Модуль подсистемы “Архивы” <FSArch>

Модуль:	FSArch
Имя:	Архиватор на файловую систему
Тип:	Архив
Источник:	arh_FSArch.so
Версия:	1.3.1
Автор:	Роман Савоченко
Описание:	Модуль архива. Предоставляет функции архивирования сообщений и значений на файловую систему.
Лицензия:	GPL

Оглавление

Введение

Модуль предназначен для архивирования сообщений и значений системы OpenSCADA на файловую систему.

Любая SCADA система предоставляет возможность архивирования собранных данных, т.е. формирование истории изменения (динамики) процессов. Архивы условно можно разделить на два типа: архивы сообщений и архивы значений.

Особенностью архивов сообщений является то, что архивируются, так называемые, события. Характерным признаком события является его время возникновения. Архивы сообщений обычно используются для архивирования сообщений в системе, т.е. ведение логов и протоколов. В зависимости от источника сообщения могут классифицироваться по различным критериям. Например, это могут быть протоколы аварийных ситуаций, протоколы действий операторов, протоколы сбоев связи и др.

Особенностью архивов значений является их периодичность, определяемая промежутком времени между двумя смежными значениями. Архивы значений применяются для архивирования истории непрерывных процессов. Поскольку процесс непрерывный, то и архивировать его можно только путём введения понятия квантования времени опроса, поскольку иначе мы получаем архивы бесконечных размеров ввиду непрерывности самой природы процесса. Кроме этого, практически мы можем получать значения с периодом ограниченным самими источниками данных. Например, довольно качественные источники данных, в промышленности, редко позволяют получать данные с частотой более 1кГц. И это без учёта самих датчиков, имеющих ещё менее качественные характеристики.

Для ведения архивов в системе OpenSCADA предусмотрена подсистема «Архивы». Данная подсистема, в соответствии с типами архивов, состоит из двух частей: архив сообщений и архивы значений. Подсистема в целом является модульной, что позволяет создавать архивы, основанные на различной природе и способах хранения данных. Данный модуль предоставляет механизм архивирования на файловую систему как для потока сообщений, так и для потока значений.

1. Архиватор сообщений

Архивы сообщений формируются архиваторами. Архиваторов может быть множество с индивидуальными настройками, позволяющими разделять архивирование различных классов сообщений.

Архиватор сообщений этого модуля позволяет хранить данные как в файлах формата языка XML, так и в формате плоского текста. Язык разметки XML является стандартным форматом, который с лёгкостью понимают многие сторонние приложения. Однако открытие и разбор файлов в таком формате требует значительных ресурсов. С другой стороны, формат плоского текста требует значительно меньше ресурсов, хотя и не является унифицированным, а также требует знания его структуры для разбора.

В любом случае поддерживаются оба формата и пользователь может выбрать любой из них, в соответствии со своими требованиями.

Файлы архивов именуются архиваторами, исходя из даты первого сообщения в архиве. Например так: <2006–06–21 17:11:04.msg>.

Файлы архивов могут ограничиваться по размеру и времени. После превышения лимита создаётся новый файл. Максимальное количество файлов в директории архиватора также может быть ограничено. После превышения лимита на количество файлов, старые файлы начнут удаляться!

С целью оптимизации использования дискового пространства архиваторы поддерживают упаковку старых архивов упаковщиком gzip. Упаковка производится после продолжительного неиспользования архива.

При использовании архивов в формате языка XML соответствующие файлы загружаются целиком! Для выгрузки неиспользуемых продолжительное время архивов применяется таймаут доступа к архиву, после превышения которого архив выгружается из памяти, а затем и пакуется.

Модулем предоставляются дополнительные параметры настройки процесса архивирования (рис.1).

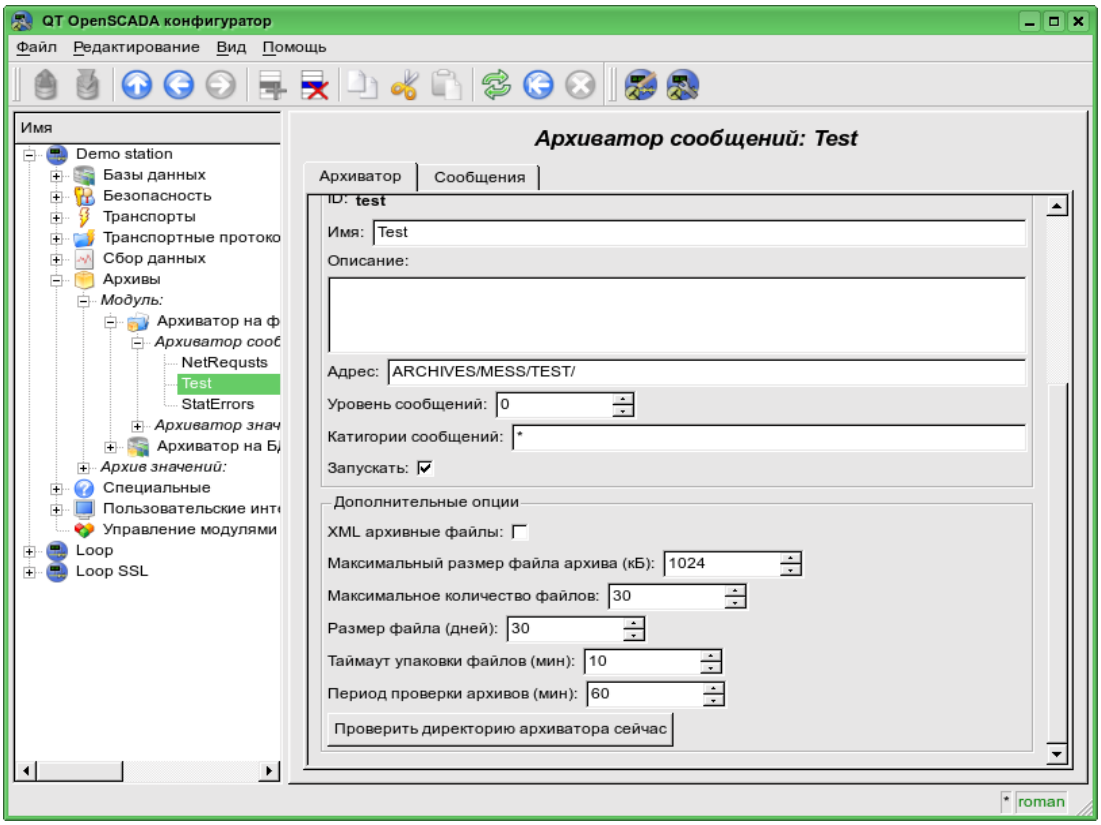

Рис.1. Дополнительные параметры настройки процесса архивирования сообщений модуля FSArch

В число этих параметров входят:

- В число этих параметров входят:
- Выбора XML-формата архивных файлов.
- Максимальный размер одного файла архива.
- Максимальное количество архивных файлов.
- Ограничение размера файла архива по времени.
- Таймаут упаковки файлов архива.
- Периодичность проверки файлов архиватором на предмет поиска новых архивов и удаление старых.
- Команда немедленной проверки директории архиватора. Может использоваться при размещения в директории архиватора файлов архивов из другой станции.

1.1. Формат файлов архива сообщений

В таблице ниже приведен синтаксис файла архива, построенного на XML-языке:

Тег	**Описание**	**Атрибуты**	**Содержит**
FSArch	Корневой элемент. Идентифицирует файл как принадлежащий данному модулю.	*Version* — версия файла архива; *Begin* — время начала архива (hex – UTC в секундах от 01/01/1970); *End* — время окончания архива (hex – UTC в секундах от 01/01/1970).	(m)
m	Тег отдельного сообщения.	*tm* — время создания сообщения (hex – UTC в секундах от 01/01/1970); *tmu* — микросекунды времени сообщения; *lv* — уровень сообщения; *cat* — категория сообщения.	Текст сообщения

Архивный файл на основе плоского текста состоит из:

- заголовка в формате: [FSArch <vers> <charset> <beg_tm> <end_tm>]

  Где:
  - <*vers*> — версия модуля архивирования;
  - <*charset*> — кодировка файла (обычно UTF8);
  - <*beg_tm*> — UTC время начала архива с эпохи 01.01.1970 в шестнадцатеричной форме;
  - <*end_tm*> — UTC время конца файла архива с эпохи 01.01.1970 в шестнадцатеричной форме.

- записей сообщений в формате: [<tm> <lev> <cat> <mess>]
  Где:
  - **<*tm*> —** время сообщения в виде: **<utc_sec:usec>**, где:
    - *utc_sec* — UTC время с эпохи 01.01.1970 в шестнадцатеричной форме;
    - *usec* — микросекунды времени в десятичной форме.
  -
  - <*lev*> — уровень важности сообщения;
  - <*cat*> — категория сообщения;
  - <*mess*> — текст сообщения.

Текст сообщения и категория кодируются с целью исключения символов разделителей (символ пробела).

1.2. Пример файла архива сообщения

Пример содержимого архивного файла в формате языка XML:

```
<?xml version='1.0' encoding='UTF-8' ?>
<FSArch Version="1.3.0" Begin="4a27dfbc" End="4a28c990">
<m tm="4a28c982" tmu="905587" lv="4"
cat="/DemoStation/sub_DAQ/mod_DiamondBoards/">Ошибка dscInit.</m>
<m tm="4a28c990" tmu="595549" lv="4"
cat="/DemoStation/sub_Transport/mod_Sockets/out_HDDTemp/">Ошибка подключения к
Internet сокету: Операция выполняется в данный момент!</m>
</FSArch>
```

Пример содержимого архивного файла в формате плоского текста:

```
FSArch 1.2.0 UTF-8 4a27dfbb 4a28c991
4a28c98f:432619 1 /DemoStation/ Запуск!
4a28c98f:432858 1 /DemoStation/sub_Transport/ Пуск%20подсистемы.
4a28c98f:455400 1 /DemoStation/sub_DAQ/mod_DAQGate/cntr_test/ Включение
   %20контроллера!
4a28c98f:457360 1 /DemoStation/sub_DAQ/mod_ModBus/cntr_testTCP/ Включение
   %20контроллера!
4a28c98f:460691 1 /DemoStation/sub_DAQ/mod_ModBus/cntr_testRTU/ Включение
   %20контроллера!
4a28c98f:464227 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node/ Включение
   %20контроллера!
4a28c98f:680767 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102cntr/ Включение
   %20контроллера!
4a28c98f:705683 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node_cntr/
   Включение%20контроллера!
4a28c98f:753659 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM101/ Включение
   %20контроллера!
4a28c98f:905073 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102/ Включение
   %20контроллера!
4a28c990:81670 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM201/ Включение
   %20контроллера!
4a28c990:206208 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM202/ Включение
   %20контроллера!
4a28c990:333471 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM301/ Включение
   %20контроллера!
4a28c990:457490 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM302/ Включение
   %20контроллера!
4a28c990:591702 1 /DemoStation/sub_DAQ/mod_System/cntr_AutoDA/ Включение
   %20контроллера!
4a28c990:595549 4 /DemoStation/sub_Transport/mod_Sockets/out_HDDTemp/ Ошибка
   %20подключения%20к%20Internet%20сокету:%20Операция%20выполняется%20в%20данный
   %20момент!
4a28c990:618617 1 /DemoStation/sub_DAQ/mod_SoundCard/cntr_test/ Включение
   %20контроллера!
4a28c990:621487 1 /DemoStation/sub_DAQ/mod_LogicLev/cntr_experiment/ Включение
   %20контроллера!
4a28c990:729323 1 /DemoStation/sub_DAQ/mod_JavaLikeCalc/cntr_testCalc/ Включение
   %20контроллера!
4a28c990:733434 1 /DemoStation/sub_DAQ/mod_Siemens/cntr_test/ Включение
   %20контроллера!
4a28c990:754368 1 /DemoStation/sub_DAQ/mod_DAQGate/cntr_test/ Включение
   %20контроллера!
4a28c990:786925 1 /DemoStation/sub_Archive/ Пуск%20подсистемы.
4a28c990:955967 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node/ Запуск
   %20контроллера!
```

2. Архиватор значений

Архивы значений формируются архиваторами значений индивидуально для каждого зарегистрированного архива. Архиваторов может быть множество с индивидуальными настройками, позволяющими разделить архивы по различным параметрам, например, по точности и глубине.

Архив значений является независимым компонентом, который включает буфер, обрабатываемый архиваторами. Основным параметром архива значения является источник данных. В роли источника данных могут выступать атрибуты параметров подсистемы «Сбор данных», а также другие внешние источники данных (пассивный режим). Другими источниками данных могут быть: сетевые архиваторы удалённых OpenSCADA систем, среда программирования системы OpenSCADA и др. Не менее важными параметрами архива являются параметры его буфера. От параметров буфера зависит возможность работы архиваторов. Так, периодичность значений в буфере должна быть не больше периодичности самого быстрого архиватора, а размер буфера не менее двойного размера для самого медленного архиватора. В противном случае возможны потери данных!

Общая схема архивирования значений наглядно изображена на рис. 2.

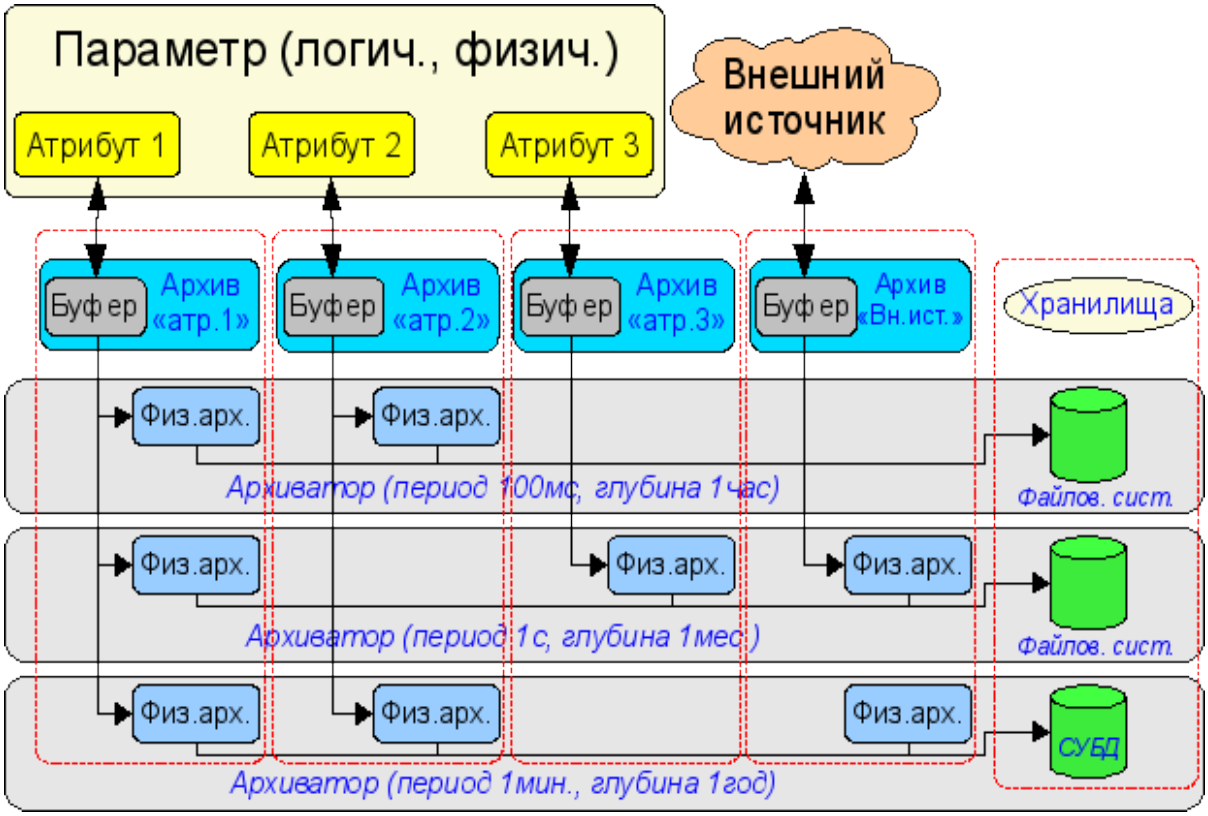

Рис.2. Общая схема процесса архивирования значений модуля FSArch.

Файлы архивов именуются архиваторами, исходя из даты первого значения в архиве и идентификатора архива. Например, таким образом: <MemInfo_use 2006–06–17 17:32:56.val>.

Файлы архивов могут ограничиваться по времени. После превышения лимита создаётся новый файл. Максимальное количество файлов в директории архиватора также может ограничиваться. После превышения лимита на количество файлов старые файлы начнут удаляться!

С целью экономии дискового пространства архиваторы поддерживают упаковку в дополнении к последовательной упаковке старых архивов упаковщиком gzip. Упаковка производится после продолжительного неиспользования архива.

Модулем предоставляются дополнительные параметры настройки процесса архивирования (рис.3).

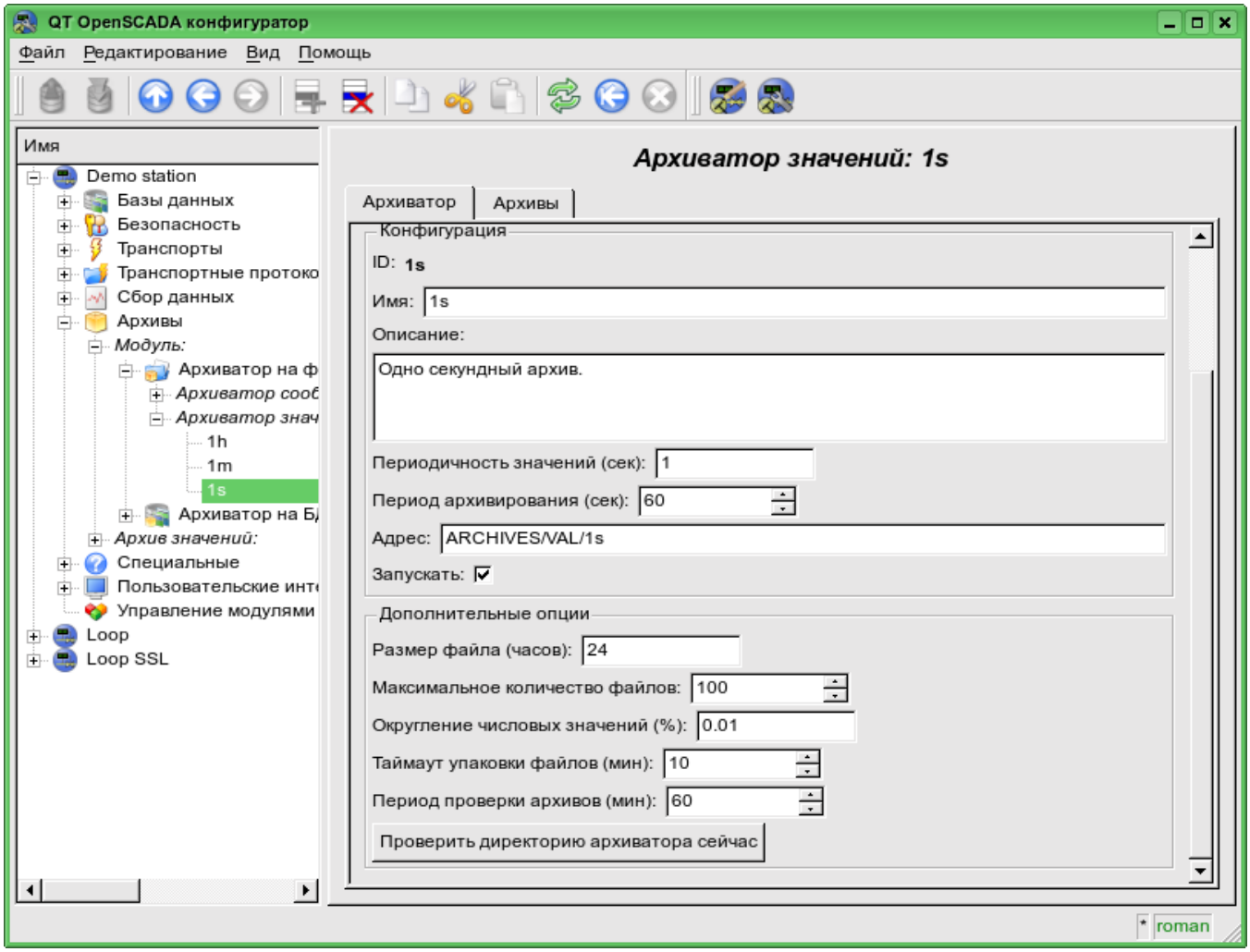

Рис.3. Дополнительные параметры настройки процесса архивирования значений модуля FSArch.

2.1. Формат файлов архива значений

Для реализации архивирования на файловую систему предъявлялись следующие требования:

- быстрый (простой) доступ на добавление в архив и чтение из архива;
- возможность изменения значений в существующем архиве (с целью заполнения дыр в дублированных системах);
- цикличность (ограничение размера);
- возможность сжатия методом упаковки последовательности одинаковых значений, сохраняющим возможность быстрого доступа (последовательная упаковка);
- возможность упаковки устаревших данных стандартными архиваторами (gzip, bzip2 ...) с возможностью распаковки при обращении.

В соответствии с вышеизложенными требованиями организовано архивирование методом множественности файлов (для каждого источника). Цикличность архива реализуется на уровне файлов, т.е. создается новый файл, а самый старый удаляется. Для быстрого сжатия используется метод притягивания к последнему одинаковому значению. Для этих целей в файле архива предусматривается битовая таблица упаковки размером один в один с количеством хранимых данных. Т.е. каждый бит соответствует одному значению в архиве. Значение бита указывает на наличие значения. Для потока одинаковых значений биты обнулены. В случае с архивом строк таблица является не битовой а байтовой и содержит длину соответствующего значения. В случае поступления потока одинаковых значений, длина будет нулевой и читаться будет первое одинаковое значение. Поскольку таблица байтовая, то архив сможет хранить строки длиной не более 255 символов. Таким образом, методики хранения можно разделить на методику данных фиксированного и нефиксированного размера. Общая структура файла архива приведена на рис.4.

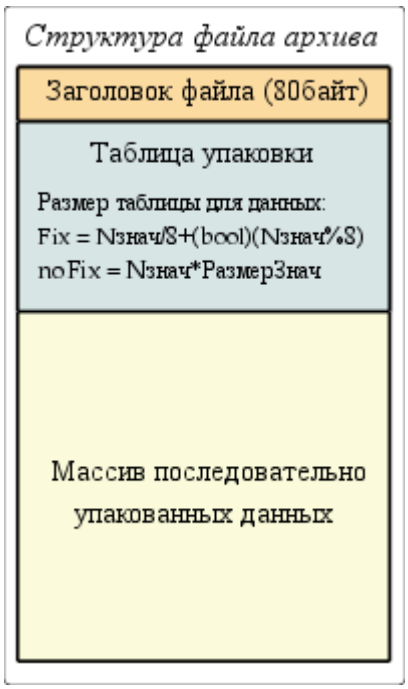

Рис. 4. Общая структура файла архива значений.

При создании нового файла архива формируется: заголовок (структура заголовка в таблице 1), нулевая битовая таблица упаковки архива и первое недостоверное значение. Таким образом, получится архив, инициализированный недостоверными значениями. В дальнейшем новые значения будут вставляться в область значений с корректировкой индексной таблицы упаковки. Из этого следует, что пассивные архивы будут вырождаться в файлы размером в заголовок и битовую таблицу.

Таблица 1. Структура заголовка файла архива

Поле	**Описание**	**Размер байт (бит)**
f_tp	Системное имя архива («OpenSCADA Val Arch.»)	20
archive	Имя архива, которому принадлежит файл.	20
beg	Время начала архивных данных (мкс)	8
end	Время конца архивных данных (мкс)	8
period	Периодичность архива (мкс)	8
vtp	Тип значения в архиве (Логический, Целый, Вещественный, Строка)	(3)
hgrid	Признак использования жёсткой сетки в буфере архива	(1)
hres	Признак использования времени высокого разрешения (мкс) в буфере архива	(1)
reserve	Резерв	14
term	Символ окончания заголовка архива (0x55)	1

Разъяснение механизма последовательной упаковки приведено на рис. 5. Как можно видеть из рисунка признак упаковки содержит длину (не фиксированные типы) или признак упаковки (фиксированные типы) отдельно взятого значения. Это значит, что для получения смещения нужного значения необходимо сложить длины всех предыдущих действительных значений. Выполнение данной операции каждый раз и для каждого значения является крайне накладной операцией. Поэтому был внедрён механизм кеширования смещений значений. Механизм кеширует смещения значений через предопределённое их количество, а также кеширует смещение последнего значения к которому производился доступ (отдельно на чтение и запись).

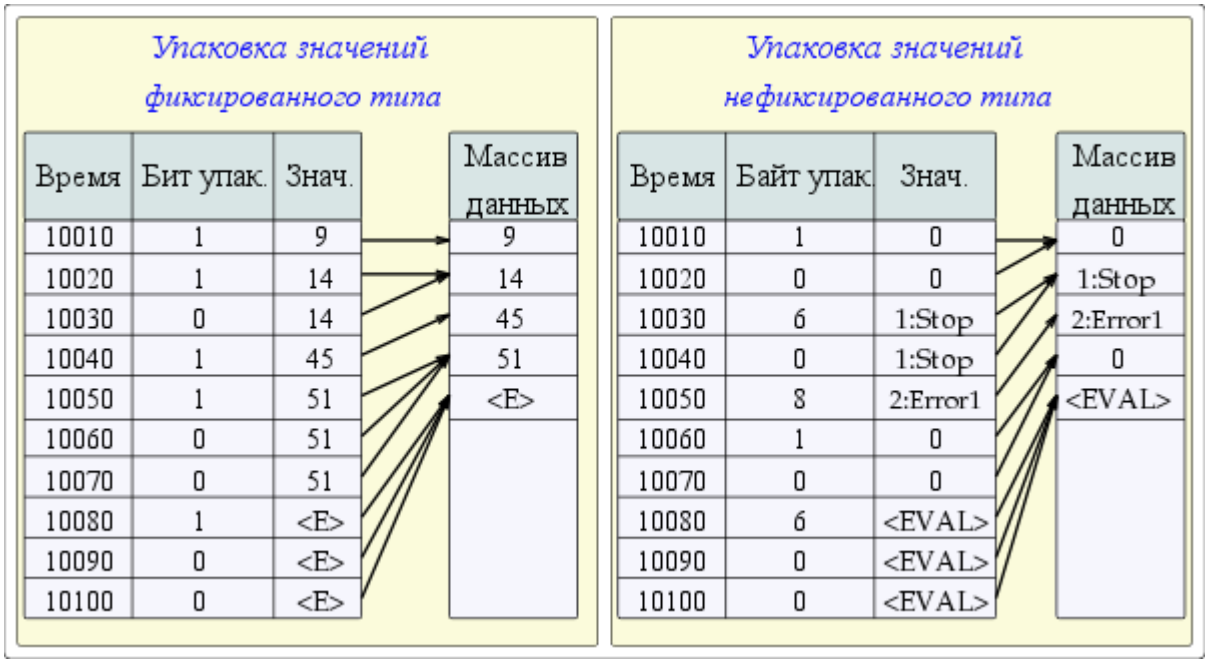

Рис. 5. Механизм последовательной упаковки значений.

Изменения значений внутри существующего архива также предусмотрено. Однако, учитывая необходимость выполнения сдвига хвоста архива, рекомендуется выполнять эту операцию как можно реже и как можно большими блоками.

3. Эффективность

При проектировании и реализации данного модуля были заложены механизмы повышения эффективности процесса архивирования.

Первым механизмом является механизм блочного(покадрового или транзактивного) помещения данных в файлы архивов значений. Такой механизм позволяет достичь максимальной скорости архивирования, а следовательно и позволяет одновременно архивировать больше потоков данных. Опыт практического применение показал, что система K8–3000 с обычным IDE жестким диском способна архивировать до 300000 потоков данных с периодичностью 1 секунда или, система K5–400 с IDE диском (2.5 дюйма) способна архивировать до 100 параметров с периодичностью 1 миллисекунда.

Вторым механизмом является упаковка как текущих значений, так и устаревших файлов архивов для оптимизации используемого дискового пространства. Реализовано два механизма упаковки: механизм последовательной упаковки (архивы значений) и механизм дожатия архивов стандартным упаковщиком (gzip). Данный подход позволил достичь высокой производительности в процессе архивирования текущих данных с эффективным механизмом последовательного сжатия. А дожатие стандартным упаковщиком устаревших архивов завершает общую картину компактного хранения больших массивов данных. Статистика практического применения в условиях реального зашумленного сигнала(худшая ситуация) показала, что степень последовательной упаковки составила 10%, а степень полной упаковки составила 71%.