SCADA-АНАРЭС

Инструкция пользователя

АННОТАЦИЯ

Руководство программиста предназначено для работы с программным обеспечением SCADA-АНАРЭС и содержит сведения для проверки, обеспечения функционирования и настройки программ.

Настоящим документом следует руководствоваться при эксплуатации, тестировании и конфигурировании программного обеспечения SCADA-АНАРЭС.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Полное наименование разработки SCADA-АНАРЭС.

1.2. Взаимодействие с оборудованием SCADA-АНАРЭС осуществляется по интерфейсу Ethernet, а также по последовательному интерфейсу для контроллера местной сигнализации.

1.3. Функции SCADA-АНАРЭС:

- отображение информации о текущем состоянии коммутационных аппаратов на схеме;
- отображение измерений на схеме;
- отображение в журнале событий аварийной и предупредительной информации при отключении коммутационных аппаратов от защит;
- отображение аварийной и предупредительной информации по терминалам защит;
- управление коммутационными аппаратами;
- сохранение журнала событий в архив;
- ведение полного архива передаваемой и принимаемой информации;
- система сохранения диагностических сообщений;

.

2. СВЕДЕНИЯ О ФОРМЕ ПРЕДСТАВЛЕНИЯ ПРОГРАММ

2.1. SCADA-АНАРЭС поставляется в электронном виде

2.3. SCADA-АНАРЭС предназначено для работы под управлением операционной системы Windows.

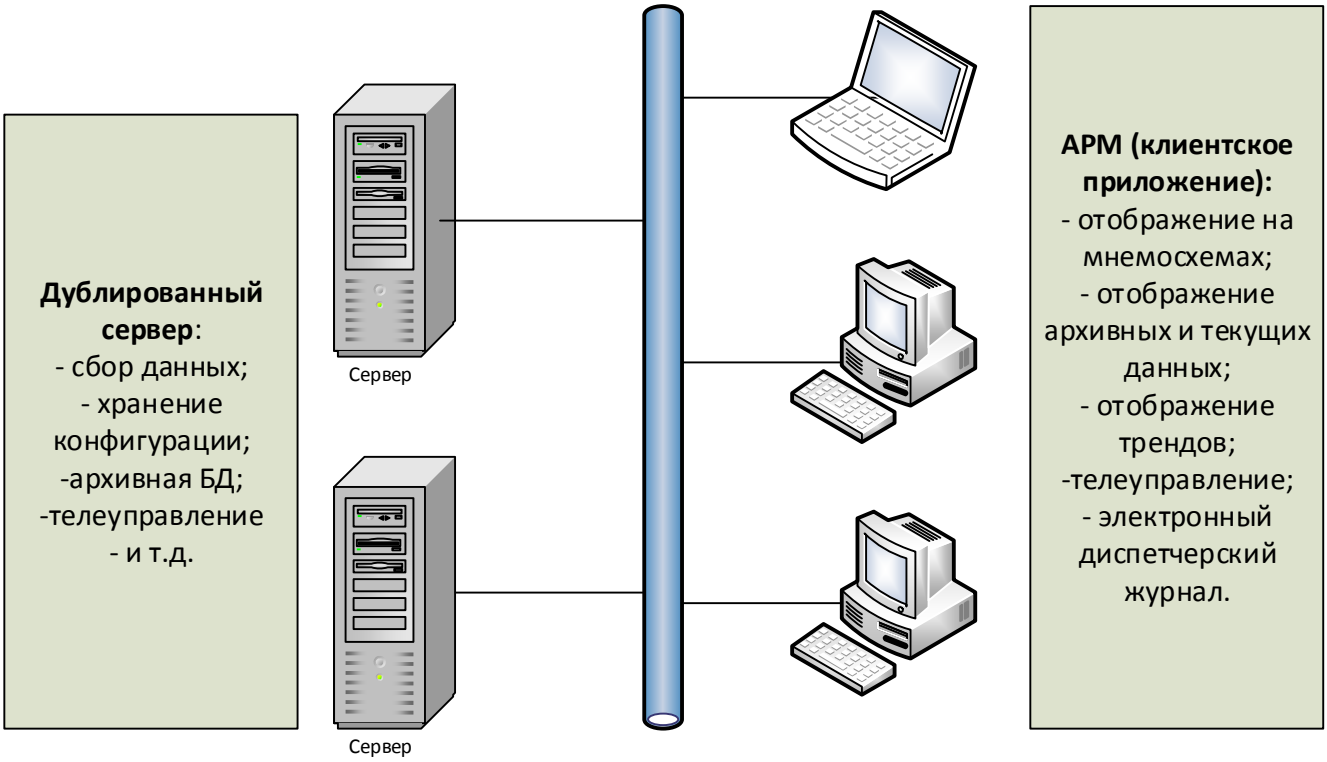
3. СТРУКТУРА SCADA-АНАРЭС

3.1. Клиент-серверная архитектура

3.1.1. SCADA-АНАРЭС выполнен по клиент-серверной архитектуре (рис. 1) и работает в тесном взаимодействии с инструментальным программно-инструментальным комплексом DicSys.

3.1.2. Клиентское приложение SCADA-АНАРЭС выполняет отображение информации на мнемосхемах; отображение архивных и текущих данных в табличном виде и в виде трендов; телеуправление; пользовательские диалоги; скриптовой язык.

3.1.3. Опционально SCADA-АНАРЭС может комплектоваться электронным диспетчерским журналом.



3.1.1. Упрощенная архитектура внутреннего устройства программного обеспечения SCADA-АНАРЭС представлена на рисунке 2.

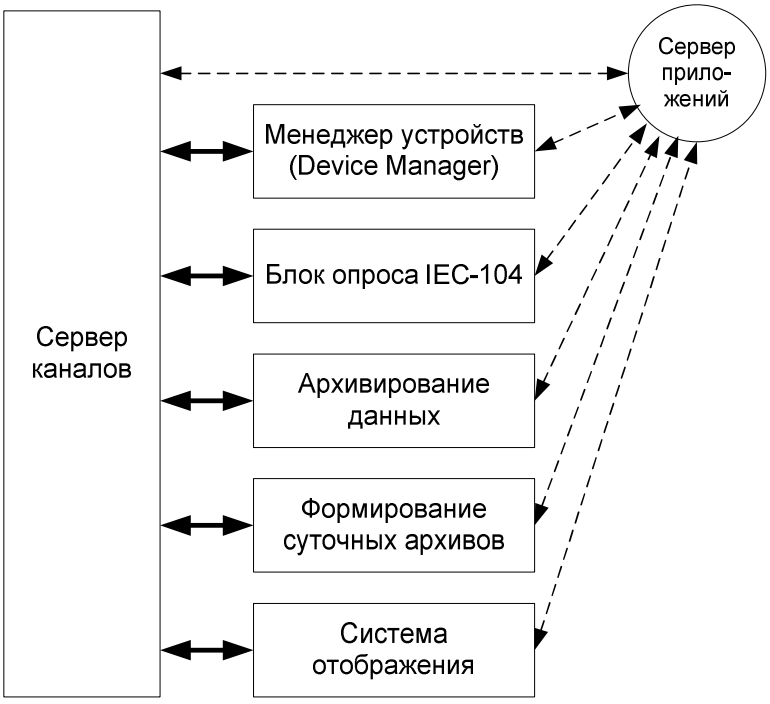


Рис.2. Структура ПО SCADA-АНАРЭС.

3.2. Сервер приложений SCADA-АНАРЭС

3.2.1. Сервер приложений предназначен:

- для обмена текстовыми сообщениями между отдельными программами (приложениями), входящими в состав SCADA-АНАРЭС;
- для запуска отдельных программ (приложений), входящими в состав SCADA-АНАРЭС;
- обмена данными между отдельными программами (приложениями), входящими в состав SCADA-АНАРЭС через механизм FileMapping (ОС Windows).

3.2.2. Сервер приложений состоит из следующих файлов:

- ServProg.exe – исполняемый программный модуль сервера приложений
- ServLib.dll – динамическая библиотека для взаимодействия отдельных программ (приложений), входящих в состав SCADA-АНАРЭС, с самим сервером приложений (более подробно о данной динамической библиотеке описано в разделе «Описание программных интерфейсов и библиотек для работы в составе SCADA-АНАРЭС»).
- ComServ.ini – текстовый файл для описания отдельных программ (приложений), входящих в состав SCADA-АНАРЭС.

3.2.3. Конфигурирование сервера приложений выполняется путем редактирования файла ComServ.ini.

3.2.4. Файл ComServ.ini является текстовым файлом в формате конфигурационных ini-файлов ОС Windows. Файл состоит из секций, название секции записано на отдельной строке и помещено в квадратные скобки, например: [ANARESChanelServer]

Где «ANARESChanelServer» – имя секции.

Для каждого из отдельных приложений SCADA-АНАРЭС, относящихся к ядру SCADA-АНАРЭС (сервер каналов) или функциональным задачам SCADA-АНАРЭС (RunTime) в файле ComServ.ini должна быть заведена отдельная секция.

Внутри секции могут быть записаны переменные в виде: имя переменной = значение переменной

Например: ProcessPriority=256

Где «ProcessPriority» – имя переменной, «256» – значение переменной.

Каждая переменная и ее значение должны находиться на одной строке. Если в секции записано несколько переменных, то каждая из них должна быть записана на отдельной строке.

3.2.5. Для сервера приложений используются следующие переменные:

- ExeName – путь и имя исполняемого файла;
- ProcessPriority – приоритет процесса приложения;
- SessionFile – имя конфигурационного ini-файла (с путем), в котором описаны ссылки на данные (используется в ПВК АНАРЭС-2000 для хранения данных расчетных схем, не используется в SCADA-АНАРЭС);
- SessionPathParam – имя переменной в конфигурационном ini-файле (см. предыдущий пункт про SessionFile), где указан каталог с данными (используется в ПВК АНАРЭС-2000 для хранения данных расчетных схем, не используется в SCADA-АНАРЭС);

- AddPath – имя подкаталога с файлами данных (см. предыдущий пункт про SessionPathParam), если файлы данных находятся в подкаталоге (используется в ПВК АНАРЭС-2000 для хранения данных расчетных схем, не используется в SCADA-АНАРЭС);

Другие переменные, если они будут записаны в ComServ.ini не будут использоваться сервером приложений.

3.2.6. Пример файла ComServ.ini:

```
[ANARESChanelServer]
ExeName=d:\#IGES_CH\ANARESChanelServer.exe
ProcessPriority=256

[Selector]
ExeName=d:\#IGES_CH\Selector.exe
ProcessPriority=128

[Arbitr]
ExeName=d:\#IGES_CH\Arbitr.exe
ProcessPriority=128

[SocketClnt]
ExeName=d:\#IGES_CH\SocketClnt.exe
ProcessPriority=128

[SocketSrv]
ExeName=d:\#IGES_CH\SocketSrv.exe
ProcessPriority=128

[ANARESChanelRandom]
ExeName=d:\#IGES_CH\ChanelRandom.exe
ProcessPriority=128

[CSInput]
ExeName=d:\#IGES_CH\CSInput.exe
ProcessPriority=128

[UserLogin]
ExeName=d:\#IGES_CH\UserLogin.exe
```

```
ProcessPriority=64

[CSArchiver]
ExeName=d:\#IGES_CH\CSArchiver.exe
ProcessPriority=128

[MakeArcFile]
ExeName=d:\#IGES_CH\MakeArcFile.exe
ProcessPriority=128

[Viewer]
ExeName=d:\#IGES_CH\ViewDisp.exe
ProcessPriority=128

[EmulControl]
ExeName=d:\#IGES_CH\EmulControl.exe
ProcessPriority=128

[Trends]
ExeName=d:\#IGES_CH\Trends.exe
ProcessPriority=128

[MSSQLArch]
ExeName=d:\#IGES_CH\MSSQLArch.exe
ProcessPriority=128
```

3.2.7. Подробнее см. раздел «Конфигурирование SCADA-АНАРЭС».

3.3. Работа сервера приложений

3.3.1. При запуске SCADA-АНАРЭС (подробнее см. раздел «Запуск и останов SCADA-АНАРЭС») или при запуске любого из отдельных приложений SCADA-АНАРЭС, относящихся к ядру SCADA-АНАРЭС (сервер каналов) или функциональным задачам SCADA-АНАРЭС (RunTime) автоматически будет запущен сервер приложений.

3.3.2. При завершении всех отдельных приложений SCADA-АНАРЭС, относящихся к ядру SCADA-АНАРЭС (сервер каналов) или функциональным задачам SCADA-АНАРЭС (RunTime) автоматически будет выгружен и сервер приложений.

3.3.2. Ручной запуск сервера приложений не приведет к запуску SCADA-АНАРЭС, при этом сервер приложений будет выгружен.

3.3.2. Попытка повторного запуска сервера приложений (при работающей SCADA-АНАРЭС) не приведет к запуску дубликата, т.к. сервер приложений содержит механизм блокировки двойного запуска.

3.4. Сервер каналов SCADA-АНАРЭС

3.4.1. Сервер каналов предназначен:

- обеспечивает обмена информацией (ТС, ТИ, сообщения) между различными функциональными блоками SCADA-АНАРЭС, установленными на одном компьютере;
- для запуска серверов данных (отдельных программ (приложений), входящими в состав SCADA-АНАРЭС), путем выдачи команд через сервер приложений SCADA-АНАРЭС;

3.4.2. Сервер приложений состоит из следующих файлов:

- ANARESChanelServer.exe – исполняемый программный модуль сервера каналов
- ANARESChanelServer.ini – текстовый файл для описания конфигурации сервера каналов
- ts.ini – текстовый файл для описания перечня телесигналов (ТС)
- ti.ini – текстовый файл для описания перечня телеизмерений (ТИ)
- Mes.ini – текстовый файл для описания перечня сообщений.

3.4.3. Информация, передаваемая через сервер каналов SCADA-АНАРЭС, подразделяется на три вида:

1. Массив телесигналов (ТС);
2. Массив телеизмерений (ТИ);
3. Очередь сообщений.

3.4.4. С точки зрения взаимодействия с сервером каналов любой функциональный блок может быть реализован в виде одного из трех типов:

1. Сервер данных;
2. Клиент данных;
3. Клиент и сервер данных.

3.4.5. Предназначение массива ТС и массива ТИ состоит в передаче между функциональными блоками текущих значений неких параметров, при этом расположение конкретного параметра в оперативной памяти не меняется во время работы системы, что упрощает обработку ТИ и ТС.

3.4.6. Предназначение очереди сообщений состоит в передаче между функциональными блоками информации по мере ее возникновения (по событиям, предупреждения, по мере выполнения определенных операций и др.).

3.4.7. Сервер данных – это программный блок, передающий в сервер каналов фрагмент массива ТС и ТИ, а также сообщения. Сервер данных может получать данные от внешних источников: каналы и протоколы обмена с внешними подсистемами, базы данных и т.п.

3.4.8. Клиент данных – это программный блок, получающий от сервера каналов весь массив или фрагмент массива ТС и ТИ, а также все или выборочные сообщения. Клиент данных может передавать данные во внешние источники: каналы и протоколы обмена с внешними подсистемами, базы данных и т.п., визуализировать данные для пользователя, выдавать значения в порты ввода/вывода.

3.4.9. Клиент и сервер данных – это программный блок, сочетающий в себе функции сервера и клиента данных. Например, такой блок может получать от сервера каналов некие данные, обрабатывает их и записывает их опять в сервер каналов, но под другими уникальными номерами.

3.4.10. Перечень ТС и ТИ, а также распределение фрагментов массива ТС и ТИ между различными серверами определяется на этапе конфигурирования сервера каналов.

3.4.11. Синхронизация программных блоков, обмен данными и командами в SCADA-АНАРЭС выполняется с помощью базовых объектов ядра ОС Windows: Mutex, Event, FileMapping. Это обеспечивает высокую производительность, высокую надежность, независимость от конкретной версии ОС и системных библиотек Windows (т.к. эти объекты реализованы во всех версиях Windows).

3.4.12. Для обеспечения надежности (защищенности) архитектура обмена данными между сервером каналов, серверами данных и клиентами данных выполнена по следующей схеме (см. рис. 3).

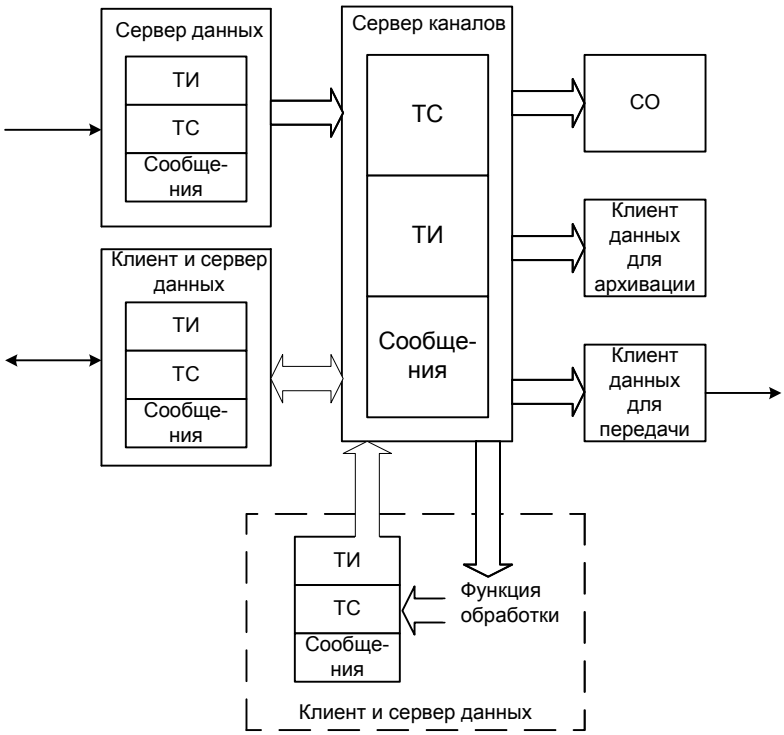


Рис. 3. Архитектура обмена информации между программными блоками.

3.4.13. Каждый сервер данных имеет свой массив ТИ и ТС и очередь сообщений. Записывать данные в локальные массивы сервера данных имеет возможность только сам сервер данных. Считывать же эту информацию может только сервер каналов, который записывает ее в свои массивы ТИ и ТС и очередь сообщений, эти блоки данных, в свою очередь, не доступны для записи извне. Клиенты данных могут в режиме только для чтения получать доступ к данным сервера каналов. Таким образом, ошибка в каком-то функциональном блоке не может привести к порче данных других блоков.

3.4.14. Конфигурирование сервера каналов выполняется путем редактирования файла ANARESChanelServer.ini.

3.4.15. Файл ANARESChanelServer.ini является текстовым файлом в формате конфигурационных ini-файлов ОС Windows. Подробнее см. раздел «Конфигурирование SCADA-АНАРЭС».

3.5. Работа сервера каналов

3.5.1. При запуске SCADA-АНАРЭС (подробнее см. раздел «Запуск и останов SCADA-АНАРЭС») или при запуске любого из отдельных приложений SCADA-АНАРЭС, относящихся к ядру SCADA-АНАРЭС (сервер каналов) или функциональным задачам SCADA-АНАРЭС (RunTime) автоматически, с помощью сервера приложений, будет запущен сервер каналов.

3.5.2. Сервер каналов автоматически проверяет работоспособность всех серверов данных, если какой-то сервер данных не запущен, то сервер каналов через сервер приложений выдает команду на запуск этого сервера данных. По этой же технологии выполняется первоначальный запуск SCADA-АНАРЭС, т.е. после запуска сервера каналов, он запускает все сервера данных.

3.5.3. Все функциональные задачи SCADA-АНАРЭС (RunTime) реализуются либо как сервера данных, либо клиент и сервер данных. Это необходимо для обеспечения автоматического запуска приложения и контроля его работоспособности. В качестве клиента данных реализуются только приложения, запуск и останов которых выполняет пользователь самостоятельно.

3.5.4. Завершение работы SCADA-АНАРЭС состоит в выдачи через сервер приложений команды серверу каналов на останов SCADA-АНАРЭС. Сервер каналов после получения команды на останов SCADA-АНАРЭС выдает через сервер приложений команды всем серверам данных на их останов. После завершения работы всех серверов данных автоматически завершается сервер каналов.

3.5.5. Ручной запуск сервера каналов приведет к запуску SCADA-АНАРЭС.

3.5.6. Попытка повторного запуска сервера каналов (при работающей SCADA-АНАРЭС) не приведет к запуску дубликата, т.к. сервер приложений содержит механизм блокировки двойного запуска.

3.6. Отслеживание состояния серверов данных

3.6.1. В сервере каналов имеется диапазон (блок) ТИ (в самом начале), каждый из ТИ определяет состояние отдельной программы (сервера каналов или серверов данных).

Начало диапазона определяется переменной BegStateTI, в секции [common] в файле ANARESChanelServer.ini.

3.6.2. ТИ с номером BegStateTI определяет состояние самого сервера каналов.

ТИ с номером BegStateTI+X означает состояние сервера [DataServX], определенного в файле ANARESChanelServer.ini.

3.6.3. Значение равное 0 (поле ТИ – Kvant), означает отсутствие каких-либо ошибок в работе этой программы. Значение этих битов определяет сервер каналов, на основании данных объектов ядра Windows или ответов от программы (SetCommandResult). Изначально значение ТИ устанавливается равным 7. После запуска сервера данных, сервер каналов начинает сбрасывать соответствующие биты.

3.6.4. Значение битов:

- 0 – установлен, если приложение не запущено (отсутствует Mutex приложения).
- 1 – сбрасывается после готовности сервера данных к полноценной работе (сбрасывается при получении ответа START, устанавливается при получении ответа STOP).

• 2 – установлен, если в приложении возникла критическая аварийная ситуация (отсутствует Event состояния приложения, либо если Event в сигнальном состоянии (SetEvent), сбрасывается, если Event в свободном состоянии (ResetEvent)).

• 3-31 – пользовательские биты, устанавливаются в серверах данных в соответствии с прикладным (пользовательским) алгоритмом (по ответу SetStateBitX, где X (от 3 до 31), сбрасываются по ответу ResetStateBitX).

3.7. Блок сервера устройств (DeviceManager)

3.7.1. Сервер DeviceManager предназначен:

- для взаимодействия с устройствами различных производителей;
- для чтения данных из устройств;

3.7.2. Сервер DeviceManager состоит из следующих файлов:

- DeviceManager.exe – собственно сам сервер
- DeviceManager.ini – текстовый файл для задания конфигурации сервера DeviceManager.
- Device.ini – конфигурация устройств взаимодействующих с DeviceManager.
- Dump_Viewer.exe – утилита просмотра файла с протоколом (дампом) обмена.

3.7.3. Основным понятием в сервере DeviceManager является шина (bus). Понятие шина представляет собой абстракцию реальной конфигурации промежуточных устройств связи и полевых шин. Шины организованы в виде иерархического дерева.

3.7.4. В сервере существуют одна виртуальная общая шина (RootBus) с которой взаимодействуют подчиненные. Шины чаще всего, организуются по принципу последовательного обмена. Т.е. если группа устройств соединена общей полевой шиной (например RS485) то обмен данными должен производиться последовательно с каждым устройством, такие устройства выделяются в общую шину и сервер обеспечивает их последовательный опрос.

3.7.5. Текущая версия DeviceManager позволяет вести обмен:

- Через порт компьютера RS-232 (485/422)
- Через Ethernet.

3.7.6. При обмене с устройствами данные, считываемые из устройства, могут быть записаны в ТИ и ТС, а также формироваться сообщения.

3.7.7. Информация по значениям дискретным и аналоговым входам, дискретным и аналоговым выходам, а также по внутренним параметрам устройств записывается в ТИ и ТС.

3.7.8. Информация по событиям, сформированным устройствами передается через сообщения в виде текстовых строк следующего формата:

- Device@Function@Block@Event Type@Event@State

Где:

- Device – идентификатор устройства (терминала РЗА)
- Function – идентификатор функции устройства («защита» в терминалах РЗА, например, «Токовые защиты»)
- Block – идентификатор блока функции устройства («ступень» в терминалах РЗА, например, «МТЗ 2 ступень»)
- Event Type – идентификатор типа события (например, «Диагностические», «Системные», «Срабатывание РЗ»)
- Event – идентификатор события («Пуск», «Отключение»)
- State – идентификатор состояния события (например, для события «Пуск» – это «Пуск реле», «Возврат реле», для события «Отключение» – это «Выдача сигнала на отключение», «Снятие сигнала на отключение»).

3.7.9. Конфигурирование сервера устройств выполняется путем редактирования файла DeviceManager.ini.

3.7.10. Файл DeviceManager.ini является текстовым файлом в формате конфигурационных ini-файлов ОС Windows. Файл состоит из секций, название секции записано на отдельной строке и помещено в квадратные скобки, например: [DataServer]

Где «[DataServer]» – имя секции.

3.7.11. Внутри секции могут быть записаны переменные в виде: имя переменной = значение переменной

Например: ProcessPriority=256

Где «MesFile» – имя переменной, «MesFileDeviceManager» – значение переменной.

3.7.12. Каждая переменная и ее значение должны находиться на одной строке. Если в секции записано несколько переменных, то каждая из них должна быть записана на отдельной строке.

3.7.13. Секции [DataServer] используются для взаимодействия с сервером каналов.

3.7.14. Секция [BusGroup] используется для задания структуры дерева Three=RootBus/Obj1,RootBus/Obj2,RootBus/Obj3 задает дерево с корневым узлом и тремя подчиненными узлами.

3.7.15. Каждый узел может содержать несколько шин. Например:

```
[Obj2]

Buses=Bus1,Bus2
```

3.7.15. В секции шины задается тип шины например Type=Serial232Common означает что шина использует 232 порт и является последовательной. В параметре Ports указаны параметры порта связи например Ports=com1(9600,7,NOPARITY,1).

3.7.16. Также указаны устройства, подключенные к шине Device=REF541_1,REF541_2.

Параметр Protocol=SPA задает используемый устройствами протокол.

3.7.17. В целях отладки при настройке комплекса можно указать параметр DumpFile=c:\dump_bus1.dmp в указанный файл будет сбрасываться весь обмен по шине. Для просмотра этого файла существует специальная утилита “dump_viewer.exe”.

3.7.18. Пример файла DeviceManager.ini:

```
[DataServer]
ServName=DeviceManager
TSFile=TSFileDeviceManager
TSCount=22
TSBeg=0
TIFile=TIFileDeviceManager
TICount=13
TIBeg=3
MesFile=MesFileDeviceManager
MesFileSize=1024
SleepTime=1000
MesTask=0
LockedTimeOut=50
ReadedTimeOut=500
CanUnRealMode=0

[ServerBlocksTS]
REF541_1=REF541_1

[ServerBlocksTI]
REF541_1=REF541_1

[BusGroup]
Three=RootBus/Obj1,RootBus/Obj2,RootBus/Obj3

[RootBus]

[Obj1]
Buses=Bus4
```

```
[Obj2]
Buses=Bus1,Bus2

[Obj3]

[Bus1]
DumpFile=c:\dump_bus1.dmp
Type=Serial232Common
Ports=com1(9600,7,NOPARITY,1)
Device=REF541_1,REF541_2
Protocol=SPA

[Bus4]
Device=Ret316_1
Protocol=SPA
Type=SerialEth
Socket=192.168.0.254(4001)

[Bus2]
Type=Serial232Common
Ports=com1(9600,7,NOPARITY,1)
Device=REF541_3,REF541_4
Protocol=SPA

[Bus3]
Type=Serial232Common
Ports=com2(9600,7,NOPARITY,1)
Device=ALPHA1200_1
Protocol=UNK

[ALPHA1200_1]
Address=1
Class=ABB
Type=ALPHA1200
```

```
[REF541_1]
Address=1
Class=ABB
Conversion= REF541_1.ini
Type=REF541

[REF541_2]
Address=2
Class=ABB
Conversion= REF541_2.ini
Type=REF541

[REF542_3]
Address=1
Class=ABB
Conversion= REF542_3.ini
Type=REF541

[REF542_4]
Address=1
Class=ABB
Conversion= REF542_3.ini
Type=REF541
```

3.7.19. Файл device.ini используется сервером DeviceManager в части связь между уровнем протокола и параметрами устройства в специальном формате, который может быть обработан верхним уровнем вне зависимости от конкретного типа устройства.

3.7.20. Пример файла Device.ini

```
[DeviceFunc]
Логика=Логика
Регистратор аварийных режимов=Регистратор аварийных режимов
Токовая функция=Токовая функция
```

```
Выдержка времени/интегрирующий усилитель=Выдержка времени/интегрирующий усилитель
ДЗТ=ДЗТ
Система=Система

[States]
Сброшено=Сброшено
Стоп=Стоп
Установлено=Установлено
Ошибка=Ошибка
Перезапуск=Перезапуск
Пуск=Пуск
Успешно=Успешно

[DeviceFuncBlock]
Логика=Логика
Регистратор аварийных режимов=Регистратор аварийных режимов
Токовая функция=Токовая функция
Выдержка времени/интегрирующий усилитель=Выдержка времени/интегрирующий усилитель
ДЗТ=ДЗТ
Система=Система

[Devices]
Ret316_1=reg

[Ret316_1]
Class=500
Object=2AT

[EventTypes]
Диагностика=Диагностика
Логика=Логика
Пуск=Пуск
Срабатывание=Срабатывание
```

```
[Events]
DitoReset_8E18=DitoReset_8E18
InitialisationSet_9E3=InitialisationSet_9E3
Protectionstopped_0E47=Protectionstopped_0E47
StartSet_26E3=StartSet_26E3
StartSet_31E3=StartSet_31E3
SupplyfaultCPUNo.4_1E44=SupplyfaultCPUNo.4_1E44
AC61OK_4E14=AC61OK_4E14
DB61failure_4E5=DB61failure_4E5
DitoReset_8E30=DitoReset_8E30
DitoReset_9E2=DitoReset_9E2
Coldprotectionstart_0E50=Coldprotectionstart_0E50
Noerror_0E1=Noerror_0E1
CPUfailure_3E2=CPUfailure_3E2
DitoReset_15E4=DitoReset_15E4
Warmprotectionstart_0E49=Warmprotectionstart_0E49


[DISPRet316_1]
Ret316_1@Система@Система@Диагностика@SupplyfaultCPUNo.4_1E44@Ошибка=Система
- Supply fault CPU No. 4
Ret316_1@Токовая                                              функция@Токовая
функция@Пуск@StartSet_25E3@Установлено=Токовая функция - Пуск установлено
Ret316_1@Токовая                                              функция@Токовая
функция@Пуск@StartSet_30E3@Установлено=Токовая функция - Пуск установлено
Ret316_1@ДЗТ@ДЗТ@Срабатывание@DitoReset_15E6@Сброшено=Диф.              защита
трансформатора - Сраб-В сброшено
Ret316_1@Система@Система@Диагностика@ADfaultCPUNo.2_1E12@Ошибка=Система  -  AD
fault CPU No. 2
```

3.8. Работа сервера DeviceManager

3.8.1. При запуске SCADA-АНАРЭС (подробнее см. раздел «Запуск и останов SCADA-АНАРЭС») или при запуске любого из отдельных приложений SCADA-АНАРЭС, относящихся к ядру SCADA-АНАРЭС (сервер каналов) или функциональным задачам SCADA-АНАРЭС (RunTime) автоматически будет запущен сервер DeviceManager.

3.8.2. При завершении всех отдельных приложений SCADA-АНАРЭС, относящихся к ядру SCADA-АНАРЭС (сервер каналов) или функциональным задачам SCADA-АНАРЭС (RunTime) автоматически будет выгружен и сервер DeviceManager.

3.8.3. Ручной запуск сервера DeviceManager не приведет к запуску SCADA-АНАРЭС, при этом сервер приложений будет выгружен.

3.8.4. Попытка повторного запуска сервера DeviceManager не приведет к запуску дубликата, т.к. сервер приложений содержит механизм блокировки двойного запуска.

3.9. Архиватор в текстовые файлы SCADA-АНАРЭС

3.9.1. Архиватор предназначен:

- для периодической фиксации изменений ТИ и ТС в сервере каналов.
- для фиксации сообщений сервера каналов.

3.9.2. Основное назначение архиватор в текстовые файлы – возможность работать при отсутствии SQL-сервера.

3.9.3. В проектах, где предусмотрен SQL-сервер, для архивации информации, архиватор в текстовые файлы может использоваться как резервное средство архивации.

3.9.4. Архиватор состоит из следующих файлов:

- CSArchiver.exe – исполняемый программный модуль архиватора
- CSArchiver.ini – текстовый файл для описания конфигурации архиватора
- CSArchiver.arc – текстовый файл – архив ТИ, ТС и сообщений
- TextMessage.arc – текстовый файл – архив текстовых сообщений

3.9.5. Архиватор выступает как клиент и сервер данных.

3.9.6. Как сервер данных архиватор может выдавать сообщения о нештатных ситуациях в архиваторе.

3.9.7. Как клиент данных архиватор считывает все ТИ и ТС. Затем поэлементно (каждый ТИ и каждый ТС) проверяется на отличие от предыдущего своего значения, если изменения есть, то в текстовый файл CSArchiver.arc в заданном формате будет записана запись со значением всех полей ТИ и ТС. В зависимости от параметра FixChangeData запись в архив ТИ и ТС будет выполняться не только при изменении значения ТИ/ТС, но и при изменении метки времени.

3.9.8. Также в архив CSArchiver.arc записываются значения всех сообщений сервера каналов.

3.9.9. Текстовые сообщения дополнительно записываются в архиве TextMessage.arc

3.9.10. Под текстовыми сообщениями принимаются сообщения следующих типов:

- ms_String=0 – строковое сообщение (просто общая информация)
- msAppError=1 – Ошибка в программе (приложении) сервере данных
- msAppWarning=2 – Предупреждение в программе (приложении) сервере данных
- msAlarm=3 – Аварийная сигнализация (технологическая сигнализация)
- msWarning=4 – Предупреждение (технологическая сигнализация)
- msInformation=5 – Информация (технологическая сигнализация)
- msControl=6 – Произведено управление (технологическая сигнализация)

3.9.11. При в файлах CSArchiver.arc и TextMessage.arc поля разделяются символом табуляции (код ASCII/ANSI – 9), а записи разделяются переводом строки.

3.9.12. Формат записи ТИ в файле CSArchiver.arc

- «TI» – признак записи по ТИ
- Метка времени – время формирования ТИ в сервере данных, в формате Windows FileTime (количество 100 наносекундных интервалов с 1 января 1601 года)
- Метка времени – время формирования ТИ в устройстве сбора данных, в формате Windows FileTime (количество 100 наносекундных интервалов с 1 января 1601 года)
- Номер ТИ в сервере каналов
- Текстовый идентификатор ТИ
- Время формирования ТИ в сервере данных в формате «dd.mm.yyyy hh:nn:ss.zzz»
- Уникальный номер ТИ
- Флаг ТИ
- Значение ТИ в квантах (целое число)
- Значение ТИ (вещественное число)
- «END» – признак конца записи

3.9.12. Формат записи ТС в файле CSArchiver.arc

- «TS» – признак записи по ТС
- Метка времени – время формирования ТС в сервере данных, в формате Windows FileTime (количество 100 нсек интервалов с 1 января 1601 года)
- Метка времени – время формирования ТС в устройстве сбора данных, в формате Windows FileTime (количество 100 нсек интервалов с 1 января 1601 года)
- Номер ТС
- Текстовый идентификатор ТС
- Время формирования ТС в сервере данных в формате «dd.mm.yyyy hh:nn:ss.zzz»
- Уникальный номер ТС
- Значение ТС
- «END» – признак конца записи

3.9.13. Формат записи сообщений в файле CSArchiver.arc

- «Message» – признак записи по сообщениям
- Метка времени – время формирования сообщения, в формате Windows FileTime (количество 100 нсек интервалов с 1 января 1601 года)
- Текстовый идентификатор сообщения
- Время формирования сообщения в сервере данных в формате «dd.mm.yyyy hh:nn:ss.zzz»
- Уникальный номер сообщения

- Задача сформировавшая сообщение
- Номер сообщения
- Размер данных сообщения
- Данные сообщения в формате 16-ричных байтов разделенных пробелами
- «END» – признак конца записи

3.9.14. Формат записи текстовых сообщений в файле TextMessage.arc

- Время формирования сообщения в сервере данных в формате «dd.mm.yyyy hh:nn:ss.zzz»
- символ «–»
- Строка сообщения

Где dd.mm.yyyy hh:nn:ss.zzz:

- yyyy – год (например, 2006)
- mm – месяц (например, 03)
- dd – день (например, 21)
- hh – час (например, 03)
- nn – минуты (например, 26)
- ss – секунды (например, 31)
- zzz – миллисекунды (например, 078)

3.9.15. При переходе через сутки архиватор переименовывает файлы текстовых архивов:

- Файл CSArchiver.arc в файл CSArchiver.arc_yyyymmdd
- Файл TextMessage.arc в файл TextMessage.arc_yyyymmdd

Где yyyymmdd:

- yyyy – год (например, 2006)
- mm – месяц (например, 03)
- dd – день (например, 21)

3.9.16. Потом создает пустые файлы: CSArchiver.arc и TextMessage.arc, и продолжает свою работу. Далее приложение «Формирование суточных архивов» может выполнить упаковку файлов (с помощью архиватора, например, WinRAR) за предыдущие сутки в отдельный каталог на диске.

3.9.17. Конфигурирование архиватора выполняется путем редактирования файла CSArchiver.ini.

3.9.18. Файл CSArchiver.ini является текстовым файлом в формате конфигурационных ini-файлов ОС Windows. Подробнее см. раздел «Конфигурирование SCADA-АНАРЭС».

3.9.19. Следует отметить, что запись изменений ТИ и ТС ведется с периодичностью, указанной в параметре RefreshTime (в мсек) в секции [Client].

3.9.20. Дополнительно в секции [CSArchiver] указаны параметры:

- FixChangeData – признак записи в архив ТИ и ТС, если не только изменилось значение ТИ/ТС, но и при изменении метки времени (FixChangeData=1). Если FixChangeData=0, то запись в архив ТИ и ТС, только в случае изменения значений ТИ/ТС.
- dFlushTime – время записи файловых буферов на диск, в мсек.

3.10. Формирование суточных архивов SCADA-АНАРЭС

3.10.1. Блок формирования суточных архивов предназначен:

- для запаковки суточных текстовых архивов (см. Архиватор в текстовые файлы) с помощью архиваторов (например, WinRAR).
- для контроля за свободным местом на диске, и удаления старых суточных архивов в случае уменьшения свободного места на диске меньше, чем объем заданный в настройках.

3.10.2. Блок формирования суточных архивов состоит из следующих файлов:

- MakeArcFile.exe – исполняемый программный модуль блока формирования суточных архивов
- MakeArcFile.ini – текстовый файл для описания конфигурации блока формирования суточных архивов

3.10.3. Блок формирования суточных архивов выступает как сервер данных.

3.10.4. Как сервер данных блок формирования суточных архивов может выдавать сообщения о нештатных ситуациях в архиваторе.

3.10.5. Блок формирования суточных архивов периодически проверяет наличие файлов текстовых архивов сформированных блоком «Архиватор в текстовые файлы»:

- Файл CSArchiver.arc_yyyymmdd
- Файл TextMessage.arc_yyyymmdd

Где yyyymmdd:

- yyyy – год (например, 2006)
- mm – месяц (например, 03)
- dd – день (например, 21)

3.10.6. При обнаружении указанных файлов, запускается архиватор (например, WinRAR) и запаковывает файл в каталог указанный в параметре OldArcPath. После успешной архивации исходный (не упакованный) файл удаляется.

3.10.7. При этом в каталоге указанном в параметре OldArcPath, будут сформированный файлы:

- Файл CSArchiver.arc_yyyymmdd.rar
- Файл TextMessage.arc_yyyymmdd.rar

3.10.8. Также блок формирования суточных архивов контролирует объем свободного места на диске, указанном в параметре OldArcPath. Если объем свободного места на диске будет меньше, чем указано в параметре SpaceLimit, то начнется удаление самых старых суточных архивов до тех пор, пока свободное место на диске не будет превышать значение, указанное в параметре SpaceLimit.

3.11. Конфигурирование блока формирования суточных архивов

3.11.1. Конфигурирование блока формирования суточных архивов выполняется путем редактирования файла MakeArcFile.ini.

3.11.2. Файл MakeArcFile.ini является текстовым файлом в формате конфигурационных ini-файлов ОС Windows. Подробнее см. раздел «Конфигурирование SCADA-АНАРЭС».

3.11.3. Дополнительно в секции [MakeArcFile] указаны параметры:

- OldArcPath – путь запакованных архивов
- SpaceLimit – минимальное место на диске, в случае достижения которого будут удаляться суточные архивы за самые старые сутки, в Гигабайтах.
- ExeName – путь исполняемого файла архиватора (например, C:\Program Files\WinRAR\WinRAR.exe)

3.12. Блок опроса по протоколу МЭК 60870-5-104 (IEC-104)

3.12.1. Блок IEC-104:

- для взаимодействия с устройствами различных производителей;
- для чтения данных из устройств;

3.12.2. Блок IEC-104 состоит из следующих файлов:

- iec104_clnt.exe – собственно сам блок опроса
- iec104_clnt.ini – текстовый файл для задания конфигурации.

3.12.3. Блок IEC-104 позволяет вести обмен информацией с устройствами по протоколу МЭК 60870-5-104 по сети Ethernet.

3.12.4. При обмене с устройствами данные, считываемые из устройства, могут быть записаны в ТИ и ТС, а также формироваться сообщения.

3.12.5. Информация по значениям дискретным и аналоговым входам, дискретным и аналоговым выходам, а также по внутренним параметрам устройств записывается в ТИ и ТС.

3.12.6. Информация по событиям, сформированным устройствами, передается также в виде текстовых сообщений с указанными в конфигурации признаками, например:

- msAlarm – аварийная сигнализация;
- msWarning – предупредительная сигнализация;
- msInformation – информация.

3.12.7. Конфигурирование блока выполняется путем редактирования файла iec104_clnt.ini или с помощью автоматического формирования этого файла блоком конфигурирования ChanelConfig

3.12.8. Файл iec104_clnt.ini является текстовым файлом в формате конфигурационных ini-файлов ОС Windows. Файл состоит из секций, название секции записано на отдельной строке и помещено в квадратные скобки, например: [DataServer]

Где «[DataServer]» – имя секции.

3.12.9. Внутри секции могут быть записаны переменные в виде: имя переменной = значение переменной

Например: ProcessPriority=256

Где «MesFile» – имя переменной, «MesFileiec104_clnt» – значение переменной.

3.12.10. Каждая переменная и ее значение должны находиться на одной строке. Если в секции записано несколько переменных, то каждая из них должна быть записана на отдельной строке.

3.12.11. В секции [main] задаются общие параметры протокола:

- t1_WaitResponse – таймаут ожидания ответа;
- t2_AcknolegeDelay – таймаут посылки подтверждения в случае отсутствия новых сообщений;
- t3_TestTimeOut – таймаут посылки сообщений для тестирования связи;
- ReadAllPeriod – период группового (полного) опроса устройства;
- WaitForInitTimeOut – таймаут ожидания перед инициализацией устройства после соединения;
- SleepTime – дискретность цикла опроса;
- QueueSize – размер очереди сообщений;
- TimeOut – таймаут при чтении из порта;
- ConnectTimeOut – таймаут ожидания соединения;
- Minimum_Acknolege_Count – количество подтверждаемых посылок.

3.12.12. В секции [TS] задаются ТС сервера каналов и соответствующий им адрес в устройстве в виде:

Идентификатор_ТС=Адрес_устройства.Адрес_объекта_информации

3.12.13. В секции [TS_ON_Msg] задаются сообщения, формируемые при изменении значения ТС на «Установленно» в виде соответствующего ТС сервера каналов и сообщения с типом:

Идентификатор_ТС=Тип_сообщения.Текст сообщения

3.12.14. В секции [TS_OFF_Msg] задаются сообщения, формируемые при изменении значения ТС на «Сброшено» в виде соответствующего ТС сервера каналов и сообщения с типом:

Идентификатор_ТС=Тип_сообщения.Текст сообщения

3.12.14. Пример файла DeviceManager.ini:

```
 [DataServer]
ServName=Iec104_clnt
TSFile=TSFileIec104_clnt
TSCount=213
TSBeg=384
TIFile=TIFileIec104_clnt
TICount=0
```

```
MesFile=MesFileIec104_clnt
MesFileSize=256
SleepTime=200
MesTask=4
LockedTimeOut=50
ReadedTimeOut=500
CanUnRealMode=0

[ServerBlocksTS]
ForLogic=ForLogic

[IPList]
172.18.241.55=55

[main]
t1_WaitResponse=15000
t2_AcknolegeDelay=10000
t3_TestTimeOut=30000
ReadAllPeriod=0
WaitForInitTimeOut=5000
SleepTime=100
QueueSize=65535
TimeOut=100
ConnectTimeOut=1000
Minimum_Acknolege_Count=1

[TS]
REL_511_--_Авария(1121115)=55.1121115
ГусГРЭС-ВЛ_--_REL_511_N5_--_Авария(1121301)=55.1121301

[TS_ON_Msg]
REL_511_--_Авария(1121115)=msAlarm. REL 511 -- Авария

[TS_OFF_Msg]
REL_511_--_Авария(1121115)=msInformation.REL 511 -- Авария - СНЯТА
```

3.13. Работа блока IEC-104

3.13.1. При запуске SCADA-АНАРЭС (подробнее см. раздел «Запуск и останов SCADA-АНАРЭС») или при запуске любого из отдельных приложений SCADA-АНАРЭС, относящихся к ядру SCADA-АНАРЭС (сервер каналов) или функциональным задачам SCADA-АНАРЭС (RunTime) автоматически будет запущен блок IEC-104.

3.13.2. При завершении всех отдельных приложений SCADA-АНАРЭС, относящихся к ядру SCADA-АНАРЭС (сервер каналов) или функциональным задачам SCADA-АНАРЭС (RunTime) автоматически будет выгружен и блок IEC-104.

3.13.3. Попытка повторного запуска блока IEC-104 не приведет к запуску дубликата, т.к. сервер приложений содержит механизм блокировки двойного запуска.

3.14. Графическая подсистема

3.14.1. Описание графической подсистемы приведено в приложениях 1,2.

4. КОНФИГУРИРОВАНИЕ

4.1. Конфигурирование SCADA-АНАРЭС

4.1.1. Адаптация программного обеспечения SCADA-АНАРЭС для использования на конкретном объекте выполняется с помощью конфигурирования.

4.1.2. В данном разделе описано конфигурирование SCADA-АНАРЭС. Конфигурирование включает в себя:

- выбор состава используемых задач;
- конфигурирование состава приложений предоставляющих или изменяющих данные (серверов данных);
- конфигурирование состава приложений получающих данные (клиентов данных);
- настройку индивидуальных параметров приложений входящих в состав комплекса;
- конфигурирование потоков данных между приложениями комплекса и состава передаваемой информации;
- подготовку мнемосхем для графического отображения информации.

4.1.2. Для хранения конфигурации SCADA-АНАРЭС используется набор текстовых файлов в формате конфигурационных ini-файлов ОС Windows:

- comserv.ini - содержит список автоматически запускаемых приложений комплекса с указанием их приоритетов
- ANARESChanelServer.ini - содержит общие настройки сервера каналов и список серверов и клиентов данных с параметрами блоков, относящихся к серверу каналов

- TS.ini - содержит список ТС с указанием их абсолютных номеров, блоков данных, которым они принадлежат, а также наименования ТС
- TI.ini - содержит список ТИ с указанием их абсолютных номеров, блоков данных, которым они принадлежат, а также наименования ТИ.
- Mes.ini - содержит список сообщений с указанием их уникальных номеров и список сообщений передаваемых по сети.

4.1.3. Заполнение и редактирование конфигурационных файлов может осуществляться, как вручную, так и с помощью конвертера конфигурации из таблиц MS Excel (ChanelConfig.exe).

4.1.4. Для подготовки графических мнемосхем используется редактор схем ПВК АНАРЭС-2000. Для привязки схем к информации, имеющейся в SCADA-АНАРЭС, используется адаптированная система отображения ПВК АНАРЭС-2000.

4.2. Конфигурирование сервера приложений

4.2.1. Конфигурирование сервера приложений выполняется путем редактирования файла ComServ.ini.

4.2.2. Файл состоит из секций, название секции записано на отдельной строке и помещено в квадратные скобки, например: [ANARESChanelServer].

4.2.3. Для каждого приложения SCADA-АНАРЭС, которое требует автоматического контроля сервером приложений в файле ComServ.ini должна быть заведена отдельная секция. В конфигурацию должны быть заведены все приложения относящихся к ядру SCADA-АНАРЭС (сервер каналов) или функциональным задачам SCADA-АНАРЭС (RunTime).

4.2.4. Внутри секции могут быть записаны переменные в виде: имя переменной = значение переменной

Например: ProcessPriority=256

Где «ProcessPriority» – имя переменной, «256» – значение переменной.

Каждая переменная и ее значение должны находиться на одной строке. Если в секции записано несколько переменных, то каждая из них должна быть записана на отдельной строке.

4.2.5. Для сервера приложений используются следующие переменные:

- **ExeName** – путь и имя исполняемого файла
- **ProcessPriority** – приоритет процесса приложения
- **SessionFile** – не используется в SCADA-АНАРЭС
- **SessionPathParam** – не используется в SCADA-АНАРЭС
- **AddPath** – не используется в SCADA-АНАРЭС

4.2.6. Другие переменные, если они будут записаны в ComServ.ini не будут использоваться сервером приложений.

4.3. Конфигурирование сервера каналов

4.3.1. Конфигурирование сервера каналов выполняется путем редактирования файла ANARESChanelServer.ini.

4.3.2. В секции [common] задаются общие настройки сервера каналов:

- **TSFile** – идентификатор блока дискретных данных (телесигналов), доступного для чтения клиентами данных (как правило, задается ANARESChanelTSFile);
- **TIFile** – идентификатор блока аналоговых данных (телеизмерений), доступного для чтения клиентами данных (как правило, задается ANARESChanelTIFile);
- **MesFile** – идентификатор очереди сообщений, доступной для чтения клиентами данных (как правило, задается ANARESChanelMesFile);
- **ServName** – идентификатор сервера каналов (как правило, задается ANARESChanelMesFile).
- **MesFileSize** – размер очереди сообщений в КБ.
- **MesTask** – цифровой идентификатор приложения (сервера каналов) в сообщениях;
- **SleepTime** – время паузы в основном цикле. Определяет скорость реакции сервера каналов;
- **SlowTimeOut** – время определения того, что клиент не успел считать изменившиеся данные и требуется перейти в медленный режим для обеспечения полного считывания данных клиентом;
- **LockedTimeOut** – время ожидания разблокирования читаемых клиентами данных для записи новых данных;
- **GetMutexTimeOut** – время ожидания получения идентификатора приложения от использующих сервер каналов приложений;
- **RunServerTimeOut** –время ожидания загрузки сервера данных;
- **AnarExit** – признак необходимости автоматически выгружать приложения при закрытии сервера каналов;
- **MesClientSize** – размер очередей сообщений для каждого клиента в КБ;
- **MainPath** – путь к головному каталогу SCADA-АНАРЭС
- **DataServCount** – полное количество серверов данных;
- **TSServCount** – количество серверов данных передающих дискретные сигналы (ТС);
- **TIServCount** – количество серверов данных передающих аналоговые сигналы (ТИ);
- **MesServCount** – количество серверов данных передающих сообщения;
- **BegStateTI** – начало блока статусных ТИ;
- **ClientCount** – количество клиентов данных.

4.3.3. Для каждого сервера данных должна быть заведена отдельная секция [DataServ<N>], где <N> порядковый номер сервера данных. В секции сервера данных задается:

- **Name** – наименование сервера данных;
- **ServerName** – идентификатор сервера данных;
- **LoadMode** – признак необходимости автоматической загрузки сервера данных;
- **ExitMode** – признак необходимости автоматической выгрузки сервера данных;
- **ExeName** – путь к исполняемому файлу.

4.3.4. Для каждого сервера данных передающего дискретную информацию (ТС) должна быть заведена отдельная секция [TIServ<N>], где <N> порядковый номер сервера данных. В секции задается:

- **ServerName** – идентификатор сервера данных;
- **FName** – имя блока дискретных данных передаваемых в сервер каналов;
- **TimeOut** – время ожидания возможности передачи данных в сервер каналов;
- **BegAddr** – начальный адрес блока;
- **Count** – количество ТС.

4.3.5. Для каждого сервера данных передающего аналоговую информацию (ТИ) должна быть заведена отдельная секция [TIServ<N>], где <N> порядковый номер сервера данных. В секции задается:

- **ServerName** – идентификатор сервера данных;
- **FName** – имя блока аналоговых данных передаваемых в сервер каналов;
- **TimeOut** – время ожидания возможности передачи данных в сервер каналов;
- **BegAddr** – начальный адрес блока;
- **Count** – количество ТИ.

4.3.6. Для каждого сервера данных передающего сообщения должна быть заведена отдельная секция [MesServ<N>], где <N> порядковый номер сервера данных. В секции задается:

- **ServerName** – идентификатор сервера данных;
- **FName** – имя очереди сообщений передаваемых в сервер каналов;
- **TimeOut** – время ожидания возможности передачи данных в сервер каналов.

4.3.7. Для каждого клиента данных должна быть заведена отдельная секция [Client<N>], где <N> порядковый номер клиента данных. В секции клиента данных задается:

- **Name** – наименование клиента данных;
- **TSEventName** – идентификатор события, передаваемого клиенту в случае изменения ТС;
- **TIEventName** – идентификатор события, передаваемого клиенту в случае изменения ТИ;
- **MesEventName** – идентификатор события, передаваемого клиенту в случае наличия новых сообщений;
- **MinRefreshTime** – минимальное время обновления данных оптимальное для клиента;
- **MaxRefreshTime** – максимально допустимое время обновления данных для клиента;
- **ExeName** – путь к исполняемому файлу.

4.4. Конфигурирование набора дискретных сигналов (ТС)

4.4.1. Набор дискретных сигналов, которыми оперирует SCADA-АНАРЭС, задается в конфигурационном файле TS.ini.

4.4.2. Все ТС разбиты на блоки для дальнейшего указания, какими блоками оперируют сервера и клиенты данных. Каждый ТС может входить одновременно в несколько блоков. Каждый ТС обязательно должен входить в блок совпадающий по названию с идентификатором сервера данных, записывающего этот ТС.

4.4.3. Каждый ТС задается идентификатором и соответствующим ему уникальным номером в секции соответствующей имени блока данных. Например:

[Имя блока]

Идентификатор ТС=Уникальный номер ТС

4.4.4. В отдельной секции [Names] задаются наименования ТС в виде:

Уникальный номер ТС=Наименование ТС

4.4.5. В секции [DataBlock] полный список блоков ТС с указанием уникального номера ТС начала блока и количества ТС в блоке. Например:

[DataBlock]

Имя_блока=Номер начального ТС в блоке

Имя_блока_Count=Количество ТС в блоке

4.5. Конфигурирование набора дискретных сигналов (ТИ)

4.5.1. Набор дискретных сигналов, которыми оперирует SCADA-АНАРЭС, задается в конфигурационном файле TI.ini.

4.5.2. Все ТИ разбиты на блоки для дальнейшего указания, какими блоками оперируют сервера и клиенты данных. Каждый ТИ может входить одновременно в несколько блоков. Каждый ТИ обязательно должен входить в блок совпадающий по названию с идентификатором сервера данных, записывающего этот ТИ.

4.5.3. Каждый ТИ задается идентификатором и соответствующим ему уникальным номером в секции соответствующей имени блока данных. Например:

[Имя блока]

Идентификатор ТИ=Уникальный номер ТИ

4.5.4. В отдельной секции [Names] задаются наименования ТИ в виде:

Уникальный номер ТИ=Наименование ТИ

4.5.5. В секции [DataBlock] полный список блоков ТИ с указанием уникального номера ТИ начала блока и количества ТИ в блоке. Например:

[DataBlock]

Имя_блока=Номер начального ТИ в блоке

Имя_блока_Count=Количество ТИ в блоке

4.6. Конфигурирование набора сообщений

4.6.1. Набор сообщений, которыми оперирует SCADA-АНАРЭС, задается в конфигурационном файле Mes.ini.

4.6.2. Каждое сообщение задается идентификатором типа сообщения и соответствующим ему уникальным номером типа сообщения в секции [Messages] в виде:

*[*Messages*]*

Идентификатор сообщения= Уникальный номер сообщения

4.6.3. В отдельной секции [Names] задаются наименования типа сообщений в виде:

[Names]

Уникальный номер сообщения=Наименование сообщения

4.6.4. В секции [Public] задаются сообщения, предназначенные для передачи по сети:

[Names]

Уникальный номер сообщения=Идентификатор сообщения

4.6.5. Набор сообщений определяется составом программных блоков SCADA-АНАРЭС, поэтому в общем случае для редактирования пользователями не предназначен.

4.7. Индивидуальная настройка серверов данных

4.7.1. Для каждого сервера данных задается его индивидуальная конфигурация в конфигурационном файле, совпадающем с именем сервера данных.

4.7.2. В секции [DataServer] задаются общие параметры сервера данных:

- **ServName** – идентификатор сервера данных;
- **TSFile** – идентификатор блока данных ТС передаваемого в сервер каналов;
- **TSCount** – количество ТС в блоке данных ТС;
- **TSBeg** – начальный номер ТС в блоке;
- **TIFile** – идентификатор блока данных ТИ передаваемого в сервер каналов;
- **TICount** – количество ТС в блоке данных ТС;
- **TIBeg** – начальный номер ТИ в блоке;
- **MesFile** – идентификатор очереди сообщений передаваемой в сервер каналов;
- **MesFileSize** – размер очереди сообщений;
- **SleepTime** – время паузы в основном цикле. Определяет скорость реакции сервера данных;
- **MesTask** – цифровой идентификатор приложения (сервера данных) в сообщениях;
- **LockedTimeOut** – время ожидания разблокирования блоков данных для записи новых данных;
- **ReadedTimeOut** – время ожидания того, что предыдущие записанные данные прочитаны;
- **CanUnRealMode** – признак того, что сервер данных может работать в режиме эмуляции;
- **CheckDuplicate** – необходимость отслеживания дубликатов сообщений от этого сервера данных (актуально для блоков передачи данных по сети).

4.7.3. Блоки данных ТС, записываемые сервером записываются в секции [ServerBlocksTS] в виде:

Идентификатор блока= Идентификатор блока

4.7.4. Блоки данных ТИ, записываемые сервером записываются в секции [ServerBlocksTI] в виде:

Идентификатор блока= Идентификатор блока

4.8. Индивидуальная настройка клиентов данных

4.8.1. Для каждого клиента данных задается его индивидуальная конфигурация в конфигурационном файле, совпадающем с именем клиента данных.

4.8.2. В секции [Client] задаются общие параметры клиента данных:

- **TSBeg** – начальный номер ТС доступных клиенту в общем блоке ТС сервера каналов;
- **TSCount** – количество ТС, доступных клиенту;
- **TIBeg** – начальный номер ТИ доступных клиенту в общем блоке ТИ сервера каналов;
- **TICount** – количество ТИ, доступных клиенту;
- **ClientTSFile** – идентификатор блока данных ТС в сервере каналов, доступного для чтения клиенту;
- **TSEventName** – идентификатор события на изменение ТС передаваемых клиенту;
- **ClientTIFile** – идентификатор блока данных ТИ в сервере каналов, доступного для чтения клиенту;
- **TIEventName** – идентификатор события на изменение ТИ передаваемых клиенту;
- **ClientMesFile** – идентификатор очереди сообщений в сервере каналов, доступной для чтения клиенту;
- **MesEventName** – идентификатор события на приход сообщений;
- **RefreshTime** – период обновления (время паузы в цикле работы с сервером каналов).
- **MesClient** – цифровой идентификатор приложения (клиента) в сообщениях;
- **TSTask** – маска клиента для блокировки буфера ТС;
- **TITask** – маска клиента для блокировки буфера ТИ;
- **MesTask** – маска клиента для блокировки буфера ТИ;
- **LockedTimeOut** – время ожидания разблокирования блоков данных для записи новых данных;
- **MainBlock** – основной блок данных клиента (опционально).

4.8.3. Блоки данных ТС, читаемые клиентом записываются в секции [DataBlockTS] в виде:

Идентификатор блока= Идентификатор блока

4.8.4. Блоки данных ТИ, читаемые клиентом записываются в секции [DataBlockTI] в виде:

Идентификатор блока= Идентификатор блока

4.9. Структура файлов конфигурации MS Excel

4.9.1. В файле конфигурации (*.xls) обязательно должны находиться несколько листов, следующих в указанном порядке и содержащих следующую информацию:

Таблица 1. Структура файла конфигурации

Название листа	**Содержание листа**
Сервера данных	Описание программных блоков (приложений), посылающих данные в сервер каналов
Клиенты	Описание программных блоков (приложений), принимающих данные из

	сервера каналов.
ТС	Описание всех телесигналов (ТС), с которыми оперирует сервер каналов
ТИ	Описание всех телесигналов (ТИ), с которыми оперирует сервер каналов
Сообщения	Описание всех сообщений, передаваемых в системе
Общие настройки	Общие настройки сервера каналов
<Совпадает с названием блока>	Индивидуальные настройки для каждого блока. Порядок расположения листов с настройками блоков безразличен. Если блок не имеет индивидуальных настроек, то лист с индивидуальными настройками для этого блока не создается

4.9.2. Первая строка на каждом листе конфигурации, за исключением листов с индивидуальными настройками блоков, содержит заголовки полей и игнорируется при загрузке конфигурации.

4.9.3. Расположение полей должно соответствовать расположению, описанному в документации. Перестановка полей не допускается.

4.9.4. Лист «Сервера данных» содержит таблицу со следующими полями:

- **Наименование** – Наименование сервера данных (используется только для отображения).
- **Идентификатор** – Уникальный идентификатор сервера данных.
- **Автоматическая загрузка** – Признак того, что сервер должен быть автоматически загружен при запуске системы. «Да» – сервер данных загружается автоматически, «Нет» – соответственно, сервер данных автоматически не загружается.
- **Автоматическая выгрузка** – Признак того, что сервер должен быть автоматически выгружен при выходе из системы. «Да» – сервер данных выгружается автоматически, «Нет» – сервер данных автоматически не выгружается.
- **Размер очереди сообщений** – Размер очереди сообщений в килобайтах передаваемых данным сервером. Размер очереди должен быть больше чем максимальное сообщение, выдаваемое этим сервером. Оптимальным будет размер, в 10 раз превышающий ориентировочное количество сообщений, которое может послать сервер за один период сервера каналов. (Настройка периода сервера каналов осуществляется на листе «Общие настройки».) Например, период работы сервера данных равен 100 мс. За один период сервер данных может сгенерировать 10 различных сообщений. Размер максимального сообщения 1КБ. Период сервера каналов равен 10мс (значительно меньше периода сервера данных). При этом максимально возможный размер сообщений за один период 10КБ. Таким образом, размер очереди сообщений можно принять равной 100КБ. В том случае, если сервер данных не предусматривает отправку сообщений, то размер очереди сообщений должен быть нулевым.
- **Таймаут блокировки»** – Таймаут ожидания возможности записи в сервер каналов (в мсек). Должен быть немного больше, чем период сервера каналов.
- **Таймаут чтения** – Время ожидания (МСек) того, что предыдущие данные (ТИ или ТС) были прочитаны сервером каналов. Если таймаут превышен, а данные не были считаны сервером каналов, то разрешается повторная запись данных.
- **Таймаут записи** – Время ожидания (в Мсек) сервером каналов возможности чтения записанных сервером данных ТИ или ТС. Время должно быть больше времени записи всего блока ТИ или ТС и зависит от быстродействия сервера данных.

- **Время задержки** – Пауза (МСек) в цикле работы сервера данных. Работа с сервером каналов всех приложений комплекса, как серверов, так и клиентов данных, является циклической. Пауза после каждого цикла работы с сервером каналов регулирует быстродействие программы и долю процессорного времени, занимаемого приложением.
- **Имя файла** – Имя исполняемого модуля сервера данных.
- **Приоритет процесса** – приоритет назначаемый процессу:
  32 – Нормальный приоритет (NORMAL_PRIORITY_CLASS).

  64 – Минимальный приоритет (IDLE_PRIORITY_CLASS). Приложение работает только во время простоя системы.

  128 – Высокий приоритет (HIGH_PRIORITY_CLASS).

  256 – Приоритет реального времени (REALTIME_PRIORITY_CLASS). Как правило не назначается, иначе приложением с приоритетом меньше 256 управление может никогда не передаться.

- **Работа в режиме эмуляции** – Признак того, что сервер может работать в режиме эмуляции. «Да» – сервер может работать в режиме эмуляции, «Нет» – не может.
- **Фильтрация дубликатов сообщений** – Признак необходимости отслеживания дубликатов сообщений от этого сервера данных (актуально для блоков передачи данных по сети). «Да» – сервер может работать в режиме эмуляции, «Нет» – не может.

4.9.5. Лист «Клиенты» содержит таблицу со следующими полями:

- **Наименование** – Наименование клиента данных (используется только для отображения)
- **Идентификатор** – Уникальный идентификатор клиента данных
- **Минимальное время обновления** – Минимальный интервал времени (МСек), через который сервер каналов сообщает клиенту о появлении новых данных. Это же время задается в качестве времени задержки в цикле клиента данных.
- **Максимальное время обновления** – Интервал времени (МСек), через который сервер каналов в случае, если данные не считаны клиентом, переходит в «медленный» режим. В «медленном» режиме сервер каналов сообщает клиентам о том, что обновились данные не через «Минимальное время обновления», а через «Максимальное время обновления». Этот режим позволяет клиентам с низким приоритетам гарантированно прочитать данные из сервера каналов. После того, как данные «медленным» клиентом прочитаны, сервер каналов переходит обратно в быстрый режим.
- **Таймаут блокировки** – Таймаут ожидания возможности чтения данных из сервера каналов (МСек). Должен быть немного больше, чем период сервера каналов.
- **Имя файла** – Имя исполняемого модуля клиента данных.
- **Базовый блок** – Блок данных, являющийся основным для клиента данных. В ЦКПА используется для OPC-сервера, являющегося клиентом по отношении к серверу каналов.
- **Блоки данных** – Указывается один или несколько блоков данных (ТИ,ТС), используемых клиентом. Каждый блок указывается в отдельной ячейке последовательно без пропуска ячеек.

4.9.6. Лист «ТС» содержит таблицу описания телесигналов со следующими полями:

- **Наименование** – Наименование телесигнала (используется только для отображения).
- **Идентификатор** – Идентификатор телесигнала. Идентификатор должен быть уникальным в пределах каждого блока данных. Ссылка на ТС производится по идентификатору блока и идентификатору ТС. В качестве идентификатора можно использовать наименование ТС. При сохранении конфигурации в ini-файлы из идентификатора автоматически удаляются все пробелы и недопустимые символы. Если в строке не указан идентификатор, строка при сохранении пропускается и может использоваться в качестве комментария.
- Пустое поле для совместимости с таблицей ТИ.
- Пустое поле для совместимости с таблицей ТИ.

- **Сервер данных** – Наименование блока, оно же наименование сервера данных, который записывает этот блок.
- **Блоки данных** – Наименование дополнительных блоков данных, в которые входит этот ТС. Дополнительные блоки данных используются для ссылки на ТС не через основной блок маршрутизации данных клиентам. Наименование дополнительного блока данных не должно совпадать с наименованием серверов данных.

4.9.7. Лист «ТИ» содержит таблицу описания телеинформации со следующими полями:

- **Наименование** – Наименование ТИ (используется только для отображения).
- **Идентификатор** – Идентификатор ТИ. Идентификатор должен быть уникальным в пределах каждого блока данных. Ссылка на ТИ производится по идентификатору блока и идентификатору ТИ. В качестве идентификатора можно использовать наименование ТИ. При сохранении конфигурации в ini-файлы из идентификатора автоматически удаляются все пробелы и недопустимые символы. Если в строке не указан идентификатор, то строка при сохранении пропускается и может использоваться в качестве комментария.
- **Минимум** – Минимальное значение ТИ. Используется при преобразовании полученного от Smart-ПУ значения в квантах в реальное значение ТИ.
- **Максимум** – Максимальное значение ТИ. Используется при преобразовании полученного от Smart-ПУ значения в квантах в реальное значение ТИ.
- **Сервер данных** – Наименование блока, оно же наименование сервера данных, который записывает этот блок.
- **Блоки данных** – Наименование дополнительных блоков данных, в которые входит этот ТС. Дополнительные блоки данных используются для ссылки на ТС не через основной блок маршрутизации данных клиентам. Наименование дополнительного блока данных не должно совпадать с наименованием серверов данных.

4.9.8. Лист «Сообщения» содержит таблицу описания сообщений со следующими полями:

- **Номер** – Уникальный номер сообщения.
- **Идентификатор** – Уникальный идентификатор сообщения.
- **Наименование** – Наименование сообщения (используется для отображения).
- **Передавать по сети** – Признак того, что это сообщение надо передавать по сети. («Да» – передавать, «Нет» – не передавать).
- **Комментарий** – Комментарий для данного сообщения.

4.9.9. Лист «Общие настройки» содержит таблицу настроек сервера каналов. Каждый настраиваемый параметр – отдельная строка с полями (см. П.4. «Конфигурирование сервера каналов»):

- **Наименование** – наименование параметра;
- **Идентификатор** – наименование параметра;
- **Значение** – наименование параметра;

4.9.9. Идентификаторы параметров изменять нельзя.

4.9.10. В листе «Общие настройки» предусмотрены следующие параметры:

- **Блок ТС в сервере каналов (TSFile) –** Менять это значение нет необходимости. По умолчанию значение равно «ANARESChanelTSFile».
- **Блок ТИ в сервере каналов (TIFile) –** Менять это значение нет необходимости. По умолчанию значение равно «ANARESChanelTIFile».
- **Блок сообщений в сервере каналов (MesFile) –** Менять это значение нет необходимости. По умолчанию значение равно «ANARESChanelMesFile».

- **Имя сервера каналов (ServName) –** Менять это значение нет необходимости. По умолчанию значение равно «ANARESChanelServer».
- **Размер очереди сообщений СК (MesFileSize)** – Размер очереди сообщений сервера каналов в КБ. Должен быть достаточным, чтобы обеспечить хранение в течение промежутка времени (от нескольких секунд до минуты) всех сообщений системы. По умолчанию значение равно 1МБ.
- **Номер в списке клиентов сообщений (MesTask)** – Менять это значение нет необходимости. По умолчанию значение равно 1.
- **Задержка в цикле сервера каналов (SleepTime) –** Задержка в цикле сервера каналов (Мсек). Определяет быстродействие сервера каналов. По умолчанию значение равно 15 МСек.
- **Таймаут отмены медленного режима (SlowTimeOut)** – Задержка времени (МСек) перед переходом из «медленного» режима в нормальный. По умолчанию значение равно 500 МСек.
- **Таймаут блокировки (LockedTimeOut)** – Время ожидания возможности чтения/записи общих данных с клиентами серверами данных.
- **Ожидание запуска (GetMutesTimeOut)** – Время ожидания (МСек) сигнала о том, что приложение, входящее в состав ПО ЦКПА, запущено. По умолчанию равно 5 Сек.
- **Пауза перед рестартом (RunServerTimeOut)** – Пауза перед повторным рестартом приложения, входящего в состав ПО ЦКПА. По умолчанию значение равно 10 Сек.
- **Максимальное количество клиентов (MesClientSize)** – Максимальное количество клиентов работающих с сообщениями. По умолчанию значение равно 100.

4.9.11. Настройки блока Интегратор производятся на листе «Integrator» (секция [Integrator]).

- dDTTS – Интервал времени интегрирования ТС (Сек). По умолчанию 60 Сек.
- dDTTI – Интервал времени интегрирования ТИ (Сек). По умолчанию 60 Сек.
- UstMaxdTI – Максимально допустимое отличие одноименных ТИ в абсолютных значениях. По умолчанию задано: 10.

4.9.12. Настройки блока приема данных из ЛВС производятся на листе «SocketClnt».

В секции [SocketMain] задаются следующие параметры:

- TimeOut – Время ожидания окончания операции при работе с Win-Socket (МСек). По умолчанию задано – 1000 МСек.
- TestPeriod – Пауза между посылкой тестовых сообщений серверу (МСек). По умолчанию задана – 5000 МСек.
- ConnectTimeOut – Пауза между TimeOut между попытками повторного соединения (МСек). По умолчанию задано 1000 МСек.

Далее указываются секции обозначающее IP-адреса серверов данных и порт по которому производится соединение в виде [ip-адрес:порт].

Для каждого сервера данных указывается набор блоков данных на стороне клиента и соответствующие им блоки на серверной стороне.

В секции [ChanelStatusTS] для каждого ip-адреса (канала) указывается идентификатор ТС, в который пишется признак неработоспособности канала.

4.9.13. Настройки блока передачи данных в ЛВС производятся на листе **«SocketSrv»** (секция [main]).

- Port – Номер порта TCP/IP по которому производится передача. По умолчанию задан порт 502.
- MinTITSSendPeriod – Минимальное время между отправкой одноименного блока данных по сети (МСек). По умолчанию задано 1000 МСек.

4.10. Преобразование файла конфигурации в ini-файлы

4.11.1. Файл конфигурации соответствующего программного модуля в набор ini-файлов преобразует утилита ChanelConfig.exe.

4.11.2. После запуске утилиты предлагается выбрать файл конфигурации формата *.xls. После выбора нужного файла следует указать каталог, в котором будут сохранены конфигурационные ini-файлы, читаемые при загрузке исполняемых модулей.

5. ОПИСАНИЕ ПРОГРАММНЫХ ИНТЕРФЕЙСОВ И БИБЛИОТЕК ДЛЯ РАБОТЫ В СОСТАВЕ SCADA-АНАРЭС

5.1. Общие сведения

5.1.1. SCADA-АНАРЭС содержит ряд библиотек обеспечивающих стыковку с внешними системами и добавление отдельных программных блоков в состав комплекса.

5.1.2. Использование интерфейсных библиотек дополняет стандартные механизмы обмена (такие, как OPC) в случае, если требуется более тесная интеграция внешних блоков.

5.1.3. В настоящее время библиотеки разработаны для Visual C++ и Borland Delphi/Builder. В дальнейшем интерфейсные библиотеки могут дополняться. В описании приведены листинги библиотек для Visual C++. Варианты библиотек для Borland Delphi/Builder аналогичны, поэтому в документации не приводятся.

5.2. Состав библиотек

- ServProc – библиотека для работы с сервером приложений;
- CSClient – библиотека для работы с сервером каналов в качестве клиента данных;
- CSDataServ – библиотека для работы с сервером каналов в качестве сервера данных;
- MsgQueue – библиотека для работы с очередями сообщений;
- CSIniFile – библиотека для работы с конфигурационными файлами SCADA-АНАРЭС.

5.3. Библиотека для работы с сервером приложений (ServProc)

5.3.1. Библиотека для работы с сервером приложений содержит функции по обмену командами и блоками не типизированных данных между приложениями комплекса. Листинг 1 содержит заголовочный файл библиотеки.

Листинг 1. Библиотека ServProc

```
#pragma once
```

```
#include <OAIDL.H>
#include <string>
#include "fmstream.h"
#include "CAnaresUnits.h"

//Загрузка сервера приложений
//Path - путь к серверу каналов
extern bool LoadComServer(const char *Path);

//Завершение работы с сервером приложений
extern void UnLoadComServer();

//Принудительная выгрузка библиотеки. Без этого библиотека будет выгружена при закрытии приложения
extern void FreeComServer();

//Получить команду от сервера приложений для открытой сессии
//Handle - дескриптор сессии
//Command - буфер в который будет выдана команда
//Len - размер буфера
extern bool GetCommand(int Handle,char *Command,size_t Len);

//Получить результат выполнения команду от сервера приложений для открытой сессии
//Handle - дескриптор сессии
//Command - буфер в который будет выдан результат
//Len - размер буфера
extern bool GetCommandResult(int Handle,char *ResStr,size_t Len);

//Класс для работы с сессией сервера приложений
class CAnarSession
{
public:
    unsigned SessionHandle;          //Дескриптор сессии
    std::string SessionName;         //Имя сессии
    std::string ServerName;          //Имя сервера
    CAnarSession(void);
    ~CAnarSession(void);
    bool Connect(                    //Открыть сессию (возвращает - успешно/неуспешно)
        const char* ServerName_,     // - имя сервера
        const char* SessionName_,    // - имя сессии
        int ServerFlag               // - признак того, что программа - сервер данных
        );
```

```
void DisConnect(void);              //Закрыть сессию
bool OpenData(                      //Открыть данные (возвращает - успешно/неуспешно).
    const char* Name,               // - имя набора данных
    int Mode,                       // - режим открытия файла
    int MaxSize,                    // - максимальный размер данных
    unsigned &ClntMask              // - возвращает маску клиента данных
                                    //(используется при закрытии)
    );
bool CloseData(                     //Закрыть данные (возвращает - успешно/неуспешно)
    const char* Name                // - имя набора данных
    );
bool QueryData(                     //Запросить набор данных за указанное время
                                    //(возвращает есть ли данные)
    const char* Name,               // - имя набора данных
    DATE QueryDateTime              // - дата и время запрашиваемых даных
    );
bool ReadQueryData(                 //Прочитать запрошенные данные
                                    //(возвращает прочитаны ли данные)
    const char* Name,               // - имя набора данных
    DATE QueryDateTime              // - дата и время запрашиваемых даных
    );
bool SetDataStatus(                 //Установить пользовательский статус данных
    const char* Name,               // - имя набора данных
    int Status                      // - статус данных
    );
bool SetCommand(                    //Отправить команду другим приложениям
    const char* Command             // - команда
    );
bool SetCommandResult(              //Отправить ответ на команду
    const char* ResStr              // - ответ
    );
bool IsCommand(void);               //Проверка: есть ли новые команды?
bool IsCommandResult(void);         //Проверка: есть ли ответы на команды?
bool GetCommand(                    //Получить команду
    char *ComStr,                   // - буфер команды
    size_t Len                      // - длина буфера
    );
bool GetCommandResult(              //Получить ответ на команду
    char *ResStr,                   // - буфер для ответа
    size_t Len                      // - лдина буфера
    );
```

```
bool ClearCommandResult(void);    //Очистить все ответы
bool SetMsgValues(                //Назначить сообщения Windows для
                                  //отправки при наличии команд или ответов
    int CommandMsg,               // - сообщение на наличие команд
    int ResCommandMsg             // - сообщение на наличие ответов
    );
bool CheckDate(                   //Проверить время набора данных
    const char* Name,             // - имя набора данных
    DATE CDate,                   // - сравниваемое время
    int &Flag                     // - возвращаемое значение: 0 - время совпадает,
                                  //   1 - не совпадает, -1 - данные заблокированы
    );
bool SetDate(                     //Установить время набора данных
    const char* Name,             // - имя набора данных
    DATE CDate                    // - время даных
    );
bool GetDate(                     //Полусить дату и время данных
    const char* Name,             // - имя набора данных
    DATE &CDate                   // - возвращает время данных
    );
bool GetDataOpenMode(             //Получить режим открытия данных
    const char* Name,             // - имя набора данных
    int &OpenMode                 // - возвращает режим открытия
    );
fmstream* OpenDataWrite(          //Открыть набор данных для записи.
                                  //Возвращает указатель на открытый fmstream
    char *RegFile,                // - имя набора данных
    int Size                      // - размер даных
    );
void  CloseDataWrite(             //Закрыть открытые для чтения данные
    fmstream* FIO,                // - указатель на открытый fmstream
    char* RegFile                 // - имя набора данных
    );
fmstream* OpenDataRead(           //Открыть данные для чтения
    char *RegFile,                // - имя набора данных
    unsigned &DataMask            // - куда возвращать маску данных (нужна для закрытия)
    );
void CloseDataRead(               //Закрыть открытые для чтения данные
    fmstream* FIO,                // - указатель на открытый fmstream
    char* RegFile,                // - имя набора данных
    unsigned DataMask             // - маска данных
```

```
        );
    unsigned  int OpenDataRW(        //Открыть данные для чтения и записи (возвращает маску)
        char* RegFile,               // - имя набора данных
        unsigned int OpenMode,       // - режим открытия файла
        int Size,                    // - размер
        fmstream* FMStream           // - возвращаемый указатель на fmstream
        );
    void  CloseDataRW(               //Закрыть даные открытые и для чтения и записи
        fmstream* FIO,               // - указатель на открытый fmstream
        char *RegFile,               // - имя набора данных
        unsigned int DataMask        // - маска данных
        );
};
```

5.3.2. Для большинства клиентов и серверов данных, работающих в составе SCADA-АНАРЭС, работа с сервером приложений ограничивается, лишь открытием сервера приложений в начале работы программы и закрытием его в конце:

```
LoadComServer(MainIniPath.c_str());
…
// Основной цикл программы
…
UnLoadComServer();
```

`MainIniPath.c_str()` – Получение основного каталога комплекса.

5.3.3. Обработка команд сервера приложений производится неявно в классах клиента и сервера данных.

5.4.Общие сведения по структуре данных

5.4.1. Информация, передаваемая через сервер каналов SCADA-АНАРЭС, подразделяется на три вида:

1. Дискретные данные (телесигналы).
2. Аналоговые данные (телеизмерения);
3. Сообщения.

5.4.2. **Телесигнал** (ТС) представляет собой следующую структуру:

```
struct TTSVed
{
    int Unical;               //Уникальный номер в сервере каналов
    SCADATIME UnitDateTime;   //Время формирования ТС (например, с устройства)
```

```
    SCADATIME WriteDateTime;  //Локальное время ТС (время считывания ТС)
    unsigned int TC;          //Значение ТС по битам:
                              //...  3  2  1 0
                              //     НД НА Р Значение
                              //Р - ручной ввод (Замещение)
                              //НА - не актуальное значение (оборван канал связи)
                              //НД - не достоверное значение
};
```

5.4.3. ТС в сервере каналов идентифицируется по уникальному номеру, заданному в поле Unical. Этот номер уникален в рамках одной рабочей станции.

5.4.4. Значение ТС содержится в первом бите поля ТС. Остальные биты представляют собой флаги:

Бит	Значение
0	Значение ТС
1	Ручной ввод ТС
2	Значение не актуально (это может быть, например, в том случае, если значение не обновлялось достаточно долгий промежуток времени).
3	Не достоверное значение
…	Резерв

5.4.5. ТС имеет также две метки времени. Одна (UnitDateTime) содержит время формирования телесигнала в устройстве, вторая (WriteDateTime) – время записи ТС в сервер каналов. Время хранится в формате FILETIME.

5.4.5. **Телеизмерение** (ТИ) представляет собой следующую структуру:

```
struct TTIVed
{
    int Unical; //Уникальный номер в АНАРЭС
    SCADATIME UnitDateTime;   //Время формирования ТС (например, с устройства)
    SCADATIME WriteDateTime;  //Локальное время ТС (время считывания ТС)
    unsigned int TIFlag;      //Флаги достоверности ТИ.
                              //...  3  2  1 0
                              //     НД НА Р ?
                              //? - недостоверное по неизвестной причине
                              //Р - ручной ввод (Замещение)
                              //НА - не актуальное значение (оборван канал связи)
                              //НД - не достоверное значение
    unsigned int Kvant;       //Значение в квантах
```

```
    double Value;            //Масштабированное значение
};
```

5.4.7. ТИ в сервере каналов идентифицируется по уникальному номеру, заданному в поле Unical. Этот номер уникален в рамках одной рабочей станции.

5.4.7. Значение ТИ содержится в поля Value. Также имеется поле для хранения значения ТИ в квантах. Все приложения в рамках комплекса, как правило, оперируют со значением в реальной величине, хранимой в поле Value, а поле Kvant, используется, как правило, для контроля.

5.4.8. Флаги достоверности хранятся в поле TIFlag:

Бит	Значение
0	Значение недостоверно по неизвестной причине
1	Ручной ввод ТИ
2	Значение не актуально (это может быть, например, в том случае, если значение не обновлялось достаточно долгий промежуток времени).
3	Не достоверное значение
…	Резерв

5.4.9. ТИ имеет также две метки времени. Одна (UnitDateTime) содержит время формирования ТИ в устройстве, вторая (WriteDateTime) – время записи ТИ в сервер каналов. Время хранится в формате FILETIME.

5.4.10. Телеизмерения и телесигналы объединяются в блоки данных. Каждый ТС/ТИ должен входить по крайней мере в один блок, совпадающий по названию с сервером данных, выдающим этот ТС/ТИ. Количество дополнительных блоков данных, в которые входит ТС/ТИ не ограничено.

5.4.11. Клиенты получают доступ к данным поблочно, поэтому дополнительные блоки данных позволяют гибко назначать возможность доступа к данным тем или иным клиентам.

5.4.12. ТС/ТИ может также идентифицироваться строковым идентификатором. Строковый идентификатор должен быть уникален в пределах блока. В разных блоках могут находиться ТС/ТИ с одинаковым идентификатором. Это позволяет достаточно просто организовывать обработку данных, имея одинаковые идентификаторы сигналов в разных блоках (один для необработанных данных, другой для обработанных). Таким образом полный идентификатор ТС/ТИ состоит из идентификатора блока и идентификатора самого сигнала. Соответствие строкового идентификатора и уникального номера хранится в конфигурации.

5.5. Работа с сообщениями

5.5.1. Для передачи между приложениями комплекса информации, не попадающей под определения телеизмерения или телесигнала, имеется механизм передачи сообщений. Сообщение представляет собой заголовок сообщения, где указан тип, метка времени и другие параметры сообщения, и блок данных произвольной длины. Формат блока данных определяется типом сообщения.

5.5.2. Сообщения передаются через очередь сообщений. Размер очереди сообщений задается в конфигурации.

5.5.3. Сообщение представляет собой структуру:

```
struct TMesRecord              //Структура базовой части сообщения
{
    unsigned int Next;         //Смещение до следующего сообщения (пользователем не задается)
                               //У самого нового = NilMsg
    unsigned int Prev;         //Смещение до предыдущего сообщения (пользователем не задается)
                               //у самого старого = NilMsg
    unsigned int Size;         //Полный размер сообщения (пользователем не задается)
    //Поля определяющие уникальность:
    int Task;                  //Код задачи, которая записала сообщение,
                               //у каждого приложения в системе должны быть уникальные кода
    int Unical;                //Уникальный номер сообщения (пользователем не задается)
    SCADATIME DateTime;        //Время возникновения сообщения в формате FILETIME
    //-------------------------
    int Msg;                   //Идентификатор сообщения (тип сообщения)
    unsigned int DataSize;     //Размер данных сообщения
};
```

5.5.4. Сообщение определяется типом сообщения (поле Msg). Набор зарезервированных сообщений представлен в таблице 1.

Таблица 1. Список зарезервированных типо вообщений

Тип (Msg)	Идентификатор	Наименование	Комментарий
1	msAppError	Ошибка в программе	Ошибка в программе (например, Exception)
2	msAppWarning	Предупреждение в программе	Нештатная ситуация в программе (обработанная ошибка), например TimeOut
3	msAlarm	Аварийная сигнализация	Технологическое предупреждение (нарушение технологического процесса, например все данные недостоверны)

4	msWarning	Предупреждение	Технологическое предупреждение не связанное с нарушением алгоритма (например, одиночный выход параметра за пределы)
5	msInformation	Информация	Технологическое информационное сообщение
6	msControl	Произведено управление	Сообщение о произведенном управлении технологическим процессом (УВ)
7	msTSInMes	Передача блока ТС	Передача блока ТС в сообщении
8	msTIInMes	Передача блока ТИ	Передача блока ТI в сообщении
9	msFileInMes	Файл в сообщении	
10	msFileInMesRecieved	Файл получен	
101	msManualTI	Перевод на ручное управление ТИ	Перевод на ручное управление ТИ
102	msAutoTI	Перевод на автоматическое обновление ТИ	Перевод на автоматическое обновление ТИ
103	msManualTS	Перевод на ручное управление ТС	Перевод на ручное управление ТС
104	msAutoTS	Перевод на автоматическое обновление ТС	Перевод на автоматическое обновление ТС
105	msGetManualStatus	Запрос на текущий статус ТИ и ТС в Арбитре	Запрос на текущий статус ТИ и ТС в Арбитре
106	msSetNewSource	Переключиться на другой источник данных в Арбитре	Переключиться на другой источник данных в Арбитре
131	msEmulTI	Моделирование ТИ	Моделирование ТИ
132	msNotEmulTI	Отмена моделирования ТИ	Отмена моделирования ТИ
133	msEmulTS	Моделирование ТС	Моделирование ТС
134	msNotEmulTS	Отмена моделирования ТС	Отмена моделирования ТС

5.5.5. Для работы с очередью сообщений имеется класс `CMsgQueue`. Заголовочный файл соответствующей библиотеки представлен в листинге 2.

Листинг 2. Заголовочный файл библиотеки MsgQueue

```
//-------------------------------------------------
//Базовый класс очереди обмена сообщениями через память.
```

```
//-----------------------------------------------------
#pragma once

#define NULLMSG                    0xFFFFFFFF

#define mq_NOERROR              0
#define mq_ERROR_SMALLSIZE      1
#define mq_ERROR_CREATEMEM      2
#define mq_ERROR_DENIWRITE      3
#define mq_ERROR_TOOBIG         4
#define mq_ERROR_DENIREAD       5
#define mq_ERROR_EMPTY          6
#define mq_ERROR_NOMSG          7
#define mq_ERROR_SMALLSTRING    8

#pragma pack (8)
#include "CScadaUnits.h"
#include "OIKDef.h"
#include <Windows.h>

//Перед каждым сообщением идет смещение на начало этого сообщения
//unsigned int MsgOffset;
struct TMesRecord             //Структура базовой части сообщения
{
    unsigned int Next;        //Смещение до следующего сообщения (пользователем не задается)
                              //У самого нового = NilMsg
    unsigned int Prev;        //Смещение до предыдущего сообщения (пользователем не задается)
                              //у самого старого = NilMsg
    unsigned int Size;        //Полный размер сообщения (пользователем не задается)
    //Поля определяющие уникальность:
    int Task;                 //Код задачи, которая записала сообщение,
                              //у каждого приложения в системе должны быть уникальные кода
    int Unical;               //Уникальный номер сообщения (пользователем не задается)
    SCADATIME DateTime;       //Время возникновения сообщения в формате FILETIME
    //-------------------------
    int Msg;                  //Идентификатор сообщения (тип сообщения)
    unsigned int DataSize;    //Размер данных сообщения
};

struct TMesDataRec:TMesRecord
{
```

```
    char *Data;                         //Данные сообщения
};
#pragma pack ()

struct MQHeader                                     //Заголовок блока памяти для CMsgQueue
{
        unsigned int First;
        unsigned int Last;
        unsigned int Size;              //Размер блока памяти без заголовка
        SCADATIME MaxDT;
        unsigned int ClientCount;   //Количество клиентов
};
struct MQHeaderClient:MQHeader  //Заголовок с усетом счетчиков клиентов
{
        unsigned int Clients[1000];   //Clients[i]  -  относительный  указатель  для  i-го  клиента  на
последнее считанное сообщение
};

//Очередь обмена сообщениями.
class CMsgQueue
{
    int CacheSize;
    MQHeaderClient *PHeader;          //Указатель на заголовок
    unsigned char *PQueue;            //Указатель на начало самой очереди
    unsigned int ReadOffset;
    void InitCache(int size);
    TMesRecord* CMsg(unsigned int offs);
    unsigned int NewUnical;           //Уникальный номер нового сообщения
    unsigned int ThisTask;
    unsigned int CurClient;

    void ClearClientRange(unsigned int PMin,unsigned int PMax);
    bool GetClientMsg();      //Устанавливает для Clients[CurClient] указатель на сообщение.
public:
    TMesRecord *CurMessage;
    unsigned int ErrorCode;
    CMsgQueue();
    ~CMsgQueue();

    bool CreateQueue( //Создание очереди сообщений с указанием размера, включая заголовок:
                void *buff,                   //Буфер в котором надо разместить очередь
```

```
            unsigned int BufSize,          //Размер буфера
            unsigned int _ClientCount,     //Количество клиентов очереди
            unsigned int _CurClient,       //Клиент - создатель очереди
            unsigned int _ThisTask         //Код задачи, создающей очередь
            );
bool OpenQueueReadOnly( //Открытие очереди только для чтения
            void *buff,                    //Буфер с очередью
            unsigned int _CurClient        //Клиент, открывающий очередь
            );
bool OpenQueue( //Открытие очереди сообщений для чтения/записи
            void *buff,                    //Буфер с очередью
            unsigned int _CurClient,       //Клиент, открывающий очередь
            unsigned int _ThisTask         //Код задачи
            );

bool WriteMsg(TMesRecord* MQMsg); //Ретрансляция сообщения в очередь
bool WriteMsgUnical(TMesRecord* MQMsg);//Добавление сообщения в очередь с удалением дублей
bool AddMsg(TMesRecord* MQMsg); //Добавить сообщение с заполнением полей: Task,Unical,DT
bool NewMsg(       //Добавление сообщения с типом и данными
            int Msg,                       //Тип сообщения
            unsigned int DataSize,         //Размер данных сообщения
            void* Data                     //Данные
            );
bool NewMsgWithTime( //Добавление сообщения с типом, данными и временем
            int Msg,                       //Тип сообщения
            unsigned int DataSize,         //Размер данных сообщения
            void* Data,                    //Данные
            SCADATIME* cFileTime           //Время в формате FILETIME
            );
bool WriteString(const char* MsgString);//Добавление нового сообщения с типом строка

bool ReadMsg();    //Извлекает сообщение из очереди последовательно.
bool ReadMsg(      //Извлекает сообщение указанного типа.
            int MCount,                    //Количество типов сообщений
            int *Msgs                      //Типы сообщений, которые надо извлечь
            );
bool ReadMsg(      //Извлекает сообщение указанного типа и задачи.
            int MCount,                    //Количество типов сообщений
            int *Msgs,                     //Типы сообщений, которые надо извлечь
            int TCount,                    //Количество кодов задач извлечь
            int *Tasks                     //Коды задач сообщений, которые надо извлечь
```

```
            );
    bool ReadString(  //Чтение сообщения с типом строка (Msg=0)
            char* MsgString,            //Строка назначения
            unsigned int MaxLen         //Максимальный размер строки
            );

    bool GetLastMsg(int Msg); //Извлекает последнее сообщение указанного типа
    bool GetNewMsg(int Msg);  //Извлекает новое сообщение указанного типа

    int GetAllMsgsSize();     //Возвращает размер буфера под все сообщения
    bool ReadAllMsgs(         //Копирует все сообщения из очереди.
            char *Buff,                 //Буфер, куда копировать сообщения
            unsigned int BufSize        //Размер выделенного буфера
            );
    int GetMsgQueueSize();    //Возвращает полный размер очереди сообщений

    bool SetCurClient(unsigned int _CurClient); //Установить номер клиента
};
```

5.5.5. Класс MsgQueue инкапсулирован в классы работы с сервером каналов в качестве сервера и клиента данных, поэтому прямая работа с MsgQueue, как правило, не предполагается.

5.5. Библиотека для работы с конфигурационными файлами (CSIniFile)

5.5.1. Клиентская библиотека для работы с конфигурационными файлами содержит класс CSIdents, обеспечивающий интерфейс для работы с идентификаторами данных сервера каналов. Описание класса приведено в листинге 3.

Листинг 3. Класс CSIdents

```
class CCSIdents
{
        std::string CurLocalIdent;
        void CheckLocalOpen();
        bool IsLastIdentTS;
        std::string TempResult;
public:
        CIniFile *LocalIni;           //Ini-файл текущего приложения
        CIniFile TSIdents;            //Для получения номеров ТИ и ТС надо вызвать
        CIniFile TIIdents;            // TSIdents.ReadInteger("DataBlockName","IdentName",-1);
```

```
                                    // если -1, то не найден.
        CIniFile MesIdents;         //Ini-файл идентификаторов сообщений
       CIniFile NameIdents;         //Ini-файл наименований
        CCSIdents();
        ~CCSIdents();
        std::string& GetFirstLocalTSIdent();        //Возвращает первый идентификатор ТС в секции
        std::string& GetFirstLocalTIIdent();        //Возвращает первый идентификатор ТИ в секции
        std::string& GetNextLocalIdent();           //Возвращает следующий идентификатор в секции
        std::string& GetCurrentGlobalIdent();       //Возвращает глобальный идентификатор для
                                                    //последнего считанного локального
        bool FindLocalTSIdent(const char * Ident);  //Ищет локальный идентификатор ТС
        bool FindLocalTIIdent(const char * Ident);  //Ищет локальный идентификатор ТИ
        int TSIndexForCurrentLocalIdent();          //Возвращает индекс ТС для последнего
                                                    //считанного локального идентификатора
                                                    //если нет соответствующего ТС-возвращает -1
        int TIIndexForCurrentLocalIdent();          //То же по ТИ
        bool IsMessagePublic(int MsgNumber);        //Возвращает признак того, что сообщение
                                                    //передается по сети
        std::string& GetCurrentCompName();          //Получить имя компьютера
        std::string& GetCurrentCompIdent();         //Получить идентификатор компьютера

        std::string& GetMesIdent(int Msg);          //Получить идентификатор сообщения по номеру
        int GetTSNum(const char * Ident);           //Возвращает номер ТС по его идентификатору
        int GetTINum(const char * Ident);           //Возвращает номер ТИ по его идентификатору
        std::string& ExtractIdentName(const char * Ident);  //Возвращает краткий идентификатор
        std::string& GetTSName(int Num);            //Возвращает наименование ТС по его номеру
        std::string& GetTIName(int Num);            //Возвращает наименование ТИ по его номеру
        std::string& GetMesName(int Num);           //Возвращает наименование сообщения по номеру
};
```

5.5.2. Каждый ТИ, ТС сообщения в сервере каналов идентифицируются уникальным номером. Также данные имеют строковый идентификатор и название.

5.5.3. Для получения номера ТИ/ТС по известному строковому идентификатору блока данных и элемента данных необходимо использовать методы: GetTSNum и GetTINum, соответственно. В качестве параметра указывается составной идентификатор, состоящий из идентификатора блока и идентификатора элемента данных, записанных через точку, например, «DataBlock1.TS1».

5.5.4. Для сообщений возникает необходимость в применении обратной функции – получение идентификатора сообщения по его номеру. Клиенты получают сообщения с типом в числовом значении

(поле Msg). Если клиенту проще идентифицировать сообщение по строковому идентификатору, то можно применить метод GetMesIdent.

5.5.5. Для получения наименований элементов данных (ТС, ТИ, сообщения) используются функции GetTSName, GetTIName, GetMesName.

5.5.5. Для получения идентификатора компьютера, на котором выполняется приложение используется функция GetCurrentCompIdent. Для получения наименования компьютера используется функция GetCurrentCompName. Имя и идентификатор компьютера указываются в конфигурации и могут отличаться от имени компьютера заданного в Windows.

5.7. Библиотека клиента данных

5.7.1. Клиентская библиотека для работы с сервером каналов содержит класс CSDataClient, обеспечивающий взаимодействие с Сервером каналов. Заголовочный файл библиотеки приведен в листинге 4.

Листинг 4. Библиотека CSClient

```
#pragma once
#include "CScadaUnits.h"
#include "fmstream.h"
#include "MsgQueue.h"
#include "OIKDef.h"
#include "ServProc.h"
#include <string>
#include "ProgDir.h"
#include "fmstream.h"
#include "IniFile.h"

class CCSDataClient
{
private:
    CAnarSession *cChanelSrv;

    std::string cTIFileName;
    std::string cTSFileName;
    std::string cMesFileName;

    std::string TIEventName;
    std::string TSEventName;
```

```
    std::string MesEventName;
    std::string ChanelServerMutexName;

    int cLockedTimeOut;
    int TSMask;
    int TIMask;
    int MesClient;
    bool ServStarting;
    unsigned int StartTime;
    unsigned int RunTime;

    bool cTSPresent;
    bool cTIPresent;
    bool cMesPresent;

    int fTICount;
    int fTSCount;
    int MesBufSize;
    int ScanMesOffs;
    int MesMask;
    HANDLE hTIEvent;
    HANDLE hTSEvent;
    HANDLE hMesEvent;
    fmstream *fMapTI;
    fmstream *fMapTS;
    fmstream *fMapMes;
    int ReadTIStatus;

    void ReConnect();
    void AddToMsgList(TMesRecord* aMsg);
    void ClearMsgList();
public:
    bool WorkFlag;                          //Признак работоспособности (не надо ли выйти)
    int RefreshTime;                        //Время обновления (Sleep)
    int TSBegin;                            //Индекс начала ТС в сервере каналов
    int TSCount;                            //Количество ТС в сервере каналов
    int TIBegin;                            //Индекс начала ТИ в сервере каналов
    int TICount;                            //Количество ТС в сервере каналов

    TTIVed* TI;                             //Буфер в который считываются ТИ
    TTSVed* TS;                             //Буфер в который считываются ТС
```

```
    std::vector <TMesDataRec*> MsgList;      //Указатели на сообщения в буфере

    CMsgQueue* MsgQueue;                     //Очередь сообщений. Создается автоматически

    CCSDataClient(void);
    ~CCSDataClient(void);

    void ReadIniClient(                      //Читаем ini-файл
              CIniFile* ini                  // - открытый ini-файл
              );
    void InitClient();                       //Загружаем сервер каналов
    bool StartClient();                      //Загружаем сервер каналов
    void FreeClient();                       //Закрывает сессию
    bool CheckWork();                        //Проверяет запущенность сервера и если что, его
                                             //перезапускает, выставляет WorkFlag
    bool ReadTS(                             //Читает блок ТС из сервера каналов
              bool IgnoreEvent               // - не проверять событие наличия новых данных
              );
    bool ReadTI(                             //Читает блок ТИ из сервера каналов
              bool IgnoreEvent               // - не проверять событие наличия новых данных
              );
    bool ReadMes(                            //Извлекает все сообщения указанного типа.
              int MCount=0,                  // - количество считываемых сообщений
              int *Msgs=NULL                 // - типы сообщений которые надо читать
              );
    void ResetReadEvents(                    //Сбрасывает события чтения
              bool TSEvent=true,             // - сбрасывать событие наличия новых ТС
              bool TIEvent=true,             // - сбрасывать событие наличия новых ТИ
              bool MesEvent=true);           // - сбрасывать событие наличия новых сообщений
    unsigned int DirectReadTSValue(          //Читает значение ТС без блокировок.
              int iTS                        // - номер ТС
              );
    unsigned int DirectReadTIValue(          //Читает значение ТИ в квантах без блокировок.
              int iTI                        // - номер ТИ
              );
    int TSBufLength();                       //Длина буфера ТС (в количестве ТС)
    int TIBufLength();                       //Длина буфера ТИ (в количестве ТИ)
    bool FullStartClient();                  //Возвращает признак того, что сервер каналов
                                             //запустился и готов к работе
    SCADATIME GetChanelServerTime();         //Возвращает время сервера каналов
};
```

```
extern CCSDataClient CSClient;
```

5.7.2. Объект CSDataClient создается статически. Перед началом работы с сервером каналов в качестве клиента данных необходимо выполнить инициализацию.

```
CSIdents=new CCSIdents(); // создание и загрузка объекта для работы с идентификаторами.

CSClient.ReadIniClient(CSIdents->LocalIni);  // чтение идентификаторов для клиента.
```

Перед началом работы надо запустить клиента данных и проверить успешность его запуска:

```
if (!CSClient.StartClient())
{
      …
}
```

5.7.2. В приложении клиента данных необходимо организовать отдельный цикл работы с сервером каналов следующего вида:

```
while (CSClient.CheckWork())
{
      if (CSClient.FullStartClient())
      {
      //Здесь выполняется работа с данными, полученными из сервера каналов
      }
      Sleep(CSClient.RefreshTime);
}
```

5.7.3. В каждом цикле работы с сервером каналов необходимо выполнять чтение телеизмерений, телесигналов и сообщений. В случае, если в конфигурации указано, что клиенту данных доступны ТИ, ТС или сообщения, то чтение соответствующего набора данных обязательно.

5.7.4. Чтение ТИ или ТС осуществляется выполнением методов ReadTI или ReadTS, соответственно. При этом в параметрах может указываться признак того, что данные надо прочитать в любом случае, даже если они не обновлялись. Если данные обновились после предыдущего их чтения, то функция возвращает true.

5.7.5. Для работы со считанными ТИ/ТС используются указатели на соответствующие массивы структур TI или TS. Индекс в массиве для конкретного ТИ/ТС можно получить используя CSIdents->GetTINum и CSIdents->GetTSNum.

Чтение сообщений осуществляется выполнением метода ReadMes. В параметрах можно указывать типы сообщений, которые надо считывать. В этом случае остальные типы сообщений считываться не будут. Если набор считываемых типов сообщений не указан, то считываются все сообщения.

5.7.5. Считанные сообщения хранятся в контейнере структур (MsgList) с типом `TMesDataRec`:

```
struct TMesDataRec:TMesRecord
{
    char *Data;                        //Данные сообщения
};
```

5.7.7. Работа с сообщениями производится в соответствии с их типом (MsgList->Msg). Доступ к данным осуществляется приведением элементов массива к пользовательскому типу сообщений или через MsgList->Data.

5.7.8. Быстродействие клиента данных определяется временем обновления, заданным в конфигурации (`CSClient.RefreshTime`). Таким образом, приложению – клиенту необходимо в цикле опроса выполнять паузу (Sleep) на время RefreshTime.

5.8. Библиотека сервера данных

5.8.1. Библиотека для работы с сервером каналов в качестве сервера данных содержит класс CSDataServ, обеспечивающий взаимодействие с Сервером каналов. Заголовочный файл библиотеки приведен в листинге 5.

Листинг 5. Библиотека CSDataServ

```
#pragma once

#include "CScadaUnits.h"
#include "fmstream.h"
#include "MsgQueue.h"
#include "OIKDef.h"
#include "ServProc.h"
#include "IniFile.h"

class CDataServ
{
    std::string sTIFileName;
    std::string sTSFileName;
    std::string sMesFileName;
    int MesFileSize;
```

```
    int ReadedTimeOut;
    int TSBegin;
    int TIBegin;
    bool RelativeAddr;
    int sLockedTimeOut;
    int sReadedTimeOut;
    fmstream *sFMapTI;
    fmstream *sFMapTS;
    fmstream *sFMapMes;
    CMsgQueue WriteMsgQueue;
    bool WorkDataServerFlag;
    bool SetUnrealStarting;
    bool SetUnrealStoping;
    bool CanUnrealMode;
    void InitBuffers();
public:
    int MesTask;                    //Идентификатор задачи
    CAnarSession *sChanelSrv;       //Ссылка на открытую сессию сервера приложений
    bool TSPresent;                 //Сервер может передавать ТС
    bool TIPresent;                 //Сервер может передавать ТИ
    bool MesPresent;                //Сервер может передавать сообщения
    std::string ServName;           //Имя сервера
    int SleepTime;                  //Пауза в цикле сервера данных
    int TSCount;                    //Количество ТС
    int TICount;                    //Количество ТИ
    bool FullStartDataServer;       /Сервре полностью запущен
    bool ExitFlag;                  //Признак того, что пришла команда "Завершить работу"

    TTSVed * TSBuffer;              //Буфер для записи ТС
    TTIVed * TIBuffer;              //Буфер для записи ТИ
    bool TSModify;                  //Признак того, что ТС изменились и их надо записать в СК
    bool TIModify;                  //Признак того, что ТИ изменились и их надо записать в СК
    std::vector <std::string> UnprocessedCommands;  //Необработанные команды

    CDataServ(void);
    ~CDataServ(void);
    void ReadIni(CIniFile* ini);    //Читаем ini-файл
    bool InitDataServer(bool UseRelativeAddr);//Загружаем сервер каналов
    void StartDataServer();         //Запускаем сервер (информируем сервер каналов)
    void StopDataServer();          //Останавливаем сервер (информируем сервер каналов)
    void FreeDataServer();          //Удаляем все. Актуально, когда класс статический
```

```
    bool ProcessCommands();          //Обрабатывает команды. false - Выход
    bool SetCommandResult(const char* ResStr);      //Послать ответ на команду
    void SetRealTimeMode();          //Сообщить о переходе в режим реального времени
    void SetUnRealTimeMode();        //Сообщить о переходе в режим нереального времени

    bool WriteTS();                  //Записывает буфер ТС
    bool WriteTI();                  //Записывает буфер ТИ
    bool WriteMsg(TMesRecord* MQMsg);       //Ретрансляция сообщения в очередь
    bool WriteMsg(                   //Запись сообщения в очередь
                int Msg,             // - тип сообщения
                unsigned int DataSize,// - размер данных
                void *Data           // - указатель на данные
                );
};
extern CDataServ CSDataServer;
```

5.8.2. Объект CSDataServ создается статически. Перед началом работы с сервером каналов в качестве сервера данных необходимо выполнить инициализацию.

```
CSIdents=new CCSIdents(); // создание и загрузка объекта для работы с идентификаторами.
```

```
CSDataServer.ReadIni(CSIdents->LocalIni);  // чтение идентификаторов.
```

5.8.2. Перед началом работы надо запустить сервера данных и проверить успешность его запуска:

```
if (!CSDataServer.StartDataServer())
{
      …
}
```

5.8.3. В приложении сервера данных необходимо организовать отдельный цикл работы с сервером каналов следующего вида:

```
while (CSDataServer.ProcessCommands())
{
      if (CSDataServer.FullStartDataServer)
      {
      //Здесь выполняется формирование ТИ,ТС и отправка сообщений
            …
```

```
            CSDataServer.WriteTS();
            CSDataServer.WriteTI();
      }
      Sleep(CSDataServer.SleepTime);
}
```

5.8.3. Быстродействие сервера данных определяется паузой в цикле работы сервера данных. Пауза в конфигурации задается в миллисекундах (`CSDataServer.SleepTime`). Таким образом, приложению – серверу данных необходимо в цикле опроса выполнять паузу (Sleep) на время `CSDataServer.SleepTime`.

5.8.4. Запись телесигналов в сервер каналов осуществляется следующим образом. В массив ТС записываются все ТС, которые надо обновить в данный момент:

```
CSDataServer.TSBuffer[iTS].TC=<Значение>;
```

5.8.5. Также необходимо для каждого ТС присвоить «время записи в сервер каналов»:

```
CSDataServer.TSBuffer[iTS].WriteDateTime=<Текущее время в формате FILETIME>;
```

5.8.5. При необходимости присваивается время устройства: `CSDataServer.TSBuffer[iTS].UnitDateTime`.

5.8.7. Фактическая запись всего объема подготовленных ТС в сервер каналов осуществляется командой CSDataServer.WriteTS() в основном цикле сервера данных.

5.8.9. Аналогичным образом осуществляется запись телеизмерений. В массив ТИ записываются все ТИ, которые надо обновить в данный момент:

```
CSDataServer.TIBuffer[iTI].Value=<Значение>;
CSDataServer.TIBuffer[iTI].TIFlag=<Признаки достоверности>;
```

5.8.9. Если есть необходимость может записываться и значение в квантах:

```
CSDataServer.TIBuffer[iTI].Kvant=<Значение в квантах>;
```

5.8.10. Также необходимо для каждого ТИ присвоить «время записи в сервер каналов»:

```
CSDataServer.TIBuffer[iTI].WriteDateTime=<Текущее время в формате FILETIME>;
```

5.8.11. При необходимости присваивается время устройства: `CSDataServer.TIBuffer[iTI].UnitDateTime`.

5.8.12. Фактическая запись всего объема подготовленных ТИ в сервер каналов осуществляется командой CSDataServer.WriteTI() в основном цикле сервера данных.

5.8.13. Получение индекса ТИ или ТС (iTI, iTS) с помощью CSIdents->GetTINum и CSIdents->GetTSNum.

5.8.14. Запись сообщений выполняется с помощью CSDataServer.WriteMsg. В параметрах указывается тип сообщения, размер пользовательских данных сообщения и ссылка на сами данные.

5.8.15. Для ретрансляции готового сообщения (например, полученного от другого источника данных) выполняется функцией CSDataServer.WriteMsg с указанием в качестве параметра указателя на сформированное сообщение.

Список сокращений

SCADA	- Supervisory Control And Data Acquisition (Диспетчерский контроль и сбор данных)
АСУ	- автоматизированная система управления
ПО	- программное обеспечение
АРМ	- автоматизированное рабочее место
ПО	- программное обеспечение
СПО	- системное программное обеспечение

Перечень ссылочных документов

1. ГОСТ 19.101-77 Виды программ и программных документов.
2. ГОСТ 19.103-77 Обозначение программ и программных документов.
3. ГОСТ 19.104-78 Основные надписи.
4. ГОСТ 19.105-78 Общие требования к программным документам.
5. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом
6. ГОСТ 24.207-80 Требования к содержанию документов по программному обеспечению.
7. ГОСТ 19.503-90 Правила внесения изменений
8. ГОСТ 19.504-79 Руководство программиста. Требования к содержанию и оформлению.

Контактная информация

Центр исследований и разработок «Интеллектуальные энергосистемы»

Адрес: г. Иркутск, ул. Лермонтова, 279/4.

Телефон: (3952) 458-098,
E-mail: info@rdc-sg.com