|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Logo-Radiy | | **НВП Радій**  **RPC Radiy** | | |
|  | |  | | |
|  | | | | |
| Project: | | Radiy Platform Configuration Tools (RPCT) | | |
|  | |  | | |
| **RPCT Product Architecture Document**  **DX.X** | | | | |
| Version | Description | Prepared by | Reviewed by | Approved by |
| 0.4 | BaseService description was added  (chapter 5.1.1) | Y. A. Beliy,  S. A. Malokhatko  November 21 2014 | V. I. Tokarev  November 21 2014 | V. V. Sklyar  November 21 2014 |
| 0.3 | Application signals editor description was added  (chapter 4.4.4) | Y. A. Beliy,  S. A. Malokhatko | V. I. Tokarev | V. V. Sklyar |
| October 31 2014 | October 31 2014 | October 31 2014 |
| 0.2 |  | Y. A. Beliy,  S. A. Malokhatko | V. I. Tokarev | V. V. Sklyar |
| October 17 2014 | October 17 2014 | October 17 2014 |
| 0.1 |  | Y. A. Beliy,  S. A. Malokhatko | V. I. Tokarev | V. V. Sklyar |
| August 29 2014 | August 29 2014 | August 29 2014 |
| 0.0 |  | Y. A. Beliy,  S. A. Malokhatko | V. I. Tokarev | V. V. Sklyar |
| May 05 2014 | May 05 2014 | May 05 2014 |
|  | | | | |
| Kirovograd, Ukraine  2014 | | | | |

Contents

[Contents 2](#_Toc396217509)

[1 Scope and Purpose of the Document 3](#_Toc396217510)

[1.1 References 3](#_Toc396217511)

[1.2 Terms and Abbreviations 3](#_Toc396217512)

[2 Introduction to FSC-Based Solution configured by RPCT 5](#_Toc396217513)

[2.1 FSC Overview 5](#_Toc396217514)

[2.2 RPCT Overview 5](#_Toc396217515)

[3 RPCT Software Architecture 7](#_Toc396217516)

[4 CAD Software Architecture 8](#_Toc396217517)

[4.1 Overview 8](#_Toc396217518)

[4.2 Application Project Database Architecture 8](#_Toc396217519)

[4.2.1 Application Project Database naming 9](#_Toc396217520)

[4.2.2 Application Project Database structure versioning and backward compatibility 9](#_Toc396217521)

[4.3 Application Project Database Structure 9](#_Toc396217522)

[4.3.1 Таблица User 9](#_Toc396217523)

[4.3.2 Таблица Version 10](#_Toc396217524)

[4.4 Integrated Development Environment Software Architecture 10](#_Toc396217525)

[4.4.1 Database Interface Module 10](#_Toc396217526)

[4.4.2 Project Management Module 10](#_Toc396217527)

[4.4.3 Hardware Configurator Module 13](#_Toc396217528)

[4.4.4 Signals Editor Module 13](#_Toc396217529)

[4.4.5 FBL Manager Module 13](#_Toc396217530)

[4.4.6 Application Logic Designer Module 13](#_Toc396217531)

[4.4.7 FSC Data Visualization Designer Module 13](#_Toc396217532)

[4.4.8 Application Logic Compiler Module 13](#_Toc396217533)

[4.4.9 Project Builder Module 13](#_Toc396217534)

[4.4.10 SCADA Configuration Editor Module 13](#_Toc396217535)

[4.4.11 Firmware Uploader Module 13](#_Toc396217536)

[5 SCADA Software Architecture 14](#_Toc396217537)

[5.1 SCADA Server Software Architecture 14](#_Toc396217538)

[5.1.1 Base Service Architecture 14](#_Toc396217539)

[5.1.2 Configuration Service Architecture 20](#_Toc396217540)

[5.1.3 FSC Data Acquisition Service Architecture 20](#_Toc396217541)

[5.1.4 FSC Tuning Service Architecture 20](#_Toc396217542)

[5.1.5 Data Archiving Service Architecture 20](#_Toc396217543)

[5.1.6 Service Control Manager Architecture 20](#_Toc396217544)

[5.2 SCADA Client Software Architecture 20](#_Toc396217545)

[5.2.1 Monitoring and Tuning Application Architecture 20](#_Toc396217546)

[6 General Requirements and Solutions 21](#_Toc396217547)

[6.1 Hardware and Software Platforms 21](#_Toc396217548)

[6.2 Development tools, frameworks and libraries 21](#_Toc396217549)

# Scope and Purpose of the Document

The scope of this PAD is to depict the architecture of the Radiy Platform Configuration Tools software created by the company RPC Radiy.

The Product Architecture Document (PAD) provides a comprehensive architectural overview of the Radiy Platform Configuration Tools software. It presents a number of different architectural views to depict different aspects of the software. It is intended to capture and convey the significant architectural decisions which have been made on RPCT software. PAD implements high level requirements presented in RPCT Requirement Specification (SRS).

While SRS is the black-box document, PAD is white-box, and defines the internal structure of the product, providing design solutions for the requirements, and allocating the parts of the solution to the RPCT software.

To Do: Section overview

## References

1. IEC 61508:2010. Functional safety of electrical/electronic/programmable electronic safety-related systems.
2. IEC 62566:2011. Nuclear power plants – Instrumentation and control important to safety – Selection and use of complex electronic components for systems performing category A functions.
3. XXX RPCT Requirement Specification (SRS)

## Terms and Abbreviations

|  |  |
| --- | --- |
| FSC | The Radiy FPGA-based Safety Controller hardware platform, meaning a chassis with all modules. |
| FSC Data | Data received from FSC which include information about FSC state and technological process data |
| Technological Process Data | The state information of the controlled object received from FSC |
| FSC module | The highest-level module within the FSC, such as Logic Module (LM), Discrete Input Module (DIM), etc. They can be identified by its position within the chassis and a serial number. Logic Module can and should be programmed and reprogrammed to support the work of programmed algorithm synthesized from the logical blocks. All modules provide specific inputs and outputs. |
| Chassis | Frame and backplane supporting the FSC modules, the inter-module connections on the backplane, and the I/O FSC module external connections via rear connectors |
| Functional Block | The logic blocks that do primitive logical operation like Boolean OR, AND, or more sophisticated operations like delays, signal comparison, etc. |
| FBL | Functional Block Library – the set of the functional blocks for application logic building. |
| Tuning | Process which allows changing of predefined values at Application Logic program run. |
| Project Database | Storage for the project data like chassis configuration, application logic, videoframes, SCADA software configuration, etc. |
| CAD | The software part for FSC platforms devices configuration, designing application logic, compilation, updating firmware, etc. |
| SCADA | The software part which provides monitoring and visualization of technological process data. |
| I&C System | Instrumentation and control system |
| Chassis configuration | The set of modules, placed in the one chassis. Chassis configuration includes the combination of modules within module position in the chassis, module type, module revision and other service data. |
| Application logic | Main program of the FSC. There are algorithms which provide input data acquisition, performs the main functions of module (data processing, application logic etc.) and diagnostic, output data conditioning, data exchange with other modules[PAD] |
| Signals Data | The Signals Data is a table of signals which includes properties, such as signal name, signal address, signal range, signal unit, etc. Signals Data is a helpful representation of data needed for design, tuning and monitoring. |

# Introduction to FSC-Based Solution configured by RPCT

## FSC Overview

Radiy FPGA-based Safety Controller (FSC) is itself a single rack-mount chassis containing all required inputs, outputs and logic processing so that it operates as a single-channel device in an I&C Systems.

In terms of its high level functionality and flexibility, the FSC is essentially a safety PLC, except that its internal logic is performed by FPGAs instead of micro-processors. The FSC is composed of a Logic Module (LM) and a number of varied I/O modules contained within a chassis, just like a general-purpose safety PLC.

In normal operation, the FSC operates in “online” mode and performs the control function defined at the application level. Self-diagnostic are performed by both the application and platform levels, although they are aimed at different types of faults. Failures detected by either level that are potentially unsafe are converted to safe failures.

In online mode, no reconfiguration and no potentially unsafe connections to other devices is permitted. Monitoring of the system is possible through non-interfering 1-way broadcast of application state and hardware status data. Application Logic parameters tuning may be performed in this mode.

In offline mode, in which the outputs are in the safe state, diagnostic equipment may be connected and reconfiguration is possible.

RPCT Software is intended for design application logic, uploading it to FSC in offline mode, monitoring and tuning FSC in online mode.

## RPCT Overview

The RPCT is the multicomponent software platform which allows user to configure FSC chassis, design application logic, monitoring and tuning FSC in online mode.

RPCT software consists of two main parts: CAD and SCADA.

The CAD part provides offline tools for FSC chassis configuration, application logic design and its compilation, firmware upload, visualization design and SCADA software configuration.

The SCADA part provides online monitoring and visualization of the whole system (stands with chassis), including information about chassis configuration, diagnostic and technological data.

Also SCADA includes online tuning component which provides ability for changing parameters in application logic of FSC.

Processes and data flow are described in the Figure 2‑1



Figure 2‑1 Processes and data flow of RPCT

# RPCT Software Architecture

# CAD Software Architecture

## Overview

CAD is offline GUI tools for FSC chassis configuration, application logic design and its compilation, firmware upload, visualization design and SCADA software configuration.

FSC-based solution development process is shown on the Figure 4–1



Figure 4–1 FSC-based solution development process

## Application Project Database Architecture

База данных проекта приложения (далее - AppProjectDB) предназначена для хранения полной информации необходимой для создания и развертывания приложений на базе Radiy FSC.

Требования к архитектуре и реализации AppProjectDB приведены в таблице 4.1.

Таблица 4.1. Требования к AppProjectDB

|  |  |  |
| --- | --- | --- |
| **[T.APDB]** | **Requirement** | **VAL** |
| APDB.01 | Для управления AppProjectDB должна использоваться объектно-реляционная СУБД PostgreSQL |  |
|  | Для каждого проекта должна создаваться отдельная база данных |  |
|  | Должна обеспечивать одновременный доступ нескольких пользователей через локальную сеть |  |
|  | Функции модификации AppProjectDB должны быть, по возможности, вынесены в хранимые процедуры |  |
|  | Функции модификации AppProjectDB должны учитывать возможность одновременного доступа нескольких пользователей |  |
|  | Должна обеспечиваться версионность и обратная совместимость структуры AppProjectDB (см. 4.2.2) |  |
|  | Структура AppProjectDB должна соответствовать описанной в разделе 4.3 |  |
| **[/T.APDB]** |  |  |
|  |  |  |

### Application Project Database naming

Для каждого проекта должна создаваться отдельная база данных с именем в формате u7\_ProjectStrID, где: u7\_ - сигнатура, обозначающая БД проекта приложения; ProjectStrID – строковый идентификатор БД, определяемый пользователем и состоящий из латинских букв нижнего регистра, цифр и символа подчеркивания.

### Application Project Database structure versioning and backward compatibility

Каждая AppProjectDB должна иметь версию, которая однозначно определяет структуру базы данных.

Все новые версии RPCT software должны обеспечивать обновление AppProjectDB до актуальной версии.

## Application Project Database Structure

AppProjectDB должна включать в себя таблицы, описание и формат которых приведен в следующих разделах.

### Таблица User

Таблица User предназначена для хранения информации о пользователях AppProjectDB. Таблица имеет следующую структуру:

|  |  |  |
| --- | --- | --- |
| Filed | Type | Description |
| UserID | integer | Числовой идентификатор пользователя, ключ |
| Date | timestamp | Дата создания пользователя |
| Username | Text | User’s login |
| FirstName | Text | User’s first name |
| LastName | Text | User’s last name |
| Password | Text | User’s password |
| Administartor | Boolean | Признак того, является ли пользователь администратором |
| ReadOnly | Boolean | Признак того, что пользователь может только просматривать проект |
| Disabled | Boolean | Признак того, что пользователь запрещен, и не может открывать, просматривать и редактировать проект |

### Таблица Version

Таблица Version предназначена для хранения информации об истории обновлений версии AppProjectDB (см. 4.2.2). Таблица имеет следующую структуру:

|  |  |  |
| --- | --- | --- |
| Filed | Type | Description |
| VersionID | Integer | Ключ |
| VersionNo | Integer | Номер версии |
| Date | Timestamp | Дата обновления AppProjectDB до версии VersionNo |
| Reasone | Text | Причина обновления |

## Integrated Development Environment Software Architecture

Overview with list of all software modules

### Database Interface Module

Должна обеспечивать обратную совместимость и поддержание актуальной версии, накатывание изменений

### Project Management Module

Модуль должен обеспечивать следующие функции:

|  |  |  |
| --- | --- | --- |
| **[T.PMM]** | **Requirement** | **VAL** |
| PMM.01 | Ввод параметров подключения к серверу PostgreSQL (см. 4.4.2.1) |  |
|  | Получение и отображение списка доступных проектов |  |
|  | Создание нового проекта |  |
|  | Открытие существующего проекта для просмотра или редактирования |  |
|  | Обновление версии существующего проекта БД до актуального, поддерживаемого данной версией IDE |  |
|  | Удаление существующего проекта |  |
|  | Создание пользователей проекта |  |
|  | Редактирование профайла пользователей |  |
|  | Назначение прав пользователей |  |
|  | Отключение пользователей |  |
| **[/T.PMM]** |  |  |

#### Ввод параметров подключения к серверу PostgreSQL

Модуль должен принимать следующие параметры подключения к серверу PostgreSQL:

IP-адрес сервера

Порт подключения

Имя пользователя PostgreSQL

Пароль пользователя

Введенные параметры должны сохраняться на компьютере пользователя и использоваться при повторном запуске IDE.

#### Получение и отображение доступных проектов

Сразу после запуска IDE модуль должен соединиться с сервером PostgreSQL, получить и отобразить список доступных БД проектов.

По запросу пользователя, модуль должен обновлять информацию о доступных БД проектов.

#### Создание нового проекта

Для каждого нового проекта должна создаваться отдельная база данных с именем в формате u7\_ProjectStrID, где: u7\_ - сигнатура, обозначающая БД проекта приложения; ProjectStrID – строковый идентификатор БД, определяемый пользователем и состоящий из латинских букв нижнего регистра, цифр и символа подчеркивания.

В новой базе данных сразу должен создаваться пользователь Administrator, имеющий полный доступ к БД проекта.

#### Открытие существующего проекта

Модуль должен обеспечивать пользователю выбор и открытие любого из доступных проектов.

При этом, у пользователя должен запрашиваться логин и пароль на доступ к выбранному проекту.

Если пользователь не прошел авторизацию должно выдаваться соответствующее сообщение.

Если пользователь успешно авторизован, модуль должен обеспечить пользователю функции просмотра и редактирования проекта в зависимости от назначенных пользователю прав.

Если БД проекта имеет более раннюю версию чем та, которую поддерживает IDE, пользователю должно быть предложено обновить БД проекта до актуальной версии (см. 4.4.2.5).

IDE не должна открывать проекты более поздних версий чем та, которую поддерживает IDE. Для открытия таких проектов должна использоваться IDE соответствующей версии.

#### Обновление версии БД проекта до актуальной версии

Модуль должен обеспечивать обновление версии БД проекта до актуальной версии, поддерживаемой IDE. Обновление может выполнить только пользователь с именем «Administrator». Перед обновлением должен запрашиваться пароль администратора.

Резервная копия БД проекта более ранней версии должна сохраняться в базе данных с именем u7uprgadeN\_\*, где N – номер версии старой БД проекта.

#### Удаление существующего проекта

По запросу пользователя модуль должен обеспечивать удаление существующего проекта.

Удалить проект может только пользователь с именем «Administrator».

Перед удалением проекта еще раз должен запрашиваться логин и пароль пользователя.

По запросу на удаление БД проекта не удаляется физически, а переименовывается в БД с именем u7deleted\_\*.

#### Создание пользователей проекта

Модуль должен обеспечить возможность создания новых пользователей проекта. Количество пользователей не ограничено.

При создании нового пользователя должны быть заполнены поля профиля, перечисленные в следующем пункте.

#### Редактирование профайла пользователя

Модуль должен предоставить возможность ввода и изменения следующих полей профайла пользователя

|  |  |  |
| --- | --- | --- |
| Поле профайла | Может быть изменено администратором | Может быть изменено пользователем |
| Логин пользователя | Нет | Нет |
| Пароль пользователя | Да | Да |
| Имя пользователя | Да | Да |
| Фамилия пользователя | Да | Да |
| Признак того, что пользователь является «Администратором» | Да | Нет |
| Признак того, что пользователь может только просматривать проект без возможности редактирования | Да | Нет |
| Признак того, что пользователь «отключен» | Да | Нет |

#### Назначение прав пользователей

Модуль должен обеспечивать пользователям-администраторам возможность изменения прав доступа других пользователей, а именно:

* предоставлять пользователям право на просмотр проекта
* предоставлять пользователя право на редактирование (изменение) проекта

#### Отключение пользователей

Модуль должен обеспечивать пользователям-администраторам возможность «отключения» других пользователей, то есть введения запрета на доступ к проекту.

### Hardware Configurator Module

Модуль должен обеспечивать следующие функции:

| **[T.AHC]** | **Requirement Statement** | **VAL** |
| --- | --- | --- |
| AHC.01 | Shall provide ability to add new racks, delete existing racks and edit its properties. |  |
| AHC.02 | Shall provide ability to add new chassis, delete existing chassis and edit its properties. |  |
| AHC.03 | Shall provide ability to define chassis configuration. |  |
| AHC.04 | Shall control modules and chassis compatibility |  |
|  |  |  |
|  |  |  |
| **[/T.AHC]** |  |  |

Состав оборудования должен храниться в виде древовидной структуры на основе файлов в БД проекта. Каждое устройство должно идентифицироваться с уникальным строковым идентификатором и представляться в БД отдельным файлом. Устройство может содержать дополнительный набор устройств (например, шасси может включать в себя набор модулей). Корневым устройством является файл с именем “hc” (hardware configuration),который должен создаваться на этапе инициализации нового проекта.

#### Создание типового оборудования

Пользователю должна быть предоставлена возможность создания типовой конфигурации оборудования. Конфигурация типового устройства так же имеет древовидную структуру, для таких устройств корневым элементом является файл с именем “hp” (hardware preset). Кроме состава устройства, пользователь должен иметь возможность редактирования свойств устройства.

#### Добавление устройства в состав оборудования

Пользователю должна быть предоставлена возможность добавления устройств в состав оборудования, для чего пользователю требуется из контекстного меню выбрать тип устройства или выбрать устройство из ранее созданного типового оборудования. Добавление оборудования должно исключать возможность нарушения иерархии устройств (модуль может быть добавлен в шасси, но не наоборот).

#### Удаление устройства из состава оборудования

По команде пользователя файл устройства должен помечаться как удаленный. Если файл ранее не был зафиксирован, то он должен полностью удалиться из БД проекта. Файл должен помечаться как удаленный при процедуре фиксирования изменений.

#### Редактирование свойств оборудования

Каждое устройство имеет ряд свойств, пользователь должен иметь возможность редактирования этих свойств и фиксации изменений в БД проекта.

#### Проверка совместимости оборудования

Для проверки совместимости оборудования, пользователь должен иметь возможность ввести проверочный скрипт в свойствах родительского устройства (поле ChildRestriction). При добавлении нового устройства, модуль должен запускать проверочный скрипт родительского элемента, и при выявлении нарушений запрещать операцию.

### Application Logic Signals Editor Module

Модуль редактирования сигналов приложения должен обеспечивать выполнение следующих функций.

Table 3‑8 Requirements to the Signals Editor

| **[T.ALSE]** | **Requirement Statement** | **VAL** |
| --- | --- | --- |
| ALSE.01 | Shall provide ability to create and edit analogue signals and theirs properties |  |
| ALSE.02 | Shall provide ability to create and edit discrete signals and theirs properties |  |
| ALSE.03 | Shall provide repository of signals for Application Logic Designer |  |
| **[/T.ALSE]** |  |  |

Визуально модуль редактора сигналов приложения должен быть представлен в IDE отдельной вкладкой «Application Signals» (далее - вкладка).

После открытия проекта на вкладке должны в табличном виде отображаться все сигналы приложения.

#### Создание сигналов

При создании сигнала программа должна запрашивать у пользователя тип создаваемого сигнала (аналоговый или дискретный) и число каналов сигнала.

Программа должна обеспечивать редактирование свойств сигнала, перечень которых приведен в следующей таблице:

Таблица. Свойства сигнала

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Analog | Discrete | Description |
| ID | X | X | Уникальный строковый идентификатор сигнала. Начинается с символа #, за которым следуют латинские буквы, цифры и символ подчеркивания. После # обязательно следует буква.  Идентификаторы многоканальных сигналов заканчиваются на \_A, \_B, \_C и т. д. |
| ExternalID | X | X | Идентификатор сигнала для отображения. Произвольная строка. |
| Name | X | X | Наименование сигнала. Например «Температура котла» |
| Channel | X | X | Номер канала сигнала, начинается с 1. |
| DataFormat | X |  | Формат данных для аналоговый сигналов:  Binary unsigned LE  Binary signed LE  Binary unsigned BE  Binary signed LE |
| DataSize | X |  | Длина данных аналогового сигнала в битах |
| LowADC | X |  | Нижнее значение сигнала в единицах АЦП |
| HighADC | X |  | Верхнее значение сигнала в единицах АЦП |
| LowLimit | X |  | Нижняя граница физического значения сигнала |
| HighLimit | X |  | Верхняя граница физического значения сигнала |
| Unit | X |  | Физические единицы измерения сигнала |
| Adjustment | X |  | Поправка значения сигнала в физических единицах |
| DropLimit | X |  | Величина фиксации «обрыва» сигнала в физических единицах |
| ExcessLimit | X |  | Величина фиксации «превышения» сигнала в физических единицах |
| UnbalanceLimit | X |  | Величина фиксации рассогласования каналов многоканального сигнала в процентах |
| InputLowLimit | X |  | Нижнее значение входного электрического сигнала |
| InputHighLimit | X |  | Верхнее значение входного электрического сигнала |
| InputUnit | X |  | Единицы измерения входного электрического сигнала |
| InputSensor | X |  | Тип датчика входного сигнала |
| OutputLowLimit | X |  | Нижнее значение выходного электрического сигнала |
| OutputHighLimit | X |  | Верхнее значение выходного электрического сигнала |
| OutputUnit | X |  | Единицы измерения выходного электрического сигнала |
| OutputSensor | X |  | Тип датчика выходного сигнала |
| Acquire | X | X | Признак - сигнал нужно регистрировать |
| Calculated | X | X | Признак – сигнал вычисляемый |
| NormalState |  | X | Нормальное состояние дискретного сигнала, 0 или 1 |
| DecimalPlaces | X |  | Количество знаков после запятой в значении аналогового сигнала |
| Aperture | X |  | Апертура для записи в архив, процентов |
| InOutType | X | X | Тип сигнала – входной, выходной, внутренний |
| DeviceID | X | X | Строковый идентификатор устройства, к которому привязан сигнал |

Программа должна контролировать правильность ввода строкового идентификатора сигнала, а также автоматически добавлять суффиксы (\_A, \_B и т. д.) к идентификаторам многоканальных сигналов.

Для новых сигналов должна создаваться рабочая копия, которая до выполнения операции «check in» должна быть доступна только пользователю, создавшему сигнал.

После выполнения операции «check in» рабочая копия сигнала должна быть записана в БД и стать доступной всем пользователям проекта.

#### Редактирование свойств сигналов

Программа должна обеспечивать редактирование (изменение) свойств уже созданных сигналов. Перечень свойств подлежащих редактированию перечислен в таблице (выше).

Если пользователь начинает редактирование сигнала, для которого ранее была выполнена операция «check in», программа должна автоматически выполнить для этого сигнала операцию «check out», то есть создать новую рабочую копию сигнала.

Рабочая копия сигнала должна быть доступна только пользователю, который начал редактирование сигнала. Все другие пользователи проекта не должны иметь доступ к этой рабочей копии.

Программа должна запрещать выполнение операции «check out» для сигналов, которые уже редактируются другими пользователями.

После выполнения операции «check in» для рабочей копии сигнала, все изменения должны записываться в БД и новая версия сигнала должна становиться доступной всем пользователям.

#### Отмена изменения свойств сигналов

Программа должна обеспечивать отмену изменения сигналов, операция «undo changes» . При этом должна удаляться ранее созданная рабочая копия сигнала и из БД должны загружаться свойства ранее сохраненного сигнала.

Отмену изменений сигнала может сделать только пользователь, который начал редактирование сигнала. То есть тот, который выполнил операцию «check out» и создал рабочую копию сигнала.

#### Удаление сигналов

Программа должна обеспечивать удаление ранее созданных сигналов.

При удалении сигнала автоматически должна выполняться операция «check out» и создаваться новая рабочая копия сигнала с пометкой «удален». После выполнения операции «check in» сигнал должен быть удален из общего списка сигналов.

### FBL Manager Module

Модуль должен обеспечивать следующие функции:

|  |  |  |
| --- | --- | --- |
| **[T.AFBLM]** | **Requirement** | **VAL** |
| AFBLM.01 | Возможность создания AFB (Application Functional Blocks) |  |
| AFBLM.02 | Возможность редактирования свойств AFB |  |
| AFBLM.03 | Возможность контроля версий AFB |  |
| AFBLM.04 | Создание репозитория AFB для Application Logic Designer |  |
| **[/T.AFBLM]** |  |  |

Модуль управления AFB должен вызываться из меню главного окна IDE. Вызов модуля должен быть возможен только тогда, когда открыт проект.

После вызова модуль должен отобразить перечень существующих AFB.

#### Создание AFB

При создании нового AFB модуль должен запросить его наименование. После ввода информации в базу данных проекта должен быть добавлен файл формата XML, имеющий расширение AFB. В файле должен содержаться готовый шаблон описания AFB.

#### Редактирование свойств AFB

Перед редактированием AFB пользователь должен выполнить команду Check Out для выбранного файла. Для редактирования свойств AFB должен вызываться диалог, отображающий содержимое AFB в формате XML.

Модуль должен обеспечивать возможность валидации свойств AFB после редактирования. Валидация должна производиться с помощью шаблона XSD, который должен храниться в базе данных проекта. Пользователь должен выбрать шаблон, с помощью которого будет проводиться валидация. После валидации программа должна сообщить, соответствует ли описание AFB шаблону или нет.

#### Контроль версий AFB

Модуль должен обеспечивать доступ как к последней рабочей версии AFB, так и к предыдущим версиям.

Для просмотра истории изменений выбранного AFB должна быть реализована команда View History.

Для редактирования AFB пользователь должен выполнить команду Check Out. Одновременное редактирование AFB несколькими пользователями должно быть запрещено.

После редактирования пользователь должен выполнить команду Check In. При выполнении команды Check In должен быть запрошен комментарий к сделанным изменениям.

Должна быть обеспечена возможность отмены сделанных пользователем изменений.

#### Репозиторий AFB

Созданные пользователем AFB должны храниться в базе данных проекта и быть доступными для добавления в Application Logic Designer.

#### Формат файла описания AFB

При создании пользователем AFB программа сохраняет его описание в базе данных проекта в формате XML. Ниже приведен пример описания AFB.

<?xml version="1.0" encoding="UTF-8"?>

<ApplicationFunctionalBlocks>

<AfbElement Guid="{6130e210-e53f-4cdd-b79d-6f3dcf22bd09}" StrId="#test" Caption="test" OpCode="0">

<InputSignals>

<AfbElementSignal Caption="SignalCaption" Type="Analog"/>

</InputSignals>

<OutputSignals>

<AfbElementSignal Caption="SignalCaption" Type="Analog"/>

</OutputSignals>

<Params>

<AfbElementParam Caption="Param1" Type="AnalogIntegral">

<Value>

<AfbParamValue Type="AnalogIntegral" Value="0"/>

</Value>

<Default>

<AfbParamValue Type="AnalogIntegral" Value="0"/>

</Default>

<LowLimit>

<AfbParamValue Type="AnalogIntegral" Value="0"/>

</LowLimit>

<HighLimit>

<AfbParamValue Type="AnalogIntegral" Value="100"/>

</HighLimit>

</AfbElementParam>

</Params>

</AfbElement>

</ApplicationFunctionalBlocks>

Документ начинается стандартного заголовка XML-файла. Тело документа начинается с тега <ApplicationFunctionalBlocks> и заканчивается тегом </ApplicationFunctionalBlocks>.

Описание AFB начинается с тега <AfbElement> и заканчивается тегом </AfbElement>.

AFB содержит следующие атрибуты:

- Guid – уникальный идентификатор AFB;

- StrId – строковый идентификатор;

- Caption – наименование AFB;

- OpCode – тип операции (число).

В описании AFB находятся следующие разделы, обрамленные тегами:

- <InputSignals>...</InputSignals> - содержат описание входных сигналов AFB в структурах типа AfbElementSignal;

- <OutputSignals>...</OutputSignals> - содержат описание выходных сигналов AFB в структурах типа AfbElementSignal;

- <Params>...</Params> - содержат описание параметров AFB в структурах типа AfbElementParam.

Структуры описания входного и выходного сигнала AfbElementSignal содержат следующие атрибуты:

- Caption – наименование сигнала;

- Type – тип сигнала. Может принимать значение «Analog» и «Discrete».

Структуры описания параметра AFB AfbElementParam содержат следующие атрибуты:

- Caption – наименование параметра;

- Type – тип параметра. Может принимать значения «AnalogIntegral» (целое число), «AnalogFloatingPoint» (число с плавающей запятой), «DiscreteValue» (дискретный параметр).

Структуры описания параметра AFB AfbElementParam содержат следующие разделы:

- <Value>...</Value> - текущее значение параметра типа AfbParamValue;

- <Default>...</Default> - значение параметра по умолчанию типа AfbParamValue;

- <HighLimit>...</HighLimit> - максимальное значение параметра типа AfbParamValue;

- <LowLimit>...</LowLimit> - минимальное значение параметра типа AfbParamValue.

Структура AfbParamValue имеет следующие атрибуты:

- Type – тип значения. Должен совпадать с типом параметра (Type в структуре AfbElementParam);

- Value – значение. Может быть целым числом (например, «100»), дробным числом (например, «1.25») или дискретным значением («0» или «1»).

### Application Logic Designer Module

### FSC Data Visualization Designer Module

### Application Logic Compiler Module

### Project Builder Module

Модуль сборки (компиляции) проекта предназначен для преобразования данных проекта в файлы, необходимые для прошивки модулей FCS и конфигурирования программного обеспечения SCAD.

Сборка проекта должна выполняться по запросу пользователя в двух режимах – debug и release. Каждая сборка имеет свой уникальный номер начиная с 1.

Debug-сборка – это тестовая сборка, предназначенная для выявления ошибок в проекте и проверки выполнения функций заложенных в проект. Debug-сборка сохраняется только на компьютере пользователя, который выполнил сборку. Не рекомендуется использование Debug-сборки для прошивки и конфигурирования систем, находящихся в промышленной эксплуатации. Если в модули FSC будет записана прошивка Debug-сборки, то программное обеспечение SCADA будет выдавать об этом диагностическое сообщение.

Release-сборка – это рабочая версия сборки, предназначенная для прошивки и конфигурирования систем, находящихся в промышленной эксплуатации. Release-сборка, если в ней отсутствуют ошибки, всегда полностью сохраняется в БД проекта. Таким образом, существует возможность восстановить любую версию прошивки системы.

#### Структура сборки

Debug- и Release-сборка имеют абсолютно одинаковую структуру. Каждая сборка – это набор файлов, необходимых для прошивки модулей FSC и конфигурирования ПО SCADA.

Все файлы сборки размещаются в каталоге с именем формата (далее – каталог сборки):

ProjectStrID-BuildType-N ,

где: ProjectStrID – строковый идентификатор проекта, задается пользователем при создании проекта; BuildType – тип сборки, debug или release; N – уникальный порядковый номер сборки.

Примеры имен каталогов сборки проекта:

usb3-debug-14

sar-release-25

В каталоге сборки должен находиться файл build.xml, содержащий всю необходимую информацию о сборке, а именно:

* номер сборки
* тип сборки
* дата и время сборки
* имя пользователя выполнившего сборку
* имя рабочей станции, на которой выполнена сборка
* перечень всех подкаталогов и файлов, входящих в сборку. Для каждого файла должен быть рассчитан и записан код MD5 и размер файла

Файл build.xml должен начинаться с стандартного XML-заголовка:

<?xml version="1.0" encoding="UTF-8"?>

За ним следует тело документа, начинающееся с тега <build>. Атрибуты тега <<build>>:

id – идентификатора (порядковый номер сборки)

type – тип сборки, debug или release

date – дата сборки

time – время сборки

user – имя пользователя

workstation – имя рабочей станции

Далее следует перечень подкаталогов и файлов сборки. Подкаталоги описываются тегом <directory>. Тег имеет один атрибут name – имя подкаталога.

Файлы описываются тегом <file>. Атрибуты тега:

name – имя файла

size – размер файла в байтах

md5 – контрольная сумма MD5 файла

Все видеокадры технологического процесса должны быть записаны в подкаталог WorkflowSchemes.

Все видеокадры диагностики должны быть записаны в подкаталог DiagSchemes.

Файлы, необходимые для прошивки модулей FSC и/или конфигурирования ПО SCADA должны быть размещены в подкаталогах, имя которых соответствует строковому идентификатору шкафа.

Пример файла build.xml:

<?xml version="1.0" encoding="UTF-8"?>

<build id="23" type="debug" date="01.01.2015" time="12:02:18" user="Malokhatko" workstation="ws1">

<directory name="server1">

<file name="diagservice.cfg" size="1234" md5="23829342934928734928734928734" />

<file name="dataservice.cfg" size="2344" md5="23123123934928734928734928734" />

</directory>

<directory name="shfs1">

<file name="lm1.cfg" size="45674" md5="23829342934928734928734928734" />

<file name="lm1.alg" size="12344" md5="23829342934928734928734928734" />

<file name="lm2.cfg" size="11112" md5="23829342934928734928734928734" />

<file name="lm2.alg" size="1234" md5="23829342934928734928734928734" />

</directory>

<directory name="WorkflowSchemes">

<file name="wscheme1.wvs" size="4322" md5="23829342934928734928734928734" />

<file name="wscheme2.wvs" size="5432" md5="78987987198792873981792873981" />

</directory>

<directory name="DiagSchemes">

<file name="dscheme1.dws" size="2334" md5="23423423423423423423423423445" />

<file name="dscheme2.dws" size="1554" md5="54646085309582039412476827637" />

</directory>

</build>

Более подробный состав файлов сборки и их формат определяются на стадии разработки ПО RPCT.

### SCADA Configuration Editor Module

Модуль редактирования конфигурации SCADA предназначен для определения состава и настроек аппаратных и программных средств SCADA.

#### Конфигуратор аппаратных средств SCADA

Конфигуратор аппаратных средства SCADA должен определять состав и идентификаторы компьютеров (серверов и рабочих станций) осуществляющих прием данных от FSC, архивирование и визуализацию этих данных.

Для каждого компьютера SCADA должны задаваться следующие параметры:

- наименование

- имя компьютера в сети (строковый идентификатор)

- перечень IP-адресов с масками подсети

#### Конфигуратор программных средств SCADA

Конфигуратор программных средств SCADA должен определять состав и настройки программного обеспечения для каждого компьютера в системе SCADA.

Для каждой программы SCADA должны определяться следующие параметры:

- строковый идентификатор программного обеспечения

- тип программы

- минимальная версия

- максимальная версия

- индивидуальные настройки программы в зависимости от типа

##### Настройки программы Data Acquisition Service

Для программы Data Acquisition Service необходимо задание следующих настроек:

- IP-адреса и порты интерфейсов приема данных от FSC

### Firmware Uploader Module

# SCADA Software Architecture

Overview, Srver to Client connections etc

## SCADA Server Software Architecture

Overview, all services, all uses TCP IP or UDP, etc

### Base Service Architecture

Base Service should implement fundamental base functions of RPCT SCADA services.

Базовый Сервис должен реализовать функционал программ-сервисов RPCT SCADA (далее - сервисов) перечисленный ниже. Базовый Сервис послужит основой для следующих сервисов:

* Configuration Service
* FSC Data Acquisition Service
* FSC Tuning Service
* Data Archiving Service

Функционал, индивидуальный для каждого сервиса, и не реализованный в Базовом Сервисе называется Основной Функцией Сервиса (Service Main Function - SMF).

Функционал Базового Сервиса включает в себя:

* инсталляцию, запуск, остановку и деинсталляцию сервисов в зависимости от аргументов командной строки
* базовые функции сетевого обмена
* обработка общих запросов предоставления информации о сервисах
* передачу/прием файлов по сети
* прием и отработку запросов на остановку, запуск и перезапуск выполнения основной функции сервиса
* ведение протокола (логирование) работы сервисов

#### Функции инсталляции, запуска, остановки и деинсталляции сервисов

Базовый Сервис должен обеспечивать запуск сервисов в виде служб Windows или демонов в Unix-системах.

Базовый Сервис должен обеспечивать общие для всех сервисов действия в зависимости от аргументов командной строки.

Таблица 1. Базовые действия сервиса в зависимости от аргументов командной строки

|  |  |
| --- | --- |
| Аргумент | Действие |
| -h | Вывод справки о доступных аргументах (ключах) командной строки |
| без аргументов | Запуск сервиса |
| -t | Остановка сервиса |
| -i | Инсталляция сервиса в системе. |
| -u | Деинсталляция сервиса |
| -v | Вывод информации о версии и состоянии сервиса |
| -e | Запуск сервиса в режиме обычного приложения |

#### Функции сетевого обмена

В рамках Базового Сервиса должны быть реализованы функции сетевого обмена построенного по принципу клиент-сервер. Сетевой обмен должен осуществляться путем передачи UDP-дейтаграмм.

Клиент – программа, которая является инициатором обмена. Клиент отправляет программе-серверу UDP-дейтаграмму, которая называется «запросом». В ответ Клиент ожидает получить UDP-дейтаграмму, которая называется «ответ». Учитывая ненадежность протокола UDP, клиент должен отслеживать таймаут получения ответа. Если Клиент не получил ответ в заданное время, он должен повторить запрос.

Сервер – программа, которая принимает запросы от Клиента, обрабатывает их и выдает соответствующие ответы.

Все «запросы» и «ответы» должны начинаться со стандартного заголовка (REQUEST\_HEADER), формат которого приведен в таблице 2.

Таблица 1. Формат стандартного заголовка запросов и ответов – REQUEST\_HEADER

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| ID | DWORD | 4 | Идентификатор запроса. Константы RQID\_\* |
| ClientID | DWORD | 4 | Уникальный идентификатор клиента. Должен быть неизменным на протяжении всего сеанса связи между клиентом и сервером. (Возможно, посчитать как хеш от GUID, генерируемого на старте каждой программы) |
| Version | DWORD | 4 | Версия протокола |
| No | DWORD | 4 | Последовательный нумератор запросов. Для каждого нового запроса клиент увеличивает значение этого поля на 1. В ответах – поле имеет тоже значение, что и в запросе. |
| ErrorCode | DWORD | 4 | Код ошибки. В запросах всегда равен 0. В ответах: 0 – нет ошибки, !0 – код ошибки индивидуальный для каждого запроса, константы RQERR\_\* |
| DataLen | DWORD | 4 | Длина секции данных запроса, без учета размера этого заголовка. |

Как видно, все запросы от одного клиента идентифицируются полем ClientID. Для обеспечения параллельной обработки запросов Сервер должен создавать отдельный поток обработки запросов для каждого клиента. Если программа Клиент не присылает запросы в течении 15 секунд – поток-обработчик должен завершаться. Коммуникация между потоком приема пакетов и потоком обработки должна осуществляться через очереди запросов (и, возможно, ответов).

#### Функции приема общих для всех сервисов запросов

Базовый Сервис должен обеспечивать прием общих для всех сервисов запросов от Service Control Manager. Распределение портов приема запросов от Service Control Manager приведено в таблице 2. Номер порта приема базовых запросов должен передаваться в качестве параметра библиотеки.

Таблица 2. Порты приема запросов от Service Control Manager.

|  |  |
| --- | --- |
| Сервис | Порт |
| BaseService | 13300 |
| Configuration Service | 13310 |
| FSC Data Acquisition Service | 13320 |
| FSC Tuning Service | 13330 |
| Data Archiving Service | 13340 |

Базовый Сервис должен обеспечивать прием, обработку и выдачу ответа на запросы, перечисленные в таблице 3.

Таблица 3.

|  |  |  |
| --- | --- | --- |
| Идентификатор запроса | Значение | Описание |
| RQID\_GET\_SERVICE\_INFO | 1000 | Запрос общей информации о сервисе |
| RQID\_SERVICE\_MF\_START | 1100 | Запустить выполнение основной функции сервиса |
| RQID\_SERVICE\_MF \_STOP | 1101 | Остановить выполнение основной функции сервиса |
| RQID\_SERVICE\_MF \_RESTART | 1102 | Перезапустить выполнение основной функции сервиса |
| RQID\_SEND\_FILE\_START | 1200 | Запрос начала передачи файла |
| RQID\_SEND\_FILE\_NEXT | 1201 | Запрос передачи очередной части файла |

Ниже описаны форматы всех запросов и ответов, которые должна обрабатывать базовая библиотека сервисов.

##### Запрос RQID\_GET\_SERVICE\_INFO

Запрос общей информации о сервисе.

Формат запроса – REQUEST\_HEADER, где ID = RQID\_GET\_SERVICE\_INFO.

Формат ответа:

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Header | REQUEST\_HEADER |  | Стандартный заголовок |
| ServiceType | DWORD | 4 | Тип сервиса, константы:  RQSTP\_CONFIG = 1,  RQSTP\_FSC\_AQUISION = 2,  RQSTP\_FSC\_TUNING = 3,  RQSTP\_ARCHIVING = 4 |
| MajorVersion | DWORD | 4 | Мажорная версия |
| MinorVersion | DWORD | 4 | Минорная версия |
| BuildNo | DWORD | 4 | Номер сборки программы |
| CRC | DWORD | 4 | Контрольная сумма исполнимого файла |
| ServiceUptime | DWORD | 4 | Время работы сервиса в секундах от запуска |
| State | DWORD | 4 | Текущее состояние выполнения основной функции сервиса, константы:  SST\_MF\_STOPPED = 0,  SST\_MF\_STARTS = 1,  SST\_MF\_WORK = 2,  SST\_MF\_STOPS = 3 |
| ServiceFunctionUptime | DWORD | 4 | Время работы основной функции сервиса в секундах от последнего запуска |

##### Запрос RQID\_SEND\_FILE\_START

Запрос начала передачи файла. Используется совместно с запросом RQID\_SEND\_FILE\_NEXT для передачи файлов между программами. Допускается параллельная передача нескольких файлов.

Формат запроса:

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Header | REQUEST\_HEADER |  | Стандартный заголовок |
| FileName | TCHAR[64] | 128 | Имя файла. 64 символа Unicode |
| FileSize | DWORD | 4 | Длина файла, байт |

Формат ответа:

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Header | REQUEST\_HEADER |  | Стандартный заголовок |
| FileID | DWORD | 4 | Уникальный идентификатор файла. Используется в последующих запросах RQID\_SEND\_FILE\_NEXT. |

Начало приема файла логируется.

##### Запрос RQID\_SEND\_FILE\_NEXT

Запрос передачи части файла. Используется совместно с запросом RQID\_SEND\_FILE\_START для передачи файлов между программами.

Формат запроса:

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Header | REQUEST\_HEADER |  | Стандартный заголовок |
| FileID | DWORD | 4 | Уникальный идентификатор файла из ответа на запрос RQID\_SEND\_FILE\_START |
| PartNo | DWORD | 4 | Текущий номер части файла, с 0 |
| PartCount | DWORD | 4 | Количество частей всего |
| DataSize | DWORD | 4 | Длина данных, байт |
| FileCRC32 | DWORD | 4 | Контрольная сумма файла, CRC32. Записывается текущее значение CRC для всех переданных до этого данных. |
| FileData | BYTE[DataSize] | DataSize | Данные файла. Массив байтов длиной DataSize |

Ответ - REQUEST\_HEADER.

Окончание приема файла логируется.

##### Запрос RQID\_SERVICE\_MF\_START

Запрос на запуск выполнения основной функции сервиса. Если сервис остановлен – будет осуществлен запуск выполнения основной функции. Если сервис работает – никаких действий не выполняется.

Запрос - REQUEST\_HEADER

Ответ - REQUEST\_HEADER

Запрос логируется.

##### Запрос RQID\_SERVICE\_MF\_STOP

Запрос на остановку выполнения основной функции сервиса. Если сервис работает - остановка выполняется в штатном режиме, как при завершении сервиса. Если сервис уже остановлен – никаких действий не выполняется.

Запрос - REQUEST\_HEADER

Ответ - REQUEST\_HEADER

Запрос логируется.

##### Запрос RQID\_SERVICE\_MF\_RESTART

Запрос на перезапуск выполнения основной функции сервиса. Если сервис работает - выполняется остановка в штатном режиме, а затем запуск выполнения основной функции. Если сервис уже остановлен – выполняется запуск.

Запрос - REQUEST\_HEADER

Ответ - REQUEST\_HEADER

Запрос логируется.

#### Функции протоколирования (логирования) работы сервисов

В Базовом Сервисе должны быть реализованы функции ведения протоколов работы сервисов.

Протокол – один или несколько текстовых файлов, в которые записывается важная информация о работе сервиса (программы). Каждая запись представляет собой отдельную строку файла, содержащую поля, разделенные точкой с запятой.

Таблица Перечень полей

|  |  |
| --- | --- |
| Поле | Описание |
| Источник | NRQ – запрос по сети  APP – программа  USR – действия пользователя |
| Тип | ERR – ошибка  WRN – предупреждение  MSG – сообщение |
| Дата | Дата записи по системному времени компьютера. Формат – ГГГГ.ММ.ДД |
| Время | Время записи по системному времени компьютера. Формат – ЧЧ:ММ:СС.ЛЛЛ |
| Сообщение | Произвольная строка БЕЗ символов точка с запятой. |
| Функция | Опционально – имя функции, из которой была произведена запись лога |
| Файл | Опционально – имя файла и строка, из которого была произведена запись |

Ведение логов должно выполняться отдельным потоком. Модуль ведения логов должен получать в качестве настроек следующие параметры:

* имя файла лога
* максимальная длина файла лога
* количество лог-файлов

Когда последний лог-файл достигает максимального размера, самый старый лог удаляется, и создается новый лог-файл.

Имя лог-файла должно иметь формат – FILE\_NNN.log, где FILE – имя лог-файла из настроек библиотеки, NNN – порядковый номер файла от 000 до 999.

### Configuration Service Architecture

### FSC Data Acquisition Service Architecture

Сервис предназначен для:

1. приема данных от FSC
2. обработки этих данных от FCS
3. передачи информации о состоянии приема данных от источников клиентскому ПО
4. ведения динамической БД состояний сигналов
5. передачу значений сигналов для архивирования
6. передачу значений сигналов клиентскому ПО

#### Receiving data from FSC

Данные от FSC передаются в виде UDP-дейтаграмм.

#### Sending information about data receiving state to client software

Для передачи информации о состоянии приема данных от FSC сервис должен обеспечивать обработку следующих запросов:

Таблица 3.

|  |  |  |
| --- | --- | --- |
| Идентификатор запроса | Значение | Описание |
| RQID\_GET\_DATA\_SOURCES\_IDS | 1250 | Запрос идентификаторов источников приема данных |
| RQID\_GET\_DATA\_SOURCES\_INFO | 1251 | Запрос информации об источниках данных |
| RQID\_GET\_DATA\_SOURCES\_STATISTICS | 1252 | Запрос информации о состоянии приема данных от источников |

##### Запрос RQID\_GET\_DATA\_SOURCES\_IDS

Запрос идентификаторов источников данных.

Формат запроса - RQID\_GET\_DATA\_SOURCES\_IDS

Формат ответа на запрос RQID\_GET\_DATA\_SOURCES\_IDS:

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Header | REQUEST\_HEADER |  | Стандартный заголовок |
| IDcount | DWORD | 4 | Количество идентификаторов в массиве |
| IDarray | DWORD[IDcount] | 4 \* IDcount | Массив идентификаторов источников данных |

##### Запрос RQID\_GET\_DATA\_SOURCES\_INFO

Запрос статической информации об источниках данных.

Формат запроса RQID\_GET\_DATA\_SOURCES\_IDS

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Header | REQUEST\_HEADER |  | Стандартный заголовок |
| IDcount | DWORD | 4 | Количество идентификаторов в массиве |
| IDarray | DWORD[IDcount] | 4 \* IDcount | Массив идентификаторов источников данных, о которых запрашивается информация |

Формат ответа на запрос RQID\_GET\_DATA\_SOURCES\_IDS:

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Header | REQUEST\_HEADER |  | Стандартный заголовок |
| Count | DWORD | 4 | Количество структур DataSourceInfo в массвие |
| DataSourceInfo | DataSourceInfo [Count] | Sizeof(DataSourceInfot) \* Count | Массив структур DataSourceInfo |

Формат структуры DataSourceInfo

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Version | WORD | 2 | Версия структуры |
| Size | WORD | 2 | Размер структуры в байтах |
| ID | DWORD | 4 | Идентификатор источника данных |
| Name | TCHAR[32] | 32 \* 2 | Наименование источника |
| IP | DWORD | 4 | IP-адрес |
| Port | DWORD | 4 | Порт |
| PartCount | DWORD | 4 | Количество ожидаемых частей данных |

##### Запрос RQID\_GET\_DATA\_SOURCES\_STATISTICS

Запрос статической информации об источниках данных.

Формат запроса RQID\_GET\_DATA\_SOURCES\_STATISTICS

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Header | REQUEST\_HEADER |  | Стандартный заголовок |
| IDcount | DWORD | 4 | Количество идентификаторов в массиве |
| IDarray | DWORD[IDcount] | 4 \* IDcount | Массив идентификаторов источников данных, о которых запрашивается информация |

Формат ответа на запрос RQID\_GET\_DATA\_SOURCES\_STATISTICS

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Header | REQUEST\_HEADER |  | Стандартный заголовок |
| Count | DWORD | 4 | Количество структур DataSourceStatistics в массвие |
| DataSourceStatistics | DataSourceStatistics[Count] | Sizeof(DataSourceStatistics) \* Count | Массив структур DataSourceStatistics |

Формат структуры DataSourceStatistics

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Длина, байт | Описание |
| Version | WORD | 2 | Версия структуры |
| Size | WORD | 2 | Размер структуры в байтах |
| Uptime | ULONGLONG | 8 | Время от начала приема данных, секунд |
| ReceivedSize | ULONGLONG | 8 | Объем принятых данных, байт |
| ReceiveSpeed | DOUBLE | 8 | Скорость приема данных, байт/с |
|  |  |  |  |
|  |  |  |  |

### FSC Tuning Service Architecture

### Data Archiving Service Architecture

### Service Control Manager Architecture

Service Control Manager (далее - SCM) - это программа, предназначенная для отображения состояния и управления сервисами SCADA (далее - сервисы). SCM должна предоставлять пользователю удобный графический интерфейс и реализовывать следующие функции:

Отображение состояния сервисов запущенных на локальном или удаленном компьютере.

Выдачу команды остановки сервиса

Выдачу команды запуска сервиса

Отображение состояния зависит от типа сервиса и будет уточняться в процессе разработки.

## SCADA Client Software Architecture

### Monitoring and Tuning Application Architecture

# General Requirements and Solutions

## Hardware and Software Platforms

Программное обеспечение RPCT предназначено для выполнения на компьютерах с процессорами архитектуры x86, x86-64 под управлением операционной системы Windows 7 и более поздних версий (добавить версию Linux).

Минимальный объем оперативной памяти для CAD - 8 Гб, для серверной части SCADA – 8 Гб, для клиентской части SCADA – 2-4 Гб.

Для запуска ПО RPCT необходимо установка динамических библиотек (dll) Qt 5.3.x и Visual Studio 2013 Redistributable Pack.

Для функционирования CAD необходима установка и конфигурация СУБД PostgreSQL 9.3 и драйвера Qt для доступа к БД PostgreSQL.

## Development tools, frameworks and libraries

C++, Qt, Qt Creator, PostgreSQL, Google Protobuf, Git, VS2013 compiler, GCC 4.8 tool chain.

‘FSC Product Architecture Document’ (D5.1) Contents Checklist

|  |  |
| --- | --- |
| Inspected Document  (title, D-number, ID, version) | FSC Product Architecture Document, D5.1,  version 1.3M |

|  |  |
| --- | --- |
| Person who performed check  (full name, position / role) | V.I. Tokarev,  Safety System Architect |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | April 22, 2014 |  | Conclusion | Document’s content meets all the requirements |  |

|  |  |
| --- | --- |
| **Document Body Text** | |
| ‘Scope and Purpose of the Document’ section clearly describes purpose of the document |  |
| Notes: | |
| ‘Reference Documents’ section doesn’t require changes and/or additions |  |
| Notes: | |
| All references in ‘Reference Documents’ section are reasonable |  |
| Notes: | |
| ‘Terms and Abbreviations’ section doesn’t require changes and/or additions |  |
| Notes: | |
| ‘Safety architecture concept’ section is present |  |
| Notes: | |
| ‘Hardware – FSC Chassis and Modules’ section is present |  |
| ‘Hardware – FSC Chassis and Modules’ section clearly describes chassis configuration, as well as all the chassis external and internal interfaces |  |
| Notes: | |
| ‘Timing Diagrams and Working Cycles’ section is present |  |
| ‘Timing Diagrams and Working Cycles’ section clearly describes operation timing diagrams of FCS single channel configuration and timing characteristics calculation technique for the FSC-based applications |  |
| Notes: | |
| ‘HW Modules Description’ section is present |  |
| ‘HW Modules Description’ section clearly describes the following modules, including their interfaces, modes, and potential safety impact: Discrete Input Module , Discrete Output Module, Analog Input Module, Analog Output Module, Logic Module |  |
| Notes: | |
| ‘Unit Architecture Descriptions’ section is present |  |
| ‘Unit Architecture Descriptions’ section clearly describes the following units, including their use (and/or operation mode) and interfaces: discrete input unit 2, discrete output unit, analog-to-digital conversion unit, digital-to-analog conversion unit, optical transceiver, LVDS transceiver unit, LAN transceiver, CLK unit, indication board interface unit, EEPROM Unit, address unit, ASPI unit, power supply & watchdog unit, and bypass control unit, Safety Override Unit, FPGA Unit, Neutron Flux Measurement Unit, Discrete Input Unit 1 |  |
| Notes: | |
| ‘Functional Block Library’ section is present |  |
| ‘Functional Block Library’ section clearly describes structure and all the functions of Functional Block Library |  |
| Notes: | |
| **Appendixes** | |
| Appendix А is present |  |
| Appendix А doesn’t require changes and/or additions |  |
| Notes: | |
| Appendix B is present |  |
| Appendix B doesn’t require changes and/or additions |  |
| Notes: | |
| Appendix B is present |  |
| Appendix B doesn’t require changes and/or additions |  |
| Notes: | |

Note: mark  means “yes”, mark  - means “no”.