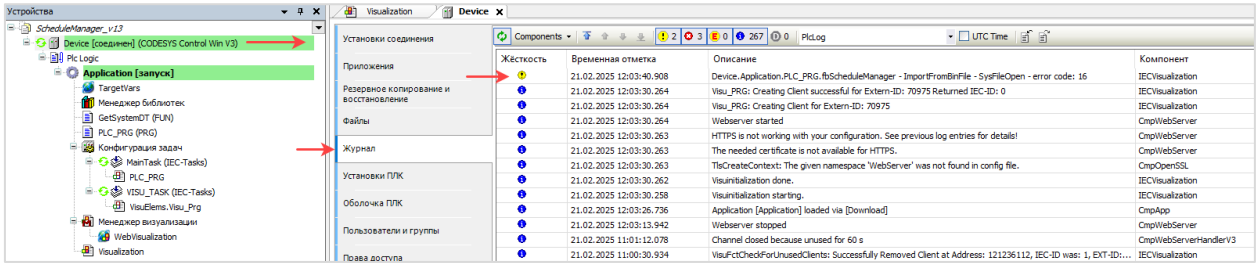

Рисунок 1.7.3 – Сообщение об ошибке в процессе операции с файлом в логе контроллера

Замечание по поводу механизма импорта

При импорте расписания из файла может произойти следующая ситуация: предположим, что сейчас в таблице настроено 4 интервала, а в импортируемом файле их 2. В этом случае первые два интервала будут перезаписаны информацией из файла, а оставшиеся два сохранятся. Чтобы избежать этого – перед импортом файла нажмите кнопку **Удалить всё**.

1.8. Загрузка и выгрузка файлов расписания из web-визуализации

Для загрузки и выгрузки файлов расписания из web-визуализации используются соответствующие кнопки в фрейме frmScheduleManager. Этот функционал будет работать только в том случае, если:

- пути к файлам на входах **sPathToBinFile** и **sPathToCsvFile** ФБ ScheduleManager заданы через файловые плейсхолдеры (при использовании абсолютных путей – функционал не будет работать);
- контроллер поддерживает функционал **Visu Transfer File** и в конфиг-файле CODESYS в секции **[CmpWebServerHandlerV3]** есть строка **AllowFileTransferServices=1**. Вы можете задать вопрос о поддержке данного функционала производителю вашего контроллера.

Данный функционал не будет работать в таргет-визуализации и сервисной визуализации среды CODESYS.

В случае ошибки при выгрузке файла – результат будет очевиден (в web-браузере не начнётся скачивание файла).

Ошибка при загрузке файла в рамках текущей реализации библиотеки не детектируется.

Если ваш контроллер не поддерживает данный функционал, то вы можете отключить отображение кнопок загрузки и выгрузки, присвоив входу **xHideFileTransferButtons** ФБ ScheduleManager значение **TRUE**.

1.9. Работа с примером

Пример создан в среде **CODESYS V3.5 SP17 Patch 3** и рассчитан на запуск на виртуальном контроллере **CODESYS Control Win V3** с версией **3.5.17.30**. Если вы планируете запускать пример на другом контроллере или другой версии виртуального контроллера – то обновите таргет-файл (**ПКМ** на узел **Device** – **Обновить устройство**) и, в случае необходимости, установите нужную версию компилятора и профиля визуализации (**Проект** – **Установки проекта** – **Опции компиляции** и **Проект** – **Установки проекта** – **Профиль визуализации**).

В рамках примера для получения системного времени виртуального контроллера используется функция **GetSystemDT**, являющаяся обёрткой над функциями из библиотеки SysTimeRtc.

Запустите виртуальный контроллер (из трея Windows), подключитесь к нему (**Device** – **Установки соединения**), загрузите проект примера (**Онлайн** – **Логин**) и запустите его (**Отладка** – **Старт**).

Нажмите на ячейку «Конфигурация…» любой строки таблицы и создайте новый интервал. При необходимости отредактируйте код вызова экземпляра ФБ ScheduleManager, размещённый в программе **PLC_PRG**.

Для работы с файлами может быть полезным знать директорию виртуального контроллера на ПК.

Долгое время она была следующей:

```
C:\ProgramData\CODESYS\тип[3]\идентификатор[4]\
```

Начиная с версий **V3.5 SP21** и **V3.5 SP20 Patch 1** путь изменился. Связанная с этим информация приведена в статье CODESYS Security Advisory 2024-02. Если коротко – если вы запускаете виртуальный контроллер из трея Windows, то его рабочей директорией может быть

```
C:\Windows\system32[5]\config\systemprofile\AppData\Roaming\CODESYS\тип\идентификатор\
```

или

```
C:\Windows\SysWOW64\config\systemprofile\AppData\Roaming\CODESYS\тип\идентификатор\
```

Для доступа к этим директориям потребуется наличие прав администратора.

[3] Например, CODESYS Control Win V3 или CODESYS Control Win V3 x64.
[4] Если у вас установлено несколько виртуальных контроллеров для одного типа – то для каждого из них будет существовать своя директория с уникальным идентификатором. Вы можете отследить директорию нужного контроллер, например, по дате её изменения или содержимому файла **CODESYSControl.cfg** – в секции **SysFile** будет указан путь установки версии среды CODESYS, которая соответствует этому контроллеру.
[5] Причём такой путь может быть для 64-битного виртуального контроллера CODESYS Control Win V3 x64.

В рабочей директории находится конфиг-файл с названием **CODESYSControl.cfg**. Если вы планируете использовать функционал загрузки и выгрузки файлов через web-визуализацию, то добавьте в его конец следующие строки:

```
[CmpWebServerHandlerV3]
AllowFileTransferServices=1
```

Сохраните файл и перезапустите виртуальный контроллер через трей Windows, чтобы изменения вступили в силу.

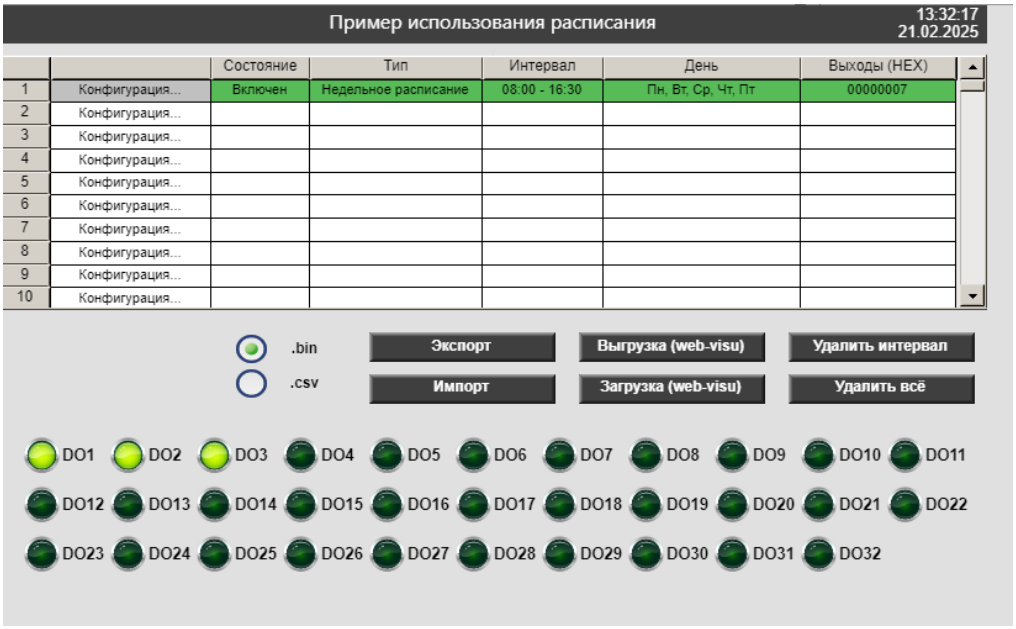

Рисунок 1.9.1 – Внешний вид визуализации примера

1.10. Методы ФБ ScheduleManager

ФБ ScheduleManager содержит ряд методов, которые используются в его коде и визуализациях библиотеки. Ряд методов виден конечному пользователю и может использоваться им в своём приложении.

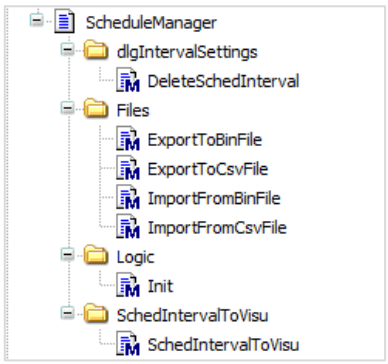

Рисунок 1.10.1 – Методы ФБ **ScheduleManager**, доступные пользователю

Эти методы позволяют из кода пользовательского приложения:

- удалить интервал (метод **DeleteSchedInterval**);
- обновить таблицу интервалов в визуализации после изменения массива бинарных интервалов (метод **Init**; см. подробнее в следующем пункте);
- конвертировать структуру бинарного интервала в структуру с текстовой информацией об интервале, отображаемой в таблице (метод **SchedIntervalToVisu**);
- произвести экспорт/импорт файла расписания (остальные методы).

Описание методов приведено в п. 2.4.

1.11. Вопросы и ответы

Вопрос 1: каково максимальное количество интервалов, поддерживаемое ФБ ScheduleManager?

Ответ: 255. Увеличить его можно только путём доработки библиотеки. Но не забывайте, что в программе можно объявить произвольное количество экземпляров ФБ.

Вопрос 2: каково максимальное количество «выходов управления» расписания?

Ответ: 32. Увеличить его можно только путём доработки библиотеки.

Вопрос 3: как создать интервал, начинающийся в один день и заканчивающийся в другой? Например, требуется, чтобы оборудование работало в будние дни, с 00:00 до 08:00 и с 20:00 до 00:00.

Ответ: в рамках библиотеки – интервал настраивается строго в пределах одного дня.

Для описанной задачи можно создать два интервала со следующими настройками:

Интервал	День
00:00 – 08:00	Пн, Вт, Ср, Чт, Пт
20:00 – 23:59	Пн, Вт, Ср, Чт, Пт

Интервал, который заканчивается в **23:59** обрабатывается **особым образом** – для такого интервала соответствующие выходы остаются активными и **в течение** последней минуты часа. Это позволяет реализовать интервал, который соответствует времени «до полуночи». Для всех остальных возможных концов интервала последняя минута не включается в состав интервала – при её наступлении связанные с ним выходы сразу отключатся.

Ещё один вариант решения описанной в вопросе задачи – создать один интервал (08:00 – 20:00), а в коде программы проверять **инвертированные** значения связанных с ним битов битовой маски; иными словами – если бит имеет значение **FALSE**, то интервал 08:00 – 20:00 сейчас неактивен, а значит, текущее время находится в интервале 00:00 – 08:00 или 20:00 – 00:00.

Вопрос 4: ФБ ScheduleManager формирует маску дискретных выходов. Но в моей задаче требуется не включать/отключать оборудование по расписанию, а менять значение его нагрузки. Как это сделать?

Ответ: реализуйте в вашем приложении требуемую логику. Например, если установлен бит 0 битовой маски – то присвойте переменной, соответствующей уставке нагрузки, например, 10, а если установлен бит 2 – то 20 и т. д. При необходимости – вы можете отслеживать в программе моменты установки и отключения бит маски (например, с помощью ФБ **R_TRIG** и **F_TRIG** из библиотеки **Standard**), чтобы, например, в момент активации интервала начать выполнять код плавного изменения нагрузки.

Вопрос 5: как определить число настроенных интервалов?

Ответ: вы можете использовать локальную переменную **usiConfiguredIntervalCount** ФБ ScheduleManager. Так как она локальная – то в CODESYS вам потребуется ввести её имя вручную, без автодополнения.

Вопрос 6: я планирую формировать расписание не через визуализацию CODESYS, а с помощью своего кода. Как это сделать?

Ответ: запишите в массив, размещённый по указателю **pastSchedIntervals**, требуемые значения. Обратите внимание, что все создаваемые интервалы должны быть размещены в массиве «вплотную» друг к другу, без «пропусков» (т. е. между создаваемыми интервалами не должно быть интервалов с **eState** = **NOT_EXIST**; по наличию такого интервала ФБ ScheduleManager определяет окончание последовательности настроенных интервалов).

После этого вызовите метод Init со значением **FALSE** на входе **xClearAll**, чтобы сформировать таблицу, отображаемую в фрейме **frmScheduleManager**, и рассчитать значения локальных переменных ФБ.

Вопрос 7: ФБ ScheduleManager формирует маску дискретных выходов. Для обращения к битам я использую побитовый доступ (dwOutputsMask.0, dwOutputsMask.1 и т. д.). Можно ли как-то задать битам понятные имена?

Ответ: это можно сделать разными способами. Два наиболее простых:

- использовать вместо числовых индексов специально объявленные константы;
- создать структуру, состоящую из 32 полей типа **BIT** с понятными именами, и «перекладывать» в неё содержимое выхода **dwOutputsMask** (см. функцию MemMove из библиотеки **CAA Memory**).

Вопрос 8: как кастомизировать визуальную часть библиотеки?

Ответ: путём её редактирования. См. подробнее в п. 2.

Вопрос 9: почему в документации упоминается, что поддерживается экспорт/импорт файла в формате csv, но в примере используется файл с расширением .txt?

Ответ: формируемый файл действительно именно формат csv (т. е. его внутренняя структура соответствует этому формату). Но для последующего импорта требуется, чтобы структура импортируемого файла в точности соответствовала экспортированному файлу. С помощью табличных редакторов (например, Microsoft Excel) этого крайне сложно добиться – часто они меняют структуру файла при его открытии и сохранении. Поэтому рекомендуется редактировать файл с помощью Notepad++ или другого аналогичного текстового редактора, не меняющего структуру файла. Именно поэтому в примере в качестве расширения файла используется **.txt**.

При редактировании файла требуется обеспечить его соответствие формату (вплоть до отсутствия лишних пробелов).

2. Руководство разработчика

2.1. Основная информация

Для редактирования библиотеки:

- полностью прочитайте п. 1 и п. 2 данного документа;
- откройте её в среде CODESYS как проект (**Файл** – **Открыть**);
- изучите встроенные комментарии к интересующим вас объектам;
- внесите нужные вам изменения;
- измените информацию о проекте (**Проект** – **Информация о проекте**; желательно изменить как минимум название и версию, чтобы избежать путаницы с исходной библиотекой);
- сохраните файл библиотеки (**Файл** – **Сохранить как**; желательно сохранить под новым именем, чтобы избежать путаницы с исходной библиотекой);
- переустановите библиотеку в репозиторий библиотек (**Файл** – **Сохранить проект и установить в репозиторий библиотек** или через Инструменты – Репозиторий библиотек).

2.2. Структура библиотеки

С точки зрения разработчика – в дополнение к описанным в п. 1.3 объектам библиотека содержит:

- ФБ **CLK_PRG** (позаимствованный из библиотеки **OSCAT Basic**), представляющий собой генератор единичных импульсов (используется в коде ФБ ScheduleManager);
- структуру **SCHED_DIALOG_VISU**;
- перечисление **MessageBoxOwen_ICON_TYPES**;
- перечисление **SCHED_LOGIC_STEP**;
- диалог **dlgIntervalSettings**;
- фреймы **frmSetInterval**, **frmSetDays**, **frmSetOutputs**, **frmMessageBox**;
- списки текстов **SchedTextList** и **GlobalTextList**;
- пул изображений **ImagePoolDialogs**.

Перечисленные объекты описываются в следующих пунктах.

Также с точки зрения разработчика ФБ ScheduleManager имеет несколько десятков методов, которые описаны в п. 2.4.

2.3. Машина состояний ФБ ScheduleManager

Код ФБ ScheduleManager довольно прост. Он включает в себя обработку изменения входа **xEnable** и реализацию машины состояний.

Шаги машины состояний описаны в перечислении **SCHED_LOGIC_STEP**:

- **IDLE** – ожидание запуска;
- **CHECK_INPUTS** – проверка значений входов;
- **MAIN_LOGIC** – выполнение основной логики (контроль расписания).

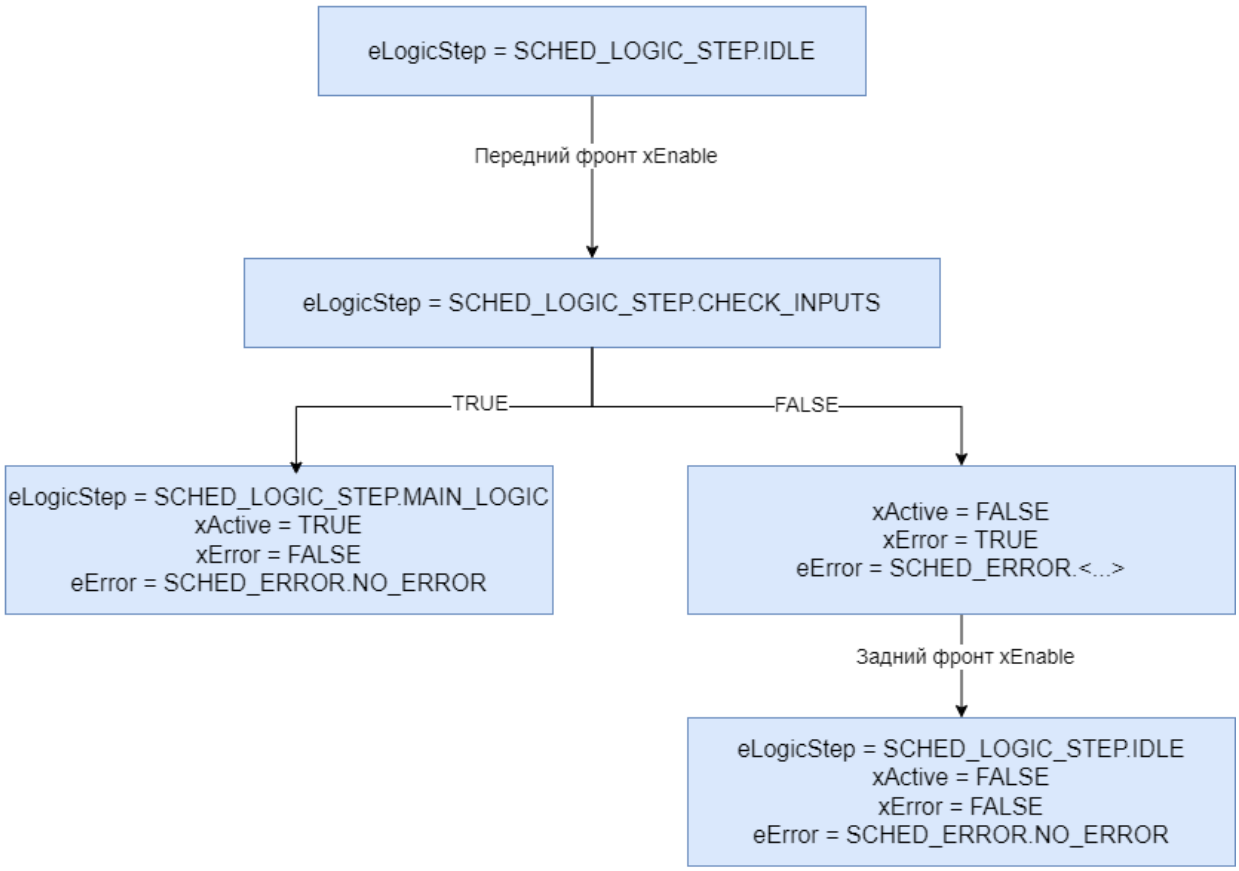

Рисунок 2.3.1 – Блок-схема машины состояний ФБ ScheduleManager

В начальный момент времени блок находится в состоянии **IDLE**. По переднему фронту входа **xEnable** выполняется переход на шаг **CHECK_INPUTS**, в котором вызывается метод IsValidInputs. В методе происходит валидация некоторых входов блока (см. подробнее в описании перечисления SCHED_ERROR). В случае успешной валидации выходу **xActive** присваивается значение **TRUE**, вызывается метод Init и происходит переход на шаг **MAIN_LOGIC**. В случае обнаружения ошибки выход **xActive** имеет значение **FALSE**, выходу **xError** присваивается значение **TRUE**, а на выход **eError** передаётся код ошибки из перечисления SCHED_ERROR; в этом случае блок остаётся на шаге **CHECK_INPUTS** до детектирования заднего фронта на входе **xEnable**. По заднему фронту данного входа выполняется переход на шаг **IDLE**, а все выходы блока инициализируются значениями по умолчанию.

На шаге **MAIN_LOGIC** запущен экземпляр ФБ **CLK_PRG**, который с периодом в **1 секунду** генерирует единичные импульсы. При формировании очередного импульса выполняется вызов:

- метода SplitDateTime для выделения из текущей даты и времени, поданной на вход блока **dtCurrentDateTime**, значений текущего месяца и дня;
- функции DayOfWeek из библиотеки **Util** для определения номера текущего дня недели;
- метода CheckForSpecialDay для определения, принадлежит ли текущий день одному из настроенных интервалов расписания с типом **Особая дата** или **Особая дата+;**
- метода Main для проверки активности настроенных интервалов и формирования битовой маски выходов.

2.4. Методы ФБ ScheduleManager

Список методов ФБ ScheduleManager приведён в таблице ниже. Описание методов приведено после таблицы. Некоторые методы доступны конечному пользователю – они упомянуты в п. 1.10.

Таблица 2.4.1 – Методы ФБ **ScheduleManager**

№	Название	Место вызова	Вызывает методы
1	Папка **dlgIntervalSettings**		
1.1	ApplySchedInterval	диалог dlgIntervalSettings, кнопка «Применить»	SchedIntervalToVisu
1.2	CheckForEmptyDaysOfWeek	диалог dlgIntervalSettings, эллипс «Вызов методов»	-
1.3	CheckForNonValidInterval	фрейм frmSetInterval, прямоугольник «Интервал задан некорректно!»	-
1.4	DeleteSchedInterval	фрейм frmScheduleManager, кнопка «Удалить интервал»	SchedIntervalToVisu
1.5	InitDialog	диалог dlgIntervalSettings, эллипс «Вызов методов»	-
2	Папка **Files**		
2.1	BitmaskToString	метод DayParamToCsvField метод ExportToCsvFile	-
2.2	DayParamToCsvField	метод ExportToCsvFile	BitmaskToString
2.3	ExportToBinFile	фрейм frmScheduleManager, кнопка «Экспорт»	SchedLog
2.4	ExportToCsvFile	фрейм frmScheduleManager, кнопка «Экспорт»	SchedLog DayParamToCsvField
2.5	ImportFromBinFile	фрейм frmScheduleManager, кнопка «Импорт»	SchedLog
2.6	ImportFromCsvFile	фрейм frmScheduleManager, кнопка «Импорт»	SchedLog
2.7	SchedLog	метод ExportToBinFile метод ExportToCsvFile метод ImportFromBinFile метод ImportFromCsvFile	-
2.8	StrIntervalToArray	метод ImportFromCsvFile	-
2.9	StrSpecialDateToArray	метод ImportFromCsvFile	-

3	Папка **Logic**		
3.1	CheckForSpecialDay	ФБ ScheduleManager	-
3.2	Init	ФБ ScheduleManager фрейм frmScheduleManager, кнопка «Импорт», кнопка «Удалить всё»	SchedIntervalToVisu
3.3	IsNowCheckedInterval	метод Main	-
3.4	IsValidInputs	ФБ ScheduleManager	-
3.5	Main	ФБ ScheduleManager	IsNowCheckedInterval
3.6	SplitDateTime	ФБ ScheduleManager	-
4	Папка **SchedIntervalToVisu**		
4.1	DayOfWeekToFormatWstring	метод DaysOfWeekMaskToFormatWstring	-
4.2	DayParamToWstring	метод SchedIntervalToVisu	DayOfWeekToFormatWstring MonthAndDayToFormatWstring
4.3	DaysOfWeekMaskToFormatWstring	метод DayParamToWstring	DayOfWeekToFormatWstring
4.4	HourAndMinuteToFormatString	метод IntervalToString	-
4.5	IntervalToString	метод SchedIntervalToVisu	HourAndMinuteToFormatString
4.6	MonthAndDayToFormatWstring	метод DayParamToWstring	-
4.7	SchedIntervalToVisu	метод ApplySchedInterval метод DeleteSchedInterval метод Init	SchedStateToWstring SchedTypeToWstring IntervalToString DayParamToWstring
4.8	SchedStateToWstring	метод SchedIntervalToVisu	-
4.9	SchedTypeToWstring	метод SchedIntervalToVisu	-

1. Методы папки **dlgIntervalSettings** вызываются в контексте визуализации

1.1. Метод **ApplySchedInterval**

Сигнатура: BOOL ApplySchedInterval (INT iVisuIndex, SCHED_DIALOG_VISU stData)

Метод создаёт новый интервал (или редактирует существующий) на основе структуры **stData**, значения которой задаются пользователем в диалоге **dlgIntervalSettings** (и переключаемых в нём фреймах). **iVisuIndex** – это номер строки таблицы фрейма frmScheduleManager, для которой был открыт данный диалог. Выход метода не используется и всегда имеет значение **FALSE**.

Вызов метода размещён в диалоге **dlgIntervalSettings** в ST-коде кнопки **Применить**.

1.2. Метод **CheckForEmptyDaysOfWeek**

Сигнатура: BOOL CheckForEmptyDaysOfWeek (ARRAY [1..7] OF BOOL axDaysOfWeek)

Метод проверяет, что для интервала типа **Недельное расписание** не выбран ни один день недели[6] (фактически – что все элементы массива **axDaysOfWeek** имеют значение **FALSE**). Метод возвращает **TRUE** в случае успешной проверки и **FALSE** – в случае неуспешной. Это же значение дублируется на выход метода **xIsEmptyDaysOfWeek** (для передачи в диалог **dlgIntervalSettings**).

Вызов метода размещён в диалоге **dlgIntervalSettings** в прямоугольнике с надписью **Вызов методов** (параметр **Переменные состояний/Отключение ввода**).

[6] То есть это «проверка на некорректную настройку».

1.3. Метод **CheckForNonValidInterval**

Сигнатура: BOOL CheckForNonValidInterval (USINT usiStartHour, USINT usiStartMinute, USINT usiStopHour, USINT usiStopMinute)

Метод проверяет, что начало и конец интервала настроены некорректно (фактически – что момент конца интервала равен или «меньше» момента начала). Метод возвращает **TRUE** в случае успешной проверки и **FALSE** – в случае неуспешной. Это же значение дублируется на выход метода **xNonValidInterval** (для передачи в фрейм **frmSetInterval**).

Вызов метода размещён в фрейме **frmSetInterval** в прямоугольнике с надписью **Интервал задан некорректно!** (параметр **Переменные состояний/Невидимый**, внутри оператора **NOT**).

1.4. Метод **DeleteSchedInterval**

Сигнатура: BOOL DeleteSchedInterval (INT iVisuIndex)

Метод удаляет интервал – как в таблице, так и в массиве бинарных данных. Если интервал был не последним в списке – то происходит смещение массива и таблицы на один элемент «вверх». **iVisuIndex** – это номер строки таблицы фрейма frmScheduleManager, которая удаляется. Выход метода не используется и всегда имеет значение **FALSE**.

Вызов метода размещён в фрейме **frmScheduleManager** в ST-коде кнопки **Удалить интервал**.

1.5. Метод **InitDialog**

Сигнатура: BOOL InitDialog (BOOL xNeedInit, INT iVisuIndex, SCHED_DIALOG_VISU stData)[7]

Метод инициализирует локальные переменные диалога **dlgIntervalSettings** – чтобы при нажатии на ячейку «Конфигурация…» уже существующего интервала (строки таблицы с номером **iVisuIndex**) – в диалоге отобразились бы значения настроек этого конкретного интервала. Флаг **xNeedInit** используется для однократного выполнения кода метода. Флаг устанавливается при нажатии на ячейку «Конфигурация…» в таблице в фрейме frmScheduleManager (см. **Конфигурация ввода – OnMouseClick – Выполнить ST-код**) и сбрасывается в коде самого метода. Метод заполняет структуру **stData**, которая передаётся в диалог **dlgIntervalSettings** (аргументы метода являются входами-выходами). Выход метод всегда имеет значение **TRUE**.

Вызов метода размещён в диалоге **dlgIntervalSettings** в прямоугольнике с надписью **Вызов методов** (параметр **Переменные состояний/Невидимый**; поскольку метод всегда возвращает **TRUE** – этот прямоугольник не будет виден пользователю).

[7] Аргументы метода объявлены в его секции VAR_IN_OUT

2. Основные методы папки **Files** вызываются в коде кнопок **Экспорт** и **Импорт**

2.1. Метод **BitmaskToString**

Сигнатура: STRING BitmaskToString (DWORD dwBitMask, USINT usiBitCount)

Метод конвертирует младшие **usiBitCount** бит переменной **dwBitMask** в строку, состоящую из нулей и единиц (первый символ строки соответствует самому младшему биту).

Вызов метода производится в методах **DayParamToCsvField** и ExportToCsvFile.

2.2. Метод **DayParamToCsvField**

Сигнатура: STRING DayParamToCsvField (INT iBinIndex)

Метод формирует для бинарного интервала, определяемого индексом **iBinIndex** (нумерация с **0**), строку для столбца «День», которая будет записана в файл формата csv. Для режима недельного расписания – это строковая интерпретация битовой маски дней недели (состоит из нулей и единиц, первый символ соответствует понедельнику). Для режима **Особая дата** и **Особая дата+** – это строка вида **sdddd.mm**, где:

- **sd** – признак особой даты (special date);
- **dd** – номер дня с ведущим нулём;
- **mm** – номер месяца с ведущим нулём.

Таким образом, формируемая методом строка всегда состоит из 7 символов.

Вызов метода производится в методе ExportToCsvFile.

2.3. Метод **ExportToBinFile**

Сигнатура: BOOL ExportToBinFile ()

Метод экспортирует настроенные интервалы в файл бинарного формата, представляющий собой «слепок» блока памяти, размещённого по указателю **pastSchedIntervals**. Размер блока памяти зависит от количества настроенных интервалов. Путь к файлу определяется значением входа **sPathToBinFile** ФБ ScheduleManager.

Метод возвращает **TRUE** при успешном завершении экспорта и **FALSE** – если в процессе экспорта возникнет ошибка. Информация об ошибке выводится в лог контроллера с помощью метода ShedLog (см. подробности).

Выполнение метода происходит синхронно, в течение одного цикла контроллера.

Вызов метода размещён в фрейме frmScheduleManager в ST-коде кнопки **Экспорт**.

2.4. Метод **ExportToCsvFile**

Сигнатура: BOOL ExportToCsvFile ()

Метод экспортирует настроенные интервалы в файл формата csv. Формат файла описан в п. 1.7.

Путь к файлу определяется значением входа **sPathToBinFile** ФБ ScheduleManager.

Метод возвращает **TRUE** при успешном завершении экспорта и **FALSE** – если в процессе экспорта возникнет ошибка. Информация об ошибке выводится в лог контроллера с помощью метода ShedLog (см. подробности).

Выполнение метода происходит синхронно, в течение одного цикла контроллера.

Вызов метода размещён в фрейме frmScheduleManager в ST-коде кнопки **Экспорт**.

2.5. Метод **ImportFromBinFile**

Сигнатура: BOOL ImportFromBinFile ()

Метод импортирует расписание из файла бинарного формата. Путь к файлу определяется значением входа **sPathToBinFile** ФБ ScheduleManager.

При импорте расписания из файла может произойти следующая ситуация: предположим, что сейчас в таблице настроено 4 интервала, а в импортируемом файле их 2. В этом случае первые два интервала будут перезаписаны информацией из файла, а оставшиеся два сохранятся. Чтобы избежать этого – перед импортом файла нажмите кнопку **Удалить всё**.

Метод возвращает **TRUE** при успешном завершении импорта и **FALSE** – если в процессе импорта возникнет ошибка. Информация об ошибке выводится в лог контроллера с помощью метода ShedLog (см. подробности).

Выполнение метода происходит синхронно, в течение одного цикла контроллера.

Вызов метода размещён в фрейме frmScheduleManager в ST-коде кнопки **Импорт**.

2.6. Метод **ImportFromCsvFile**

Сигнатура: BOOL ImportFromCsvFile ()

Метод импортирует расписание из файла формата csv. Путь к файлу определяется значением входа **sPathToCsvFile** ФБ ScheduleManager.

При импорте расписания из файла может произойти следующая ситуация: предположим, что сейчас в таблице настроено 4 интервала, а в импортируемом файле их 2. В этом случае первые два интервала будут перезаписаны информацией из файла, а оставшиеся два сохранятся. Чтобы избежать этого – перед импортом файла нажмите кнопку **Удалить всё**.

Метод возвращает **TRUE** при успешном завершении импорта и **FALSE** – если в процессе импорта возникнет ошибка. Информация об ошибке выводится в лог контроллера с помощью метода ShedLog (см. подробности).

Выполнение метода происходит синхронно, в течение одного цикла контроллера.

Вызов метода размещён в фрейме frmScheduleManager в ST-коде кнопки **Импорт**.

2.7. Метод **SchedLog**

Сигнатура: BOOL SchedLog (STRING sPouName, STRING sMethodName, STRING sOpName, STRING sErrorCode)

Метод объединяет четыре своих аргумента в строку формата

```
#POU# - #METHOD# - #OP# - error code: #ERROR#
```

где символом «#» обрамлены заполнители, в которые будут подставлены значения соответствующих входов метода, и выводит её в лог контроллера в виде предупреждения от имени компонента **IECVisualization**. Подробности см. в п. 1.7.

Выход метода не используется и всегда имеет значение FALSE.

Вызов метода производится в методах ExportToBinFile, ExportToCsvFile, ImportFromBinFile, ImportFromCsvFile.

2.8. Метод **StrIntervalToArray**

Сигнатура: ARRAY [0..3] OF USINT StrIntervalToArray (STRING sInterval)

Метод преобразует строковое представление интервала времени в массив из 4 отдельных целочисленных значений – например, строка '09:12 - 13:15' будет преобразована в массив [9, 12, 13, 15].

Валидация исходной строки не выполняется.

Выход метода не используется и всегда имеет значение **FALSE**.

Вызов метода производится в методе ImportFromCsvFile.

2.9. Метод **StrSpecialDateToArray**

Сигнатура: ARRAY [0..1] OF USINT StrSpecialDateToArray (STRING sSpecialDate)

Метод преобразует строковое представление особой даты в массив из 2 отдельных целочисленных значений – например, строка 'sd01.02' будет преобразована в массив [2, 1].

Валидация исходной строки не выполняется.

Выход метода не используется и всегда имеет значение **FALSE**.

Вызов метода производится в методе ImportFromCsvFile.

3. Методы папки **Logic** вызываются в коде ФБ **ScheduleManager**

3.1. Метод **CheckForSpecialDay**

Сигнатура: BOOL CheckForSpecialDay ()

Метод проверяет, является ли текущий день одной из особых дат, настроенных в расписании. Это нужно по той причине, что в особую дату недельное расписание не применяется.

Метод возвращает **TRUE** в случае успешной проверки и **FALSE** – в случае неуспешной.

Вызов метода производится в ФБ ScheduleManager на шаге **MAIN_LOGI**C.

3.2. Метод **Init**

Сигнатура: BOOL **Init** (BOOL xClearAll)

Метод заполняет массив интервалов расписания, отображаемый в визуализации, на основе массива бинарных данных интервалов.

Если вход метода имеет значение **TRUE** – то вместо этого производится очистка таблицы.

Выход метода не используется и всегда имеет значение **FALSE**.

Вызов метода производится:

- в ФБ ScheduleManager на шаге **CHECK_INPUTS** (в случае успешной валидации входных значений блока) со значением **FALSE** на входе **xClearAll**;
- в фрейме frmScheduleManager в ST-коде кнопки **Импорт** со значением **FALSE** на входе **xClearAll**;
- в фрейме frmScheduleManager в ST-коде кнопки **Удалить всё** со значением **TRUE** на входе **xClearAll**.

3.3. Метод **IsNowCheckedInterval**

Сигнатура: BOOL IsNowCheckedInterval (INT iVisuIndex)

Метод проверяет, соответствует ли текущий момент времени интервалу из строки таблицы с номером **iVisuIndex**.

Метод возвращает **TRUE** в случае успешной проверки и **FALSE** – в случае неуспешной.

Вызов метода производится в методе Main.

3.4. Метод **IsValidInputs**

Сигнатура: BOOL IsValidInputs ()

Метод производит валидацию входных значений ФБ ScheduleManager.

Метод возвращает **TRUE** в случае успешной валидаци и **FALSE** – в случае неуспешной. Код ошибки возвращается на выход метода **eError** (см. список кодов ошибок в описании SCHED_ERROR).

Вызов метода производится в ФБ ScheduleManager на шаге **CHECK_INPUTS**.

3.5. Метод **Main**

Сигнатура: BOOL Main ()

Метод выполняет основную логику работы по расписанию: устанавливает биты маски выходов **dwOutputsMask** для активных текущих интервалов и сбрасывает для неактивных. При этом допустимо создание «вложенных» и «перекрывающихся» интервалов, с которыми связаны одни и те же биты маски. Эта ситуация корректно обрабатываются с помощью использования локальной переменной метода **dwOutputsMaskSetThisCycle**, соответствующей битовая маске выходов, установленных в данном цикле контроллера. Метод не производит сброс бит, установленных в текущем цикле – даже если есть связанный с ними интервал, который уже неактивен – потому что они могут быть связаны с другим активным интервалом.

Выход метода не используется и всегда имеет значение **FALSE**.

Вызов метода производится в ФБ ScheduleManager на шаге **MAIN_LOGIC**.

3.6. Метод **SplitDateTime**

Сигнатура: BOOL SplitDateTime (DT dtCurrentDateTime)

Метод выделяет из значения даты и времени типа **DT** значения месяца и дня и возвращает их на своих выходы **uiCurrentMonth** и **uiCurrentDay** типа **UINT**. Представляет собой обёртку над функцией SplitDateTime из библиотеки **Util**.

Выход метода не используется и всегда имеет значение **FALSE**.

Вызов метода производится в ФБ ScheduleManager на шаге **MAIN_LOGIC**.

4. Основным методом папки **SchedIntervalToVisu** является, собственно, метод **SchedIntervalToVisu**. Все остальные методы являются вспомогательными и вызываются только в коде данного метода – поэтому их описание опущено.

4.1. Метод **SchedIntervalToVisu**

Сигнатура: SCHED_INTERVAL_VISU SchedIntervalToVisu (SCHED_INTERVAL stInterval)

Метод конвертирует интервал расписания **stInterval** из бинарного формата в строковый.

Вызов метода производится в методах ApplySchedInterval, DeleteSchedInterval и Init.

2.5. Визуализации библиотеки и их код

В данном пункте описывается код, размещённый в визуализациях библиотеки.

Основной визуализацией библиотеки является фрейм frmScheduleManager.

Размер экрана фрейма: **800x480**.

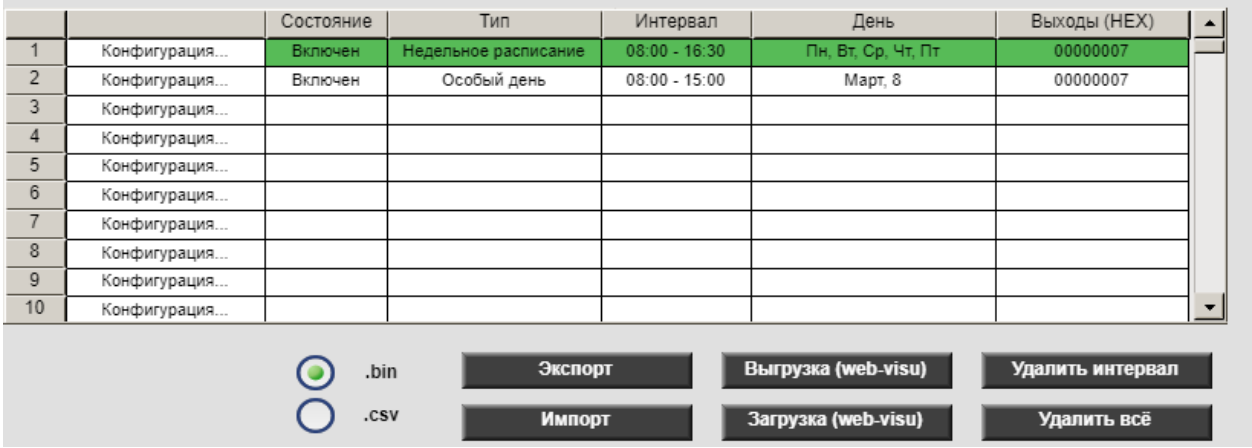

Рисунок 2.5.1 – Внешний вид фрейма **frmScheduleManager**

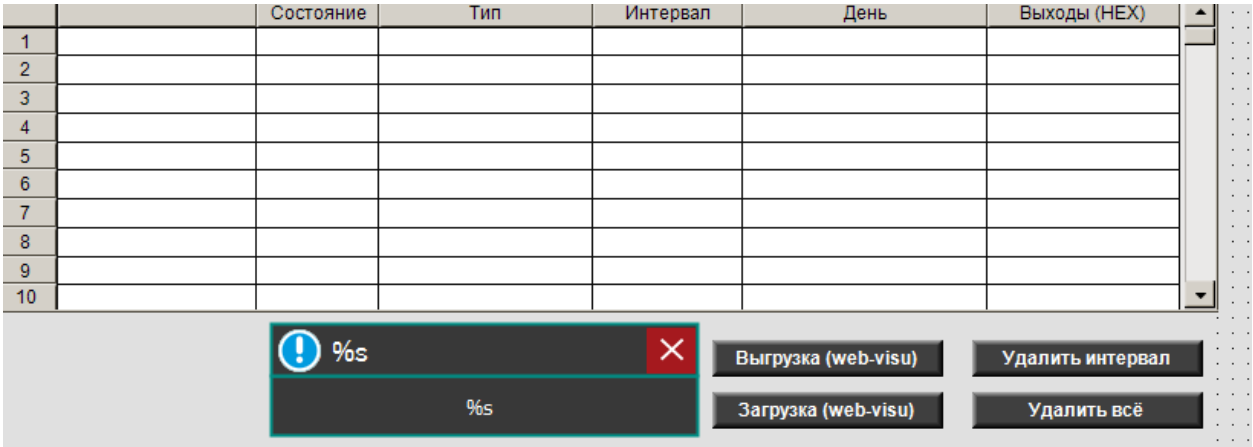

Рисунок 2.5.2 – Внешний вид фрейма **frmScheduleManager** в редакторе визуализации

В редакторе визуализации поверх кнопок **Экспорт**, **Импорт** и радиокнопки выбора формата файла размещён фрейм **frmMessageBox**, в котором отображается сообщение с информацией о результате экспорта или импорта.

Входом-выходом фрейма **frmScheduleManager** является экземпляр ФБ ScheduleManager.

Основным элементом **frmScheduleManager** является таблица, используемая для отображения и редактирования расписания. Для самого левого столбца таблицы во вкладке **Конфигурация ввода** для события **OnMouseClick** добавлены действия **Выполнить ST-код** и **Открыть диалог**.

В ST-коде выполняется присвоение значения **TRUE** локальной переменной фрейма **xNeedInit**. В действии **Открыть диалог** настроено открытие диалога **dlgIntervalSettings** с передачей ему:

- экземпляра ФБ ScheduleManager;
- флага **xNeedInit**;
- номера выбранной в таблице строки **INDEX** – эта особая переменная, которая существует только в контексте таблицы.

Информация про диалог dlgIntervalSettings будет приведена далее.

Для всех столбцов таблицы, кроме первого, к параметру **Переменные цвета/Переключить цвет** привязана **переменная fbScheduleManager.pastVisuSchedule^[INDEX].xCurrentInterval**. За счёт этого происходит выделение активных интервалов цветом, определяемым константой фрейма frmScheduleManager с названием **c_dwActiveIntervalColor** – она привязана к параметру **Переменные цвета/Состояние тревоги/Цвет заливки**. Константа имеет тип **DWORD** и содержит значение цвета в формате ARGB.

Так как первый столбец таблицы не выделяется цветом – это позволяет всегда (даже для активных интервалов) определить выделенную в таблице строку. Цвет выделения определяется параметром таблицы **Выбор/Цвет выбора**.

К параметру **Выбор/Переменная для выбранной строки** привязана локальная переменная фрейма **iSelectedLineIndex** – она определяет номер выделенной в таблице строки. Значение этой переменной используется в коде кнопок, размещённых под таблицей – потому что там нельзя использовать переменную **INDEX** (как уже упоминалось, переменная **INDEX** существует только в контексте таблицы).

К радиокнопке выбора формата файла привязана локальная переменная фрейма **usiFileFormat**. Значение **0** соответствует бинарному файлу, значение **1** – файлу формата csv.

Для кнопок **Экспорт** и **Импорт** во вкладке **Конфигурация ввода** для события **OnMouseClick** добавлено действие **Выполнить ST-код**. В этом коде происходит вызов соответствующих методов и установка значения **TRUE** для локальной переменной фрейма **xShowMessage**. Эта переменная привязана к переменной невидимости (**Переменные состояний/Невидимый,** внутри оператора **NOT**) фрейма **frmMessageBox**, что приводит к отображению окна с информацией о результате операции экспорта/импорта. Содержимое окна (текст сообщения и иконка) формируется в упомянутом чуть выше действии **Выполнить ST-код**. При нажатии на кнопку закрытия сообщения – переменной **xShowMessage** присваивается значение **FALSE**, что приводит к скрытию фрейма **frmMessageBox**.

На кнопках **Выгрузка (web-visu)** и **Загрузка (web-visu)** во вкладке **Конфигурация ввода** для события **OnMouseClick** добавлены действия **Выполнить ST-код** и **Передача файла**. В коде в зависимости от значения локальной переменной фрейма **usiFileFormat** (привязанной к радиокнопке выбора формата файла) происходит формирование пути к выгружаемому или загружаемому файлу (путём подстановки значения входа ФБ ScheduleManager с названием **sPathToBinFile** или **sPathToCsvFile** соответственно) и его запись в локальную переменную фрейма **sFileTransferPath**. Эта же переменная привязана в действии **Передача файла**. Для

кнопки **Загрузка (web-visu)** в этом же действии привязаны локальные переменные фрейма **dwFileTransferFlags** (битовая маска флагов передачи; используется значение

VisuElems.VisuEnumFileTransferControlFlags.ConfirmFileOverwriteInPlc – при существовании файла с заданным именем в веб-визуализации будет запрошено подтверждение на его перезапись) и **xFileDownloadInProgress** (флаг «выполняется загрузка файла»).

Кнопки **Выгрузка (web-visu**) и **Загрузка (web-visu)** являются невидимыми, если вход **xHideFileTransferButtons** ФБ ScheduleManager имеет значение **TRUE** (см. параметр **Переменные состояний/Невидимый** этих кнопок).

Для кнопки **Удалить интервал** во вкладке **Конфигурация ввода** для события **OnMouseClick** добавлено действие **Выполнить ST-код** с вызовом метода DeleteSchedInterval.

Для кнопки **Удалить всё** во вкладке **Конфигурация ввода** для события **OnMouseClick** добавлено действие **Выполнить ST-код** с вызовом функции MemFill из библиотеки **CAA Memory** для затирания нулями всего массива бинарных данных интервалов и метода Init со значение **TRUE** на входе **xClearAll** для очистки таблицы.

***

Теперь перейдем к диалогу **dlgIntervalSettings**, открытие которого происходит при клике на ячейку столбца «Конфигурация…».

В этом диалоге расположен элемент **Фрейм**, в конфигурации которого настроено три экрана – **frmSetInterval**, **frmSetDays** и **frmSetOutputs**. Локальная переменная диалога **usiFrameState** соответствует текущему отображаемому в фрейме экрану (**0** – **frmSetInterval**, **1** – **frmSetDays**, **2** – **frmSetOutputs**).

Локальная переменная **stData** (типа **SCHED_DIALOG_VISU**) содержит переменные, привязанные к элементам всех экранов. Привязки переменных сделаны во вкладке **Ссылки** в параметрах фрейма.

За пределами фрейма находится кнопка закрытия диалога, кнопки **Назад**, **Далее** и **Применить**, и прямоугольник с надписью **Вызов методов** (он не является частью фрейма).

Прямоугольник используется для циклического вызова методов InitDialog и CheckForEmptyDaysOfWeek – они привязаны к его параметрам **Переменные состояний/Невидимый** и **Переменные состояний/Отключение ввода** соответственно. Так как метод **InitDialog** всегда возвращает значение **TRUE** – этот прямоугольник является невидимым для конечного пользователя.

Кнопка **Назад** является неактивной при отображении в фрейме экрана **frmSetInterval** (см. её параметр **Переменные состояний/Отключение ввода**). В её вкладке **Конфигурация ввода** для события O**nMouseClick** добавлены действия **Выполнить ST-код (**usiFrameState := usiFrameState - 1;) и **Переключить визуализацию фрейма** (Переключить локальную визуализацию по шагам – на предыдущий).

Кнопка **Далее** может являться неактивной в одном из 3 случаев:

- если сейчас в фрейме отображается последний экран (**frmSetOutputs**);
- если интервал, заданный на экране **frmSetInterval**, является некорректным (см. описание метода CheckForNonValidInterval);
- если на экране **frmSetDays** выбран режим недельного расписания и при этом не выделен ни один чекбокс дней недели (см. описание метода CheckForEmptyDaysOfWeek).

Для кнопки **Далее** во вкладке **Конфигурация ввода** для события **OnMouseClick** добавлены действия **Выполнить ST-код** (usiFrameState := usiFrameState + 1;) и **Переключить визуализацию фрейма** (Переключить локальную визуализацию по шагам – на следующий).

Кнопка **Применить** является активной только в случае отображения в фрейме экрана **frmSetOutputs** (см. её параметр **Переменные состояний/Отключение ввода**). В её вкладке Конфигурация ввода для события **OnMouseClick** добавлены действия **Переключить визуализацию фрейма** (на **frmSetInterval**), **Выполнить ST-код** и **Закрыть диалог**.

В ST-коде выполняется обнуление переменной **usiFrameState** и вызов метода ApplySchedInterval.

Для кнопки закрытия диалога (расположенной в его правом верхнем углу) настроены те же самые действия – только без вызова метода **ApplySchedInterval**.

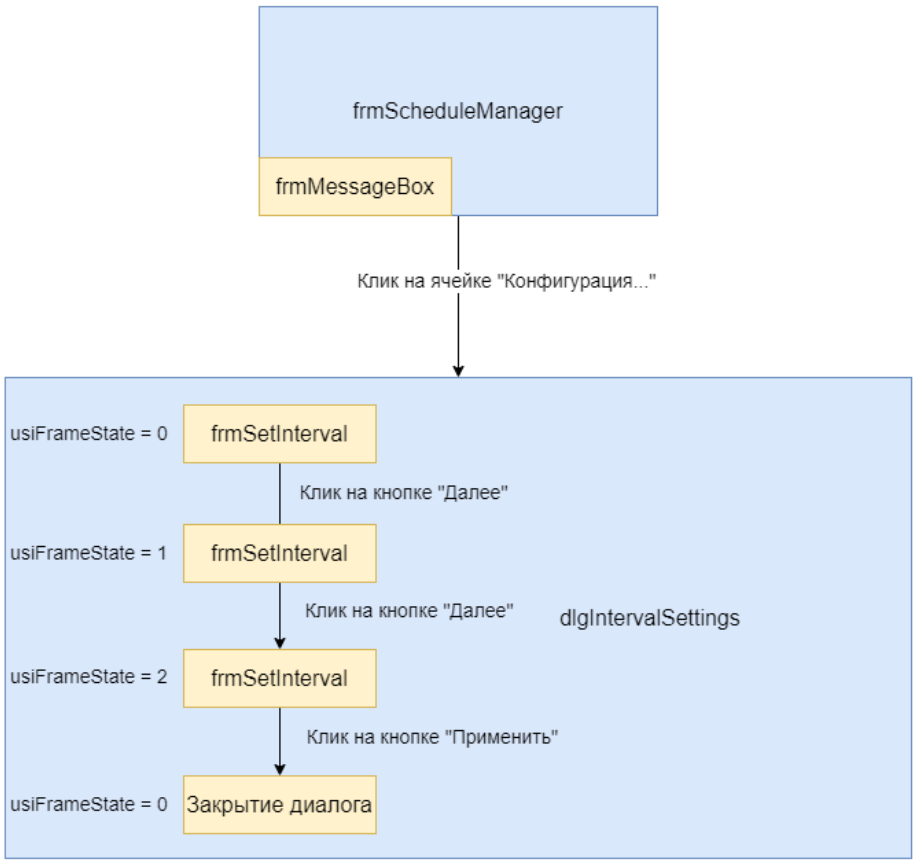

Рисунок 2.5.3 – Взаимосвязь визуализаций библиотеки

В фрейме **frmSetInterval** в прямоугольнике с надписью «**Интервал задан некорректно**!» в параметре **Переменные состояний/Невидимый** привязан вызов метода CheckForNonValidInterval с инвертированным значением выхода.

Для прямоугольников ввода часов и минут во вкладке **Конфигурация ввода** для события **OnMouseClick** добавлено действие **Записать переменную** с ограничением диапазона (**0-23** для часов и минут **0-59** для минут).

В фрейме **frmSetDays** для прямоугольников ввода месяца и дня во вкладке **Конфигурация ввода** для события **OnMouseClick** добавлено действие **Записать переменную** с ограничением диапазона (**1-12** для месяца и минут **1-31** для дня).

В фрейме **frmSetOutputs** нет ничего, что заслуживало бы отдельного упоминания.

2.5. Вопросы и ответы

Вопрос 1: как выполнить локализацию библиотеки?

Ответ: для локализации потребуется заменить тексты в следующих объектах:

- список текстов **GlobalTextList** (тексты, отображаемые в визуализации библиотеки);
- список текстов **SchedTextList** (текст, отображаемый в столбце «Конфигурация…» таблицы);
- строковые константы фрейма **frmScheduleManager** (сообщения о результате экспорта/импорта);
- код метода **DayOfWeekToFormatWstring** (сокращённые обозначения дней недели);
- код метода **MonthAndDayToFormatWstring** (названия месяцев);
- код метода **SchedStateToWstring**;
- код метода **SchedTypeToWstring**.

Вопрос 2: как увеличить максимальное число интервалов, обслуживаемое ФБ ScheduleManager?

Ответ: используемое в библиотеке ограничение на **255** интервалов является осознанным и кажется разумным. Но, в целом, увеличение максимального числа интервалов до значений, близких к верхней границе типа **INT** (**32767**), не должно быть особенно трудоемким.

Потребуется исправить:

- тип и значение константы **ScheduleManager.c_usiMaxIntervalsCount**;
- верхнюю границу переменной **ScheduleManager.pastVisuSchedule**;
- тип переменной **ScheduleManager.usiCurrentInterval**;
- тип переменной **ScheduleManager.usiPossibleIntervalsCount**;
- тип переменной **ScheduleManager.usiPossibleIntervalsCount**;
- верхнюю границу переменной **frmScheduleManager.iSelectedLineIndex**;
- тип переменной **i** в методе CheckForSpecialDay;
- тип переменной **i** в методе Main.

В фрейме frmScheduleManager потребуется привязать к параметру массив данных переменную **pastVisuSchedule**.

Вопрос 3: как увеличить число выходов, обслуживаемое ФБ ScheduleManager?

Ответ: в рамках текущей реализации библиотеки несложно увеличить число выходов до **64**. Для этого достаточно изменить типы переменных **dwOutputsMask** (ФБ **ScheduleManager**) и **dwOutputsMaskSetThisCycle** (метод Main) на **LWORD**.

Увеличение числа выходов сверх этого значения потребует более существенных изменений; конкретный вариант должен быть выбран автором изменения.