Сбор данных в OpenSCADA.

Оглавление

Введение

Сбор данных SCADA(Supervisory Control and Data Acquisition)-системы является её неотъемлемой частью, которая занимается получением данных из источников различного происхождения. Природа данных, с которыми работает SCADA, характеризуется сигналами базовых типов значений (целое, вещественное, логическое и строка). Сигналы изменяются во времени и обладает историей, жизнью. В теории управления технологическими процессами (ТП) под сигналом понимается значение датчика установки ТП в коде АЦП, 'сырой' сигнал или в реальном значении. Сигналы могут объединяться в группы по смысловой нагрузке, часто называемые параметрами. Например, развитые источники данных могут предоставлять структуры параметров с предопределённым набором связанных сигналов. Кроме непосредственного сбора данных в функции этого механизма также входит и передача воздействий на исполнительные устройства управления ТП; обычно это задвижки, насосы и регулирующие клапана. В совокупности этот процесс получил название Устройство Сопряжения с Объектом (УСО).

Источники данных характеризуются большим разнообразием, которое можно условно разделить на три группы.

- Источники 'сырых' данных, предоставляющие код АЦП или уровни дискретных сигналов, а также включающие простейшую обработку. Обычно это модули рассредоточенного УСО или простейшие промышленные программируемые логические контроллеры (ПЛК).
- Мощные промышленные ПЛК, обладающие значительной вычислительной мощностью и возможностью формирования сложных параметров с различной структурой.
- Локальные или сопутствующие источники данных. Например, УСО в виде плат расширения, а также данные аппаратного и программного окружения, в котором функционирует система.

Разнообразие источников данных породило большой спектр механизмов доступа к ним. Локальные источники данных различаются интерфейсами программирования приложения (API), а сетевые источники, в свою очередь, транспортным и протокольным уровнями взаимодействия. В целом это привело к тому, что добавление поддержки нового источника данных требует создание модуля сопряжения или драйвера. Учитывая же большое разнообразие источников, это крайне накладно, и фактически нереально охватить весь спектр рынка таких устройств. Ситуация несколько упрощается с сетевыми источниками благодаря наличию ряда стандартных и свободных протоколов взаимодействия, однако многие источники всё же используют собственные протоколы: закрытые коммерческие или протоколы, завязанные на закрытые механизмы ограниченного круга коммерческих операционных систем (ОС).

В терминах системы OpenSCADA предоставляются следующие объекты для обслуживания механизма сбора данных:

- Атрибут — объект отражения данных сигнала, включает текущее значение с типом сигнала и историю изменения значений;
- Параметр — объект группы атрибутов (сигналов) со структурой, соответствующей особенностям отдельно взятого источника данных;
- Контроллер — объект отдельного устройства данных. Как правило, это отдельный модуль УСО или устройство промышленного ПЛК.

Для учёта особенностей различных устройств сбора данных, а также различных механизмов взаимодействия в OpenSCADA предусмотрена подсистема 'Сбор данных', которая является модульной. В качестве модуля подсистемы выступает драйвер для сопряжения с источником данных отдельного типа. Каждый модуль может содержать конфигурацию нескольких устройств этого типа в виде объектов 'Контроллер' системы OpenSCADA. Общая схема объектов подсистемы 'Сбор данных' изображена на рисунке 1.

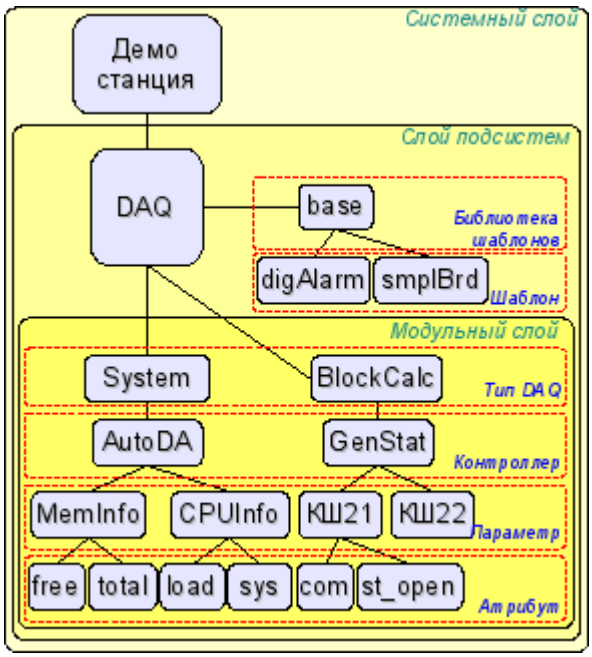

Рис. 1. Схема подсистемы 'Сбор данных'.

1. Методы сбора данных

Учитывая различие свойств источников данных, а также возможные варианты взаимодействия, методы сбора данных можно разделить на простой синхронный, простой асинхронный, пакетный и пассивный.

В рассмотрении механизмов ниже будут участвовать следующие объекты:

- ObjectSCADA — любой объект SCADA-системы, обращающийся за значением сигнала; например, архивы и визуализаторы;
- DAQParamAttribute — атрибут параметра подсистемы 'Сбор данных', выступающий посредником в доступе к значению сигнала источника данных;
- DAQParamAttributeArch — объект архива атрибута;
- HardwarePLC — объект источника данных, например, модули рассосредоточенного УСО или промышленные ПЛК.

1.1. Простой синхронный механизм сбора

Механизм характеризуется запросами к источнику данных синхронно с запросом к атрибуту параметра (рис.2). Данный механизм обычно применяется при работе с локальными источниками данных, характеризующимися низкой латентностью т.е. задержкой в ответе на запрос. С помощью этого метода можно получить актуальные данные непосредственно с запросом, однако время запроса объекта будет включать время транспортировки и обработки запроса источником данных.

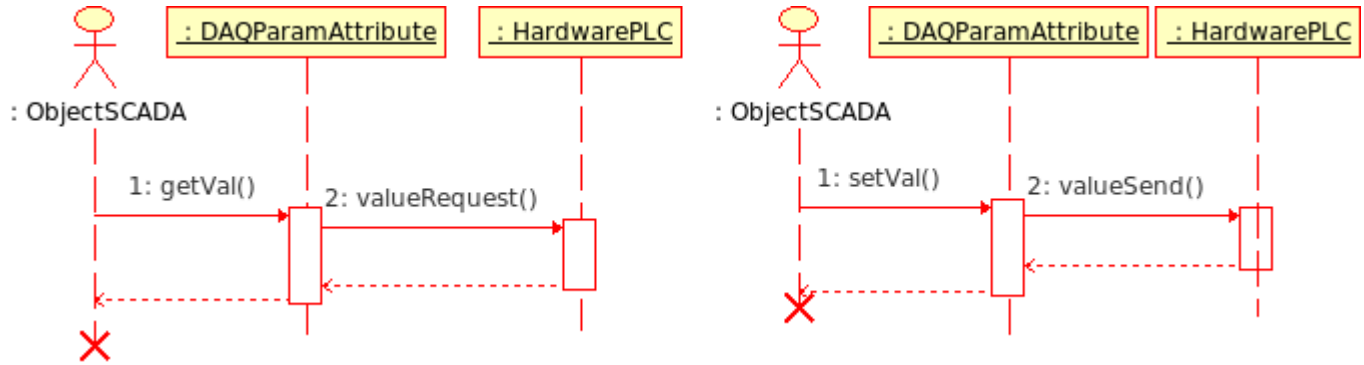

Рис. 2. Диаграмма последовательности взаимодействия при синхронных запросах.

В соответствии с диаграммой выше мы получаем следующую последовательность запросов получения данных и их передачи:

- объект SCADA-системы шлёт запрос значения к объекту атрибута параметра DAQParamAttribute::getVal();
- объект атрибута параметра, получив запрос, шлёт его источнику данных HardwarePLC::valueRequest();
- источник данных, обработав запрос, возвращает результат;
- объект атрибута параметра, получив результат, возвращает его объекту SCADA-системы.

В OpenSCADA такой механизм реализуют следующие модули подсистемы 'Сбор данных'.

- *ModBus* — модуль доступа к данным источников посредством семейства протоколов ModBus. В модуле реализован синхронный режим для записи данных.
- *DiamondBoards* — модуль доступа к данным PC/104 плат фирмы Diamond Systems. Платы PC/104 размещаются на ISA-шине, следовательно являются локальными и доступны сравнительно быстро. В режиме сбора данных не по прерыванию доступ к значениям АЦП осуществляется синхронно. Режим записи значения ЦАП всегда работает синхронно.
- *DAQGate* — модуль отражения объектов контроллеров удалённых OpenSCADA-станций на локальную. В модуле реализован синхронный режим для записи данных.
- *BlockCalc* — вычислитель на языке блочных схем. В качестве источника данных в нём выступает пользовательская блочная схема. Атрибуты параметров модуля синхронно обращаются к входам/выходам блоков блочной схемы.
- *JavaLikeCalc* — вычислитель на Java-подобном языке высокого уровня. В качестве

источника данных в нём выступает пользовательская программа на Java-подобном языке. Атрибуты параметров модуля синхронно обращаются к входам/выходам пользовательской вычислительной функции.

- *LogicLev* — модуль логического уровня параметров сбора данных, детальнее о нём в разделе 2. В качестве источника данных этого модуля выступают другие параметры подсистемы 'Сбор данных' и контекст исполнения шаблона параметров. Атрибуты параметров модуля синхронно обращаются к атрибутам других параметров в режиме отражения параметров подсистемы 'Сбор данных' или к входам/выходам контекста исполнения шаблона в режиме работы по шаблону.

1.2. Простой асинхронный механизм сбора

Механизм характеризуется запросами к источнику данных независимо от запроса к атрибуту параметра (рис.3). Обычно запросы к источнику данных осуществляются периодически в собственной задаче опроса отдельно взятого контроллера и блоками по несколько сигналов. При этом запросом к атрибуту параметра возвращается значение, полученное последним сеансом связи с источником данных. Данный механизм обычно применяется при работе с удалёнными (сетевыми) источниками данных, характеризующимися высокой латентностью, то есть задержкой в ответе на запрос.

С помощью этого метода можно обеспечить оптимизацию временного ресурса, затраченного на один сигнал, и тем самым увеличить максимальное количество опрашиваемых сигналов за интервал времени опроса.

В качестве примера рассмотрим промышленный ПЛК Siemens S7–315 при опросе его по шине Profibus (1,5 Мбит/с). Среднее время обработки MPI-запроса этим контроллером составляет 30 мс. Если использовать синхронный механизм для каждого сигнала, т.е. один запрос на каждый сигнал, то в течении одной секунды мы сможем получить около 33 сигналов. А если применить асинхронный механизм, т.е. в одном MPI-пакете получать до 220 байт или 110 сигналов целочисленного типа на 16-разрядов, то мы сможем за одну секунду получить до 3630 сигналов. Как можно видеть, эффективность асинхронного механизма в данном случае составляет 110 раз, а именно значение максимальной ёмкости MPI-пакета.

Недостатком асинхронного механизма является то, что запрос значения атрибута параметра возвращает не актуальное на момент запроса значение, а значение последнего сеанса опроса контроллера. Впрочем, если учесть, что источник данных может обновляться с периодичностью аппаратных ограничений АЦП, да и сами датчики могут иметь определённые ограничения в скорости реакции, то применение асинхронного механизма сбора может иметь серьёзные основания.

Применение асинхронного механизма для записи значений в ПЛК является достаточно редким явлением, поскольку запись значений обычно подразумевает воздействие оператора на ТП. Оператор по факту достаточно редко вносит коррективы в процесс, следовательно, запись можно выполнять синхронно. Однако существуют ситуации, например, управление ТП регуляторами на SCADA-системе, выполняющей функции среды исполнения ПЛК.

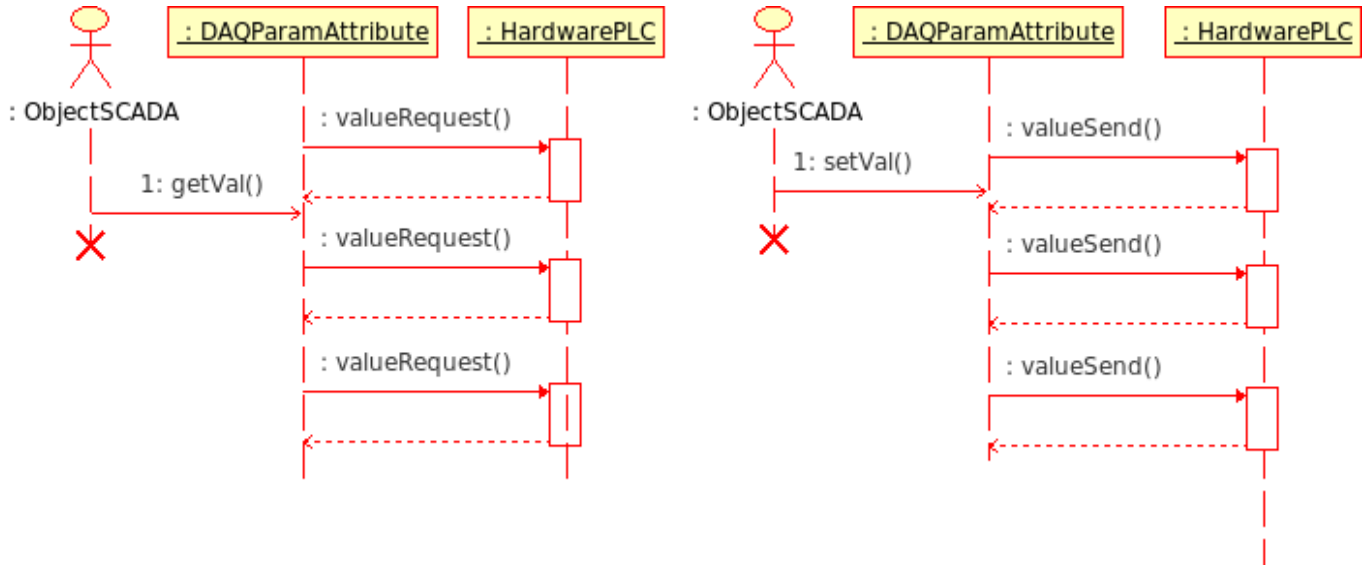

Рис. 3. Диаграмма последовательности взаимодействия при асинхронных запросах.

В соответствии с диаграммой выше мы получаем следующую картину:

- объект атрибута параметра (или вышестоящий объект контроллера) выполняет периодические запросы HardwarePLC::valueRequest() для получения значения сигнала или группы сигналов;
- полученные значения сигналов размещаются в объектах атрибутов параметров локально;
- объект SCADA-системы шлёт запрос значения к объекту атрибута параметра DAQParamAttribute::getVal() и получает сохранённое локально значение предыдущего сеанса опроса источника данных.

В OpenSCADA такой механизм реализуют следующие модули подсистемы 'Сбор данных'.

- *Siemens* — модуль доступа к данным контроллеров фирмы Siemens серии S7. В данном модуле асинхронный режим реализован как для чтения данных, так и для их записи (опционально) на ПЛК.
- *ModBus* — модуль доступа к данным источников посредством семейства протоколов Mod Bus. В модуле реализован асинхронный режим чтения данных.
- *SNMP* — модуль доступа к данным устройств сети посредством Simple Network Management Protocol. В модуле реализован асинхронный режим чтения данных.
- *System* — модуль доступа к данным окружения исполнения OpenSCADA. В модуле реализован асинхронный режим чтения данных.
- *DAQGate* — модуль отражения объектов контроллеров удалённых OpenSCADA-станций на локальную. В модуле реализован асинхронный режим чтения данных.

1.3. Пакетный механизм сбора

Пакетный механизм сбора данных характерен сбором данных каждого сигнала пакетом, включающим историю его изменения. Т.е. за один сеанс опроса данных получается несколько значений истории сигнала. Пакетный механизм работает совместно с синхронным и асинхронными механизмами.

В случае работы с синхронным механизмом выполняется фактический проброс архива источника данных для оперативной работы в системе (рис. 2). Как и простой синхронный механизм, его желательно применять только на низколатентных источниках данных или с источниками, работа которых является сеансовой, например, в сфере коммерческого учёта для чтения значений счётчиков.

При работе совместно с асинхронным механизмом история полученых сигналов обычно прямо помещается в архивы (рис. 4), а текущее значение атрибута параметра устанавливается в последнее значение пакета. Данная комбинация эффективна при сборе быстрых данных или при синхронизации архивов после потери связи с удалённым источником данных.

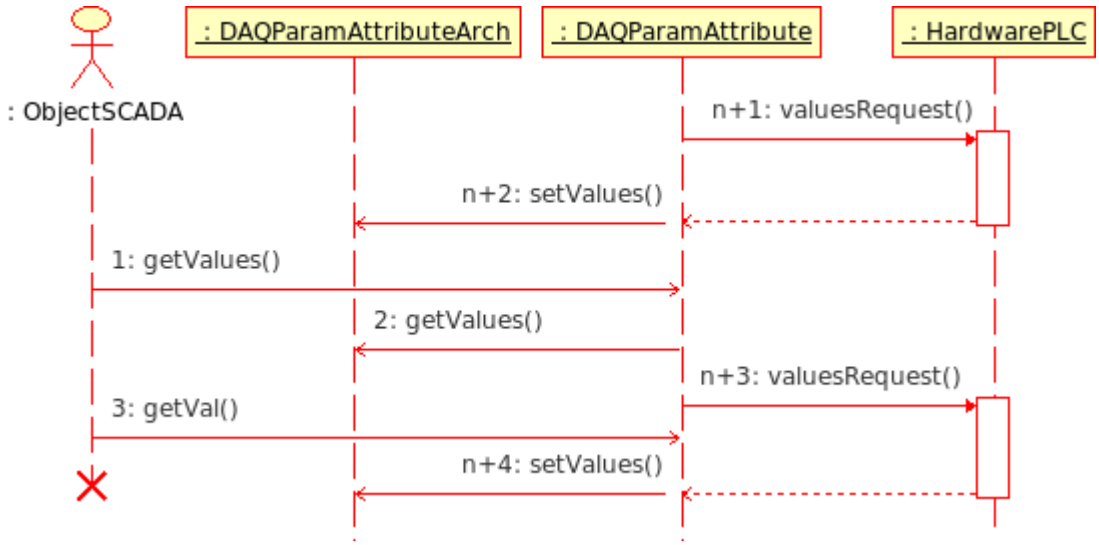

Рис. 4. Диаграмма последовательности взаимодействия при асинхронных запросах пакетного механизма.

В соответствии с диаграммой выше мы получаем следующее поведение пакетного механизма при асинхронных запросах:

- объект атрибута параметра или вышестоящий объект контроллера выполняет периодические запросы HardwarePLC::valuesRequest() для получения пакетов значений сигнала или группы сигналов;
- полученные пакеты значений сигналов помещаются в архив запросом DAQParamAttributeArch::setValues(), а последнее значение пакетов размещается в объектах атрибутов параметров;
- объект SCADA-системы шлёт запрос фрагмента архива к объекту атрибута параметра DAQParamAttribute::getValues(), а тот перенаправляет запрос к архиву DAQParamAttributeArch::getValues(). В результате возвращается фрагмент архива, доступный после предыдущего сеанса опроса источника данных;
- объект SCADA-системы шлёт запрос последнего значения к объекту атрибута параметра DAQParamAttribute::getVal() и получает сохранённое локально значение предыдущего сеанса опроса источника данных.

В OpenSCADA такой механизм реализуют следующие модули подсистемы 'Сбор данных'.

- *DiamondBoards* — модуль доступа к данным PC/104 плат фирмы Diamond Systems. Платы PC/104 размещаются на ISA-шине, следовательно, являются локальными и доступны сравнительно быстро. В режиме сбора данных по прерыванию осуществляется ожидание пакетов быстрых значений (до 200 кГц) за одну секунду (до 200000 значений в пакете) и последующее размещение данных пакетов в архивах атрибутов параметров DAQ.
- *DAQGate* — модуль отражения объектов контроллеров удалённых OpenSCADA-станций на локальную. Реализует синхронный и асинхронный пакетный режимы отражения архивов удалённых OpenSCADA-станций.

1.4. Пассивный механизм сбора

Пассивный механизм сбора данных характерен инициативой предоставления данных в SCADA-систему со стороны источника данных. Этот механизм является достаточно редким явлением, однако может иметь место в случае определённых условий или ограничений в возможности использования прямых механизмов сбора данных, рис. 5. Примером такой ситуации могут служить географически рассредоточенные системы сбора данных посредством мобильных сетей GPRS/EDGE. В таких сетях наделение клиентских узлов отдельными/реальными IP-адресами или формирование корпоративной мобильной сети может оказаться дорогим удовольствием, поэтому доступнее оказывается инициатива сеанса передачи данных с клиентских динамических IP-адресов на один реальный IP-адрес сервера SCADA-системы. Хотя возможна работа и через сетевую СУБД-посредника.

Воздействия на модификацию передаются источнику данных в момент сеанса передачи данных источником.

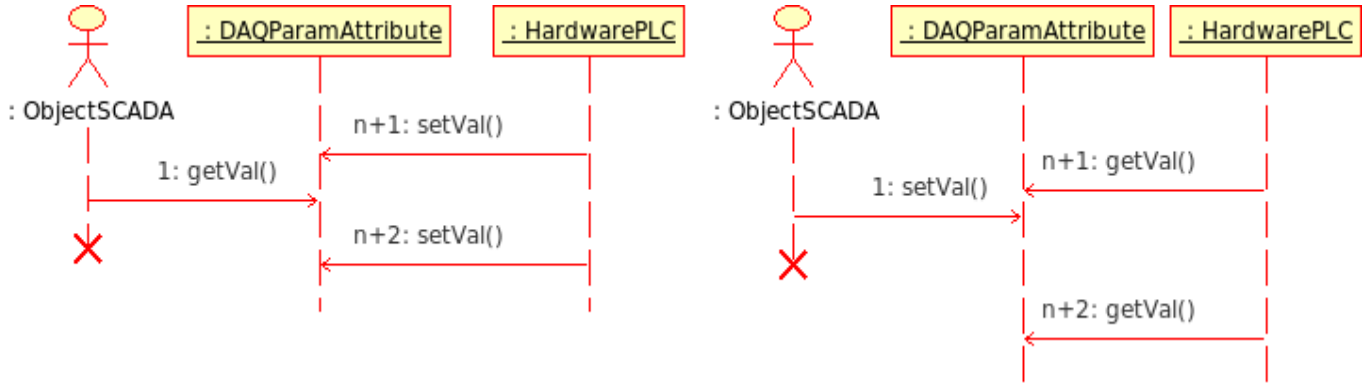

Рис. 5. Диаграмма последовательности взаимодействия при пассивном механизме работы.

В соответствии с диаграммой выше мы получаем следующее поведение пассивного механизма:

- объект источника данных осуществляет периодические сеансы связи с объектом атрибута параметра DAQParamAttributeArch::setVal() для передачи своих данных и получения команд воздействия;
- объект SCADA-системы шлёт запрос последнего значения к объекту атрибута параметра DAQParamAttribute::getVal() и получает сохранённое локально значение предыдущего сеанса связи источника данных.

В OpenSCADA такой механизм ещё не использован, однако принципиальная возможность его реализации в системе есть.

2. Виртуальные источники данных

Кроме сбора физических данных актуальной является функция виртуального сбора данных. Виртуальные данные представляют собой данные, полученные внутри системы как независимо, так и на основе физических данных. Практически механизмы формирования виртуальных данных реализуются совместно с механизмом пользовательских вычислений. В среде промышленных контроллеров и SCADA-систем используются различные языки программирования. В случае с контроллерами в качестве таких языков часто используются языки низкого уровня (ассемблеры), однако в последнее время всё чаще используются языки высокого уровня (C, Pascal и другие), а также формальные языки МЭК 61131–3 (схемы потоков состояний SFC, блочные схемы FBD, релейные схемы LD и текстовые ST, IL). В случае со SCADA-системами вычисления чаще обеспечиваются языками программирования высокого уровня и формальными языками.

В системе OpenSCADA могут быть реализованы интерфейсы программирования и виртуальных источников данных на основе различных языков в отдельных модулях подсистемы «Сбор данных». На момент версии 0.6.3.2 доступны модули виртуальных вычислителей:

- Вычислитель на Java-подобном языке: /Doc / Java Like Calc,
- Блочный вычислитель: /Doc / Block Calc.

В ядро OpenSCADA интегрирован механизм пользовательских функций или API пользовательского программирования. Пользовательские функции могут предоставляться любым объектом системы, в том числе и модулями в соответствии со своей функциональностью, тем самым предоставляя пользователю некий набор функций для контроля за тем или иным объектом. Функции пользовательского API могут быть как статическими, т.е. реализующими фиксированную функциональность отдельного объекта, так и динамическими, т.е. формируемые пользователем под нужную ему задачу на языке пользовательского программирования высокого уровня.

Модуль /Doc / Java Like Calc предоставляет в систему механизм создания динамических пользовательских функций и их библиотек на Java-подобном языке. Описание функции на Java-подобном языке сводится к обвязке параметров функции алгоритмом. Кроме этого модуль наделен функциями непосредственных вычислений путём создания вычислительных контроллеров с ассоциированной вычислительной функцией. Модулем предоставляется механизм прекомпиляции контекстно-зависимых функций, что используется для встраивания пользовательских алгоритмов непосредственно в контекст различных компонентов OpenSCADA. Например, это механизм шаблонов параметров подсистемы «Сбор данных» и движок среды визуализации и управления (СВУ).

Модуль /Doc / Block Calc предоставляет в систему OpenSCADA механизм создания пользовательских вычислений. Механизм вычислений основывается на формальном языке блочных схем (функциональных блоков). Языки блочного программирования основываются на понятии блочных схем (функциональных блоков). Причём в зависимости от сущности блока блочные схемы могут быть: логическими схемами, схемами релейной логики, моделью технологического процесса и другое. Суть блочной схемы состоит в том, что она содержит список блоков и связи между ними. С формальной точки зрения блок – это элемент (функция), который имеет входы, выходы и алгоритм вычисления. Исходя из концепции среды программирования, блок – это кадр значений, ассоциированный с объектом функции. Входы и выходы блоков нужно соединять для получения цельной блочной схемы.

С целью наполнения API пользовательского программирования пользовательскими функциями созданы следующие специализированные модули статических функций API пользовательского программирования:

- Библиотека функций совместимости со SCADA Complex1: /Doc / F Lib Complex 1,
- Библиотека стандартных математических функций: /Doc / F Lib Math,
- Библиотека функций системного API: /Doc / F Lib SYS.

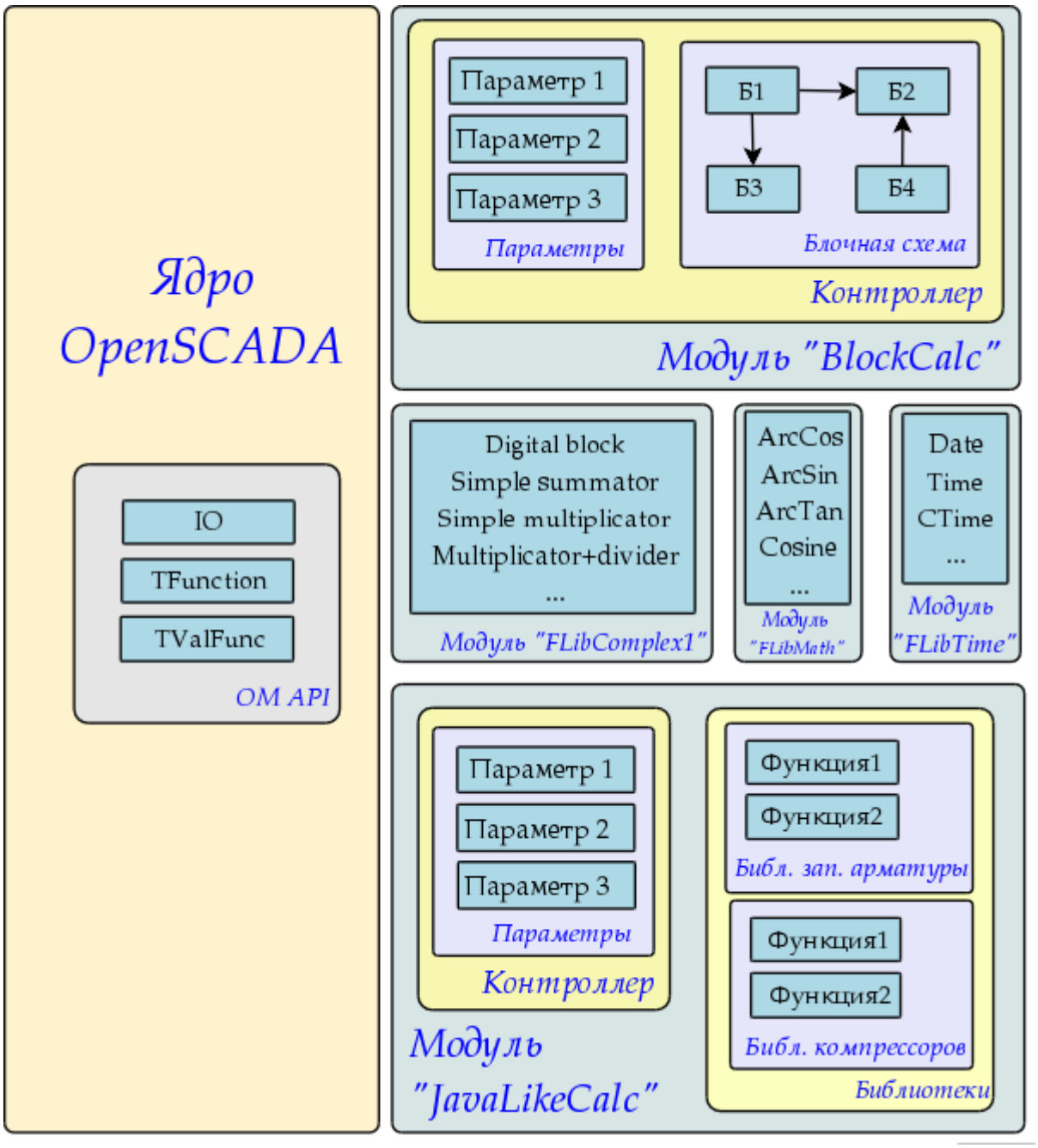

Рис. 6. Общая структура компонентов среды программирования

3. Логический уровень обработки данных

Выше мы говорили о том, что тип источника данных может колебаться от «сырого» до комплексного. Под «сырым» подразумевается источник, который предоставляет только элементарные сигналы (целое, вещественное, логическое, строка, ...), причём отдельно. Под комплексным подразумевается источник, который группирует сигналы и в параметре подсистемы 'Сбор данных' предоставляет атрибуты дополнительного назначения, покрывающие практически все диагностические задачи, т.е. параметр является законченным объектом, нетребующим дополнения.

Учитывая такой разброс, может возникнуть ситуация, когда информации в объекте параметра контроллера источника данных недостаточно для описания реального объекта ТП в целом и нужен производный объект более высокого уровня абстракции. Решением такой ситуации может быть формирование дополняющих параметров, что является ненаглядным и вносит путаницу. Более правильным решением является использование прослойки, так называемого «Логического уровня», выполняющего функции гибкого формирования параметров, контейнеров сигналов, необходимой структуры и включающего пост-обработку.

Функционально «Логический уровень» предназначен для предоставления в системе OpenSCADA механизма свободного формирования объектов параметров, контейнеров сигналов, нужной структуры.

Эксплуатационным назначением «Логического уровня» является:

- расширение сферы применения системы OpenSCADA за счёт увеличения гибкости описания объектов параметров подсистемы 'Сбор данных';
- сокращение затрат труда на создание сложных автоматизированных систем.

Концепция «Логического уровня» основана на шаблонах параметров, для которых в подсистеме 'Сбор данных' предусмотрен контейнер библиотек шаблонов (рис. 1). Каждая библиотека содержит шаблоны параметров, которые могут использоваться модулями подсистемы 'Сбор данных' для реализации параметров на основе шаблонов. Модулями системы OpenSCADA, которые используют шаблоны в своей работе, являются:

- LogicLev — модуль реализации классической концепции 'Логического уровня'.
- Siemens — модуль сбора данных контроллеров фирмы Siemens серии S7. В виду высокой гибкости и функциональности контролеров фирмы этой серии, которая позволяет формировать комплексные типы данных различной структуры, все параметры этого модуля работают по шаблонам.

Общий механизм работы 'Логического уровня' на примере модуля LogicLev изображён на рис. 7.

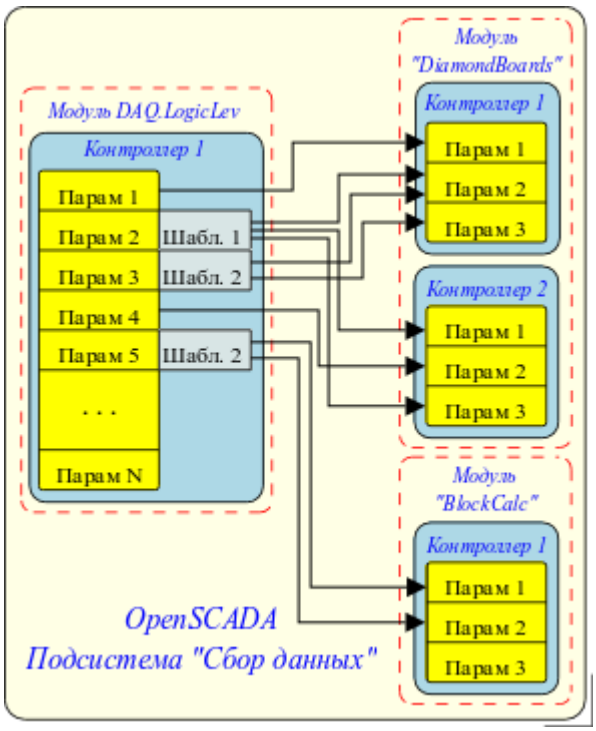

Рис. 7. Механизм работы 'Логического уровня' на примере модуля LogicLev.

Исходя из изображения видно, что параметры контроллера логического уровня выполняют функцию отражения других параметров подсистемы 'Сбор данных' (на примере параметров 1 и 4) и произвольное формирование параметров на основе шаблонов 1, 2 и других параметров подсистемы 'Сбор данных' (на примере параметров 2, 3 и 5).

Структура параметров с шаблоном в основе имеет структуру, изображённую на рис. 8.

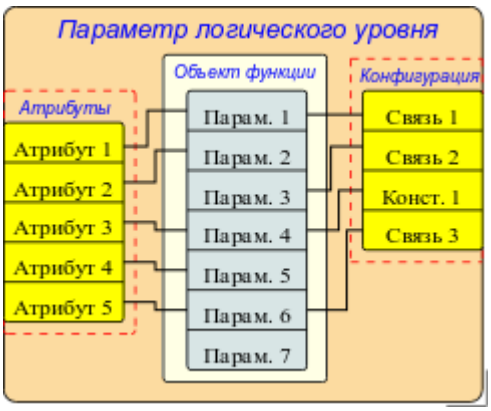

Рис. 8. Структура параметров, с шаблоном в основе.

Как можно видеть из структуры, параметр логического уровня состоит из объекта функции, атрибутов и конфигурации шаблона. Объект функции это экземпляр исполнения функции шаблона с набором входов/выходов и программой вычисления шаблона на одном из языков пользовательского программирования, обычно это Java-подобный язык пользовательского программирования модуля DAQ.JavaLikeCalc. Впрочем шаблон может быть вообще без программы, предоставляя только структуру проброса входов/выходов. Атрибуты в структуре изображают перечень атрибутов результирующего параметра в соответствии с шаблоном. Конфигурация в структуре предоставляет конфигурацию свойств шаблона и его внешних связей.

Логику работы логического уровня параметров можно записать следующим образом:

- Параметр связывается с шаблоном из которого получается структура атрибутов, в соответствии с функцией шаблона.
- В момент связывания параметра с функцией выполняется связывание объекта экземпляра функции параметра с функцией из шаблона.

- Далее, в соответствии с шаблоном функции, формируется структура связей. Исходя из структуры связей формируется форма связывания параметра и пользователем устанавливаются связи.
- При доступе к атрибутам полученного параметра производится проверка на наличие прямой связи. В случае наличия прямой связи запрос перенаправляется по этой связи, в противном случае значение берётся из объекта экземпляра функции параметра.
- В этот момент работает вычисление функции шаблона, по объекту функции параметров. При этом, перед вычислением производится чтение значений по связям, а после вычисления запись результатов по этим связям.

Шаблон параметров в целом предоставляет следующее:

- структуру входов/выходов функции шаблона;
- признаки конфигурации и связывания шаблона (константа, связь);
- предварительные значения конфигурации постоянных и шаблонов конфигурации связей;
- признаки атрибутов результирующего параметра логического уровня типов: не атрибут, атрибут с полным доступом, атрибут с доступом только на чтение;
- механизм вычисления входов/выходов функции шаблонов с использованием языка пользовательского программирования OpenSCADA.

На рис. 9 представлено изображение вкладки конфигурации шаблона параметров подсистемы 'Сбор данных' в виде таблицы с конфигурацией входов/выходов и текста программы пользовательского программирования.

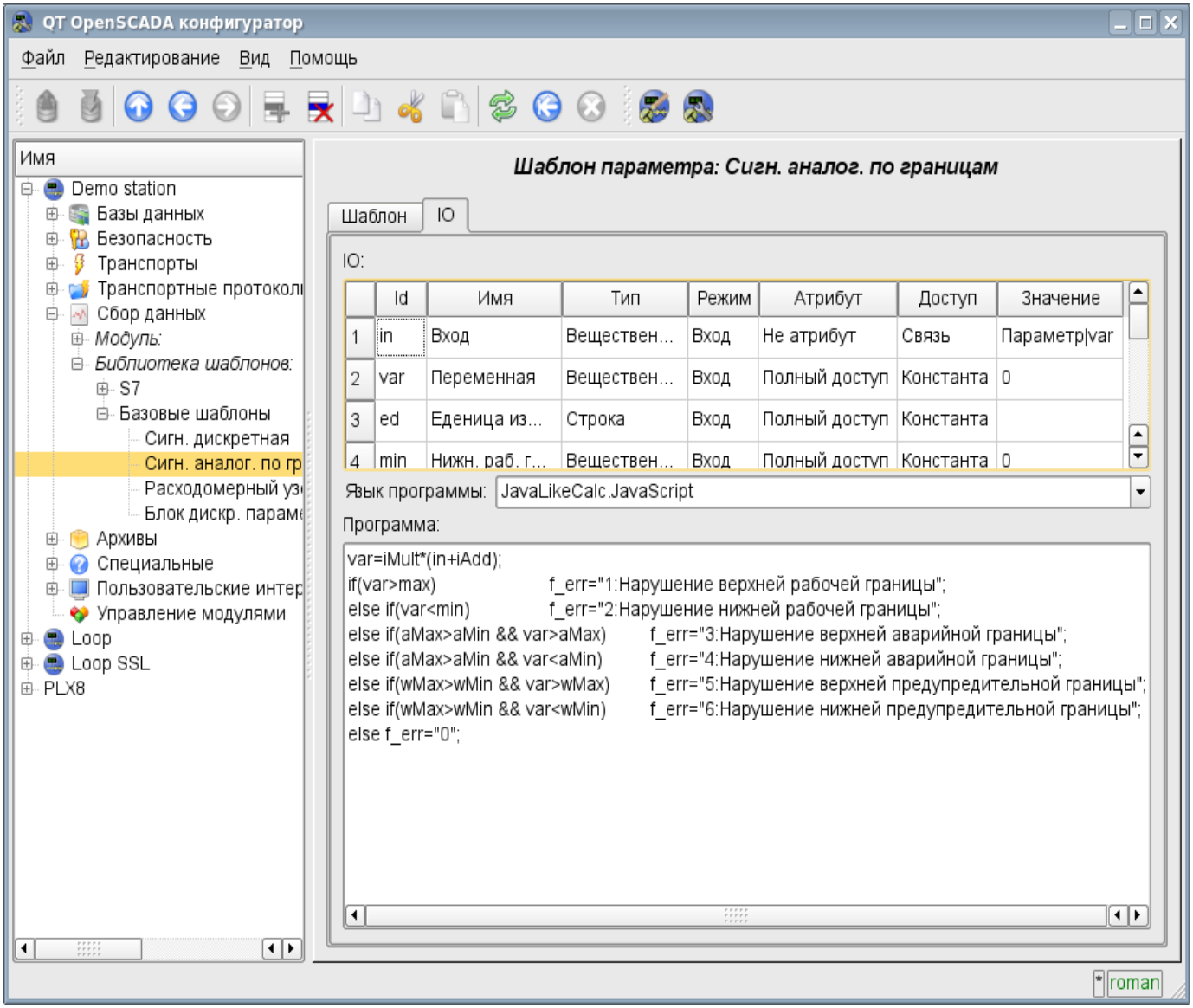

Рис. 9. Вкладка конфигурации шаблона параметров подсистемы 'Сбор данных'.

Полем входа/выхода шаблона параметра предусмотрены следующие свойства специализированного назначения: «Атрибут», «Доступ» и «Значение».

Свойство «Атрибут» выступает признаком отражения входа/выхода шаблона на результирующий атрибут параметра. Предусмотрены следующие варианты этого свойства:

- *не атрибут* — вход/выход функции шаблона не отражается на атрибут;
- *только чтение* — вход/выход функции шаблона отражается на атрибут с доступом только на чтение;
- *полный доступ* — вход/выход функции шаблона отражается на атрибут с полным доступом.

Свойство «Доступ» выступает признаком, указывающим на использование входа/выхода функции шаблона в результирующей конфигурации шаблона на логическом уровне. Предусмотрены следующие варианты этого свойства:

- *константа* — доступен для установки только на уровне конфигурации шаблона параметра в виде постоянной;
- *публичная константа* — доступен для установки на уровне параметра логического уровня в разделе конфигурации шаблона в виде постоянной;
- *Связь* — доступен для установки на уровне параметра логического уровня в разделе конфигурации шаблона в виде связи.

Поле «Значение» описывает предустановленное значение для постоянных и шаблон конфигурации внешних связей. Шаблон конфигурации внешних связей используется в целях описания механизма группировки и автоматического распределения внешних связей. Структура шаблона конфигурации внешних связей специфична для каждого модуля подсистемы «Сбор данных», который использует механизм шаблонов. В случае с модулем логического уровня распределение производится над внешними атрибутами параметров с шаблоном конфигурации внешней связи вида: <Параметр>|<атрибут>. Где «Параметр» используется для объединения параметров и помещения на форму конфигурации, а атрибут для ассоциативного связывания атрибутов при назначении параметра.

В качестве примера использования шаблона на рис.10 приведём изображения параметра модуля логического уровня 'F3'. На рис.10 представлена вкладка 'Конфигурация шаблона' для конфигурации, включая связывание, шаблона параметра. На рис.11 представлена вкладка 'Атрибуты' с перечнем атрибутов и их значений, созданных посредством шаблона.

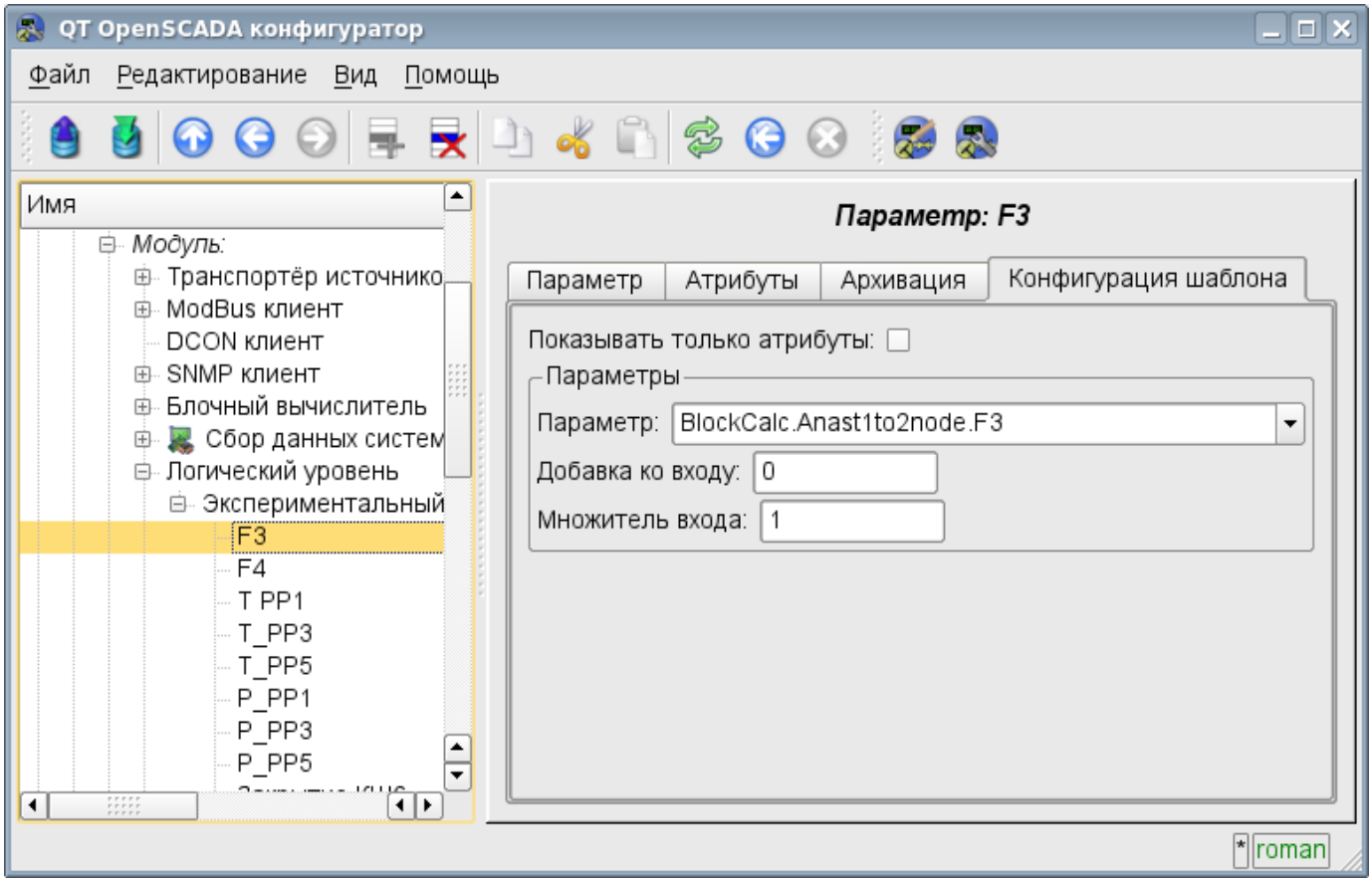

Рис. 10. Вкладка 'Конфигурация шаблона' параметра 'F3' модуля логического уровня.

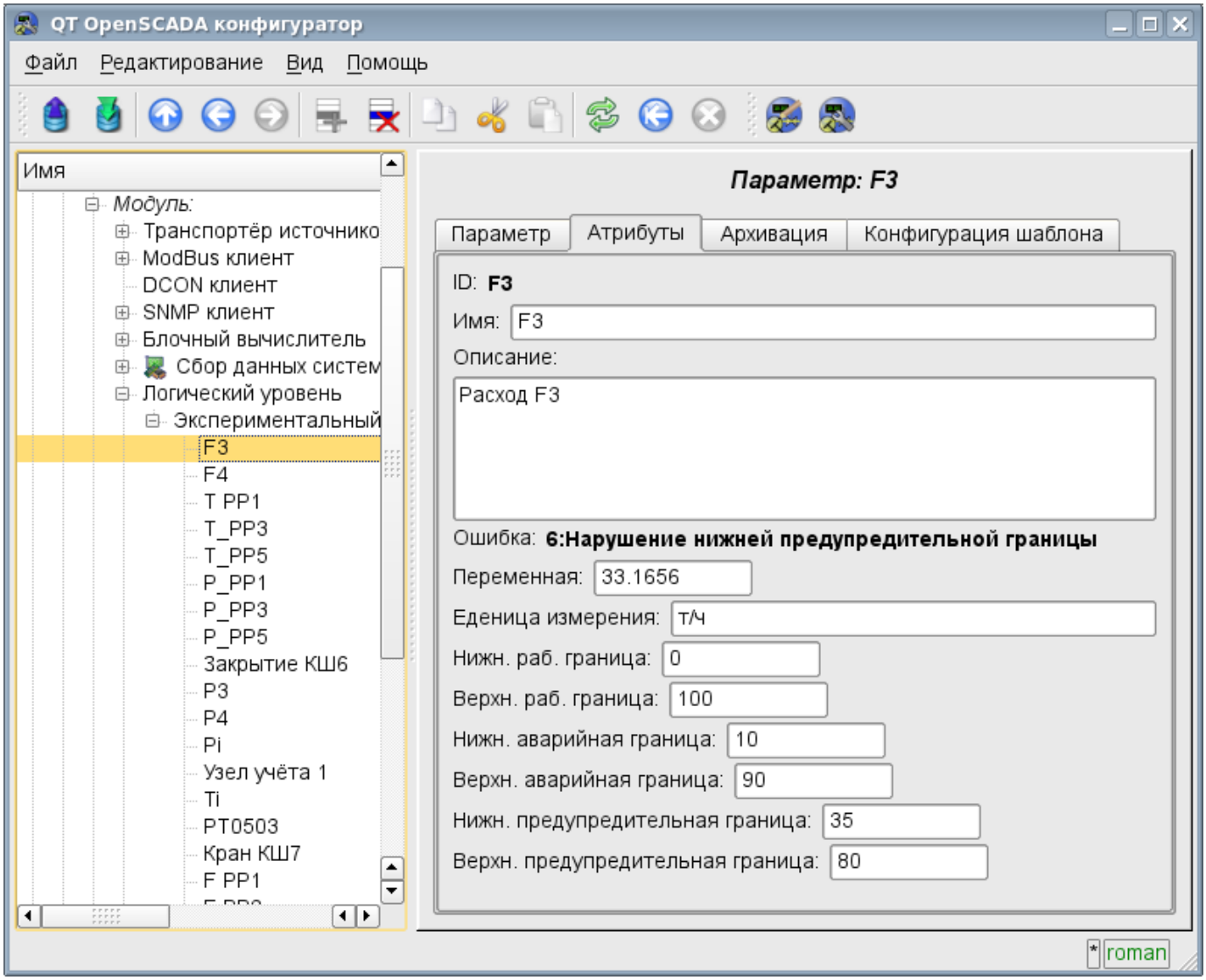

Рис. 11. Вкладка 'Атрибуты' параметра 'F3' модуля логического уровня.

4. Резервирование источников данных

Резервирование вообще и источников данных в частности служит для повышения общего уровня отказоустойчивости решения путём включения дублирующих узлов в совместную работу с основным узлом. В случае сбоя основного узла происходит подхват функций основного узла резервным. Часто схема резервирования может работать и в режиме распределения нагрузки между совместно работающими узлами.

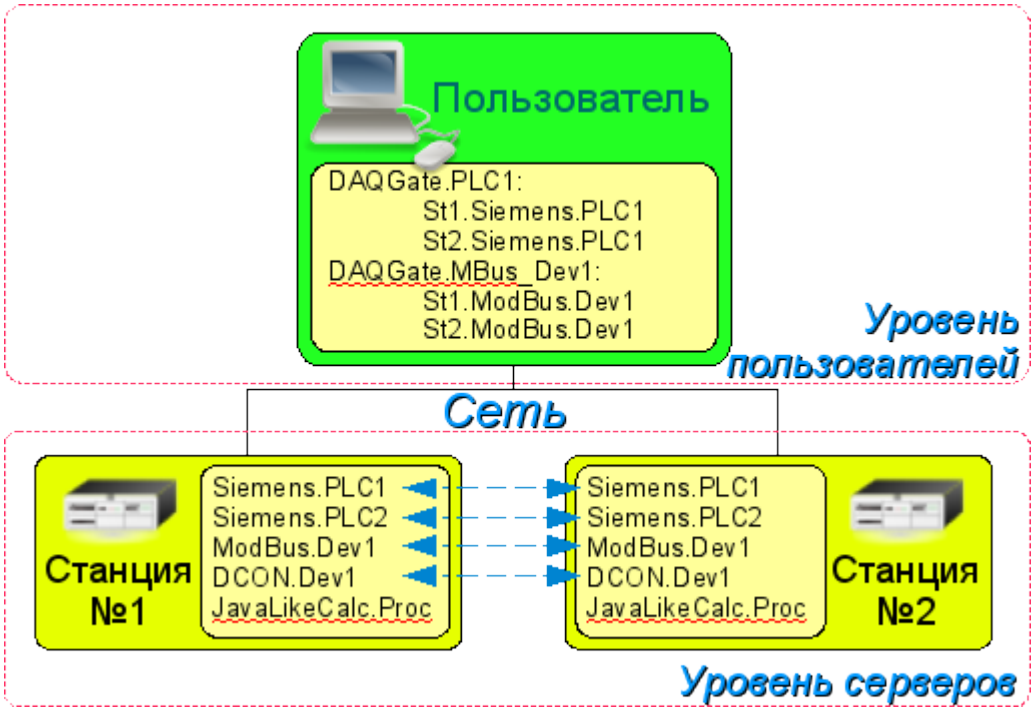

Рис. 12. Горизонтальное и вертикальное резервирование.

В случае с подсистемой «Сбор данных» системы OpenSCADA резервирование данных (рис.12) выполняет функции:

- Резервирование механизма сбора данных. Обычно эта функция реализуется без особых механизмов путём простого запуска параллельных резервных станций с одинаковой конфигурацией и работающих независимо. Однако в случае выполнения станцией функции ПЛК такое поведение недопустимо по причине одновременной выдачи управляющих воздействий и отсутствия синхронизации данных вычислителей.
- Компенсация потери данных на время простоя узла за счёт архива резервного узла. Предусмотрены два механизма компенсации. Первый и основной механизм осуществляет загрузку участков архива из резервной станции в момент запуска станции в целом или отдельных контроллеров подсистемы 'DAQ'. Участок архива запрашивается с момента последней записи в локальном архиве и по текущее время. Глубина запроса при этом ограничивается указанием предельного времени в конфигурации резервирования. Второй, дополняющий механизм, осуществляет заполнение дыр в архиве в момент фактического запроса пользователя к этим данным. Такой подход с одной стороны позволяет осуществить прогнозируемую по времени синхронизацию при старте, а с другой стороны фактически исключает потерю данных при условии работы хотя бы одной станции в течение всего рабочего времени.
- Распределение нагрузки по сбору данных между узлами. При создании сложных распределённых систем может оказаться важным вопрос прогнозирования и оптимизации общей производительности системы. С учётом таких задач механизм резервирования предусматривает исполнение задач сбора данных отдельных источников (контроллеров OpenSCADA) только на одной станции. При этом задачи остальных станций переходят в режим синхронизации данных с исполняющей станцией. В случае потери связи с исполняющей станцией запускается задача локального сбора данных. Предусмотрена также

возможность оптимального распределение нагрузки исполнения задач сбора данных группы контроллеров между станциями.

- Оптимизация нагрузки на внешние источники данных за счёт запроса данных у внешнего источника только одним узлом. В практике часто встречаются высоконагруженные источники данных или интерфейсы доступа к источникам данных, для которых даже сбор данных одной станцией может быть проблемой и потребует снижения периодичности сбора, т.е. качества данных. Механизм резервирования, кроме распределения нагрузки между станциями по описанной выше схеме позволяет снять дополнительную нагрузку на источник данных и его интерфейсы, тем самым повысив качество данных.
- Предотвращение некоторого отличия данных на разных узлах, связанное с несовпадением моментов времени при независимом сборе данных отдельными узлами путём получения данных у станции с активным контроллером. В системах высокой отчётности с резервированием должно быть исключено или сведено к минимуму расхождение в данных на разных станциях, что подразумевает реальный сбор данных одной станцией и синхронизацию с этими данными остальных станций.

Настройка резервирования начинается с добавления резервных станций в список системных станций OpenSCADA на вкладке 'Подсистема' подсистемы 'Транспорты' (рис.13). Далее вся конфигурация резервирования производится во вкладке 'Резервирование' подсистемы 'Сбор данных' (рис.14).

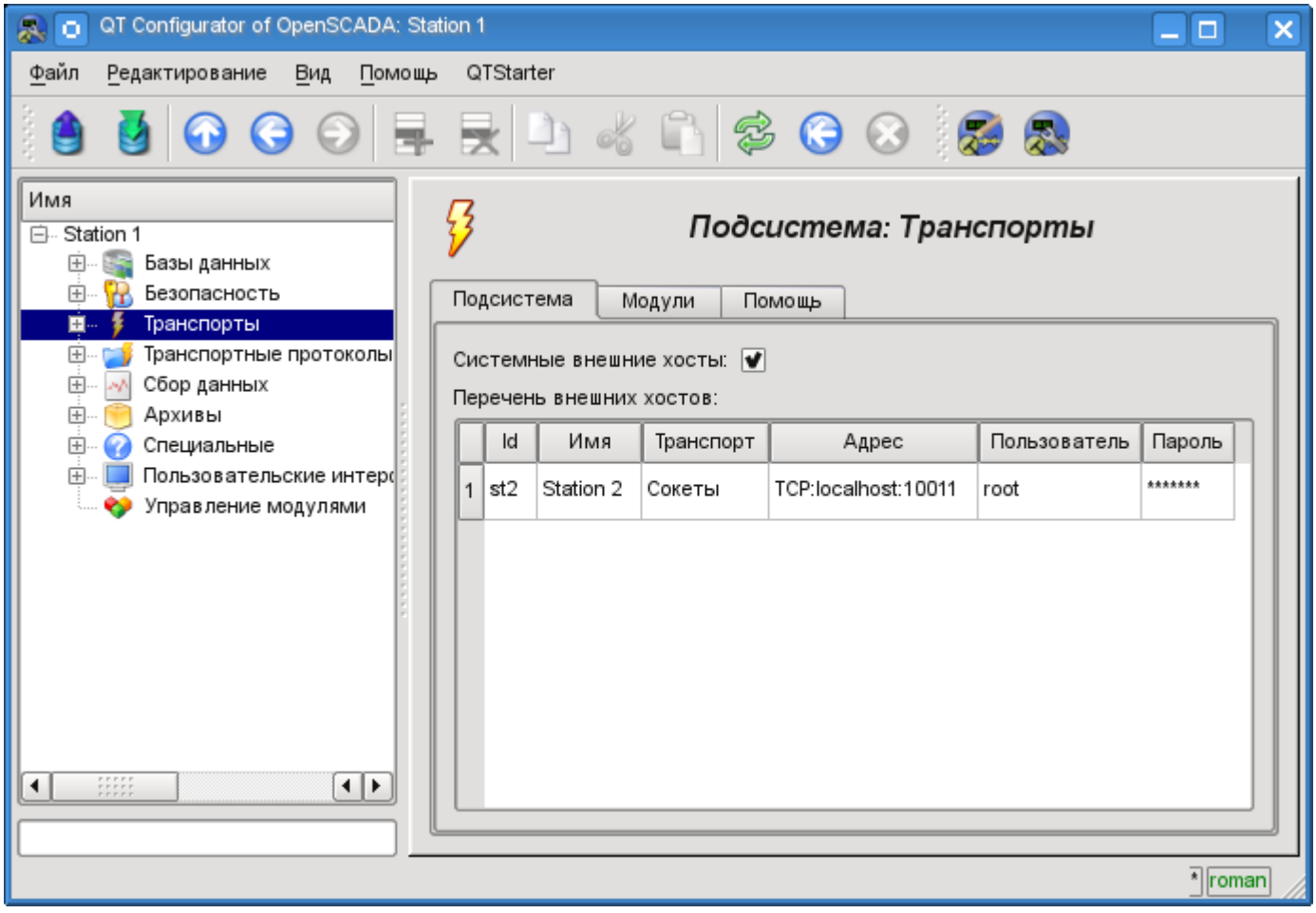

Рис. 13. Вкладка 'Подсистема' подсистемы 'Транспорты'.

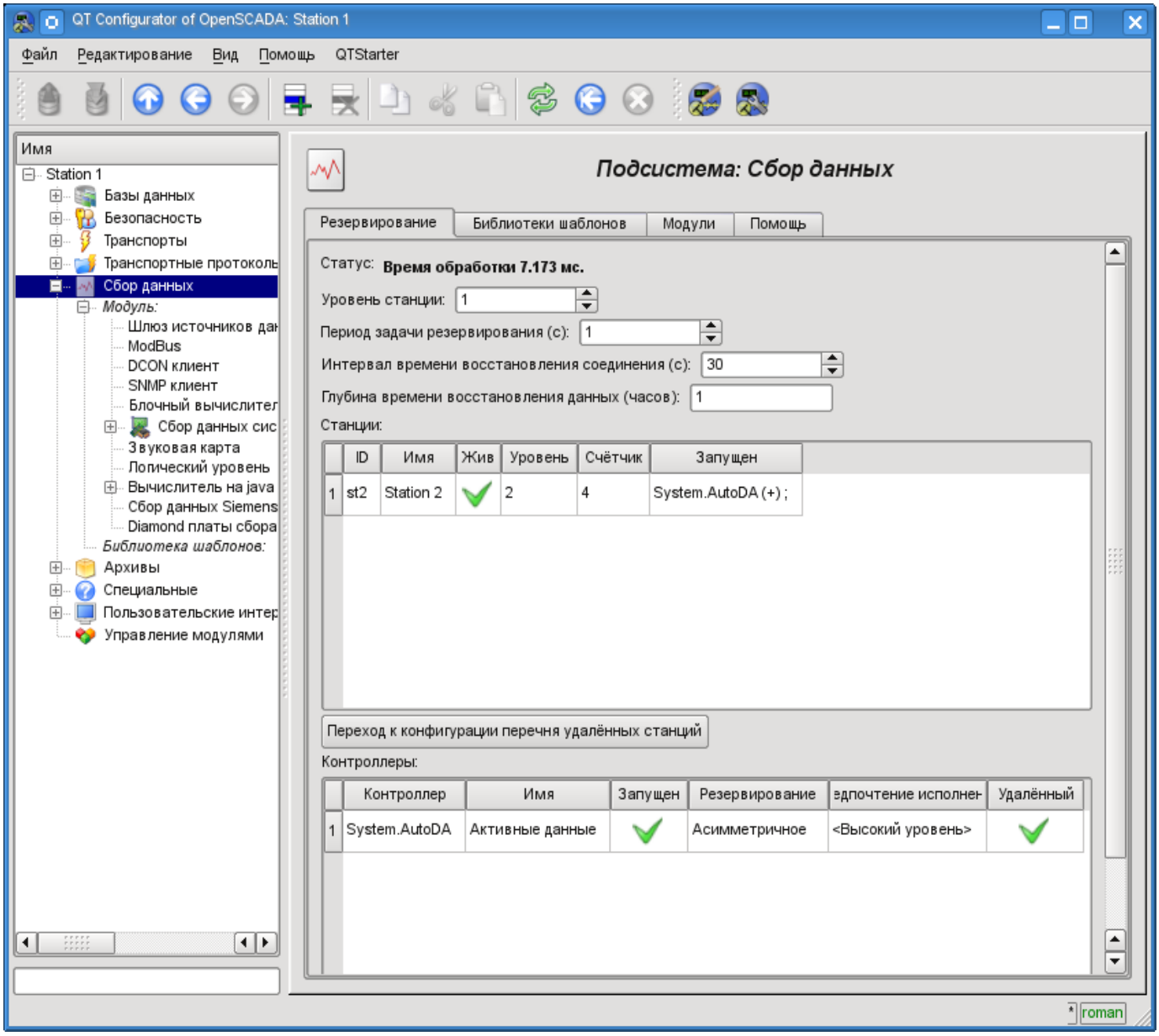

Рис. 14. Вкладка 'Резервирование' подсистемы 'Сбор данных'.

Задача обслуживания механизма резервирования запускается всегда и исполняется с периодичностью, установленной в соответствующем конфигурационном поле. Реальная работа по осуществлению резервирования осуществляется при наличии хотя бы одной резервной станции в списке станций и предполагает:

- Контроль за соединением с внешними станциями. В процессе контроля осуществляются запросы к удалённым станциям за обновлением информации о них и проверке связи. В случае потери связи со станцией повтор подключения к ней осуществляется через промежуток времени, указанный в конфигурационном поле интервала времени восстановления соединения. В поле 'Жив' станции отображается текущее состояние связи. В поле 'Счётчик' представлено количество запросов, осуществлённых к удалённой станции, или же время, оставшееся для осуществления следующей попытки соединения с потерянной станции. В поле 'Запущен' приводится перечень активных контроллеров на удалённой станции с признаком локального исполнения.
- Локальное планирование исполнения контроллеров в резерве. Планирование осуществляется в соответствии с уровнями станций и предпочтениями исполнения контроллеров.
- Вызов функции синхронизации данных для локальных контроллеров, работающих в режиме синхронизации данных из внешних станций. В процессе вызова осуществляется подготовка запроса к данным удалённой станции для параметров в контроллере и

отталкиваясь от времени прошлого запроса. По запросу возвращаются только значения модифицированных атрибутов и последовательность значений из архива в случае потери нескольких циклов значений.

Для контроля за временем, затраченным на выполнение цикла задачи обслуживания резервирования, предусмотрено поле статуса. При приближении реального времени выполнения к циклу задачи обслуживания резервирования рекомендуется увеличить периодичность исполнения этой задачи!

Для контроллера подсистемы 'Сбор данных' предусмотрены режимы асимметричного и симметричного резервирования. Асимметричное резервирование работает с той конфигурацией контроллера удалённой станции, какая есть и не пытается её обобщать. Симметричный режим подразумевает синхронизацию конфигурации контроллеров станций с конфигурацией станции наивысшего уровня, а также предполагает внесение изменений в конфигурацию всех контроллеров станций при изменении её на одной из станций. На данный момент этот режим не реализован!