Акционерное общество
«Экспериментальный завод научного приборостроения
со Специальным конструкторским бюро
Российской академии наук»

УТВЕРЖДЕН
КУНИ.505200.023-01.01 31-ЛУ

SCADA-СИСТЕМА "СОНАТА"

Описание применения
КУНИ.505200.023-01.01 31

Листов 66

Инв. № подл.	Подпись и дата	Взам. инв. №	Инв. № дубл.	Подпись и дата

2025

Литера О1

АННОТАЦИЯ

Данный документ является описанием функций и возможностей SCADA-системы «Соната». В документе представлена её архитектура, модули и механизм их взаимодействия.

СОДЕРЖАНИЕ

1. Назначение 5
1.1. Назначение комплекса 5
1.2. Описание комплекса 5
1.3. Возможности комплекса 7
1.3.1. Возможности средств разработки 7
1.3.2. Возможности среды исполнения 7
2. Условия применения 9
2.1. Технические средства, необходимые для выполнения комплекса "Соната" 9
2.1.1. Технические средства для организации контроллера ввода-вывода 9
2.1.2. Технические средства для организации вычислительного узла 10
2.1.3. Технические средства для организации АРМ 10
2.1.4. Технические средства для организации архивной станции 10
2.2. Программные средства, необходимые для работы комплекса "Соната" 11
3. Описание задачи 12
3.1. Архитектура системы "Соната" 12
3.1.1. Особенности SCADA-системы "Соната" 12
3.1.2. Структура проекта 15
3.1.3. Сигналы и типы данных 18
3.1.4. Архитектура приложений системы 24
3.1.5. Виды приложений системы 30
3.2. Работа узла 32
3.3. Сбор и первичная обработка аналоговых и дискретных сигналов 33
3.4. Регистрация и архивирование событий и тревог 34
3.4.1. Регистрация событий 34
3.4.2. Управление тревогами 35
3.4.3. Архивирование событий 37
3.5. Архивирование сигналов 37
3.6. Шифрование данных в SCADA "СОНАТА" 38
3.7. Диагностика системы 39
3.7.1. Автоматический контроль целостности файлов проекта 39
3.7.2. Автоматическая диагностика запуска и работы приложений. 39
3.8. Отображение технологической информации 40
3.9. Автоматическое управление 41
3.9.1. Циклические программы ST (IEC-61131) 41
3.9.2. Событийные программы FBD (IEC-61499) 42
3.9.3. Смешанные программы LUA 5.3 44
3.10. Резервирование 44
3.10.1. Автоматическое резервирование сети 44
3.10.2. Автоматическое резервирование архивов сигналов и архивов тревог 44
3.10.3. Автоматическое резервирование узлов 44
3.10.4. Резервирование на проектном уровне 45
3.11. Синхронизация времени 45
3.12. Многооконный режим 46
3.13. Межпроектное взаимодействие 49
3.14. Руководство по созданию проектов 49
4. Метрологически значимая часть 51
4.1. Описание структуры ПО и выполняемых функций 51

4.2. Последовательность обработки данных 51
4.3. Описание функций и параметров ПО, подлежащих метрологическому контролю 52
4.4. Описание реализованных в ПО расчётных алгоритмов 52
4.4.1. Алгоритм работы блока (функции) кусочно-линейного преобразования 52
4.5. Описание модулей ПО 53
4.6. Перечень и описание интерфейсов 53
4.6.1. Влияние через интерфейс пользователя 53
4.6.2. Влияние через интерфейс связи 54
4.7. Список, значение и действие всех команд, получаемых от клавиатуры, мыши и других устройств ввода 54
4.8. Получение идентификационных признаков SCADA-системы "Соната" и проверка метрологических библиотек 54
4.9. Защита программного обеспечения и данных 55
4.9.1. Защита от случайных или непреднамеренных изменений 55
4.9.2. Защита от преднамеренных изменений программного обеспечения 56
4.10. Описание интерфейсов пользователя, всех меню и диалогов 56
4.11. Описание хранимых или передаваемых наборов данных 56
5. Обеспечение информационной безопасности в SCADA системе "Соната" 57
5.1. Функция создания, редактирования и контроля учетных записей пользователей 57
5.2. Функция шифрования передаваемых данных между узлами 59
5.3. Функция контроля целостности среды исполнения SCADA системы «Соната» и прикладного программного обеспечения 60
5.4. Функция разграничения передачи информации по IP-адресам и маскам подсети между узлами системы 61
5.4.1. Режим зеркалирования 62
5.4.2. Режим маршрутизации 62
5.5. Функция ведения журнала событий информационной безопасности и диагностики 64
5.6. Функция контроля версий прикладного программного обеспечения 64

1. НАЗНАЧЕНИЕ

1.1. Назначение комплекса

Название класса программных систем, к которым относится система "СОНАТА" - SCADA, представляет собой аббревиатуру (Supervisory Control And Data Acquisition), буквально переводимую на русский язык как «Диспетчерское управление и сбор данных» (далее SCADA).

SCADA-система "СОНАТА" состоит из двух частей: средств разработки и среды исполнения.

Средства разработки – набор программ, предоставляющих разработчикам эффективный инструмент для создания проекта автоматизации технологического процесса.

Среда исполнения включает набор программ, выполняющих вспомогательные функции, и набор программ, работающих в реальном времени на контроллерах, АРМах (автоматизированных рабочих местах), архивных станциях и т.д.

Программы, выполняющие вспомогательные функции, могут запускаться в процессе работы системы и, по выполнению своих функций, завершаются. К вспомогательным функциям относятся печать на принтер, считывание конфигурации контроллеров, вывод информации о работающем дистрибутиве SCADA "СОНАТА", поверка входных аналоговых каналов, администрирование пользователей работающей системы и др.

Программы, работающие в реальном времени, выполняют функции сбора данных, исполнение технологических алгоритмов, архивирование и отображение технологической информации.

1.2. Описание комплекса

Таблица 1.1 - Таблица, описывающая принадлежность модулей SCADA системы "СОНАТА" к соответствующей части (средства разработки и средства исполнения в режиме реального времени)

Имя приложения	**Средства разработки**	**Среда исполнения**	
		Программы, выполняющие вспомогательные функции	**Программы реального времени**
AlertArchive			+
Archive			+
ArchiveViewer		+	
Bridge			+
ControlCenter	+	+	
Debugger	+		
Distributer	+	+	
DTS_Editor	+		
EventLogger			+
HTMLEditor	+		
IEC60870_Editor	+		

Имя приложения	Средства разработки	Среда исполнения	
		Программы, выполняющие вспомогательные функции	Программы реального времени
IEC61850			+
IEC61850_Editor	+		
IECConsoleEngine			+
IECEditor	+		
IECWindowEngine			+
ImagePrinter		+	
Katren_LOCALBUS_Editor	+		
Katren_MODBUS			+
Katren_MODBUS_Editor	+		
KM04			+
KM04_Editor	+		
Loader			+
LuaEngine			+
MODBUS			+
MODBUS_Editor	+		
OPCUA			+
OPCUA_Editor	+		
OPCUA_Server			+
OPCUA_Server_Editor	+		
ProjectManager	+		
ReadSwitch		+	
ReportEditor	+		
ReportEngine			+
SignalViewer	+	+	
SNMP			+
SNMP_Editor	+		
SonataVer		+	
Sonet_Failover_Editor	+		
Sonet_LOCALBUS_Editor	+		
Sonet_MODBUS			+
Sonet_MODBUS_Editor	+		
Sound			+
SoundEditor	+		
SourceEditor	+		

Имя приложения	Средства разработки	Среда исполнения	
		Программы, выполняющие вспомогательные функции	Программы реального времени
SPABUS			+
SPABUS_Editor	+		
SQL			+
SQL_Editor	+		
Telekont2_MODBUS			+
Telekont2_MODBUS_Editor	+		
UserListEditor	+	+	
Verification		+	
WebServer			+
WebBrowser		+	

1.3. Возможности комплекса

1.3.1. Возможности средств разработки

Средства разработки реализуют следующие функции:
- создание таблицы сигналов проекта;
- привязка сигналов к физическим каналам ввода-вывода;
- настройка параметров архивирования;
- создание технологических алгоритмов на языках FBD, ST, LUA;
- разработка интерфейса пользователя;
- создание шаблонов отчётов.

1.3.2. Возможности среды исполнения

Средства исполнения системы «Соната» реализуют следующие функции, типичные для систем подобного класса:

- сбор информации с устройств нижнего уровня (датчиков, контроллеров);
- прием и передача команд оператора/диспетчера на контроллеры и исполнительные устройства (дистанционное управление объектами);
- сетевое взаимодействие с информационной системой предприятия (с вышестоящими службами);
- автоматизированное управление объектом автоматизации;
- отображение параметров технологического процесса и состояния оборудования с помощью мнемосхем, таблиц, графиков и т.п. в удобной для восприятия форме;

- оповещение эксплуатационного персонала об аварийных ситуациях и событиях, связанных с контролируемым технологическим процессом и функционированием программно-аппаратных средств автоматизированной системы управления технологическим процессом (далее АСУ ТП) с регистрацией действий персонала в аварийных ситуациях;
- хранение полученной информации в архивах;
- представление текущих и накопленных (архивных) данных в виде графиков (тренды);
- вторичная обработка информации;
- формирование сводок и других отчетных документов по созданным на этапе проектирования шаблонам.

2. УСЛОВИЯ ПРИМЕНЕНИЯ

2.1. Технические средства, необходимые для выполнения комплекса "Соната"

SCADA-система "Соната" может функционировать на компьютерах и контроллерах, построенных на следующих современных архитектурах процессоров:

- x86, x86_64: AMD, Intel, Vortex;
- ARM: v7, v8, v9, v11;
- MIPS: MIPS32, MIPS64;
- Эльбрус: Эльбрус-2С+, Эльбрус-4С, Эльбрус-8С;

Проект автоматизации на системе "Соната" может функционировать на разных типах процессоров и разных типах операционных систем одновременно. Например, проект может содержать контроллеры, выполненные на процессорах Vortex86, работающие под управлением операционной системы QNX 6.5, и АРМ, работающие на процессорах "Эльбрус-2С+" и операционной системе "Эльбрус". Таким образом, система "Соната" позволяет строить гетерогенные системы управления.

ВНИМАНИЕ! При эксплуатации проекта загрузка процессора на любом из узлов системы не должна превышать 70%. Однако, для контроллеров "Сонет", работающих под управлением операционной системы QNX 6.5.0, допускается загрузка процессора до 100 % при условии отсутствия потери данных.

2.1.1. Технические средства для организации контроллера ввода-вывода

Контроллер ввода-вывода предназначен для опроса аналоговых и дискретных каналов, выполнения примитивных алгоритмов управления и блокировки, передачи полученных данных вычислительному узлу или АРМ.

Минимальные требования:

- процессор: 200 МГц;
- оперативная память: 64 МБ;
- дисковая память: 256 ГБ;
- сетевой интерфейс: 100 МБит/с.

Контроллер в минимальной конфигурации может обеспечить опрос до 128 каналов (физ. каналы и статусы) и обработку порядка 1000 элементарных ячеек (логический сигнал, элемент массива или поле структуры).

Рекомендуемые требования:

- процессор: 1 ГГц;
- оперативная память: 512 МБ;
- дисковая память: 1 Гб;
- сетевой интерфейс: 100 МБит/с.

Контроллер в рекомендуемой конфигурации может обеспечить опрос до 512 каналов (физ. каналы и статусы) и обработку порядка 10000 элементарных ячеек (логический сигнал, элемент массива или поле структуры).

В вышеуказанной конфигурации загрузка процессора контроллера составляет не более 60%.

2.1.2. Технические средства для организации вычислительного узла

Вычислительный узел представляет собой промышленный компьютер без монитора, с постоянным питанием, на котором выполняются основные технологические алгоритмы проекта. Вычислительный узел используется в том случае, когда не хватает вычислительной мощности контроллеров.

Минимальные требования:

- процессор: 1 ГГц;
- оперативная память: 1 Гб;
- дисковая память: 8 Гб;
- сетевой интерфейс: 100 МБит/с.

Рекомендуемые требования:

- процессор: 2 ГГц, 4 ядра;
- оперативная память: 2 Гб;
- дисковая память: 8 Гб;
- сетевой интерфейс: 1 ГБит/с.

Вычислительный узел в рекомендуемой конфигурации может обрабатывать порядка 100000 элементарных ячеек (логический сигнал, элемент массива или поле структуры).

2.1.3. Технические средства для организации АРМ

АРМ (автоматизированное рабочее место) – компьютер с монитором, предназначенный для визуального отображения состояния проекта и приёма команд управления от оператора. Если в качестве АРМ используется постоянно включенный компьютер, то на него можно возложить функции вычислительного узла проекта. В этом случае нужно объединять минимальные требования вычислительного узла с минимальными требованиями АРМ.

Минимальные требования:

- процессор: 1 ГГц;
- оперативная память: 2 Гб;
- дисковая память: 8 Гб;
- сетевой интерфейс: 100 МБит/с.

Рекомендуемые требования:

- процессор: 3 ГГц, 4 ядра;
- оперативная память: 8 Гб;
- дисковая память: 32 Гб;
- сетевой интерфейс: 1 ГБит/с.

2.1.4. Технические средства для организации архивной станции

Архивная станция представляет собой промышленный компьютер с постоянным питанием, осуществляющий запись на диск архивов значений сигналов, событий и тревог. В качестве архивной станции может выступать вычислительный узел или АРМ (при наличии постоянного питания). Архивная станция, прежде всего, требовательна к надёжности и быстродействию дисков. Рекомендуется использование твердотельных дисков (SSD) таких производителей, как Intel, Kingston, SanDisk.

В настоящее время система "Соната" использует строго периодическую запись на диск. Под этим подразумевается, что даже если сигнал не изменяет своего значения (например, дискретный), то запись данных всё равно производится с указанным интервалом. Данная архитектура используется для обеспечения возможности создания дублированных архивов.

Рекомендуемые требования:

- процессор: 2 ГГц, два или более ядра;
- оперативная память: 4 ГБ;
- дисковая память: SSD 256 ГБ со скоростью записи не менее 250 МБ/с;
- сетевой интерфейс: 1 ГБит/с.

Архивная станция в рекомендуемой конфигурации обеспечивает запись порядка 100000 записей в секунду. Таким образом, если планируется архивировать 1000 сигналов, то 100000/1000 = 100 раз в секунду можно успевать сохранять каждый сигнал.

2.2. Программные средства, необходимые для работы комплекса "Соната"

Для SCADA системы "Соната" поддерживаются следующие платформы:

- Microsoft Windows: 7 32/64 bit, 8 32/64 bit, 10 32/64 bit, server 2012;
- Linux x86, x86_64: Ubuntu 14.4 и старше:
  - необходимо наличие библиотек Qt 4.8.6 или Qt 4.8.7;
- Linux ASTRA - CommonEdition и SpecialEdition (gcc v.4.7.2):
  - необходимо наличие библиотек Qt 4.8.7;
  - проверьте наличие библиотеки libQtSvg нужной версии (она часто отсутствует в общем репозитории Qt) и установите её, в случае отсутствия. Проверить наличие данной библиотеки можно командой: **sudo ldconfig -p | grep libQtSvg**. Если вывод данной команды пустой, то в операционной системе не установлена библиотека libQtSvg. Для установки из репозиториев с помощью стандартных утилит apt-get или Synaptic необходим пакет libqt4-svg.
- Linux для ARM v.7 (gcc v.4.4.0):
  - необходимо наличие библиотек Qt 4.8.6;
- Linux Elbrus: e2k-2c+-linux, gcc v.4.4.0:
  - необходимо наличие библиотек Qt 4.8.6;
- QNX: x86 v. 6.5.0 (ЗОСРВ "Нейтрино"), gcc v.4.4.2:
  - необходимо наличие библиотек Qt 4.8.7;

Для остальных платформ система "Соната" может быть скомпилирована по требованию заказчика. Необходимым условием является наличие компилятора языка C++, совместимого с GCC версии 4.4.0 и старше, а также графической библиотеки Qt версии 4.8.4, 4.8.5, 4.8.6 или 4.8.7.

3. ОПИСАНИЕ ЗАДАЧИ

3.1. Архитектура системы "Соната"

3.1.1. Особенности SCADA-системы "Соната"

В настоящее время на рынке АСУ ТП представлено множество программных продуктов SCADA-систем. Рассмотрим их структуры и разберём основное отличие SCADA-системы "Соната" от них.

На рис. 3.1 представлена типичная схема АСУ ТП с центральным сервером сигналов. Проект состоит из нескольких управляющих программ, расположенных на нескольких узлах. Для передачи значения сигнала от одной программы другой используется центральное хранилище значений сигналов, как правило, это отдельный сервер, на котором работает специализированная система управления базами данных (далее СУБД).

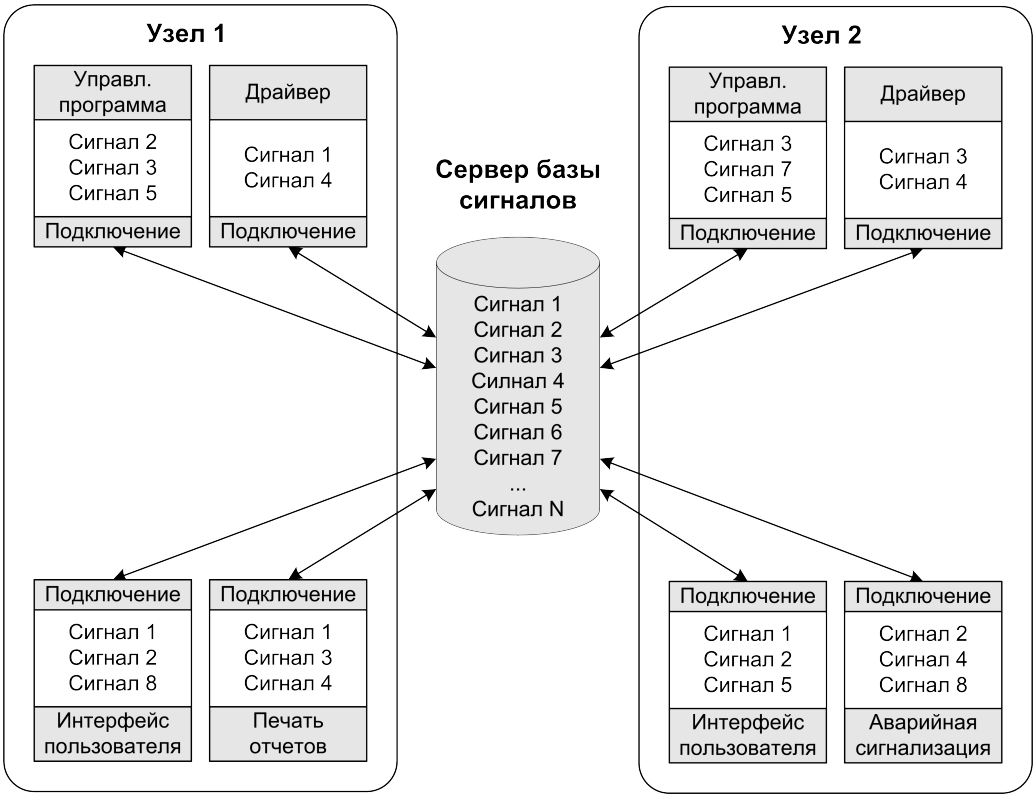


Рисунок 3.1 - Схема SCADA-системы с единой базой сигналов на выделенном сервере с индивидуальным подключением приложений к базе данных

В других системах для снижения нагрузки на центральный сервер сигналов на узлах применяют специальное приложение — диспетчер сигналов (см. рис. 3.2). В его задачу входит мультиплексирование запросов приложений к серверу, отправка единого запроса, получение ответа, демультиплексирование ответа по приложениям.

Недостатком таких систем является низкая живучесть в целом. При отказе самого сервера сигналов или канала связи с ним АСУ ТП становится неработоспособной. Для повышения надёжности системы применяют дублирование серверов с базами сигналов и дублирование линий связи.

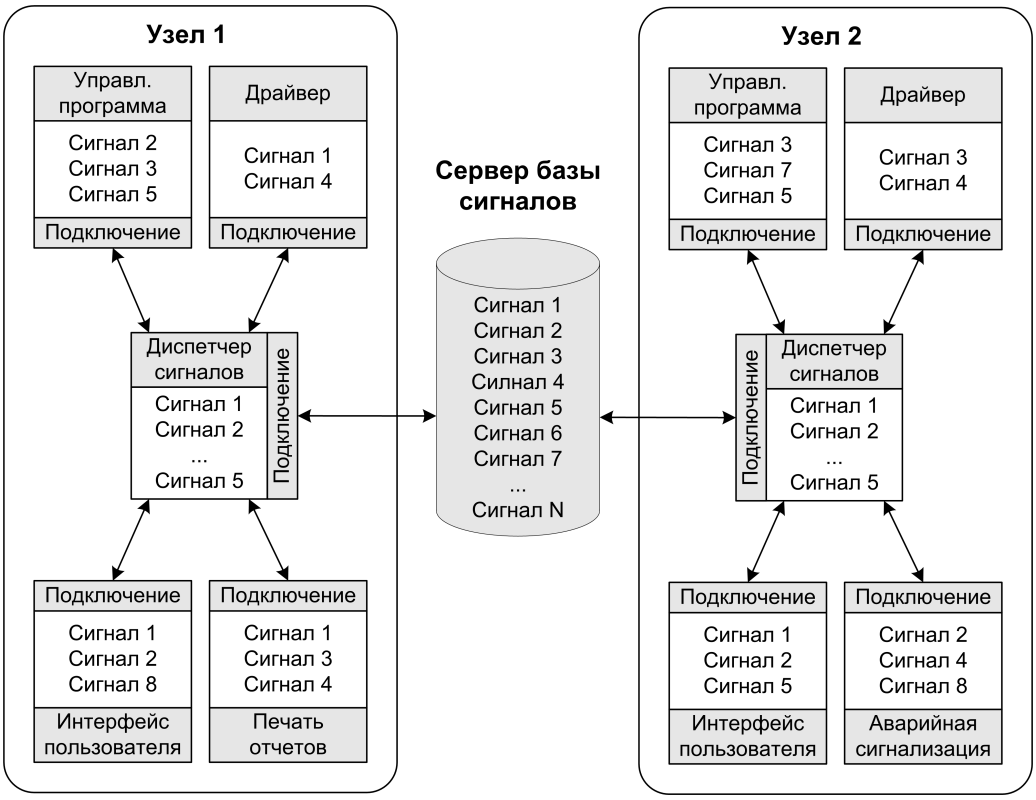


Рисунок 3.2 - Схема SCADA-системы с единой базой сигналов на выделенном сервере с групповым подключением приложений к базе данных через диспетчер сигналов

Для повышения живучести АСУ ТП были созданы SCADA-системы с "распределённой" базой сигналов (см. рис. 3.3). В них отсутствовал, как таковой, центральный сервер с базой сигналов. Необходимые для работы приложений узла сигналы хранились локально, в диспетчере сигналов узла. Связь между узлами осуществлялась диспетчерами сигналов. При отказе какого-либо узла система продолжала функционировать. На таком принципе основана SCADA-система "ОКО", поставляемая ФГУП ЭЗАН.

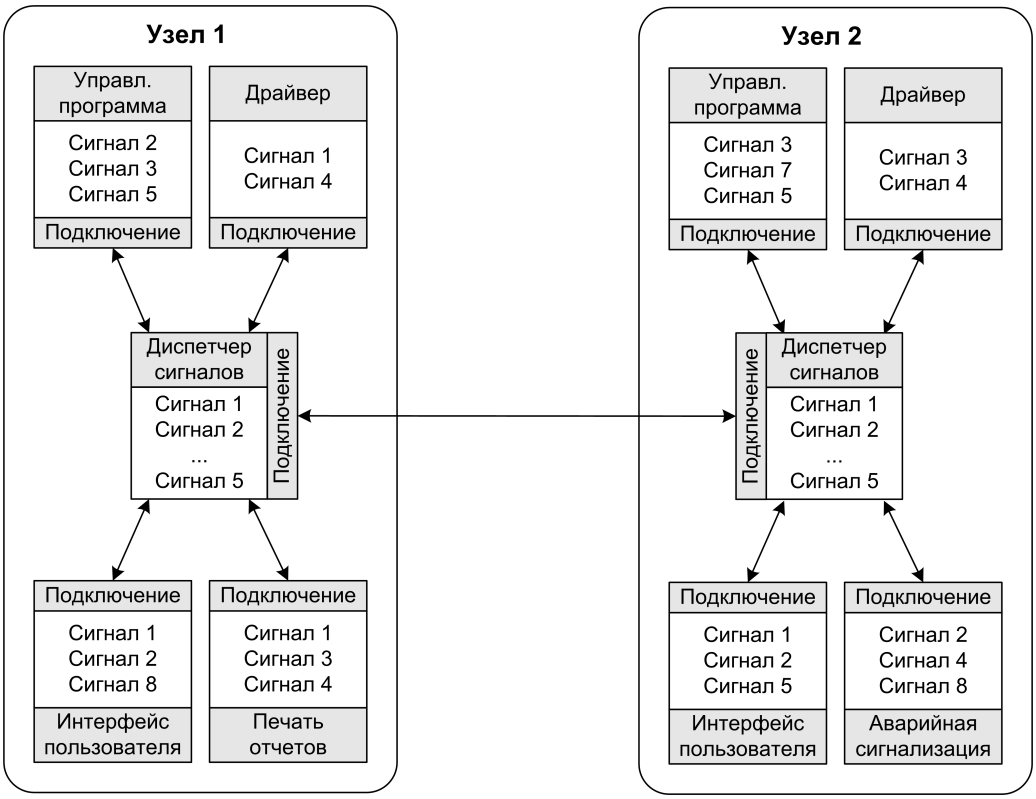


Рисунок 3.3 - Схема SCADA-системы с распределённой
по узлам базой сигналов внутри диспетчеров сигналов

Дальнейшим развитием идеи построения максимально децентрализованной SCADA-системы явилось создание SCADA-системы "Соната". Каждое приложение содержит в себе таблицу необходимых сигналов, самостоятельно независимо синхронизируя их с другими приложениями системы (см. рис. 3.4). В случае отказа какого-либо приложения, узла или линии связи, оставшаяся часть АСУ ТП продолжит функционирование. Каждое приложение в системе, будь то приложение-драйвер, управляющая программа, интерфейс пользователя или архив, является равнозначным. Такая организация системы позволяет организовывать любые схемы дублирования и резервирования.

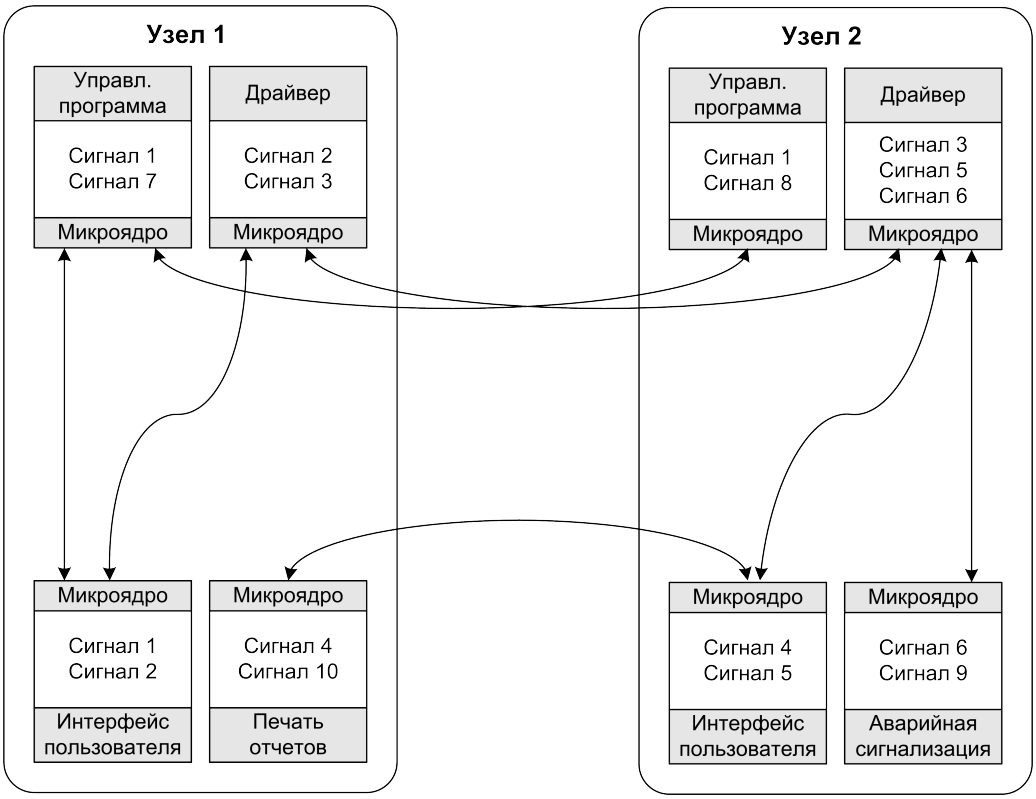


Рисунок 3.4 - Схема SCADA "Соната" с распределённой по приложениям базой сигналов

3.1.2. Структура проекта

Проект автоматизации, созданный при помощи SCADA-системы "Соната", представляет собой совокупность приложений, исполняемых на узлах, обменивающихся между собой информацией (сигналами) в реальном времени. С точки зрения обмена информацией все приложения (драйверы, алгоритмы, мнемосхемы и т.п.) являются равноправными. Проект требует для своего исполнения аппаратных средств, которые могут быть компьютерами или специализированными контроллерами. В проекте может использоваться множество контроллеров и компьютеров, обьединенных в компьютерную сеть. Для проекта каждый такой компьютер или контроллер является узлом компьютерной сети. Если на узле нет ни одного приложения SCADA-системы "Соната", то он не является частью проекта.

Контроллер - устройство управления в электронике и вычислительной технике.

Приложение – представляет собой программу (драйвер, алгоритм, мнемосхема т.п.), исполняемую на узле системы. На этапе проектирования разработчик создает или настраивает различные типы приложений, после чего размещает экземпляры приложений нужного типа на узлах системы.

Узел – компьютер или контроллер, на котором могут быть запущены приложения среды исполнения SCADA-системы «Соната». Основными характеристиками узла являются имя, набор IP-адресов сетевых интерфейсов, через которые осуществляется обмен данными между приложениями и перечень приложений. Если количество IP- адресов больше одного, то данные отправляются через все каналы передачи данных одновременно, организуя тем самым дублирование сетевых интерфейсов.

Минимальный по составу рабочий проект, созданный в SCADA-системе «Соната», должен состоять хотя бы из одного узла и одного приложения (то есть, хотя бы из одного аппаратного и

одного программного средства). Такой проект может, например, выполняться на контроллере, на котором запущено приложение, обрабатывающее физические сигналы (приложение-драйвер).

Физические сигналы – это сигналы с аналоговых (напряжения, тока, сопротивления и т.п.) и дискретных датчиков, а также выходные аналоговые (напряжения и тока) и дискретные (например, сигнальные лампочки или реле) сигналы.

Проект, состоящий из одного приложения и выполняемый на одном узле, может служить только для целей отладки и не имеет практической ценности. Минимальный проект, имеющий практическую ценность, должен, помимо приложения-драйвера, включать в себя хотя бы еще одно приложение, например, управляющую программу. Если на узле запущены два приложения с различной функциональностью, то необходимо решить задачу обмена информацией между этими приложениями в рамках единого проекта. Носителем информации в проекте является сигнал. Приложения могут обмениваться значениями сигналов. Полный список сигналов приложения называется интерфейсом приложения.

Сигнал – носитель информации в проекте.

Интерфейс приложения – полный список сигналов приложения.

Физические сигналы преобразуются в сигналы проекта с помощью приложения-драйвера.

В ряде случаев необходимо преобразование значения физического сигнала в другие физические единицы (например, преобразование сигнала от термопары, который измеряется в вольтах, в градусы Цельсия, или Кельвина, которым соответствуют измеряемые в вольтах значения). Для этой цели в SCADA-системе «Соната» вводится специальный элемент – преобразователь значения сигнала.

Преобразователь значения сигнала – описание монотонно возрастающей или монотонно убывающей функции, позволяющей однозначно осуществлять преобразование аргумента в значение и обратно.

Преобразователь значения сигнала позволяет приложению использовать значения сигналов, соответствующие значениям физических величин, а не только тем значениям, которые непосредственно от датчиков получает приложение-драйвер.

В SCADA-системе «Соната» сигналы могут быть локальными и глобальными:

- локальный сигнал доступен только внутри работающего приложения;

- глобальный сигнал доступен для любого приложения, работающего в составе данного проекта.

В SCADA-системе «Соната» приложения могут обмениваться значениями локальных сигналов путем создания связей приложений.

Связь приложений – прямое указание на обмен значениями сигнала одного работающего приложения и сигнала другого работающего приложения.

Другой способ передать значение сигнала приложениям – объявить его глобальным.

Приложение-драйвер, управляющая программа, графическое приложение и т.п. выполняют разные функции. Для выполнения похожих задач в проекте могут существовать приложения со схожей функциональностью. Для быстрого создания приложений со схожей функциональностью в SCADA-системе «Соната» существуют виды приложений.

Вид приложения – набор определённых свойств и функциональностей приложения, заданных SCADA-системой «Соната». Все приложения, относящиеся к данному виду, обладают заданными SCADA-системой «Соната» свойствами и функциональностью.

Примерами видов приложений могут быть приложение-драйвер, среда исполнения программы на технологическом языке, приложение-архив и т.п. Некоторые виды приложений обладают только теми свойствами и функциональностью, которые заданы SCADA-системой «Соната», другие, помимо заданных SCADA-системой «Соната» свойств, могут быть дополнены новыми качествами при создании и редактировании проекта.

Одним из важных видов приложения является графическое приложение, т.к. оно позволяет создавать человеко-машинный интерфейс.

При добавлении графического приложения к минимальному рабочему проекту на узле появляется человеко-машинный интерфейс. Для получения доступа к функциям, предоставляемым графическим приложением, необходима авторизация пользователя.

Пользователь – оператор, зарегистрированный в проекте. Каждый пользователь имеет индивидуальный логин и пароль.

Пользователи в проекте могут быть объединены в группы с различными правами доступа.

Список пользователей – полный перечень всех пользователей в проекте.

Минимальный рабочий проект, пусть он даже и включает в себя человеко-машинный интерфейс, (т.е. проект, состоящий из приложения-драйвера, управляющей программы и графического приложения) не требует, как правило, использования столь мощного средства программирования, как SCADA-система, и может быть выполнен на простом алгоритмическом языке программирования. Использование SCADA-систем оправдано при большом количестве узлов, имеющих близкую функциональность и типовое оборудование, т.е. в том случае, когда требуется обработка большого объема информации, поступающего от типовых стандартных объектов. Для такого случая характерно использование большого количества однотипных приложений-драйверов, управляющих программ и графических приложений. Для облегчения работы по созданию однотипных приложений в SCADA-системе «Соната» введено понятие типа приложения.

Тип приложения (шаблон) – это унифицированное описание интерфейса приложения и его конфигурации/схемы/программы. Каждый тип приложения относится к определённому виду приложения.

Экземпляр приложения – приложение, выполняемое на одном из узлов проекта и относящееся к одному (порождённое от одного) из типов приложения; все экземпляры приложений обладают одинаковым интерфейсом приложения и поведением.

Создание приложения в проекте начинается с создания типа приложения (см. Руководство оператора). Каждое приложение, выполняемое на каком-либо узле, является экземпляром приложения, порожденным определенным типом приложения. Соответственно, с учетом того, что каждое работающее приложение является экземпляром определенного типа приложения, нужно уточнить понятие связи приложений.

Связь приложений – прямое указание на обмен значениями сигнала одного экземпляра приложения и сигнала другого экземпляра приложения; типы приложений для этих экземпляров приложений могут не совпадать.

Проект, созданный с использованием всех возможностей SCADA-системы «Соната», содержит следующие элементы:

- типы приложений (шаблоны) различных видов приложений;
- узлы;
- экземпляры приложений различных типов приложений, выполняемые на узлах;
- преобразователи значения сигнала;
- связи между экземплярами приложений;
- глобальные сигналы;
- пользователи.

В процессе создания проекта разработчик создаёт сначала типы приложений (шаблоны). Затем указывает перечень узлов, образующих АСУ ТП. Затем на каждом узле размещает экземпляры приложений, порождённые от тех или иных типов приложений. В конце осуществляет связывание экземпляров приложений между собой связями или глобальными сигналами и создает пользователей.

3.1.3. Сигналы и типы данных

Сигнал представляет собой информационную единицу, хранящую значение определенного типа. Сигнал может быть скаляром или вектором определённого типа.

Тип может быть элементарным либо составным.

Данные сигналов хранятся в микроядре приложения. Сигналы элементарного типа передаются между микроядрами приложений в виде одного сетевого пакета. Сигналы составных типов передаются поэлементно.

Данные элементарного сигнала (скаляра) включают в себя следующие компоненты:

- тип данных (может быть элементарным либо составным);
- собственно значение;
- отметку времени, когда данное значение было зафиксировано;
- флаги, описывающие полученное значение.

3.1.3.1. Виды сигналов

3.1.3.1.1. Глобальные и локальные сигналы

На начальном этапе разработки проекта пользователь формирует глобальную таблицу сигналов, включающую в себя те сигналы, которые относятся к проекту в целом. Типичным примером глобального сигнала может служить сигнал, соответствующий физическому каналу ввода-вывода. На этапе разработки типов приложений глобальные сигналы добавляются в типы приложений из таблицы глобальных сигналов.

Помимо глобальных сигналов, каждый тип приложения может включать в себя локальные сигналы, специфичные для конкретного типа приложения.

3.1.3.1.2. Системные сигналы приложений

Набор сигналов приложения определяется множеством сигналов, задаваемых пользователем при разработке типа приложения. Помимо данных сигналов, каждое приложение изначально обладает набором системных сигналов, множество которых определяется видом приложения. Описание системных сигналов приведено в табл. 3.1.

Таблица 3.1 - Системные сигналы приложений

Имя сигнала	**Тип сигнала**	**Описание**
@PID	STRING	Идентификатор процесса приложения
@STATE	STRING	Состояние приложения
@COMMAND	STRING	Команда приложению
@COMMAND_VALUE	STRING	Данные команды приложению, если таковые требуются

Имя сигнала	**Тип сигнала**	**Описание**
@MESSAGE	STRING	Сообщение приложения
@MESSAGE_FRAMEWORK	STRING	Сообщения каркаса приложения
@EVENT	STRING	События приложения. Через данный сигнал архив событий извлекает события от приложения. Также через данный сигнал происходит запрос событий из архива
@RESERVED	BOOL	Состояние резерва приложения. Если значение данной переменной равно TRUE, то приложение находится в резерве
@RESERVE	BOOL	Команды вывода приложения в резерв. Если установить значение данной переменной в TRUE, то приложение начнет процесс вывода в резерв. По завершении процесса приложение выставит значение сигнала @RESERVED в TRUE. Если установить значение данной переменной в FALSE, то приложение начнет процесс вывода из резерва. По завершении процесса приложение выставит значение сигнала @RESERVED в FALSE
@LICENSE	STRING	Уникальный номер лицензии данного приложения
@ALARM	STRING	Через данный сигнал происходит запрос тревог из архива событий
@WINDOW	TWindow[32]	Для графических приложений значение данной переменной определяет положение, размер, флаги и видимость окон приложения
@NODE_ROLE	INT	Диагностический сигнал, формируемый драйвером БПР, указывающий на роль данного узла при резервировании. 0 - роль не определена, 1 -

Имя сигнала	Тип сигнала	Описание
		активный по умолчанию, 2 - резервный по умолчанию
@NODE_PEER_ROLE	INT	Диагностический сигнал, формируемый драйвером БПР, указывающий на роль узла партнера по резерву: 0 - роль не определена, 1 - активный по умолчанию, 2 - резервный по умолчанию
@NODE_ERROR	BOOL	Диагностический сигнал, указывающий на ошибки на узле
@NODE_PEER_ERROR	BOOL	Диагностический сигнал, указывающий на ошибки не узле партнёре по резерву
@NODE_RUN	BOOL	Диагностический сигнал, указывающий, что все приложения на узле запустились
@NODE_PEER_RUN	BOOL	Диагностический сигнал, указывающий, что все приложения на узле партнере по резерву запустились
@FAILOVER_PRESENCE	BOOL	Данный сигнал формирует драйвер БПР и сообщает о присутствии в схеме БПР: TRUE - в схеме есть БПР, FALSE - в схеме нет БПР
@FAILOVER_ERROR	BOOL	Данный сигнал формирует драйвер БПР и сообщает об ошибках БПР: TRUE - есть ошибки БПР, FALSE - нет ошибок БПР
@FAILOVER_RESERVE	BOOL	Данный сигнал отключает меандр от контроллера к БПР, что для БПР значит, что контроллер не отвечает и БПР произведет переключение на другой контроллер, если другой контроллер в порядке: TRUE - отключить меандр к БПР, FALSE - включить меандр к БПР
@FAILOVER_RESERVED	BOOL	Данный сигнал формирует драйвер БПР и сообщает о состоянии активный/резервный на БПР для данного контроллера: TRUE -

Имя сигнала	Тип сигнала	Описание
		состояние резервный, FALSE - состояние активный

3.1.3.1.3. Связывание сигналов

Для того чтобы приложения в ходе работы могли обмениваться данными, их сигналы должны быть связаны между собой. Глобальные сигналы связываются автоматически при компиляции проекта. Локальные сигналы экземпляров приложений связываются между собой в ходе разработки проекта.

Во время выполнения программы разница между глобальными и локальными сигналами приложений отсутствует.

3.1.3.2. Типы данных

3.1.3.2.1. Элементарные типы данных

Значение сигнала может иметь один из следующих элементарных типов (см. табл. 3.2).

Таблица 3.2 - Элементарные типы данных

Тип данных	Описание
Битовые типы данных	
BOOL	Логический тип сигналов (0 или 1, FALSE или TRUE: за истину полагается единица, за ложь – ноль)
BYTE	Байт, размер – 8 бит, значения от 0 до 255
WORD	Слово, размер – 16 бит, значения от 0 до 65 535
DWORD	Двойное слово, размер – 32 бита, значения от 0 до 4 294 967 295
Целочисленные типы данных	
USINT	Беззнаковый, размер – 1 байт, значения от 0 до 255
SINT	Знаковый, размер – 1 байт, значения от -128 до 127
UINT	Беззнаковый, размер – 2 байта, значения от 0 до 65 535
INT	Знаковый, размер – 2 байта, значения от -32 768 до 32 767

Тип данных	**Описание**
UDINT	Беззнаковый, размер – 4 байта, значения от 0 до 4 294 967 295
DINT	Знаковый, размер – 4 байта, значения от -2 147 483 648 до 2 147 483 647
ULINT	Беззнаковый, размер – 8 байт, значения от 0 до 18 446 744 073 709 551 615
LINT	Знаковый, размер – 8 байт, значения от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
Вещественные типы данных	
REAL	Размер – 4 байта, значения от 1,4×10^−45 (1,2×10^−38) до 3,4×10^+38
LREAL	Размер – 8 байт, значения от 5.0×10^−324 (2,3×10^−308) до 1,7×10^+308
Типы данных, содержащие значение даты и времени	
DT	Содержит значения даты и времени, размер 8 байт
DATE	Содержит значения даты, размер 8 байт
TOD	Содержит значения времени суток, размер 8 байт
TIME	Содержит значение длительности – продолжительности промежутка времени, размер 8 байт
Строковые типы данных	
STRING	Строка длиной до 511 байт; строка хранится в формате UTF8, поэтому каждый кириллический символ занимает 2 байта

3.1.3.2.2. Составные типы данных

Составной тип данных представляет собой массив данных определенного типа либо структуру.

3.1.3.2.2.1. Массивы

Массив представляет собой тип данных, включающий последовательность данных определенного типа. Для описания массива разработчик задает тип данных ячейки массива и размер массива. Если размер массива сигнала равен 0, то такой сигнал представляет собой скаляр.

3.1.3.2.2.2. Структуры

Структура представляет собой тип данных, представляющий собой последовательность полей разного типа. Для описания структуры задается множество полей, для каждого из которых указывается тип и размер данных. Как и в случае массивов, если поле имеет размер 0, то оно представляет собой скаляр. Размер, больший 0, означает, что поле структуры представляет собой массив.

3.1.3.2.2.3. Предопределенные структурные типы данных

Тип TWindow предназначен для хранения данных об окне приложений. Описание типа данных приведено в табл. 3.3.

Таблица 3.3 - Поля типа данных TWINDOW

Имя поля	**Тип данных**	**Описание**
VISIBLE	BOOL	Признак видимости окна. Окно видимо, если значение данного поля равно TRUE
X	DINT	X-координата верхнего левого угла окна в пикселях
Y	DINT	Y-координата верхнего левого угла окна в пикселях
WIDTH	DINT	Ширина окна в пикселях
HEIGHT	DINT	Высота окна в пикселях
FLAGS	DINT	Флаги окна
CAPTION	STRING	Заголовок окна

Значение поля, задающего флаги окна, представляет собой сумму значений, определяющих отдельные характеристики окна. Перечень значений приведен в табл. 3.4.

Таблица 3.4 - Флаги окна

Значение	**Характеристика**
1	Окно не имеет рамки и, соответственно, кнопок свертывания, развертывания и закрытия
2	Окно нельзя перемещать
4	Нельзя изменять размер окна
8	Окно нельзя закрыть
16	Окно нельзя свернуть
32	Окно нельзя развернуть на весь экран
64	Окно не может восстанавливать свой размер из свёрнутого состояния
128	Окно располагается поверх всех остальных окон

3.1.3.2.2.4. Флаги сигнала

Флаги сигнала предоставляют дополнительную информацию о значении сигнала. Описание флагов сигнала приведено в табл. 3.5."

Таблица 3.5 - Флаги сигнала

Бит	**Описание**
0-9	Пользовательские флаги сигнала
10-12	Не используются (зарезервированы)
13	Флаг невалидности (недостоверности) сигнала
14	Изменение сигнала заблокировано (значение сигнала изменить нельзя)
15	Игнорировать изменение флагов

3.1.4. Архитектура приложений системы

3.1.4.1. Микроядро приложения

Каждое приложение, функционирующее в среде исполнения системы "Соната", содержит в себе микроядро, хранящее значения сигналов приложения и осуществляющее синхронизацию сигналов с другими приложениями. Приложения, являющиеся внешними по отношению к среде исполнения системы, микроядра не имеют. Структурная схема микроядра приложения приведена на рис. 3.5

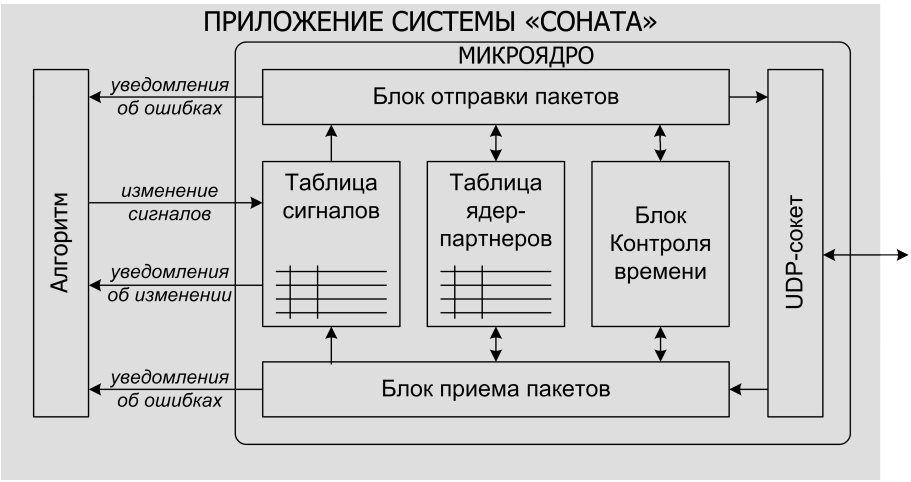


Рисунок 3.5 - Схема микроядра приложения

Микроядро приложения включает в себя следующие основные компоненты:

- таблицу сигналов, в которой хранятся значения, метки времени, флаги и другие характеристики сигналов;

- таблицу ядер-партнеров, в которой хранится информация о ядрах-партнерах, с которыми осуществляется синхронизация сигналов;

- блок контроля времени, отслеживающий отметки времени отправленных и принятых пакетов;

- блок отправки пакетов, формирующий UDP-пакеты измненения сигналов, откликов на запросы и др.;

- блок приема пакетов, разбирающий пришедшие UDP-пакеты и вносящий изменения в таблицы сигналов и ядер-партнеров;

- сокет, через который осуществляется передача данных.

Микроядро приложения выполняет следующие функции:

- хранение списка приложений-партнеров;

- хранение актуальных значений сигналов;

- получение оповещений об изменении значений сигналов и оповещение приложения об изменении;

- отслеживание работоспособности приложений-партнеров;

- прием информации об изменении значения сигнала от приложения и оповещение приложений – партнеров об изменении значений;

- отслеживание функционирования приложений-партнеров;

- оповещение приложения о возникающих проблемах.

Все приложения в системе "Соната", как в пределах одного узла, так и между узлами, обмениваются друг с другом данными, используя сетевой протокол на основе UDP (передача дейтаграмм).

Протокол UDP имеет ряд достоинств по сравнению с протоколом TCP:

- не требует установки соединения и не имеет внутреннего состояния. Приём и передача начинается непосредственно с пакета данных;

- данные передаются атомарным сообщением, а не потоком. Вследствие этого не требуются проверки для определения конца сообщения;

- не содержит какую-либо буферизацию пакетов. Пакеты из сетевого протокола UDP сразу передаются в драйвер сетевой карты;

- один сетевой сокет используется на приём и передачу данных со всеми приложениями. Это позволяет экономить ресурсы операционной системы и иметь неограниченное количество связей с другими приложениями.

Перечисленные достоинства протокола UDP позволяют гораздо точнее и с меньшими накладными расходами на стороне приложения осуществлять передачу данных и контролировать её таймауты.

Недостатками протокола UDP являются:

- отсутствие контроля пропажи сетевого пакета;

- сетевые пакеты могут приходить не в том порядке, в котором они были отправлены;

- максимальный объём атомарного сообщения не может превышать MTU (Maximum Transmission Unit = 1500 байт).

В системе «Соната» используется собственный протокол верхнего уровня, который устраняет перечисленные недостатки UDP. В протоколе системы “Соната» реализованы следующие приёмы:

- для обеспечения гарантированной доставки данных используется обязательное подтверждение каждого переданного пакета принимающей стороной. Сетевой протокол повторяет передачу исходного пакета до наступления одного из двух событий: либо приходит пакет-подтверждение приёма, либо истекает таймаут связи с данным сторонним приложением. Время повторной передачи пакета является настраиваемым параметром, по умолчанию его значение равно 50 мс;

- для ограничения использования сети протокол осуществляет отправку одновременно строго определённого количества сетевых пакетов. По умолчанию одномоментно отправляется не более 20 пакетов. Для отправленных пакетов организован пул, в котором они находятся, ожидая подтверждения. При подтверждении пакета место в пуле освобождается для нового отправляемого пакета;

- для уменьшения количества сетевых пакетов используется механизм упаковки значений нескольких изменившихся сигналов в один сетевой пакет. В текущей версии протокола в сетевой пакет может поместиться до 57 сигналов до достижения MTU. Поскольку сигналы могут изменяться очень часто, в протоколе реализовано накопление изменений в течение 5 мс при передаче данных в пределах узла и 10 мс при передаче на соседние узлы;

- любой пакет содержит отметку времени на момент отправки;

- каждое значение сигнала в пакете также имеет отметку времени его изменения. При приёме сетевого пакета происходит сравнение отметок времени значений сигналов в пакете с отметками времени значений сигналов в приложении. Если отметка времени в пакете более поздняя, то происходит изменение значения сигнала в приложении на новое. Тем самым решается проблема изменения порядка передачи сетевых пакетов;

- если два приложения не обмениваются значениями сигналов, то протокол отправляет специальные пакеты для контроля линии связи. По умолчанию интервал отправки контролирующих пакетов составляет 300 мс.

Время передачи Ethernet кадра размером в MTU по сети 100 Мб/с составляет менее 0.2 мс.

3.1.4.2. Жизненный цикл приложения

Жизненный цикл приложения начинается с его запуска. При этом приложение читает файл запуска, анализирует его, после чего запускает микроядро и переходит в состояние NONE. Приложение находится в состоянии NONE до тех пор, пока не получит команду инициализации либо команду остановки.

Инициализация приложения может происходить в один или два этапа. Способ инициализации зависит от команды, полученной приложением. Если приложение получает команду COLD, то оно начинает «холодный» старт, первым этапом которого является сброс значений RETAIN-переменных в начальное состояние. Для этого приложение переходит в состояние «холодного старта» (COLD). Если сброс значений retain-переменных прошел успешно, то приложение переходит в состояние "горячего старта", инициализирует переменные ядра начальными значениями (для обычных переменных) либо сохраненными значениями (для RETAIN и PERSISTENT переменных), инициализирует внутренние структуры элемента среды исполнения и готовит его к запуску. По окончании инициализации приложение переходит в состояние READY (готов).

Если же вместо команды COLD приложение получает команду HOT, то оно сразу же входит в режим «горячего» запуска, минуя стадию стирания значений RETAIN-переменных.

Находясь в состоянии READY, приложение ожидает прихода команды CONTINUE, при получении которой оно начинает процесс собственно запуска. Приложение переходит в состояние STARTING (запускается). Завершив процесс запуска, приложение переходит в состояние RUN (работа), в котором и проводит основное время своего жизненного цикла.

Для завершения работы приложения необходимо отдать ему команду TERMINATE. Приложение перейдет в состояние TERMINATE, завершит работу элемента среды исполнения, остановит работу механизмов ядра, перейдет в состояние TERMINATED, после чего завершит работу.

Схема жизненного цикла приложения приведена на рис. рис. 3.6

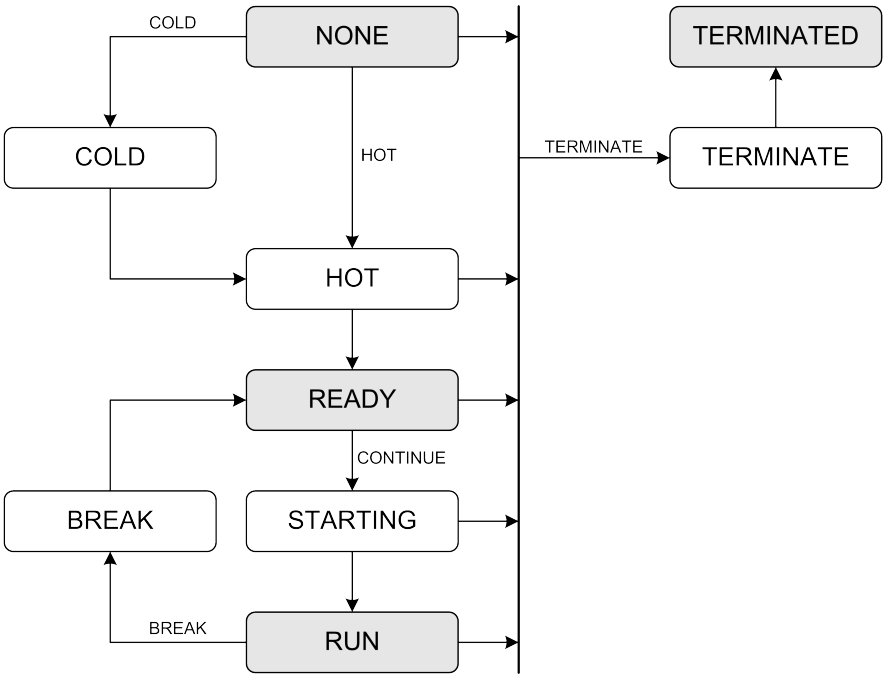


Рисунок 3.6 - Жизненный цикл приложения

Приложение, находясь в состоянии RUN, может быть на время приостановлено. Для того, чтобы приостановить приложение, необходимо отдать ему команду BREAK. По получении команды приложение перейдет в состоянии BREAK (останавливается), остановит вычислительные механизмы, после чего перейдет в состояние READY.

Из состояния READY приложение может перейти в три других состояния, используемых для отладки: STEP, STEP_OVER, CYCLE. В состояниях STEP, STEP_OVER приложение выполняет очередной шаг алгоритма. Состояние CYCLE используется только циклическими приложениями, которые выполняют в нем следующий цикл своего алгоритма.

Дополнительные состояния приложения приведены на рис. 3.7

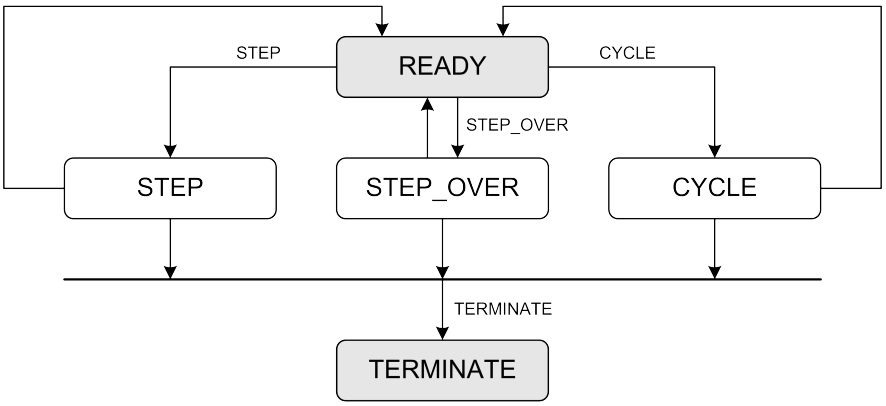


Рисунок 3.7 - Отладочные состояния приложения

3.1.4.3. Состояния приложения

Приложение, микроядро которого запущено, пребывает в одном из состояний, перечисленных в табл. 3.6. Состояние может быть устойчивым, выход из которого возможен при приходе команды извне, либо неустойчивым, выход из которого осуществляется приложением самостоятельно, после завершения выполняемой операции. Имя состояния приложения хранится в системной переменной @STATE микроядра.

Таблица 3.6 - Состояния приложения

Состояние	**Описание**
NONE	Начальное состояние, в котором приложение оказывается после запуска микроядра. Также в данное состояние приложение переходит из неустойчивых состояний в случае ошибочного завершения выполняемой операции
COLD	Состояние "холодного старта". В данном состоянии происходит сброс retain-переменных в начальное состояние. После сброса значений приложение переходит в состояние HOT. Если в ходе инициализации возникнут ошибки, то приложение перейдет в состояние NONE
HOT	Состояние "горячего старта". Обычные переменные получают начальные значения, а retain и persistent переменные - ранее сохраненные значения. По окончании данного этапа производится инициализация внутренних структур приложения. После инициализации приложение переходит в состояние READY. Если в ходе инициализации возникнут ошибки, то приложение перейдет в состояние NONE
READY	Состояние законченной инициализации и готовности приложения к запуску
STARTING	Происходит запуск приложения на выполнение (получена команда CONTINUE). По окончании запуска приложение перейдет в состояние RUN. Если в ходе запуска возникнет ошибка, то приложение перейдет в состояние NONE
RUN	Приложение запущено и выполняется
BREAK	Выполнение приложения останавливается (получена команда BREAK). По окончании остановки приложение перейдет в состояние READY. Если в ходе остановки возникнет ошибка, то приложение перейдет в состояние NONE
STEP	Приложение выполняет один шаг алгоритма, возможен вход в подпрограмму. По окончании шага приложение переходит в состояние READY
STEP_OVER	Приложение выполняет один шаг алгоритма. Вход в подпрограмму не производится, подпрограмма выполнятеся в ходе выполнения шага. По окончании шага приложение переходит в состояние READY

Состояние	**Описание**
CYCLE	Приложение выполняет один цикл алгоритма. По окончании шага приложение переходит в состояние READY
TERMINATE	Приложение завершает свою работу, происходит разрушение внутренних структур данных (получена команда TERMINATE). После успешного завершения процесса приложение перейдет в состояние TERMINATED. В случае возникновения ошибки приложение перейдет в состояние NONE
TERMINATED	Приложение завершило свою работу. Выход из данного состояния невозможен

3.1.4.4. Команды приложению

Команды служат для перевода приложения из одного состояния в другое. Команда передается приложению путем установки значения системной переменной @COMMAND. Перечень команд приведен в табл. 3.7.

Таблица 3.7 - Команды приложению

Команда	**Описание**
COLD	"Холодный" старт приложения. Вызывает переход в состояние COLD. Доступна в состояниях NONE, READY, RUN
HOT	"Горячий" старт приложения. Вызывает переход в состояние HOT. Доступна в состояниях NONE, READY, RUN
CONTINUE	Запуск приложения на выполнение. Если данная команда передана приложению, находящемуся в состоянии NONE, то приложение перейдет в состояние HOT, а после перехода в состояние READY - в состояние STARTING. Если данная команда передана приложению, находящемуся в состоянии READY, то приложение перейдет в состояние STARTING. В других состояниях команда будет проигнорирована
BREAK	Приостановка приложения. Если приложение находится в состоянии RUN, то оно перейдет в состояние BREAK. В других состояниях данная команда будет проигнорирована
STEP	Команда выполнения одного шага (с возможностью входа в подпрограмму). При

Команда	Описание
	получении данной команды приложение, находящееся в состоянии READY, перейдет в состояние STEP. В других состояниях данная команда будет проигнорирована
STEP_OVER	Команда выполнения одного шага (без возможности входа в подпрограмму). При получении данной команды приложение, находящееся в состоянии READY, перейдет в состояние STEP_OVER. В других состояниях данная команда будет проигнорирована
CYCLE	Команда выполнения одного цикла. При получении данной команды приложение, находящееся в состоянии READY, перейдет в состояние CYCLE. В других состояниях данная команда будет проигнорирована
TERMINATE	Завершение работы приложения. Приложение, приняв данную команду, перейдет в состояние TERMINATE. Данная команда доступна во всех состояниях, кроме состояния TERMINATED

3.1.5. Виды приложений системы

Вид приложения определяет то, к какой группе относится тип приложения, то есть, по сути, задает группу функций приложения. Примером вида может быть архив сигналов, мнемосхема, технологическая программа, драйвер какой-либо разновидности и т.п.

Базовая поставка системы «Соната» включает в себя виды приложений, описанные в табл. 3.8.

Таблица 3.8 - Виды приложений

Вид приложения	Описание
Внешние приложения	
APPLICATION	Внешнее неуправляемое приложение
APPLICATION.CONSOLE	Внешнее консольное управляемое приложение
APPLICATION.WINDOW	Графическое управляемое приложение системы
Приложения для построения технологических программ	
APPLICATION.IEC.CONSOLE	Консольное управляемое событийное приложение, написанное на языках стандарта IEC-61131 или IEC-61499
APPLICATION.LD.CONSOLE	Консольное управляемое циклическое приложение, написанное на языке LD (в данный момент не используется)

Вид приложения	**Описание**
APPLICATION.ST.CONSOLE	Консольное управляемое циклическое приложение, написанное на языке ST
APPLICATION.LUA	Управляемое событийно/циклическое приложение, написанное на языке LUA
Приложения для построения человеко-машинного интерфейса	
APPLICATION.IEC.WINDOW	Графическое управляемое событийное приложение, написанное на языках стадарта IEC-61131 или IEC-61499
Приложения архивации	
ARCHIVE	Приложение-архив, для хранения значений сигналов системы
EVENT LOGGER	Архив событий проекта (автоматически регистрирует события проекта)
Приложения для работы с WEB	
WEB BROWSER	Приложение для просмотра WEB - страниц, как с диска, так и с удалённого сервера
WEB SERVER	Приложение-сервер для отдачи по HTTP - запросу статических файлов
Приложения-драйверы	
DRIVER.DTS	Приложение-драйвер, выполняющее функции шлюза СВБУ "ПОРТАЛ"
DRIVER.MODBUS	Универсальное приложение-драйвер для любых устройств, поддерживающих обмен данными по протоколу MODBUS RTU или TCP
DRIVER.OPCUA	Приложение-драйвер, предназначенное для взаимодействия внешних OPCUA-серверов со SCADA-системой "Соната"
DRIVER.OPCUA.SERVER	Приложение-драйвер, работающее как OPCUA-сервер и предназначенное для предоставления доступа к сигналам SCADA-системы "Соната" для внешних программ OPCUA-клиентов
DRIVER.PCI	Приложение-драйвер КМ-04 (контроллера многофункционального производства ЭЗАН)
DRIVER.SNMP	Приложение-драйвер для работы с устройствами, предоставляющими доступ к данным по протоколу SNMP
DRIVER.SONET.FAILOVER	Приложение-драйвер для работы с модулем переключения резерва или БПР (блок переключения резерва)
DRIVER.SONET.LOCALBUS	Приложение-драйвер модулей ввода/ вывода контроллера СОНЕТ-Мастер (производства ЭЗАН)

Вид приложения	Описание
DRIVER.SONET.MODBUS	Приложение-драйвер для опроса по MODBUS контроллеров удаленного ввода-вывода СОНЕТ (производства ЭЗАН)
DRIVER.TELECONT.MODBUS	Приложение-драйвер для опроса по протоколу MODBUS RTU контроллеров ТЕЛЕКОНТ (производства ЭЗАН) - в настоящее время мало применяется
DRIVER.TELECONT2.MODBUS	Приложение-драйвер для опроса по протоколу MODBUS RTU контроллеров ТЕЛЕКОНТ (производства ЭЗАН) - применяется в настоящее время
Прочие приложения	
BRIDGE	Приложение для настройки межпроектной связи (можно передавать значения сигналов, события и тревоги между разными проектами SCADA-системы "Соната")
EVENT VIEWER	Приложение для просмотра событий проекта (данное приложение уже не используется и присутствует в списке приложений для совместимости с АСУ ТП, которые работают с более старой версией SCADA-системы "Соната")
REPORT ENGINE	Приложение для формирования, просмотра и печати отчётов
SOUND	Приложение для воспроизведения звука
TREND VIEWER	Приложение для просмотра графиков (данное приложение уже не используется и присутствует в списке приложений для совместимости с АСУ ТП, которые работают с более старой версией SCADA-системы "Соната")

3.2. Работа узла

Для превращения компьютера или контроллера в узел проекта нужно распаковать архив с системой "Соната" в нужную папку, и добавить в автозагрузку приложение Loader.Для получения подробной информации см. Руководство системного программиста.

Приложение Loader выполняет следующие функции:

- контроль целостности файлов проекта;
- пуск и останов приложений узла проекта;
- контроль работоспособности приложений узла;
- переключение приложений узла в резерв и обратно;
- приём и обновление файлов при рассылке проекта на узлы.

3.3. Сбор и первичная обработка аналоговых и дискретных сигналов

В системе "Соната" любое приложение может быть источником сигналов. Это может быть драйвер, у которого логические сигналы привязаны к физическим каналам ввода-вывода. Это может быть алгоритм, который формирует сигнал управления. Это может быть мнемосхема, которая по действиям пользователя изменяет тот или иной сигнал.

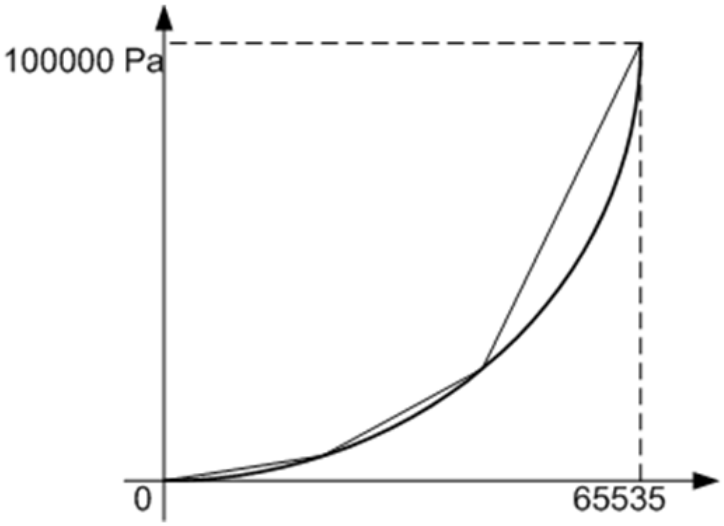
Для любого сигнала приложения можно задать так называемый "преобразователь сигнала" (DSP). Преобразователь осуществляет обработку значения сигнала из представления внутри приложения в значение, которое отправляется другим приложениям. Чаще всего преобразователи сигналов используются для преобразования кодов АЦП, получаемых драйвером от аппаратуры, в физические единицы измерения, а также для обратного преобразования значения в коды перед записью в ЦАП.

В текущей версии "Сонаты" в преобразователе сигнала можно настраивать две операции: преобразование значения при помощи монотонной функции и квантование.

Монотонная функция преобразования задаётся в виде кусочно-линейной функции с неограниченным количеством точек.

Пример 1. Драйвер опрашивает модуль 4-20 мА, к которому подключён датчик давления с нелинейной характеристикой 0-100000 Ра. Драйвер получает значение тока в кодах. 4 мА соответствует код 0, току 20 мА соответствует код 65535. Требуется осуществлять преобразование из внутреннего преставления 0-65535 кодов во внешнее представление 0-100000Па.

На рис. 3.8 представлена грубая интерполяция по 4 точкам нелинейной характеристики датчика.



Code	Pa
0	0
21845	10000
43690	30000
65535	100000

Рисунок 3.8 - Кусочно-линейная функция для интерполяции нелинейной характеристики датчика

Величина квантования (загрубления) задаётся в преобразователе сигнала указанием необходимой точности значения. Квантование используется для сокращения нагрузки на сеть и событийные вычислительные алгоритмы.

Пример 2. На объекте используется 1000 датчиков температуры, которые опрашиваются со скоростью 10 раз в секунду, для реализации быстрого аварийного отключения. Вследствие шума температурных датчиков происходит постоянное изменение их значений, что приводит к передаче по сети 1000*10 = 10000 значений в секунду. Для САУ не требуется высокая точность показаний этих датчиков, поэтому их значения можно загрубить до десятой доли градуса.

В дополнении к преобразователям сигналов, которые могут быть использованы в любом приложении Сонаты, в драйверах Сонаты можно использовать апериодический НЧ-фильтр первого порядка, для которого указывается постоянная времени в секундах.

3.4. Регистрация и архивирование событий и тревог

Событие, зарегистрированное системой "Соната", представляет собой запись, содержащую в себе информацию о месте, времени и сути возникновения какого-то явления. Событие попадает в архив, который, в свою очередь, предоставляет информацию всем остальным приложениям.

Тревога (Alarm) представляет собой информационный объект, описывающий состояние объекта автоматизации, требующее реакции оператора. Состояние тревоги и фаза ее жизненого цикла описывается при помощи последовательности тревожных событий. Соответственно, информация о тревоге хранится в архиве событий.

3.4.1. Регистрация событий

Событие описывается при помощи следующих параметров:

- отметки времени;
- идентификатора тревоги;
- состояния;
- категории;
- адреса события;
- источника события;
- имени пользователя;
- сообщения;
- дополнительных данных.

Отметка времени события хранится в архиве событий в формате UTC.

Идентификатор тревоги представляет собой некое текстовое поле, которое однозначно идентифицирует тревогу. По данному полю архив выделяет события, относящиеся к одной и той же тревоге.

Состояние события хранит в себе информацию о собственно состоянии события либо состоянии тревоги на момент наступления данного события. Фактически, состояние тревоги представляет собой состояние последнего по времени события, связанного с данной тревогой. Состояние событий, не связанных с тревогами, может принимать два состояния:

- 0 - исчезновение;
- 1 - появление.

Категория события представляет собой число, значение которого служит для группировки событий и тревог, а также определяет поведение тревоги. События, значение категории которых находится в интервале от 0 до 9999, не относятся к тревогам. Перечень категорий событий приведен в табл. 3.9.

Адрес события представляет собой текстовое поле, хранящее имя узла, заданного при проектировании, а также имя приложения, разделенные точкой. Адрес назначается автоматически при регистрации события.

Источник события логически является еще одним компонентом адреса события. Как правило, в данное поле записывается информация об объекте автоматизации, к которому относится возникшее событие.

В поле имени пользователя записывается имя оператора, прошедшего авторизацию на рабочей станции, на которой зарегистрировано событие. Если событие зарегистрировано не на рабочей станции либо пользователь не прошел авторизацию, то данное поле остается пустым.

Сообщение хранит в себе информацию, описывающую фактически произошедшее явление.

Дополнительная информация представляет собой текстовое поле, хранящее произвольную информацию, связанную с событием.

3.4.2. Управление тревогами

Фактически тревога представляет собой совокупность событий, объединенных в группу, определяемую идентификатором тревоги. События можно разделить на две группы - события, относящиеся к какой-либо тревоге (имеют непустое значение идентификатора тревоги) и события, с тревогами никак не связанные. Система "Соната" отличает первую группу событий от второй как по наличию непустого значения идентификатора тревоги, так и по категориям тревожных событий, определяющих поведение тревоги. Перечень категорий событий приведен в табл. 3.9.

Таблица 3.9 - Категории событий

Категория	**Описание**
0	Системное информационное сообщение
1	Системное предупреждение
2	Системное сообщение об аварии
3	Системное событие безопасности, относящееся к действиям пользователя
4	Системное информационное сообщение лицензирования
5	События настройки подсистемы информационной безопасности
6-999	Зарезервированные системные события
1000-9499	Пользовательские события, не относящиеся к какой-либо тревоге
9500-9999	Пользовательские события, относящиеся к информационной безопасности
10000-19999	Пользовательские тревожные события, относящиеся к обязательно квитируемым тревогам
20000-29999	Пользовательские тревожные события, относящиеся к необязательно квитируемым тревогам
30000-39999	Пользовательские тревожные события, относящиеся к неквитируемым тревогам

Проектировщик может выбирать произвольный номер категории тревоги в пределах указанных диапазонов. Разные номера тревог могут быть использованы для группировки тревог по определённому классу в пределах проекта, с возможностью последующей фильтрации.

Категория событий, относящихся к одной и той же тревоге, определяет поведение тревоги. Система "Соната" относит тревогу к одной из трех групп:

- обязательно квитируемая тревога;
- необязательно квитируемая тревога;
- неквитируемая тревога.

В зависимости от поведения, состояние тревоги может принимать одно из следующих значений:

- 0 - тревога отсутствует;
- 1 - тревога появилась;
- 2 - тревога квитирована; данное состояние достижимо только для обязательно и необязательно квитируемых тревог;
- 3 - тревога исчезла, но не была квитирована; данное состояние достижимо только для обязательно квитируемых тревог.

Обязательно квитируемая тревога для своего исчезновения требует снятия условия возникновения и квитирование оператором.

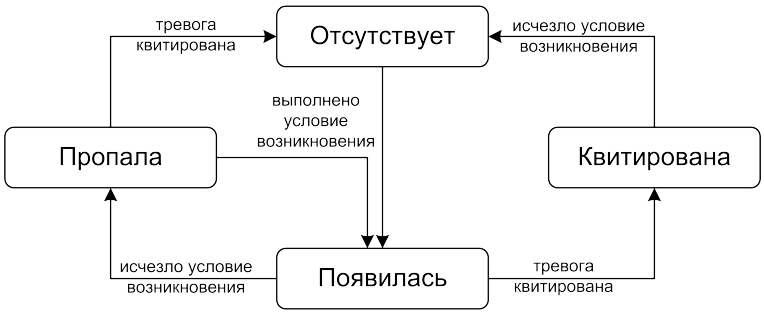


Рисунок 3.9 - Обязательно квитируемая тревога

Необязательно квитируемая тревога для своего исчезновения требует снятия условия возникновения тревоги.

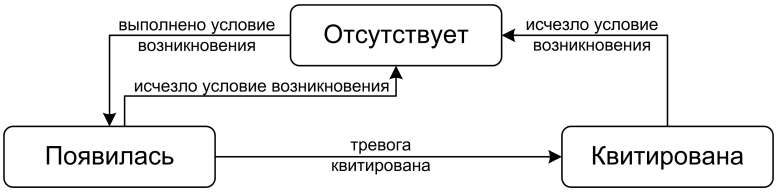


Рисунок 3.10 - Необязательно квитируемая тревога

Неквитируемая тревога для своего исчезновения требует снятия условия возникновения тревоги. Квитирование оператором никакого влияния на состояние тревоги не оказывает.

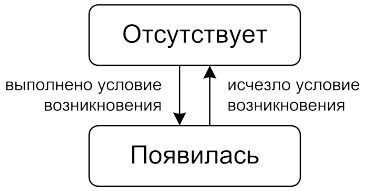


Рисунок 3.11 - Неквитируемая тревога

Каждое приложение системы "Соната" может порождать события и управлять состоянием тревог, для этого в языках программирования ST, FBD и LUA реализованы соответствующие функции для генерации события или изменения состояние тревоги.

3.4.3. Архивирование событий

Каждое приложение содержит в себе очередь, куда поступают все возникшие события. По умолчанию глубина очереди равна 2000 событий. При достижении этого количества событий старые события удаляются.

Обслуживанием всех событий проекта занимается приложение EventLogger. Оно с заданной периодичностью производит опрос очередей событий всех приложений проекта и извлекает оттуда новые данные. Сейчас скорость извлечения из очереди приложений составляет 20 событий в секунду.

Внимание! Если приложение будет генерировать более 20 событий в секунду и их совокупное количество превысит глубину очереди, то начнутся потери.

Регистратор событий - EventLogger хранит список событий в файле Data.el, используя формат данных sqlite 3.x (см. www.sqlite.org).Данный формат хранения является устойчивым к сбоям питания. Для чтения этого файла могут быть использованы любые СУБД, поддерживающие данный формат. По умолчанию в файле сохраняется 500000 событий. Старые события автоматически удаляются.

Приложения системы "Соната" могут запрашивать у EventLogger список накопленных событий. Так, приложение IECWindowEngine, осуществляющее отображение мнемосхем проекта, имеет в своём составе компонент для просмотра списков событий и тревог. Данные компоненты позволяют сохранить список на диск, а также напечатать его.

3.5. Архивирование сигналов

Архивированием текущих значений сигналов занимается приложение Archive. В текущей версии системы "Соната" архивы сигналов являются строго периодическими. Под этим подразумевается, что даже если сигнал не меняет своего значения (например, дискретный), то запись данных всё равно производится с указанным интервалом. Минимальный интервал времени сохранения составляет 20 мс. Каждый сигнал сохраняется в индивидуальном файле.

Формат архивного файла представлен на рис. 3.12. Архивный файл состоит из 5 частей. В начале файла размещены два заголовка, в конце файла размещены копии заголовков. В середине файла располагаются записи со значениями сигнала. Первый неизменный заголовок содержит информацию об архивируемом сигнале. Второй заголовок содержит текущую позицию, куда будет производиться запись.

При достижении времени очередного сохранения сигнала приложение Archive сначала производит запись значения в текущую свободную позицию, затем передвигает указатель на следующую позицию и сохраняет об этом информацию в заголовке в начале файла, затем дублирует этот заголовок в конце файла.

В случае сбоя питания теряется только последняя запись в файл. Если повреждается один из заголовков, то Archive автоматически восстанавливает файл по резервной копии заголовков.

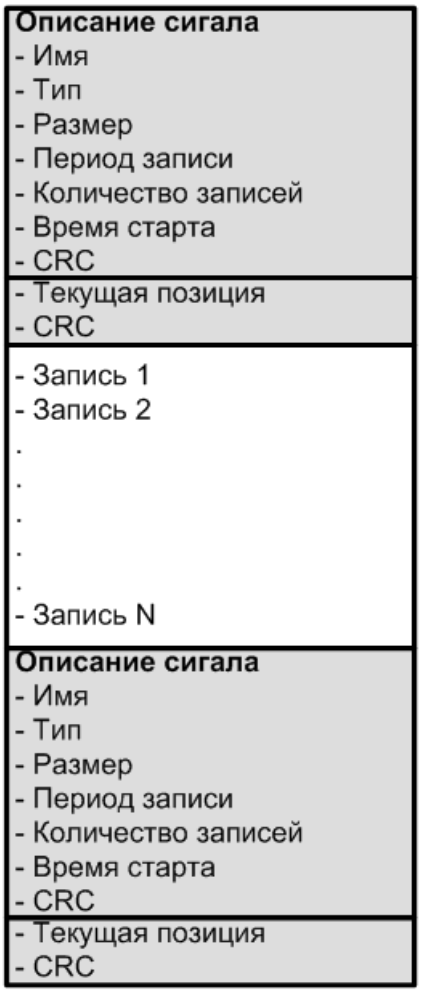


Рисунок 3.12 - Структура архивного файла

3.6. Шифрование данных в SCADA "СОНАТА"

Шифрование данных в SCADA "СОНАТА" применяется для обеспечения безопасности передачи данных проекта между узлами, которые общаются между собой по сети и в пределах одного узла.

Настройки шифрования данных описаны в документе SCADA-СИСТЕМА "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95.

Работа приложения Loader при включённом шифровании:

- По умолчанию, когда приложение Loader только стартует, шифрования нет, так как он не считал проект и не знает, есть ли пароль или нет;
- Когда проект загружается, приложение Loader получает информацию и, если включено шифрование данных, переключается на шифрование, с использованием пароля проекта;
- Если остановить работающий проект с шифрованием данных, то приложение Loader снова выключает шифрование;
- Чтобы шифрование было включено всегда, т.е. даже когда проект остановлен, нужно добавить в аргументы командной строки приложения Loader параметр -password=NNNNNNNN, где NNNNNNNN – это хеш от пароля проекта. Данный хеш отображается в диалоге свойств проекта в ProjectManager.

Шифрование данных нужно учитывать при работе следующих приложений:

- **Loader** - описано выше;
- **ControlCenter** - для работы с шифрованием данных может потребоваться ввести пароль на проект, который используется при шифровании данных;
- **Distributer** - для работы с шифрованием данных может потребоваться ввести пароль на проект, который используется при шифровании данных;
- **SignalViewer** - для работы с шифрованием данных может потребоваться ввести пароль на проект, который используется при шифровании данных;
- **ArchiveViewer** - для работы с шифрованием данных требуется ввести пароль на проект, который используется при шифровании данных.

Общие принципы работы приложений с шифрованием данных:

Когда на узле включено шифрование данных и пароль в приложениях ControlCenter, Distributor, SignalViewer, ArchiveViewer не совпадает с паролем на узле, то для этих программ узел будет выглядеть, как недоступный, так как все сетевые пакеты будут отброшены. Такое поведение узлов могло бы привести к тому, что нельзя было бы связаться с узлами, на которых установлен пароль (для шифрования данных), отличный от текущего пароля (для шифрования данных) проекта.

Чтобы выйти из этой ситуации приложения ControlCenter, Distributor, SignalViewer, ArchiveViewer работают следующим образом. При распространении проекта данные приложения применяют эвристику: сначала приложения пытаются подключиться к узлу с паролем проекта, если это не удаётся, то пробуют подключиться без пароля (с отключенным шифрованием), если это не удаётся, то запрашивают новый пароль у пользователя. В приложении ControlCenter для любого узла можно принудительно задать пароль для связи с ним, отличный от заданного в проекте пароля. Для этого нужно в дереве проекта выбрать необходимые узлы и нажать кнопку Пароль. Далее ввести новый пароль.

3.7. Диагностика системы

3.7.1. Автоматический контроль целостности файлов проекта

В Loader (загрузчике узла) реализован механизм контроля целостности файлов проекта. Для контроля целостности проекта используется специальный файл !VERSION.info, расположенный в папке проекта. В этом файле указаны контрольные суммы файлов проекта. Для самого файла также указана контрольная сумма. В качестве контрольной суммы используется алгоритм ГОСТ Р 34.11-2012 "Стрибог" длиной SHA-256 бит.

При старте узла Loader считывает файл !VERSION.info, проверяет его корректность, а затем проверяет контрольные суммы всех файлов. Если для хотя бы одного файла контрольная сумма не совпадает, то запуск узла отклоняется.

Для активации контроля целостности проекта нужно через программу ProjectManager в свойствах проекта активировать флаг "Подписать проект".

3.7.2. Автоматическая диагностика запуска и работы приложений.

При старте узла Loader осуществляет контроль за последовательностью запуска приложений. Если какое-либо приложение не стартовало или стартовало с фатальной ошибкой, то дальнейший запуск приложений останавливается.

Для контроля состояния приложений Loader использует опрос их трёх сигналов: @STATE, @MESSAGE_FRAMEWORK, @MESSAGE. При нормальной работе приложения сигнал @STATE должен иметь строковое значение "RUN", а сигналы @MESSAGE_FRAMEWORK, @MESSAGE должны иметь значение пустой строки. Если какое-либо приложение не отвечает (не отдаёт сигналы) или имеет статус, отличный от "RUN", то Loader сообщает о неисправности данного приложения.

Помимо изменения значения системных сигналов каждое приложение порождает диагностические сообщения, которые сохраняются локально на узле в индивидуальные LOG файлы, а также поступают в регистратор событий - EventLogger. В сообщениях указываются все переходы приложения из одного состояния в другое, а также факты возникновения каких-либо проблем.

Для контроля состояния узлов проекта и их приложений можно воспользоваться приложением ControlCenter (Центр управления) и EventViewer (Просмотрщиком событий) см. документ SCADA-система "Соната" Руководство пользователя КУНИ.505200.023-01.01 95.

3.8. Отображение технологической информации

Мнемосхемы пользовательского интерфейса в системе "Соната" представляют собой событийное приложение, построенное, помимо функциональных блоков, предназначенных для технологических алгоритмов, еще и с использованием специальных графических базовых и композитных графических функциональных блоков. Базовые блоки предназначены для отображения как графичесих примитов, таких как прямоугольники, эллипсы, линии, текст и т.д., так и более сложных элементов - таблиц, списков, графиков и т.п. Композитные графические блоки служат для объединения других графических блоков. Возможность вложения одних композитных блоков в другие зависит от подвида блока, используемого в качестве контейнера. Приложение, предоставляющее графический интерфейс оператору, представляет собой экземпляр приложения вида APPLICATION.IEC.WINDOW.

Мнемосхема пользовательского интерфейса представляет собой одно главное окно, в котором размещается как минимум один объект-страница главного окна, и произвольное количество дополнительных окон, предназначенных для размещения вторичных видеокадров, таких как мнемосхемы управления технологическим оборудованием, всплывающие окна графиков и т.п. Если в главном окне располагаются несколько страниц, то необходимо организовать переключение страниц на верхнем уровне иерархии событийных блоков, поскольку в один момент времени может быть видима только одна страница главного окна.

В качестве графических композитных блоков может использоваться три вида контейнеров:

- страница главного окна - контейнер, экземпляры которого должны быть объектами блока - программы, то есть находиться на верхнем уровне иерархии; блоки - страницы, расположенные на других уровнях иерархии не трактуются как страницы главного окна;
- окно - контейнер, экземпляры которого должны могут быть расположены на любом уровне иерархии, но родительским блоком должен быть какой-нибудь графический контейнер;
- графический композитный блок - контейнер, экземпляры которого должны могут быть расположены на любом уровне иерархии, но родительским блоком должен быть какой-нибудь графический контейнер.

Графический композитный блок служит для формирования сложных графических объектов, таких как графическое отображение технологического элемента, например, насоса, клапана, вентилятора и т.п. либо организации каких-либо других составных графических элементов.

Элементы, из которых строятся графические объекты, представляют собой экземпляры базовых графических типов либо графических композитных функциональных блоков. Базовые блоки можно разделить на следующие группы:

- графические примитивы, представляющие собой объекты для отрисовки прямоугольников, эллипсов, линий, текста, многоугольников, векторных (svg) и растровых (png, jpg, bmp) изображений;

- простые элементы управления, такие как кнопки, редакторы целых, вещественных, строковых значений, выпадающий список строк и т.п;

- сложные элементы управления, такие как таблицы, деревья, графики, объекты просмотра HTML-страниц и т.п.

При старте графического приложения на экране появится главное окно и окно ввода входного имени и пароля, которые должен ввести пользователь для получения доступа к функциям мнемосхемы. Перечень пользователей задается при разработке проекта.

Однако в ряде случаев ввод пользователя или пароля не требуется или авторизация вообще не требуется. Система "Соната" позволяет указать пользователя и пароль, которые будут использованы для автоматического входа при старте приложения, а также указать приложению, что вход должен производиться вообще без авторизации. Указанные возможности требуют задания дополнительных параметров командной строки графического приложения в ходе настройки узла.

Разработчик может указать следующие дополнительные параметры:

- -user=имя_пользователя. Данный параметр задает предустановленное имя пользователя. Если в командной строке параметр -password не указан, то указанное имя пользователя будет заранее проставлено в диалог авторизации;

- -password=пароль. Данный параметр используется только вместе с параметром -user и означает требование входа пользователя с указанным пользователем и паролем;

- -noauth. Данный параметр требует входа в систему без авторизации.

3.9. Автоматическое управление

Для реализации алгоритмов автоматического управления в проект автоматизации включаются событийные и циклические технологические программы, разработанные на языках программирования ST (стандарт IEC 61131), FBD (стандарт 61499) и LUA 5.3. Для подробного изучения языков программирования обратитесь к руководству разработчика. Разработчик выбирает подходящий ему язык программирования и создаёт в проекте тип приложения (шаблон), в котором реализует необходимый алгоритм автоматизации. Затем на одном или нескольких узлах размещает экземпляры приложений разработанного типа.

3.9.1. Циклические программы ST (IEC-61131)

Циклическая программа представляет собой перечень инструкций языка ST, выполняемый сверху вниз с заданной периодичностью.

Для включения подобной программы в проект необходимо создать тип приложения, относящегося к виду APPLICATION.ST.CONSOLE, создать необходимые функции и функциональные блоки, называемые элементами программы, после чего разработать собственно программу.

Функцией называется элемент программы, содержащий некий алгоритм, но не сохраняющий свое состояние от вызова к вызову. Типичным примером может служить функция вычисления

синуса, поскольку результат ее работы зависит исключительно от значения входного аргумента, но не зависит от результатов предыдущих вызовов.

Функциональным блоком называется элемент программы, содержащий некий алгоритм и сохраняющий свое состояние от вызова к вызову. Примером может служить таймерный блок TP, значение выходной переменной которого зависит не только от значения входов блока, но и от того, сколько времени прошло от момента появления высокого уровня логического сигнала на входе блока.

При разработке элемента программы необходимо описать его интерфейс. В теле элемента программы могут использоваться входные, выходные, внутренние, а также глобальные переменные. Глобальными являются переменные, входящие в интерфейс программы, то есть переменные типа приложения.

Периодичность выполнения программы задается для типа создаваемого приложения. Перед началом каждого цикла модуль выполнения циклических ST-программ засекает текущее время, выполняет все инструкции, после чего еще раз засекает текущее время и смотрит, сколько времени реально потрачено на выполнение алгоритма. Если реальное время выполнения цикла меньше заданного, то программа приостанавливается на неиспользованный остаток времени цикла.

3.9.1.1. Замена логики циклической программы без остановки приложения

Для циклических приложений реализована функция замены алгоритмов без остановки приложения и сброса значений внутренних переменных. Ограничением работы данной функции является то, что для замены логики необходимо, чтобы набор внешних переменных приложения остался неизменным. Если же предполагается, что замена логики вовлечет в работу алгоритма незадействованные ранее внешние (глобальные) переменные, то такие переменные должны быть добавлены в список внешних переменных приложений заранее (зарезервированы).

Для замены логики без перезапуска приложения необходимо выполнить распространение проекта, после чего при помощи модуля просмотра сигналов (SignalViewer) установить значение системной переменной "@COMMAND" приложения в значение "reload_logic". Приложение проверит новую логику, и, если замена возможна, завершит вычислительный цикл, произведет замену и начнет новый цикл уже с новой логикой.

3.9.2. Событийные программы FBD (IEC-61499)

Событийная программа представляет собой множество иерархически организованных событийных функциональных блоков, связанных между собой. Вычисление каждого блока происходит при изменении входных переменных каждого блока либо при возникновении событий на его входе. За счет использования событийного механизма пересчета достигается снижение нагрузки на процессор компьютера или контроллера.

Функциональные блоки разделяются на базовые и композитные. Базовые блоки определяют логику вычислительного алгоритма. Композитный блок представляет собой набор связанных между собой экземпляров других функциональных блоков, как базовых, так и композитных. Событийная программа является композитным функциональным блоком.

Для включения событийной консольной программы в проект необходимо создать тип приложения, относящегося к виду APPLICATION.IEC.CONSOLE, создать необходимые типы функциональных блоков, после чего разработать тип функционального блока верхнего уровня - тип приложения, экземпляры которого будут в будущем размещены на узлах проекта.

Создание любого функционального блока начинается с его интерфейса. Разработчик описывает перечень событий и переменных блока, после чего переходит к описанию внутренней структуры блока.

Описывая внутреннюю структуру композитного блока, разработчик формирует перечень экземпляров блоков, после чего связывает их между собой, формируя тем самым FDB-диаграмму композитного блока. Внешний вид экземпляра блока на диаграмме приведен на рис. 3.13.

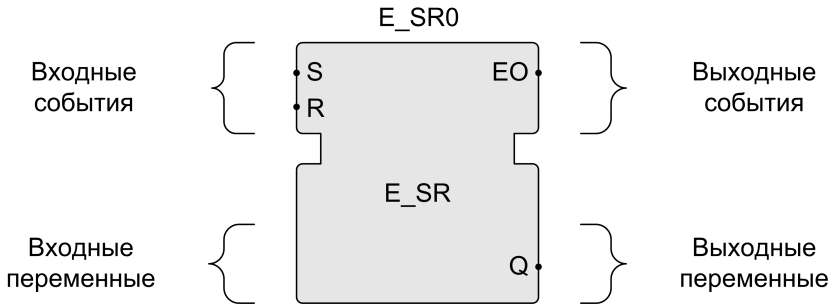


Рисунок 3.13 - Событийный функциональный блок

Базовый функциональный блок, разработанный пользователем, представляет собой алгоритм, описанный при помощи диаграммы управления выполнением (Execution Control Chart, ECC). ECC представляет собой разновидность диаграммы состояний, на которой размещены состояния блока и переходы между ними, каждому из которых сопоставляется условие. С некоторыми состояниями сопоставляется упорядоченный набор действий, каждое из которых представляет собой пару алгоритм-событие. Пример ECC для базового блока, реализующего функциональность событийного SR-триггера, приведен на рис. 3.14.

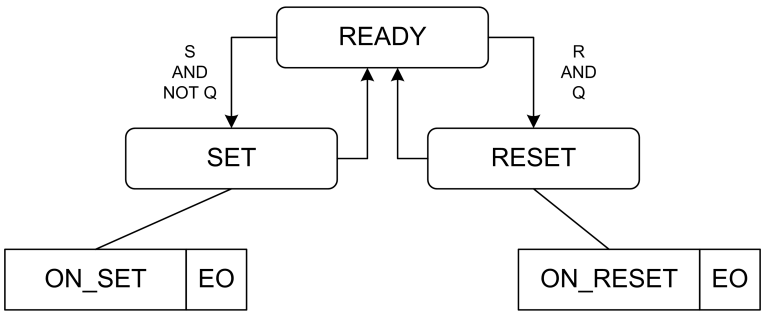


Рисунок 3.14 - Диаграмма управления выполнением событийного SR-триггера

Рассмотрим работу базового функционального блока, используя в качестве примера блок, приведенный на рис. 3.14. При инициализации приложения все базовые функциональные блоки переходят в начальное состояние (в нашем примере роль начального состояния отводится состоянию IDLE). Затем, при возникновении событий либо при изменении входных переменных запускается процесс пересчета каждого блока. Программа пересчета перебирает все переходы из текущего состояния. Для каждого из переходов вычисляется условие и, если результат вычисления истинный, блок переходит в новое состояние. После перехода программа вычисления смотрит, есть ли у нового состояния действия и, если есть, то выполняет их. Действия выполняются сверху вниз, первым выполняется алгоритм (некая подпрограмма), затем инициируется выходное событие. Затем вновь перебираются переходы и т.д. Данный процесс заканчивается, если из текущего состояния нет переходов, либо блок перешел в состояние, в котором он в данном цикле вычисления уже побывал.

Предположим, что функциональный блок E_SR находится в начальном состоянии, значение выходной переменной Q установлено в FALSE. При возникновении события S блок переходит в состояние SET, выполняет алгоритм ON_SET (Q := TRUE), после чего инициирует событие EO. Если событие S придет еще раз, то функциональный блок из состояния IDLE не выйдет, поскольку возможных переходов с событием S и переменной Q, установленной в TRUE, не существует.

Аналогичными будут действия при приходе события R, когда будет выполнен алгоритм RESET (Q := FALSE).

3.9.3. Смешанные программы LUA 5.3

Язык Lua позволяет разработчику создавать произвольные программы, в которых можно смешивать как циклическое исполнение алгоритма, так и реализовывать обработчики событий изменения сигналов и таймеров, которые прерывают последовательное исполнение кода. После завершения обработки события исполнение программы продолжается с прерванного места.

3.10. Резервирование

3.10.1. Автоматическое резервирование сети

В системе "Соната" заложена возможность многократного резервирования сети. Для её активации достаточно в настройке каждого из узлов проекта указать по несколько IP адресов, принадлежащих разным сетевым адаптерам компьютера или контроллера (см. SCADA-Система "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01.95).

ВАЖНО!!! При указании IP адресов на узлах соблюдать порядок перечисления адресов, относящихся к разным подсетям.

ВАЖНО!!! Система "Соната" использует полное зеркалирование траффика, то есть информация синхронно отправляется по всем линиям связи. Увеличение количества резервных линий связи кратно увеличивает загрузку процессора.

3.10.2. Автоматическое резервирование архивов сигналов и архивов тревог

В системе "Соната" реализована возможность организации многократно резервированных архивов сигналов и архивов тревог. Для реализации резервирования достаточно создать необходимое количество экземпляров архивов на разных узлах проекта. Архивы автоматически найдут друг друга в сети. При отключении архива и последующем включении он запросит пропущенные данные у других архивов. Прочие приложения будут автоматически выбирать доступный архив и запрашивать данные оттуда.

3.10.3. Автоматическое резервирование узлов

В системе "Соната" при использовании контроллеров "Сонет-Мастер" и аппаратного Блока-Переключения-на-Резерв (БПР) возможно создавать автоматически резервируемые узлы проекта.

При создании резервируемого узла два контроллера "Сонет-Мастер" подключаются через специальный разъём (РЕЗЕРВ) к БПР. В конфигурации узла одной из исполняемых программ должен быть драйвер Sonet.Failover. БПР контролирует наличие меандра, формируемого драйвером,

на входах от обоих контроллеров. При старте контроллеров БПР выбирает тот контроллер активным, от которого первым начал поступать сигнал. При пропадании меандра от активного контроллера и наличии меандра от резервного контроллера БПР производит переключение на резерв. При этом помимо подачи команды активации на резервный контроллер БПР осуществляет снятие питающего напряжения с выходных модулей бывшего активного контроллера и подачу питающего напряжения на выходные модули активируемого резервного контроллера. В текущей версии системы "Соната" время переключения на резерв составляет не более 100 мс.

При работе в режиме резервирования все драйвера и алгоритмы исполняются одновременно на обоих контроллерах. Постоянно происходит синхронизация значений сигналов, переменных и внутреннего состояния алгоритмов от активного контроллера к резервному.

ВАЖНО!!! Существуют особенности и ограничения на возможности синхронизации при резервировании:

- сигналы всех приложений (драйверов, алгоритмов, мнемосхем и т.п.) синхронизируются полностью и без ограничений;
- внутренние переменные и состояние циклических программ на языке ST синхронизируются полностью при завершении цикла;
- внутренние переменные и состояние событийных программ на языке FBD синхронизируются полностью при выполнении каждого блока;
- внутренние переменные и состояние программ на языке Lua не синхронизируются;
- открытые файлы, сетевые сокеты, порты в любых приложениях не синхронизируются.

3.10.3.1. Типовые схемы резервирования контроллеров CONET с применением БПР (Блока переключения на резерв)

3.10.4. Резервирование на проектном уровне

Разработчик может самостоятельно реализовать на проектном уровне резервирование отдельных приложений или узлов. Любое приложение системы "Соната" имеет два локальных системных сигнала: @RESERVE и @RESERVED. Значение сигнала @RESERVE, равное true, служит для указания приложению, что оно должно находиться в резерве; значение false указывает, что приложение должно быть активным. Сигнал @RESERVED служит для подтверждения того, что приложение перешло или вышло из резерва.

Таким образом, если разработчику нужно переключать состояние двух приложений, одно переводя в состояние активности, а другое в состояние резерва, то ему нужно подключить к двум сигналам управляющего алгоритма два сигнала @RESERVE от управляемых приложений. Подавая на два сигнала управляющего алгоритма противоположные значения (true, false) разработчик обеспечивает переключение активности управляемых приложений.

3.11. Синхронизация времени

В Сонате есть два режима синхронизации времени: ударный и безударный. При ударном режиме синхронизации полученное от источника точное время сразу же устанавливается на узле. При безударном режиме синхронизации текущее время плавно подтягивается к точному времени.

При остановленом проекте и при старте проекта происходит ударная синхронизация времени. При работающем проекте происходит безударная синхронизация времени (за 1 минуту работы подтягиваются примерно 2 секунды).

Все узлы запрашивают точное время друг у друга согласно приоритету. Приоритет настраивается в свойствах узла в Менеджере проектов (приложение ProjectManager). Чем меньше число, тем выше приоритет узла, как источника точного времени. Узлы запрашивают время друг у друга раз в 4 секунды - это минимально допустимое время согласно стандарту NTP. Точность удержания синхронизации времени между узлами не хуже 10 мс.

В настройках узла есть специальная опция - Ждать узел с высоким приоритетом времени (см. SCADA-система "Соната" Руководство пользователя КУНИ.505200.023.01.01 95), если выполнить данную настройку, то узел не будет запускаться пока не получит время от узла с более высоким приоритетом.

Для диагностики, почему синхронизация времени работает неправильно есть диагностический режим. Для его включения нужно добавить Loader аргумет -sntp_log, тогда он будет писать в свой лог результаты синхронизации.

ВНИМАНИЕ! В SCADA-системе "Соната" есть возможность запускать несколько узлов проекта на одном физическом узле. Данный функционал описан в документе SCADA-система "Соната" Руководство пользователя КУНИ.505200.023.01.01 95 п. Настройка нескольких узлов проекта на одном АРМ. Данный функционал требует запустить на одном физическом узле несколько приложений Loader на разных портах. Для корректной работы синхронизации времени на данных узлах, необходимо чтобы только один из узлов мог синхронизировать время, а для остальных узлов нужно отключить возможность синхронизации времени. Для этого необходимо у узлов, которым не нужно синхронизировать время, при старте их приложений Loader указать параметр **-sntp_client_disable**.

3.12. Многооконный режим

Мнемосхема пользовательского интерфейса (приложение APPLICATION.IEC.WINDOW) может содержать в себе от одного до 32 отдельных окон, каждое из которых может быть размещено на отдельном мониторе. Приложение-мнемосхема может быть настроено как для использования строго одного окна, так и для работы в многооконном режиме. Последний способ, впрочем, не исключает возможности работы также в режиме одного окна.

Для работы в первом режиме, когда мнемосхема содержит строго одно окно, список окон приложения не формируется. При этом интерфейс главного окна определяется набором страниц – экземпляров объектов типа «TPage - Тип страница главного окна».

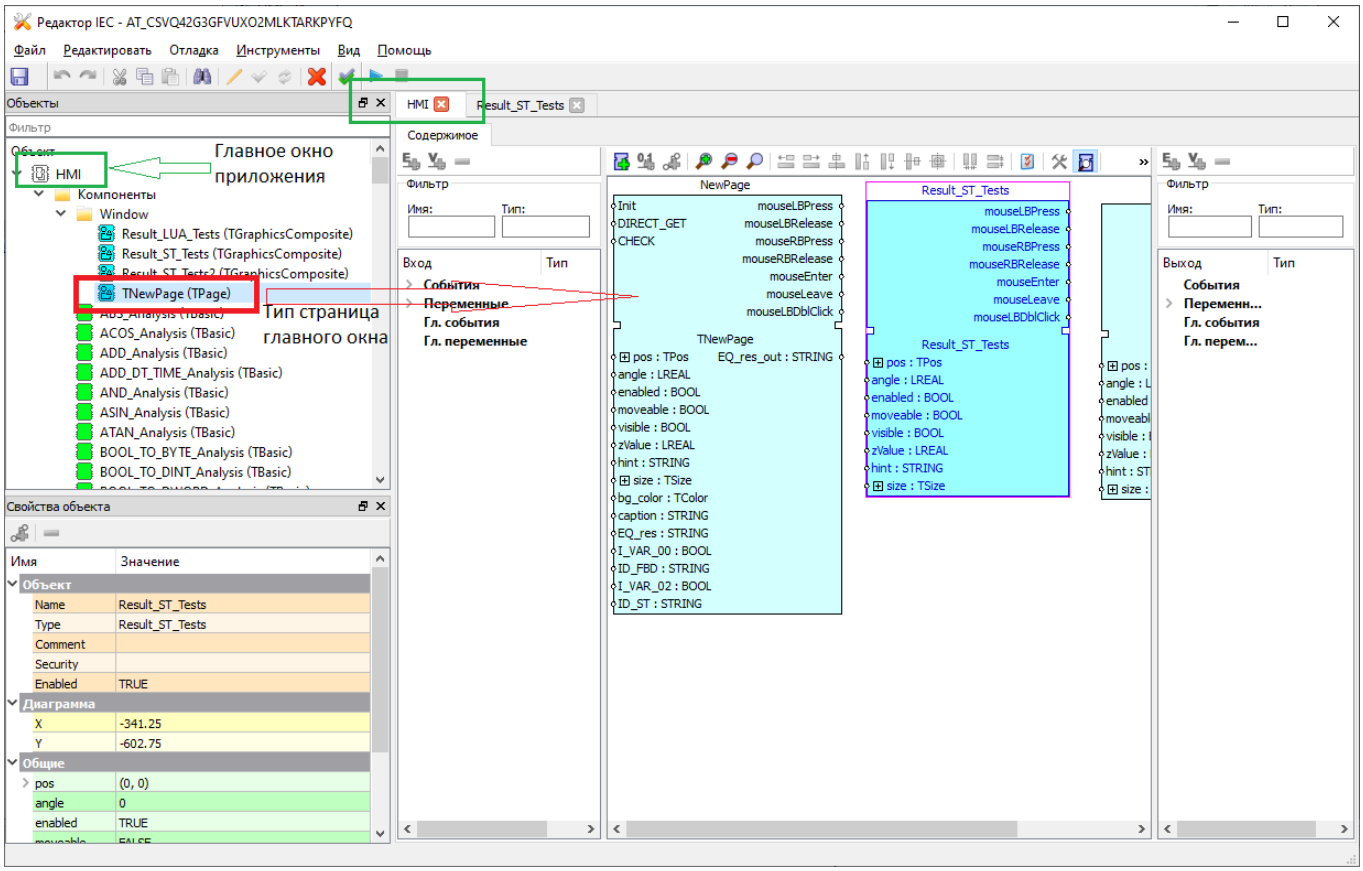


Рисунок 3.15 - Вариант настройки в режиме одного окна

В один момент времени в окне приложения видна только одна страница. Помимо главных страниц, приложение может содержать произвольное количество дополнительных окон, предназначенных для размещения вторичных видеокадров, таких как мнемосхемы управления технологическим оборудованием, всплывающие окна графиков и т.п. Переключение страниц главного окна, если страниц несколько, должно быть организовано разработчиком мнемосхемы (см. рис. 3.16).

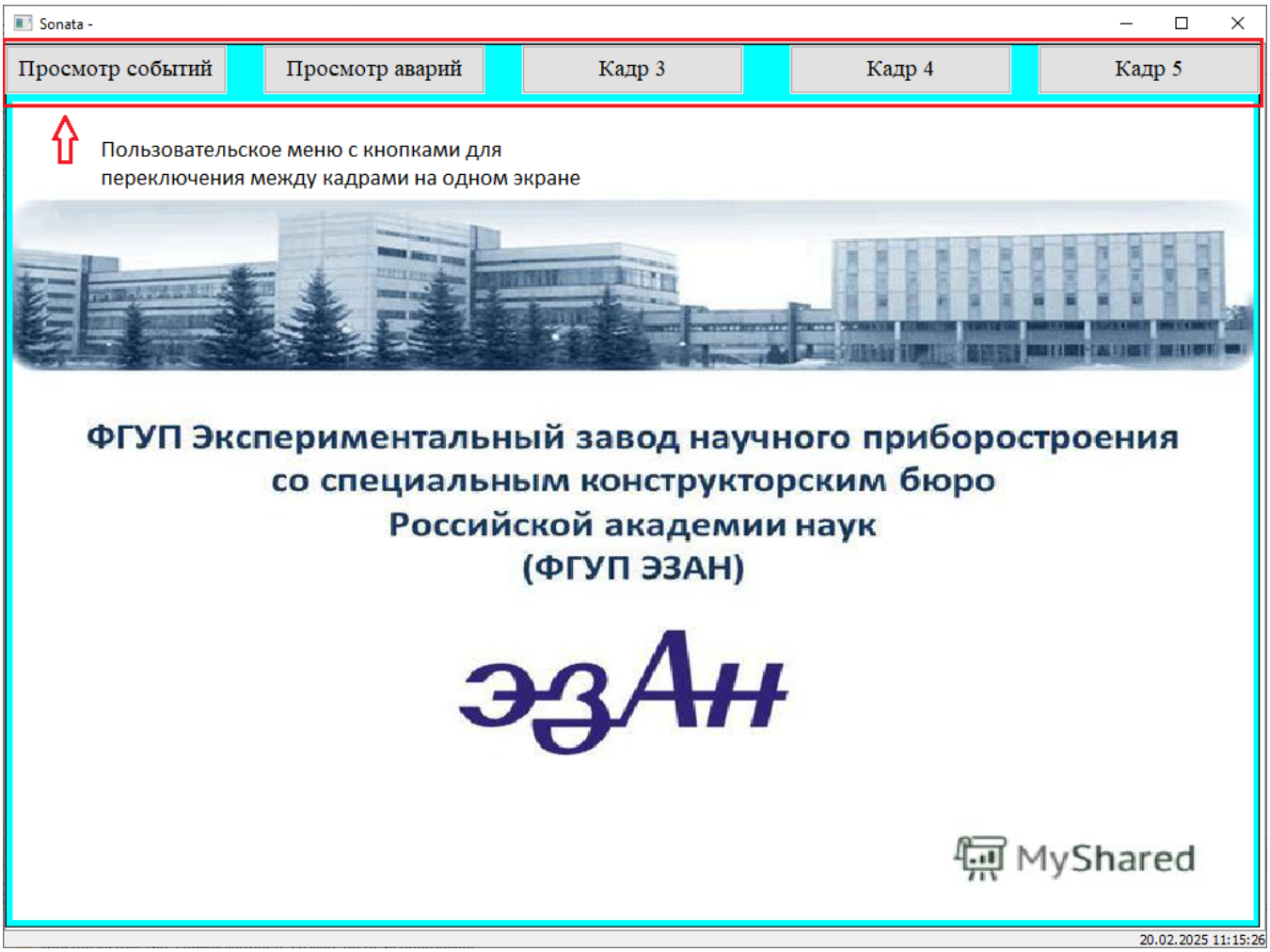


Рисунок 3.16 - Основное окно и меню с кнопками для переключения между кадрами

Для работы во втором, многооконном, режиме разработчик формирует перечень окон и задает структуру каждого окна. Окна имеют иерархическую структуру – для каждого окна создается корневой кадр, который может содержать в себе объекты. Объект может быть одиночным, то есть представляющим собой экземпляр графического композитного функционального блока, так и стеком (стопкой) других кадров. Тем самым формируется иерархическая структура графического интерфейса. Каждому объекту, входящему в иерархию, может быть задано имя. Для переключения кадров на какой-либо объект необходимо использовать объект-навигатор, которому задается имя объекта, который необходимо показать на экране. Переключение кадров в этом случае будет осуществлено самостоятельно приложением-мнемосхемой без каких-либо дополнительных настроек. Пример создания многооконного проекта приведён в документе SCADA-СИСТЕМА "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95 ??? (см. рис. 3.17).

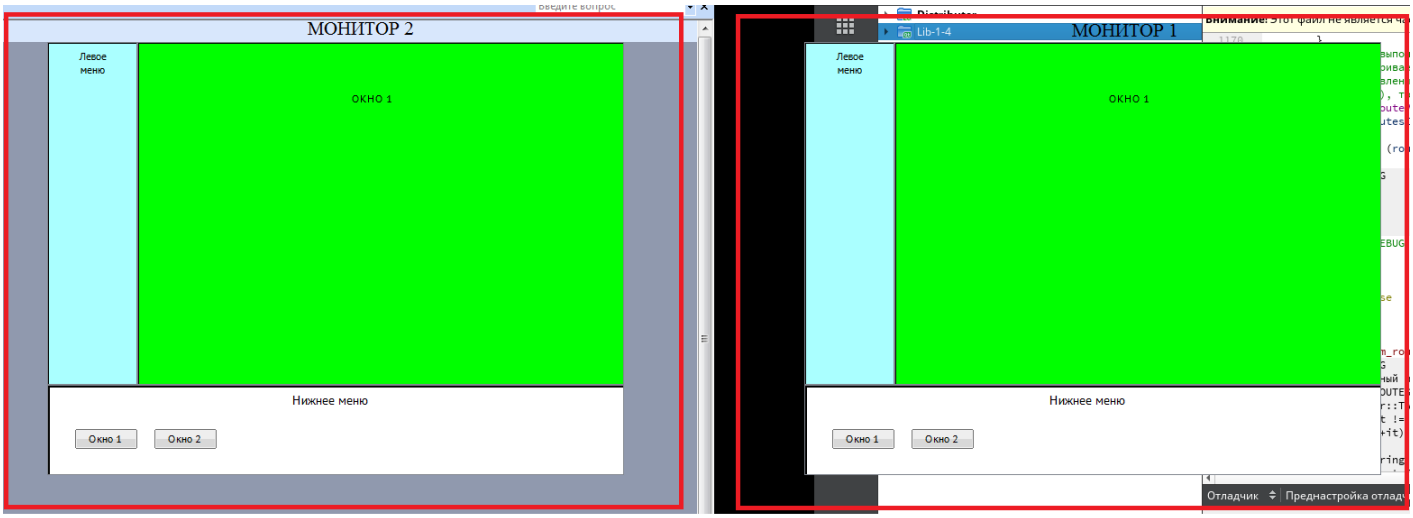


Рисунок 3.17 - Пример многооконного проекта с двумя мониторами

3.13. Межпроектное взаимодействие

Для организации обмена данными со сторонними проектами в системе "Соната" предусмотрены следующие механизмы:

- драйвер MODBUS обеспечивает работу в режиме Master/Slave по протоколам RTU или TCP;
- драйвер OPCUA обеспечивает работу в режиме клиента или сервера;
- драйвер SNMP обеспечивает работу в режиме клиента;
- WEBсервер обеспечивает отдачу статического контента, а также чтение и запись сигналов проекта через REST запросы;
- драйвер проприетарного протокола СВБУ "Портал".

Для получения подробной информации обратитесь к руководству оператора.

3.14. Руководство по созданию проектов

В ходе проектирования разработчик проделывает следующие операции:

- создает глобальную таблицу сигналов;
- создает типы приложений: драйверов, алгоритмов, мнемосхем;
- привязывает сигналы к физическим каналам ввода-вывода;
- создает правила преобразования сигналов;
- создает узлы и размещает на них экземпляры приложений созданных типов приложений;
- связывает при необходимости локальные сигналы приложений;
- распространяет проект на узлы системы;
- запускает и отлаживает проект.

ВНИМАНИЕ! При изменении в проекте: структурных типов, интерфейсов приложений, порядка приложений на узлах, перечня узлов и перечня глобальных сигналов требуется перезапуск всего проекта. Недопустимо перезапускать только часть узлов. Часть узлов допустимо

перезапускать при работающих остальных узлах, если были изменены только: тексты программ, конфигурации драйверов, конфигурация архива, конфигурация мнемосхемы.

4. МЕТРОЛОГИЧЕСКИ ЗНАЧИМАЯ ЧАСТЬ

В данном разделе описаны вопросы функционирования программ, выполняющих функции сбора, передачи, обработки, хранения и представления измерительной информации. Также рассматриваются вопросы защиты программного обеспечения и обрабатываемой, в том числе измерительной, информации от непреднамеренных и преднамеренных изменений.

4.1. Описание структуры ПО и выполняемых функций

Описание структуры и выполняемых функций SCADA-системы "Соната" приведено выше (см. раздел 1.1).

4.2. Последовательность обработки данных

На рис. 4.1 представлена схема последовательности обработки данных.

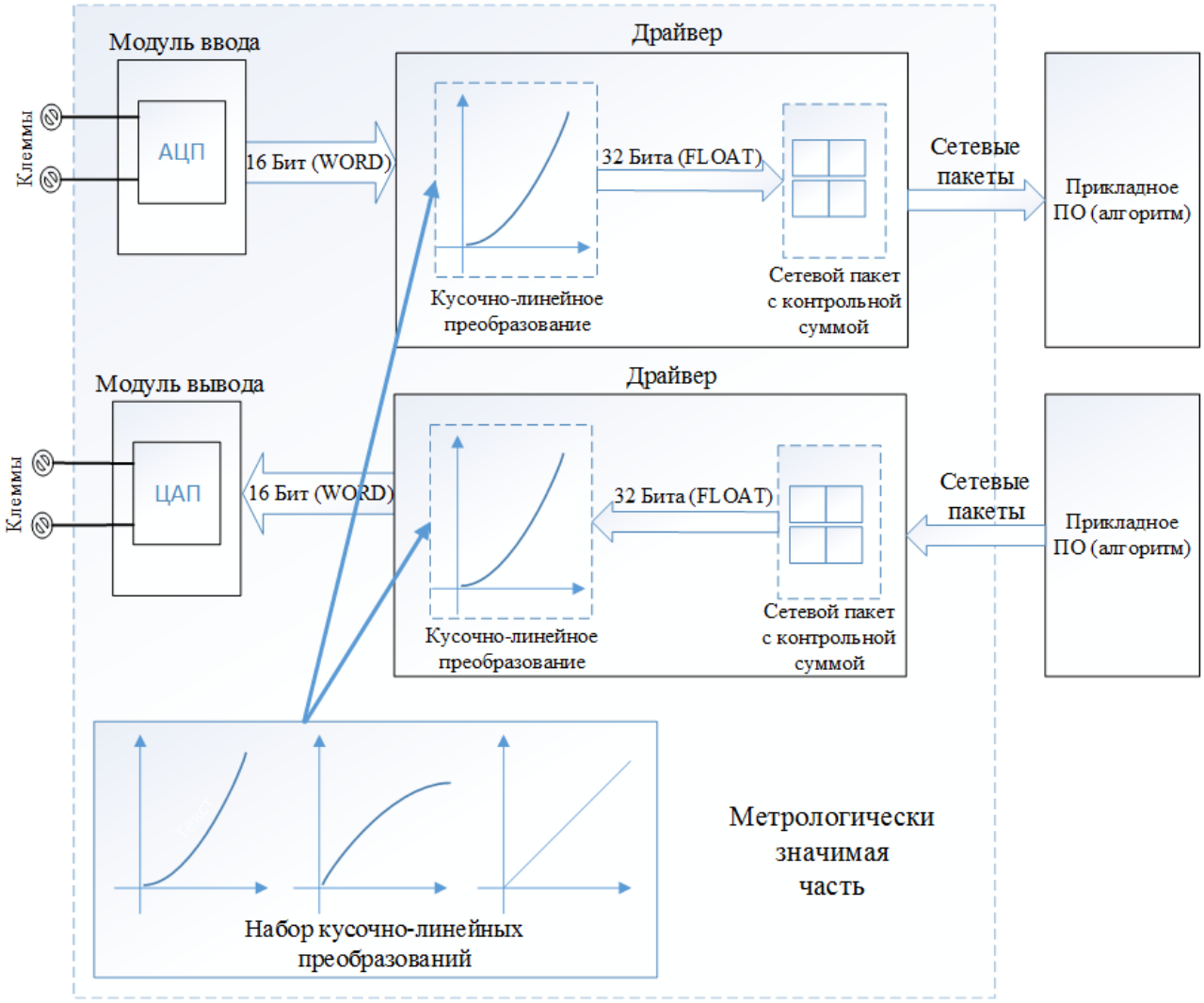


Рисунок 4.1 - Последовательность обработки данных

Модули ввода/вывода считаются поверенными за счет предварительной калибровки. Обработку данных при приеме/передаче берут на себя драйвера. Достоверность передачи данных от драйвера до прикладного ПО и в обратном направлении обеспечивается контрольными суммами передаваемых пакетов. Преобразование входных данных драйвера в выходные обеспечивается кусочно-линейными преобразованиями, которые настраивает проектировщик на этапе разработки проекта (см. раздел 3.3 и SCADA-система "Соната" Руководство пользователя КУНИ.505200.023-01.01 раздел 3.2.2.6 Редактирование преобразователей значения сигнала).

Так же существует набор предопределённых кусочно-линейных преобразователей сигналов, которые составлены по ГОСТ 6651-2009 для термопреобразователей сопротивления и по ГОСТ 8.585-2001 для термопар.

4.3. Описание функций и параметров ПО, подлежащих метрологическому контролю

Метрологическому контролю подлежат следующие функции и параметры ПО, реализованные в SCADA-системе "Соната":

- получение 16 битного цифрового сигнала от АЦП;
- кусочно-линейное преобразование в драйвере;
- преобразование в вещественное число 32 Бита (IEEE 754 Float) из 16 Бит (Word);
- упаковка в сетевой пакет с контрольной суммой;
- распаковка сетевого пакета и проверка контрольной суммы;
- преобразование из 32 Бита (Float) по кусочно-линейному закону;
- преобразование вещественного числа 32 Бита (Float) в 16 Бит (Word);
- выдача цифрового сигнала 16 Бит на ЦАП;
- преобразование цифрового сигнала в аналоговый.

4.4. Описание реализованных в ПО расчётных алгоритмов

4.4.1. Алгоритм работы блока (функции) кусочно-линейного преобразования

Для вычисления кусочно-линейного преобразования используются числа в двойной точности IEEE 754 double.

Кусочно-линейное преобразование y=f(x) задаётся в виде массива точек P={p(x, y)}. Перед началом работы программа проверяет, чтобы точки в массиве блока кусочно-линейного преобразования образовывали монотонно возрастающую или монотонно убывающую функцию.

Блок кусочно-линейного преобразования может осуществлять как прямое x->y, так и обратное преобразование y->x. Алгоритм работы прямого преобразования совпадает с алгоритмом обратного преобразования за исключением того, что оси x и y меняются местами.

Рассмотрим алгоритм.

Пусть P – массив точек { p[1]…p[N] }, x – входная величина. Требуется найти y.

Перед основным циклом алгоритма проверяем, не выходит ли входная величина x за граничные значения p[1].x и p[N].x.

```
Если x≤ p[1].x то y = p[1].y. Вычисление завершено.
Если x>= p[N].x то y = p[N].y. Вычисление завершено.
```

После проверки выхода из диапазона допустимых значений аргумента x производим поиск пары точек p[i-1] и p[i] таких что p[i-1].x≤ x≤ p[i].x.

```
i=2
Цикл пока i≤N
Если x≤p[i].x прервать цикл
Конец цикла
```

Пара точек p[i-1] и p[i] образуют концы отрезка, используемого для линейного преобразования x->y.

```
y = ( p[i].y - p[i-1].y )*(x- p[i-1].x )/(p[i].x- p[i-1].x )+ p[i-1].y
```

4.5. Описание модулей ПО

Модули ПО SCADA-системы "Соната" описаны в документе SCADA-система "Соната" Руководство системного программиста КУНИ.505200.023-01.01 32.

4.6. Перечень и описание интерфейсов

Метрологически значимое ПО систем измерения (далее в тексте СИ) разработано таким образом, чтобы его невозможно было подвергнуть искажающему воздействию через пользовательские и другие интерфейсы.

4.6.1. Влияние через интерфейс пользователя

Интерфейс пользователя SCADA-системы "Соната" не содержит команд оказывающих влияние на метрологически значимое ПО и данные при штатной эксплуатации комплекса (в режиме реального времени).

Существуют команды пользователя, которые могут оказать влияние на метрологически значимое ПО и данные, доступные только в режиме отладочных работ и при подготовке программного проекта системы управления. Данное ПО доступно лишь сертифицированному для этих работ персоналу.

Полный список таких команд интерфейса пользователя используемых при подготовке проекта и наладочных работах и их описание приведен в документах:

- SCADA-система "Соната" Руководство пользователя КУНИ.505200.023-01.01 95;
- SCADA-система "Соната" Руководство программиста КУНИ.505200.023-01.01 33.

На метрологически значимые данные, так же, можно влиять из программ созданных пользователем с помощью языков программирования, описанных в документе SCADA-система "Соната" Руководство программиста КУНИ.505200.023-01.01 33. Однако ответственность за

это лежит на проектировщиках, создающих программы. Влияние из данных программ на метрологически значимые данные относится к ошибкам проектирования.

4.6.2. Влияние через интерфейс связи

В программном обеспечении, реализующем интерфейс связи, нет команд, способных влиять на метрологически значимые данные. Контроль целостности информационных пакетов производится путем подсчета контрольных сумм пакетов.

Команды и данные, получаемые через интерфейс связи СИ, не могут оказывать недопустимого влияния на метрологически значимые данные, поскольку проверяются на целостность и принадлежность текущему проекту. Обновление ПО не может быть осуществлено через интерфейс связи СИ.

Программой, отвечающей за передачу информации по ЛВС, является микроядро, встроенное в каждое приложение. Описание микроядра приведено ранее в данном документе (см. раздел 3.1.4).

4.7. Список, значение и действие всех команд, получаемых от клавиатуры, мыши и других устройств ввода

В SCADA-системе "Соната" реализован контроль доступа пользователей, работающих с готовым проектом АСУ ТП. За первоначальную настройку данного процесса отвечает проектировщик в ходе разработки проекта. Стандартные устройства ввода, такие как клавиатура и мышь, используются только для навигации и ввода текста. Другие устройства ввода не поддерживаются.

Процесс настройки контроля доступа описан в документе SCADA-система "Соната" Руководство пользователя КУНИ.505200.023-01.01 95.

4.8. Получение идентификационных признаков SCADA-системы "Соната" и проверка метрологических библиотек

Для получения идентификационных признаков программного обеспечения служит утилита SonataVer. Данная утилита представляет собой консольную программу. Формат вызова программы должен быть следующим:

SonataVer [файл | **-f** путь_к _папке] [**-cp866**|**-cp1251**|**-cpUTF8**] [**-out** файл_вывода]

Ключи **–cp866**, **-cp1251**, **-cpUTF8** служат для указания кодировки вывода программы (кодировка 866, Windows 1251 и UTF8 соответственно). Ключ не является обязательным. По умолчанию вывод осуществляется в кодировке 866.

Ключ **–out** служит для перенаправления вывода из консоли в файл, путь к которому передается следующим параметром. Если данный ключ не указан, то вывод осуществляется в консоль.

Если программа запускается без указания проверяемого файла, либо каталога с файлами, то она выводит идентификационные признаки системы «Соната» (Наименование: SCADA-система «Соната» (КУНИ.505200.023), Версия 1.4, контрольная сумма метрологической библиотеки 0x43569245).

Если программа запускается для одиночного файла, либо для каталога, то она проверяет идентификационную информацию модуля, выводя наименование ПО, версию пакета (системы), контрольную сумму метрологической библиотеки (либо информацию о том, что данный модуль не использует метрологическую библиотеку), а также версию модуля программы.

Соответственно, по результатам, полученным в ходе выполнения утилиты **SonataVer**, можно судить о текущей версии SCADA-системы "Соната" и о неизменности кода метрологических библиотек, что будет видно из контрольной суммы. Если контрольная сумма метрологических библиотек не равна 0x43569245, то это говорит о нарушении целостности программного обеспечения. Данное ПО не следует использовать, а необходимо связаться со службой технической поддержки для получения корректного ПО.

4.9. Защита программного обеспечения и данных

ПО СИ содержит средства обнаружения, отображения и устранения сбоев (функциональных дефектов) и искажений, которые нарушают целостность ПО и данных.

4.9.1. Защита от случайных или непреднамеренных изменений

1) Метрологически значимое ПО СИ и данные защищены от случайных или непреднамеренных изменений.

В SCADA-системе "Соната" предусмотрена защита от запуска поврежденного ПО, например, вследствие частичного повреждения файловой системы накопителя содержащего ПО СИ. Система устойчива к нарушению сетевого обмена, каждый информационный пакет содержит контрольную сумму, которая проверяется получателем. Комплекс устойчив к искажению информации полученной с удаленных контроллеров ввода-вывода посредством полевой шины MODBUS, так как каждый пакет так же содержит контрольную сумму.

ПО СИ содержит средства защиты обрабатываемой информации и данных от изменения или удаления в случае возникновения непредсказуемых физических воздействий.

2) В SCADA-системе "Соната" предусмотрена защита информации при передаче ее по сети, путем контроля канала передачи на разрыв и контроля посредством контрольных сумм. Предусмотрено несколько попыток передачи информации перед появлением сообщения о разрыве канала связи. Так же предусмотрено восстановление архивной информации вследствие внезапного отключения питания с потерей тех данных, которые не были сброшены на долгосрочный накопитель из памяти. Также имеется защита от несанкционированного обмена информацией между разными проектами, работающими в одной ЛВС.

3) В SCADA-системе "Соната" предусмотрен запрос подтверждения всех действий пользователя в случае, если действия пользователя могут привести к удалению информации или не сохранению вновь введенной информации.

4) Меры, принимаемые для защиты метрологически значимого ПО и измеренных данных от случайных или непреднамеренных изменений заключаются в следующих действиях. Если обнаружено несоответствие измеренных данных реальным значениям, следует проверить и исправить правильность настройки диапазонов изменения сигнала в проекте (см. документ SCADA-система "Соната" Руководство пользователя КУНИ.505200.023-01.01 95). Если возникло подозрение об изменении ПО, следует запустить проверку ПО с помощью программы программы SonataVer, как описано в ранее в данном документе (см. раздел 4.8), и если программа обнаружит нарушения в ПО, его следует восстановить из архива.

4.9.2. Защита от преднамеренных изменений программного обеспечения

1) Для предотвращения удаления и замены запоминающего устройства другим, содержащим фальсифицированное ПО или данные, конструкцией шкафов для размещения программируемых контроллеров и столов для размещения системных блоков АРМ, предусмотрена защита от несанкционированного вторжения.

2) Защита метрологически значимой части ПО СИ от несанкционированной модификации осуществляется путем расчета контрольной суммы при компиляции модулей ПО из исходников. Полученная контрольная сумма должна совпадать с эталонной, записанной при сертификации ПО.

4.10. Описание интерфейсов пользователя, всех меню и диалогов

Описание интерфейсов пользователя, всех меню и диалогов SCADA-системы "Соната", а так же характеристики требуемых системных и аппаратных средств, необходимых для работы SCADA-системы "Соната", описаны в документе SCADA-система "Соната" Руководство пользователя КУНИ.505200.023-01.01 95.

4.11. Описание хранимых или передаваемых наборов данных

Описание хранимых или передаваемых наборов данных приведено ранее в данном документе (см. раздел 3.1.3).

5. ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ В SCADA СИСТЕМЕ "СОНАТА"

Задача по обеспечению защиты информации является одной из важных составляющих при создании и эксплуатации систем управления. Функции информационной безопасности в SCADA системе «Соната» обеспечивают доступность, целостность и конфиденциальность обрабатываемой информации и не оказывают отрицательного влияния на штатный режим функционирования автоматизированной системы управления.

К основным функциям обеспечения информационной безопасности SCADA системы «Соната» относятся:

1) Функция создания, редактирования и контроля учетных записей пользователей;
2) Функция шифрования передаваемых данных между узлами;
3) Функция контроля целостности среды исполнения SCADA системы «Соната» и прикладного программного обеспечения;
4) Функция разграничение передачи информации между узлами системы по IP адресам и маскам подсети;
5) Функция ведения журнала событий информационной безопасности;
6) Функция контроля версий прикладного программного обеспечения.

5.1. Функция создания, редактирования и контроля учетных записей пользователей

Данная функция доступна к использованию:

- при первоначальном создании проекта или редактировании существующего проекта с использованием программного обеспечения «Менеджер проектов» (см. рис. 5.1 и описание в документе SCADA система "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95 п. Выполнение комплекса программ → п. Описание программы ProjectManager (Менеджер проекта));

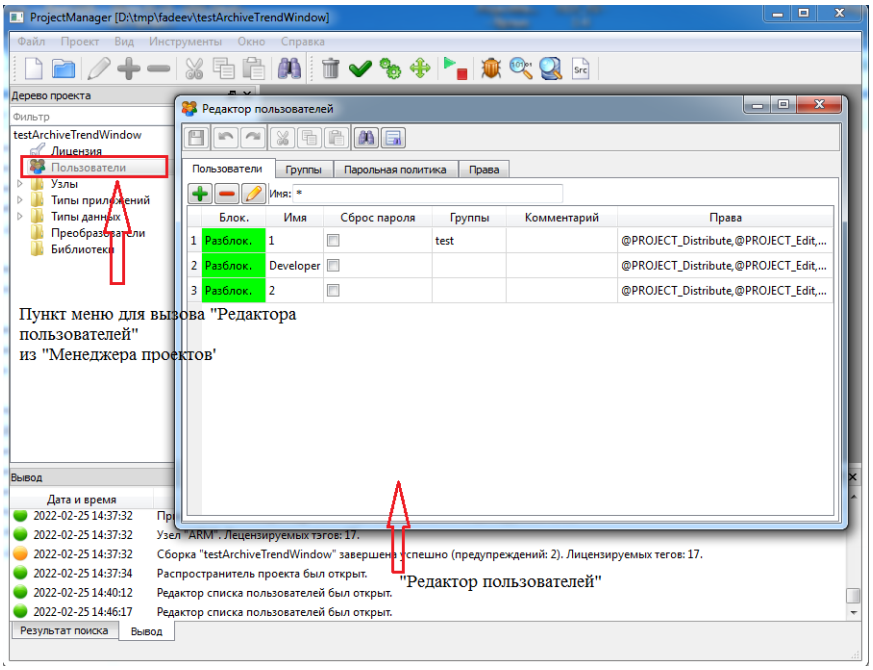


Рисунок 5.1 - Вызов Редактора пользователей из приложения Менеджер проектов

- в ходе штатной работы системы управления (для реализации используются графические функциональные блоки USERS_EDITOR и USERS_EDITOR_WINDOW, описанные в документе SCADA система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01 33).

При создании или редактировании существующего проекта в программном обеспечении «Менеджер проектов» для этих целей используется встроенный редактор «Пользователи».

В режиме штатной работы системы управления для создания и редактирования пользователей используется встроенное окно программы «Редактор пользователей» доступное к управлению для привилегированных пользователей с необходимыми правами.

Учетные записи пользователей хранятся в файле Users.usr на каждом вычислительном узле системы (на каждом узле своя копия файла). Данный файл, имеет контрольные суммы, исключающие возможность несанкционированной подмены файла или внесения в него изменений. При редактировании учетных записей администратором в режиме штатной работы системы, данный файл синхронизируется между всеми узлами. По умолчанию период синхронизации 60 секунд. Этот период можно изменить. Для этого существует специальный ключ **-settings_update_interval=NNN** у приложения **Loader** (см. описание в документе SCADA система "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95). Минимально возможный интервал 5 секунд.

ВНИМАНИЕ! Если происходит одновременное редактирование одного и того же пользователя на нескольких узлах, то в конечном итоге в файле со списком пользователей будут сохранены изменения, сделаные последним редактором. Синхронизация файла пользователей между узлами происходит при сохранении не мгновенно, а так как это описано выше. Если при последнем сохранении изменений уже успела произойти синхронизация файла со списком пользователей, то редактору будет выдано окно с предупреждением, если синхронизация не успела произойти, то сохранены будут изменения с узла, который последний по времени внес правки. Соответственно не стоит делать период синхронизации большим, но нужно учитывать, что при малом периоде синхронизации происходит увеличение сетевого трафика.

Функция контроля, валидации и верификации пользователей в режиме работы системы управления осуществляется внутренними механизмами SCADA системы «Соната», формируя события для журнала информационной безопасности (далее ИБ) в случае каких-либо изменений или действий, связанных с учетными записями пользователей.

Функция создания, редактирования и контроля учетных записей позволяет использовать следующие возможности в SCADA системе «Соната»:

- Возможность создания пользователя с уникальным именем и паролем;

- Сброс пароля пользователя при первом входе пользователя в систему;

- Смена пароля пользователя в режиме штатной работы системы по запросу администратора;

- Сброс пароля пользователя в режиме штатной работы системы по запросу администратора;

- Блокировка учетной записи пользователя на заданный временной интервал при превышении установленного количества неправильных вводов пароля, либо по запросу администратора;

- Создание групп пользователей с типовым набором прав и временными ограничениями на работу с прикладным программным обеспечением;

- Настройка уникальных прав пользователя на управление программными модулями SCADA системы «Соната»;

- Настройка уникальных прав пользователя на управление прикладными программными модулями системы управления;

- Возможность ограничения продолжительности сеанса работы пользователя;

- Возможность ограничения множества узлов системы управления, на которых может авторизоваться пользователь;

- Возможность ограничения времени и даты работы пользователя;

- Возможность ограничения множественной авторизации пользователя с одной учетной записью на нескольких узлах системы управления;

- Применение парольной политики с указанием минимальной длины пароля, специальных символов, регистров и шаблонов паролей.

ВНИМАНИЕ! Пользователь может обладать своими правами, а так же может быть включен в группу, в которой выполнены свои настройки прав. Результирующие права у пользователя получаются по правилу логического ИЛИ.

Настройка функционала реализуется в программном модуле «Редактор пользователей», входящим в состав программного обеспечения SCADA системы «Соната». Подробная информация по работе с данным модулем представлена в документе (см. SCADA система "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95 п. Выполнение комплекса программ → п. Описание программы ProjectManager (Менеджер проекта) → п. Работа в программе ProjectManager → п. Редактирование списка пользователей).

5.2. Функция шифрования передаваемых данных между узлами

В целях предотвращения перехвата сетевого трафика между узлами SCADA системы «Соната» используется функция шифрования передаваемых данных.

В системе Соната используются два собственных закрытых протокола. Первый основан на UDP и используется для передачи значений сигналов и событий в реальном времени. Второй протокол основан на TCP и предназначен для распространения прикладного программного обеспечения и для передачи сервисной информации.

Шифрование сетевых протоколов является опциональной функций, так как её использование может снижать быстродействие вычислительного узла системы управления. Для шифрования сетевых протоколов используется симметричный алгоритм AES XTS с диной ключа 256 бит. В качестве ключа шифрования выступает "солёный хэш" (см. рис. 5.2) с использованием пароля на прикладное программное обеспечение.

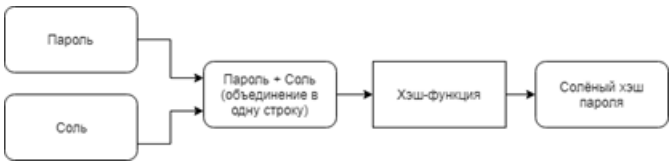


Рисунок 5.2 - "Солёный хэш"

Для получения "солёного хэша" используется встроенная в SCADA систему "Соната" специальная кодовая последовательность – соль, которая объединяется с указанным в проекте паролем перед вычислением хэша. Использование соли предотвращает атаки, основанные на заранее просчитанных таблицах для паролей.

Для увеличения надежности и достоверности передаваемых данных сетевые пакеты SCADA системы «Соната» содержат отметку времени и контрольную сумму CRC-16. Сетевые пакеты, у которых некорректная контрольная сумма или отметка времени не соответствующая текущему времени системы управления не обрабатываются.

Для активации шифрования трафика необходима дополнительная настройка прикладного программного обеспечения:

- В среде разработки в программном обеспечении «Менеджер проектов» установить пароль, используемый при шифровании (см. SCADA система "СОНАТА" Руководство пользователя

КУНИ.505200.023-01.01 95 п. Выполнение комплекса программ → п. Описание программы ProjectManager (Менеджер проекта) → п. Работа в программе ProjectManager → п. Редактирование настроек проекта и связей приложений);

- Для всех узлов системы управления в системной задаче или скрипте запуска программы «Loader» указать дополнительный аргумент командной строки -password=NNNNNNNN, где NNNNNNNN – сформированный редактором проекта солёный хеш, который будет использован для шифрования трафика (см. SCADA система "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95 п. Выполнение комплекса программ → п. Другие программы SCADA системы "Соната" → п. Программа "Loader" для загрузки проекта на узлах системы "Соната").

5.3. Функция контроля целостности среды исполнения SCADA системы «Соната» и прикладного программного обеспечения

SCADA система "Соната" предоставляет механизм контроля целостности прикладного программного обеспечения и среды исполнения. Контроль целостности может выполняться, как при старте узла, так и периодически.

Контроль целостности – длительная и ресурсоёмкая операция, в зависимости от размера прикладного программного обеспечения и вычислительной мощности процессора узла, поэтому является опциональной функций. Для ее активации необходимо выполнить действия в программном модуле «Центр управления», входящим в состав программного обеспечения SCADA системы «Соната», описанные в п. Выполнение комплекса программ → п. Другие программы SCADA системы "Соната" → п. Программа управления "ControlCenter" или Центр управления (смотрите описание функционала кнопки Подписать) документа SCADA система "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95.

При её активации загрузчик узла Loader, перед стартом проекта, будет осуществлять проверку контрольных сумм файлов. В случае несовпадения контрольной суммы одного и более файлов прикладного программного обеспечения и среды исполнения будут выполнены следующие действия:

- запрет на запуск узла системы управления;
- генерация информационного сообщения в диагностический файл узла;
- генерация события для журнала информационной безопасности.

ВНИМАНИЕ! При установке цифровой подписи на проект выполняется создание файлов с контрольными суммами файлов проекта и среды исполнения Соната. Одним из таких файлов является файл списка пользователей Users.usr. Но, так как данный файл может изменяться в процессе работы системы, к примеру, администратор добавил нового пользователя, то изменение данного файла не считается нарушением целостности системы. Этот файл имеет независимую встроенную цифровую подпись и, в случае её нарушения, мнемосхема не позволит авторизоваться пользователям.

Для контроля целостности используются хэш суммы SHA256 от содержимого файлов. Контрольные суммы файлов проекта и среды исполнения располагаются в файле !VERSION.info. Данный файл также имеет свою контрольную сумму, в нём содержится уникальный идентификатор UUID текущего проекта, что позволяет исключить возможность несанкционированной подмены файла с цифровой подписью. Дополнительно данный файл обфусцирован.

Обновление прикладного программного обеспечения на узлах системы управления, посредством программного модуля «Менеджер проектов», сбрасывает существующую цифровую подпись. Для активации проверки целостности проекта и среды исполнения необходимо вновь воспользоваться программой «Центр управления» (см. п. Программа управления "ControlCenter"

или Центр управления документа SCADA система "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95). Для этого в окне программы «Центр управления» необходимо в дереве проекта выбрать весь проект или узлы, для которых требуется установить контроль целостности прикладного программного обеспечения и среды исполнения, и нажать кнопку "Подписать". Центр управления отправит команду подписи всем программным модулям «Loader» на всех узлах системы управления. В ходе операции создания цифровой подписи произойдет сканирование папки прикладного программного обеспечения и среды исполнения, затем сформируются индивидуальные файлы !VERSION.info с контрольными суммами. При следующем запуске узла программный модуль «Loader» произведет проверку целостности прикладного программного обеспечения и среды исполнения.

Для активации периодической проверки целостности прикладного программного обеспечения и среды исполнения в режиме штатной работы системы управления необходимо в системной задаче или скрипте запуска программы «Loader» задать дополнительный аргумент командной строки -verification_interval=NNN, где NNN - интервал в секундах периодической проверки (см. SCADA система "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95 п. Выполнение комплекса программ → п. Другие программы SCADA системы "Соната" → п. Программа "Loader" для загрузки проекта на узлах системы "Соната"). После выполнения периодической проверки программным модулем «Loader» будут выполнены следующие действия:

- генерация информационного сообщения в диагностический файл узла о результатах проверки контрольной суммы;
- генерация события для журнала информационной безопасности о результатах проверки контрольной суммы.

Для исключения излишней нагрузки на аппаратные средства задан минимальный интервал проверки - 600 секунд.

Для активации/деактивации проверки целостности требуются соответствующие права у пользователя (см. SCADA система "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95 п. Выполнение комплекса программ -> Описание программы ProjectManager (Менеджер проекта) -> п. Работа в программе ProjectManager -> п. Редактирование списка пользователей -> п. Вкладка пользователи -> п. Вкладка Общие).

В штатном режиме работы системы управления с использованием программы «Центр управления» можно запросить текущий статус целостности файлов. Информация о контрольных суммах может быть представлена в нескольких видах:

- краткая информация – перечень файлов с измененными контрольными суммами;
- полная информация – список всех контролируемых файлов с эталонной и реальной контрольными суммами

5.4. Функция разграничения передачи информации по IP-адресам и маскам подсети между узлами системы

Система "Соната" обладает двумя взаимоисключающими (все узлы могут работать только в одном режиме) встроенным механизмами рагранічения передачи информации по ip-адресам и маскам подсети между узлами системы:

- **Зеркалирование** - используются все указанные IP адреса узлов, по которым выполняется синхронная рассылка сетевых пакетов. Маршрутизацией занимается операционная система;
- **Маршрутизация** - в сети должен присутствовать хотя бы один узел с двумя сетевыми картами и двумя непересекающимися подсетями. На данном узле Соната выполняет функцию передачи сетевых пакетов из одной подсети в другую. ВНИМАНИЕ! Количество маршрутов в

сети равно 2*N, где N - количество узлов с двумя IP адресами (сетевыми картами). С увеличением количества маршрутов будет увеличиваться и нагрузка на узлах.

Настройка узлов для работы в данных режимах описана в документе SCADA-система "СОНАТА" Руководство пользователя КУНИ.505200.023-01-01 95 п. Редактирование конфигурации узла.

5.4.1. Режим зеркалирования

В этом режиме используются все указанные IP адреса узлов, по которым выполняется синхронная рассылка сетевых пакетов. Маршрутизацией занимается операционная система. При отказе одной линии связи с узлом данные доставляются по другой линии. В случае двойного отказа, например, когда на узле 1 отказала первая сеть, а на узле 2 отказала вторая сеть, связь между узлами 1 и 2 будет потеряна. В другом режиме (режиме маршрутизации), в подобном случае связь будет выполнятся через третий узел (с двумя ip-адресами), который будет выступать в роли маршрутизатора.

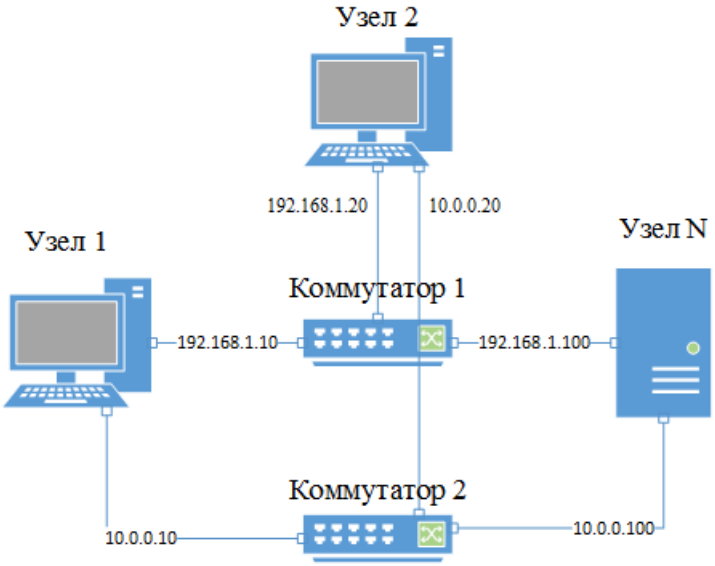


Рисунок 5.3 - Схема системы узлов, имеющих по 2 ip-адреса и работающих в режиме зеркалирования

5.4.2. Режим маршрутизации

Система "Соната" обладает встроенным механизмом поддержки работы проекта в нескольких непересекающихся подсетях без использования системного или стороннего механизма трансляции сетевых пакетов (без шлюзов и мостов). Этот механизм поддерживается загрузчиком узла (приложение Loader). Обеспечивается передача сигналов, их архивных значений и событий.

Внимание! Если маска подсети в свойствах узлов проекта указывается 0.0.0.0, то используется системная маршрутизация, иначе используется маршрутизация СКАДА системы "Соната" (обычно нужно указывать маску 255.255.255.0).

В СКАДА системе "Соната" используются две типовые схемы при разделении узлов проекта по подсетям. Схема с одним сервером (см. рис. 5.4), который должен иметь доступ к обоим подсетям, и схема с двумя серверами (см. рис. 5.5), которые так же должны иметь доступ к обоим подсетям.

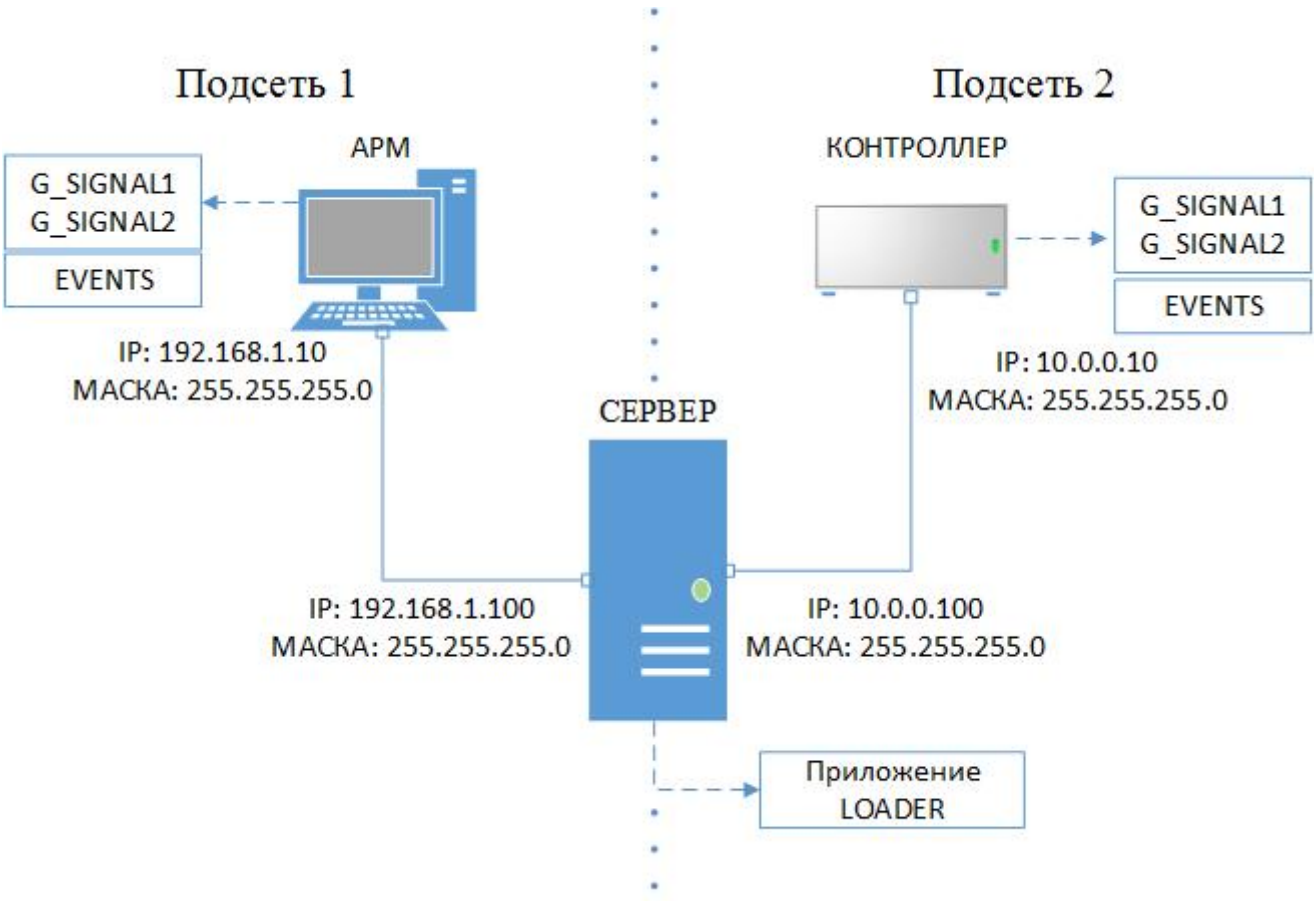


Рисунок 5.4 - Схема разделения на подсети с одним сервером

Рассматриваемый пример состоит из трёх узлов:

- автоматизированного рабочего места (АРМ), расположенного в подсети 192.168.1.X,
- программируемого контроллера (ПК), расположенного в подсети 10.0.0.X
- сервера, у которого установлены две сетевые карты, имеющие адреса двух независимых подсетей (192.168.1.X, 10.0.0.X).

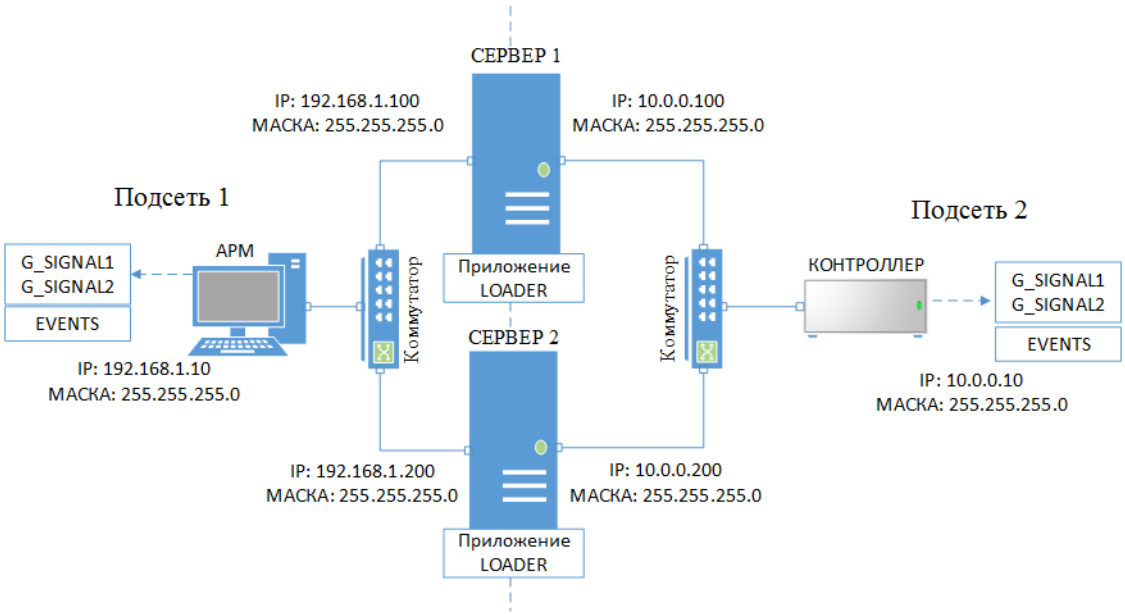


Рисунок 5.5 - Схема разделения на подсети с двумя серверами

Рассматриваемый пример состоит из четырёх узлов:

- автоматизированного рабочего места (АРМ), расположенного в подсети 192.168.1.X;
- программируемого контроллера (ПК), расположенного в подсети 10.0.0.X;
- двух серверов, у которых установлены две сетевые карты, имеющие адреса двух независимых подсетей (192.168.1.X, 10.0.0.X).

В ходе работы проекта приложение «Loader» на сервере будет выполнять функцию ретрансляции сетевых пакетов протоколов Соната из одной подсети в другую. Сетевые пакеты других протоколов игнорируются.

5.5. Функция ведения журнала событий информационной безопасности и диагностики

SCADA система "Соната" поддерживает функции формирования и фиксации событий в журнале событий информационной безопасности, возникающих в режиме штатной работы системы управления. Для каждого события или тревоги в архиве регистрируются следующие параметры: дата и время возникновения события; сведения об узле (АРМ, сервере, программируемом логическом контроллере); прикладном ПО, на которых зарегистрировано событие; пользователе (если применимо).

Журнал информационной безопасности регистрирует следующие события:

- регистрацию входа/выхода пользователей, включая неуспешные попытки доступа, с указанием идентификатора пользователя, даты и времени события;
- регистрацию событий создания, удаления, изменения привилегий пользователей;
- регистрацию совершаемых технологических операций в системе управления, включая дату и время совершения операции
- регистрацию действия администраторов системы управления;
- регистрация всех действий по созданию учетных записей (идентификаторов), присвоения и изменения прав доступа к компонентам системы управления;
- регистрация контроля запуска и останова узла\узлов системы управления;
- контроль целостности прикладного программного обеспечения и среды исполнения

Для просмотра событий информационной безопасности требуются соответствующие права пользователя (см. SCADA система "СОНАТА" Руководство пользователя КУНИ.505200.023-01.01 95 п. Выполнение комплекса программ -> Описание программы ProjectManager (Менеджер проекта) -> п. Работа в программе ProjectManager -> п. Редактирование списка пользователей -> п. Вкладка пользователи -> п. Вкладка Общие).

5.6. Функция контроля версий прикладного программного обеспечения

Функционал программного обеспечения «Центр управления», входящего в состав SCADA система "Соната", позволяет отслеживать информацию по текущим версиям прикладного программного обеспечения, запущенного на узлах системы управления. В качестве информации для оператора, либо администратора системы в окне программы «Центр управления» в поле «Дополнительной информации» указывается текущая версия. Если данные по версиям прикладного программного обеспечения между узлами различаются, то внизу окна программы для контроля оператором или администратором системы появляется надпись «Версии проекта на узлах различаются» (см. документ SCADA система "Соната" Руководство пользователя КУНИ.505200.023-01.01 95 п. Выполнение комплекса программ -> п. Другие программы SCADA системы "Соната" -> п. Программа управления "ControlCenter" или Центр управления).

Версия прикладного программного обеспечения хранится в файле !VERSION.info на каждом вычислительном узле системы управления. Данный файл, в случае активной функции цифровой подписи, имеет контрольную сумму, что исключает возможность несанкционированной подмены файла. Дополнительно данный файл обфусцирован.

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопр. документа и дата	Подп.	Дата
Изм.	измененных	замененных	новых	аннулированных					
1		все			50	35001-19			02.19