Module of subsystem “Archives”<FSArch>

Module:	FSArch
Name:	Arhivator on the file system
Type:	Archive
Source:	arh_FSArch.so
Version:	1.3.0
Author:	Roman Savochenko
Description:	Archive module. Provides archiving functions for messages and values on the file system.
License:	GPL

Table of contents

Introduction

The module is designed for archiving messages and values of OpenSCADA on the file system.

Any SCADA system provides the ability to archive the collected data, i.e. formation of history of the changes (dynamics) of processes. Archives conditionally can be divided into two types: archives of messages and archives of values.

A feature of the archives of messages is that so-called events are archived. The characteristic feature of the events is its time of occurrence. The archives of messages are usually used for archiving, messages in the system, i.e. conducting of logs and reports. Depending on the source the messages can be classified according to different criteria. For example, this may be the reports of emergency situations, the reports of actions of the operators, reports of the glitches of connection and others.

A feature of the archives of values is their frequency, measured in the time lag between two adjacent values. Archives of values are used for archiving the history of continuous processes. As the process is continuous, it can only be archived by introducing the notion of quantization of time interviewing, because otherwise we get the archives of infinite dimensions in view of continuity of the nature of the process. In addition, practically, we can get value from the time limited by the data sources. For example, a fairly high-quality data sources in the industry, are rarely allowed to receive data at a frequency of more than 1kHz. And this is without taking into account of the sensors themselves, which have even less qualitative characteristics.

For conducting of archives in the system OpenSCADA the subsystem «Archives» is provided. This subsystem, according to the types of archives, consists of two parts: an archives of messages and archives of values. The subsystem, in general, is a module that allows you to create archives based on the different nature and methods of storing of data. This module provides a mechanism for the archiving on the file system for both: for the flow of messages, and for the flow of values.

1. Message Archiver

Archives of messages are formed by archiver. There can be the set of archivers, with individual settings, allowing to share archiving of different classes of messages.

The archiver of messages of this module allows you to store data in XML files or in the flat-text format. Markup language XML is a standard format that is easily understood by a lot of exterior applications. However, opening and reviewing of the files in this format requires considerable resources. On the other hand, the flat-text format requires far fewer resources, although not uniform, but also requires knowledge of its structure to deal with.

In any case, both formats are supported and the user can select any of them in accordance with his requirements.

Files of the archive are named by archivers based on the date of the first messages in the archive. For example so: <2006–06–21 17:11:04. Msg>.

Files of the archive can be limited in size and time. After exceeding the limit a new file is created. Maximum number of files in a directory of the archiver can also be restricted. After exceeding the limit on the number of files old files will be deleted!

In order to optimize the use of disk space archivers support package of old archives by gzip packer. Packaging is made after a long non-use of the archive.

When you are using the archives in the form of XML, appropriate files are loaded entirely! For a long time unused archives unloading timeout of access to the archive is used, after the exceeding of which the archive is unloaded from memory and then is packaged.

Module provides additional settings for the archiving process (Fig. 1).

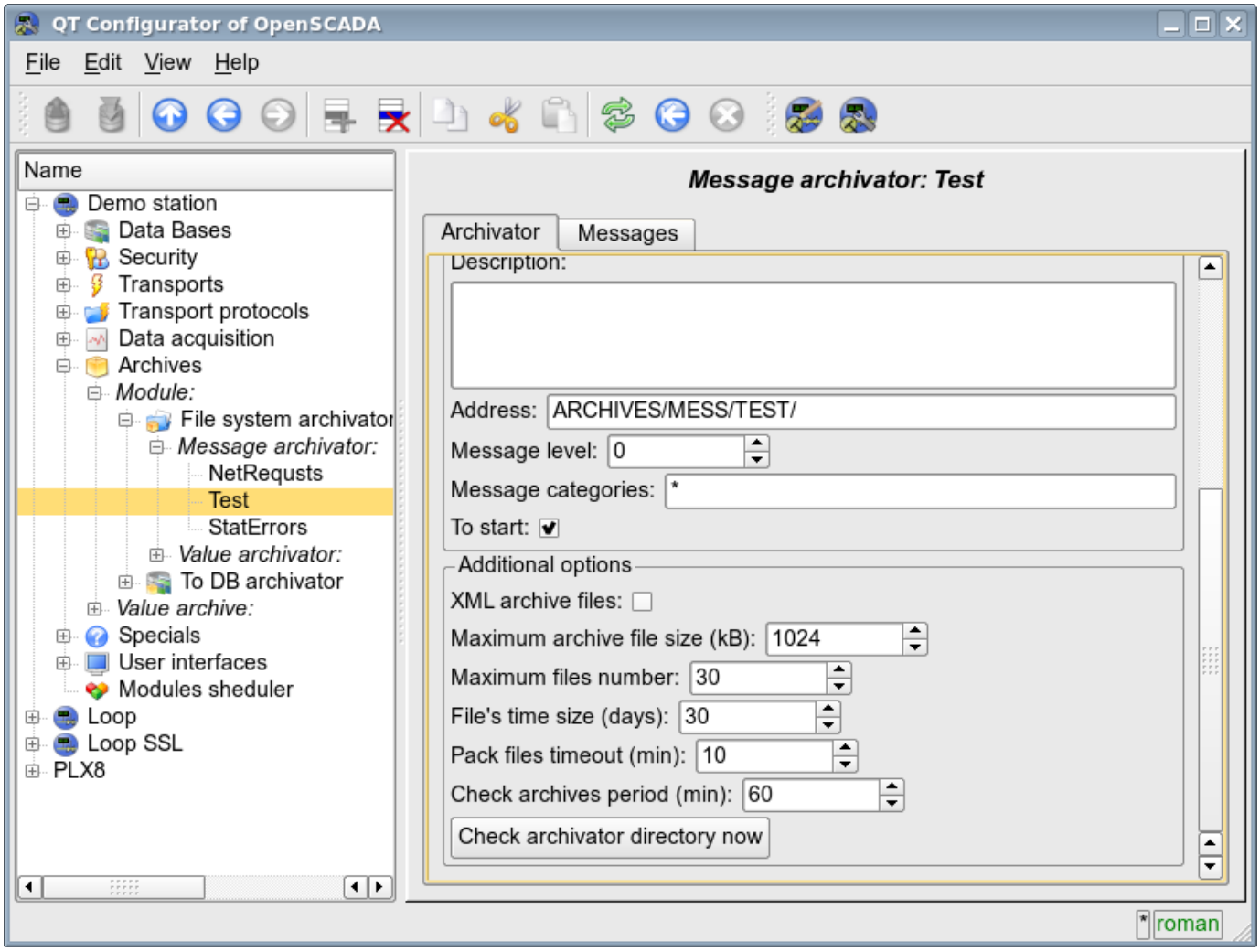

Fig.1. Additional settings of an archiving process of messages by module FSArch.

Those parameters include:

- Selecting of XML-format archive files.
- The maximum size of a single archive file.
- The maximum number of archived files.
- Limiting the size of the archive over time.
- Timeout of package of archive files.
- The frequency of inspection of files by archiver to search for new archives and deleting the old ones.
- The command of immediately verification of directory of archiver. It can be used with the placement in the directory of archiver of files of archives from other stations.

1.1. File format of archive messages

The table below shows the syntax of the archive file based on the XML-language:

Tag	Description	Attributes	Contains
FSArch	The root element. Identifies the file as belonging to the module.	*Version* — version of the archive file; *Begin* — the start time for the archive (hex – UTC in seconds from 01/01/1970); *End* — the end time for the archive (hex – UTC in seconds from 01/01/1970).	(m)
m	Tag of the single message.	*tm* — time of creation of the message (hex – UTC in seconds from 01/01/1970); *tmu* — microseconds of message's time; *lv* — message level *cat* — category of message.	Text of message

Archive file on the basis of the flat text consists of:

- header in the format: [FSArch <vers> <charset> <beg_tm> <end_tm>]
  Where:
  - <*vers*> — version of the archiving module;
  - <*charset*> — code page of the file (usually UTF8);
  - <*beg_tm*> — UTC start time for the archive from 01.01.1970, in hexadecimal form;
  - <*end_tm*> — UTC end time for the archive 01.01.1970, in hexadecimal form.
- records of the messages in the format: [<tm> <lev> <cat> <mess>]
  Where:
  - **<*tm*> — message time in format <utc_sec:usec>**, where:
    - *utc_sec* — UTC time from 01.01.1970, in hexadecimal form;
    - *usec* — microseconds of time, in decimal form.
  -
  - <*lev*> — the level of importance of the message;
  - <*cat*> — category of the message;
  - <*mess*> — text of the message.

Text of the message and its category are coded to exclude separator symbols (space character).

1.2. Example of the archive of messages file

Example of the contents of an archive file in format of the XML language:

```
<?xml version='1.0' encoding='UTF-8' ?>
<FSArch Version="1.3.0" Begin="4a27dfbc" End="4a28c990">
<m tm="4a28cd01" tmu="942937" lv="4"
   cat="/DemoStation/sub_DAQ/mod_DiamondBoards/">dscInit error.</m>
<m tm="4a28cd12" tmu="466631" lv="4"
   cat="/DemoStation/sub_Transport/mod_Sockets/out_HDDTemp/">Connect to Internet
   socket error: Operation now in progress!</m>
</FSArch>
```

Example of the contents of the archive file in the format of flat text:

```
FSArch 1.3.0 UTF-8 4a27dfbb 4a28cd12
4a28cd11:295857 1 /DemoStation/ Start!
4a28cd11:296091 1 /DemoStation/sub_Transport/ Start%20subsystem.
4a28cd11:304391 1 /DemoStation/sub_DAQ/mod_DAQGate/cntr_test/ Enable%20controller!
4a28cd11:306362 1 /DemoStation/sub_DAQ/mod_ModBus/cntr_testTCP/ Enable%20controller!
4a28cd11:310956 1 /DemoStation/sub_DAQ/mod_ModBus/cntr_testRTU/ Enable%20controller!
```

```
4a28cd11:313845 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node/ Enable
  %20controller!
4a28cd11:531765 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102cntr/ Enable
  %20controller!
4a28cd11:557546 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node_cntr/ Enable
  %20controller!
4a28cd11:616320 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM101/ Enable%20controller!
4a28cd11:770404 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102/ Enable%20controller!
4a28cd11:935745 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM201/ Enable%20controller!
4a28cd12:64148 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM202/ Enable%20controller!
  4a28cd12:212514 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM301/ Enable
  %20controller!
4a28cd12:331423 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM302/ Enable%20controller!
4a28cd12:462627 1 /DemoStation/sub_DAQ/mod_System/cntr_AutoDA/ Enable%20controller!
4a28cd12:466631 4 /DemoStation/sub_Transport/mod_Sockets/out_HDDTemp/ Connect%20to
  %20Internet%20socket%20error:%20Operation%20now%20in%20progress!
4a28cd12:499705 1 /DemoStation/sub_DAQ/mod_SoundCard/cntr_test/ Enable%20controller!
4a28cd12:502482 1 /DemoStation/sub_DAQ/mod_LogicLev/cntr_experiment/ Enable
  %20controller!
4a28cd12:620560 1 /DemoStation/sub_DAQ/mod_JavaLikeCalc/cntr_testCalc/ Enable
  %20controller!
4a28cd12:624907 1 /DemoStation/sub_DAQ/mod_Siemens/cntr_test/ Enable%20controller!
4a28cd12:644620 1 /DemoStation/sub_DAQ/mod_DAQGate/cntr_test/ Enable%20controller!
4a28cd12:665980 1 /DemoStation/sub_Archive/ Start%20subsystem.
4a28cd12:843813 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node/ Start
  %20controller!
4a28cd12:845059 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102cntr/ Start
  %20controller!
4a28cd12:845555 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_Anast1to2node_cntr/ Start
  %20controller!
4a28cd12:845983 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM101/ Start%20controller!
4a28cd12:846778 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM102/ Start%20controller!
4a28cd12:847440 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM201/ Start%20controller!
4a28cd12:849979 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM202/ Start%20controller!
4a28cd12:850851 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM301/ Start%20controller!
4a28cd12:851417 1 /DemoStation/sub_DAQ/mod_BlockCalc/cntr_KM302/ Start%20controller!
4a28cd12:852073 1 /DemoStation/sub_DAQ/mod_System/cntr_AutoDA/ Start%20controller!
4a28cd12:854718 1 /DemoStation/sub_DAQ/mod_LogicLev/cntr_experiment/ Start
  %20controller!
4a28cd12:889380 1 /DemoStation/sub_Archive/ Start%20subsystem.
4a28cd12:909319 1 /DemoStation/sub_UI/mod_VCAEngine/ Start%20module.
```

2. Values Archiver

Archives of values are formed particularly by archivers of the values for each registered archive. There cen be a lot of archivers with individual settings that allow to divide the archives by various parameters, such as the accuracy and depth.

Archive of values is an independent component, which includes buffer processed by archivers. The main parameter of archive of value is a source of data. As a source of data may make the attributes of the parameters of subsystem “Data acquisition”, as well as other external data sources (passive mode). Other sources of data could be: network archivers of remote OpenSCADA systems, environment of programming of systems OpenSCADA etc. No less important parameters are the parameters of the archive buffer. From the parameters of the buffer the opportunity of working of archivers depends on. Thus, the frequency of values in the buffer should be no more than the frequency of the fastest archiver, a buffer size not less than double the amount for the slowest archiver. Otherwise, the possible loss of data!

The overall scheme of archival of values vividly depicted in Fig. 2.

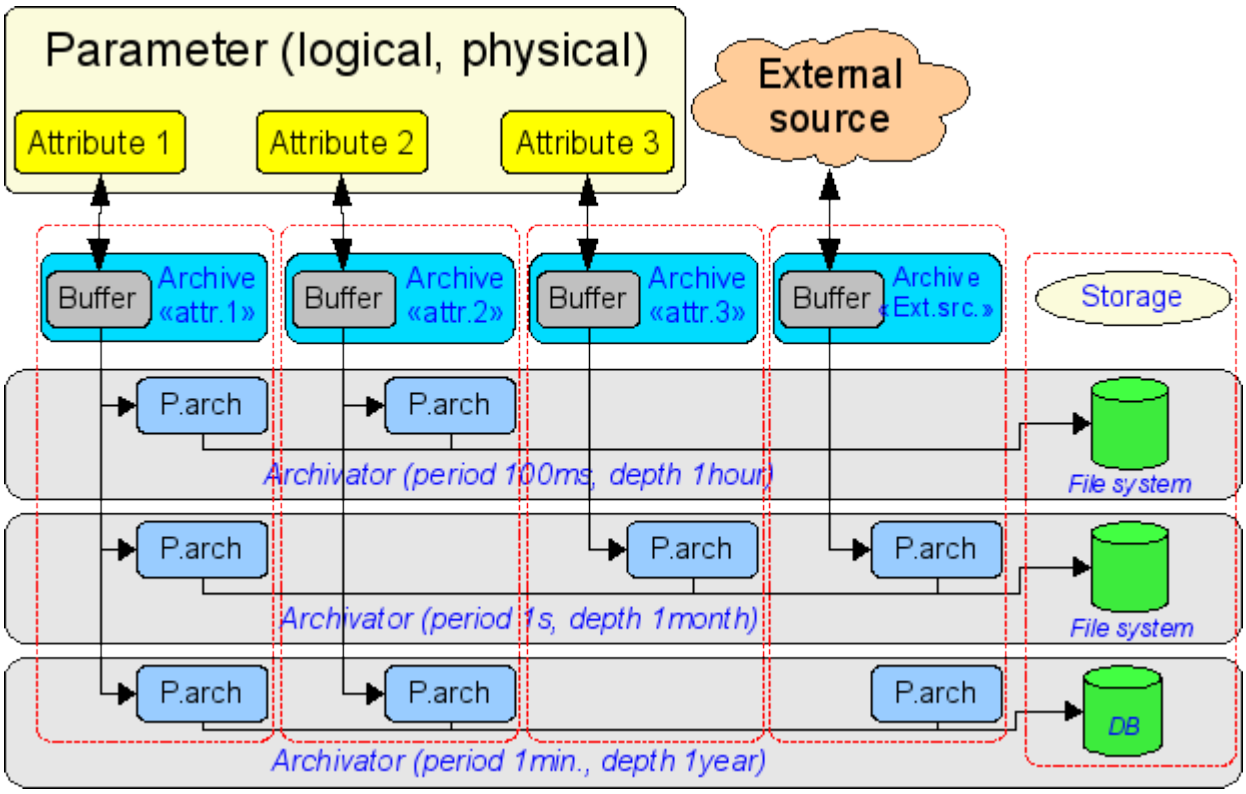

Fig.2. The overall scheme of process of archival values of module FSArch.

Files of archives are named by archivers based on the date of the first value in the archive and archive identifier. For example in this way: <MemInfo_use 2006–06–17 17:32:56.val>.

Files of archives can be limited in time. After exceeding the limit the new file is created. Maximum number of files in a directory of archiver also may be limited. After exceeding the limit on the number of files old files will be deleted!

In order to optimize the use of disk space archivers support package of old archives by gzip packer. Packaging is made after a long non-use of the archive.

Module provides additional settings for the archiving process (Fig. 3).

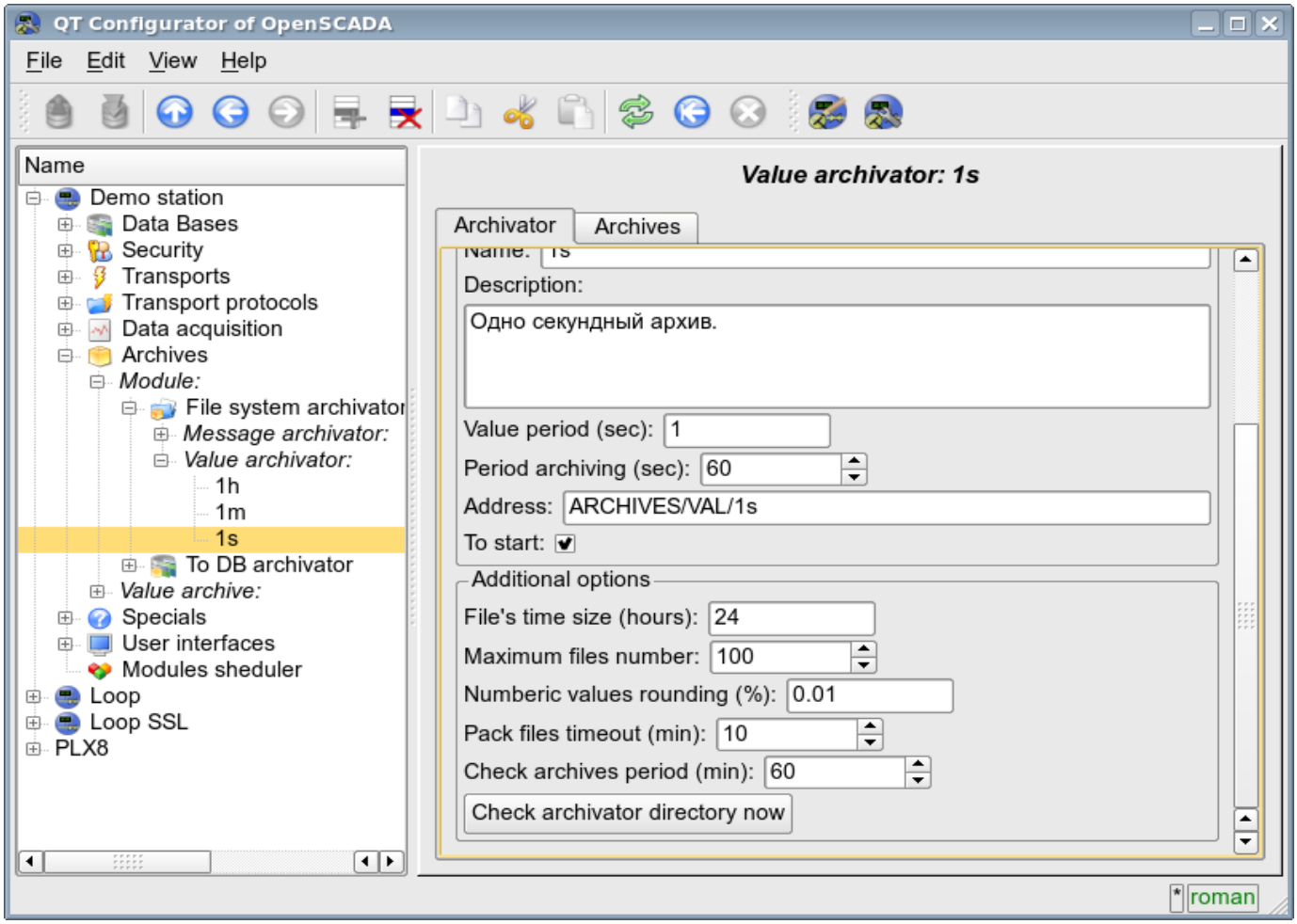

Fig.3. Additional settings of an archiving process of values by module FSArch.

2.1. File format of archive values

To implement the archiving to the file system the following requirements are to be done:

- quick (easy) access to add to the archive and reading from the archive;
- the possibility of changing the values of the existing archive (to fill holes in duplicate systems);
- cycle (size restrictions);
- the possibility of the compression by the method of packaging the same values sequence that preserves the possibility of quick access (consistent packaging);
- the possibility of packaging obsolete data by standard archivers (gzip, bzip2 ...), with the possibility of extracting on access.

In accordance with the above requirements archiving is organized by method of plurality of files (for each source). Cyclical of archive sold at the file level, ie a new file is created, and the oldest one is removed. For fast compression the method of tightening to the last equal value is used. For this purpose, the bit archiving table is provided with the size of one to one with the number of stored data. Ie each bit corresponds to the single value in the archive. The presence of bit indicates the presence of value. For the thread of the same values bits reduced to zero. In the case of the string archive the table is not a bit but the byte one and contains the length of the appropriate value. In the case of reception of the thread of equal values, the length will be zero and the first same value will be read. As the table is bite one, the archive will be able to keep strings with the length more than 255 characters. Thus, the methods of storage can be divided into a method of fixed and not fixed data size. The overall structure of the archive is shown in Fig. 4.

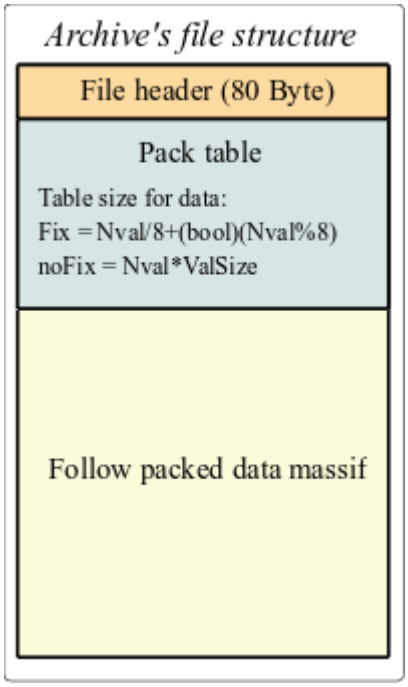

Fig. 4. The overall structure of the value archive.

When you create a new archive file there is formed: the title (the structure of the title is in the table 1), zero bit table of package of the archive and the first false value. Thus, the archive will be initialized with false values. In the future, the new values will be inserted in the area of values with adjustment of index table of packaging. It follows that the passive archives will dwindle in the files with the size of the title and the bit table.

Table 1. The structure of the header of archive file

Field	**Description**	**Size in bite(bit)**
f_tp	System name of the archive («OpenSCADA Val Arch.»)	20
archive	Name of the archive to which the file belongs.	20
beg	Start time of the archive data (мкс)	8
end	End time of the archive data (мкс)	8
period	Periodicity of the archive (мкс)	8
vtp	Type of value in the archive (Boolean, Integer, Real, String)	(3)
hgrid	Criterion of using of hard grid in the buffer of the archive	(1)
hres	Criterion of using of time of high resolution (mcs) in the buffer of the archive	(1)
reserve	Reserve	14
term	The symbol of the end of the header of file (0x55)	1

Explaining of the mechanism of consistent packaging is given in Fig. 5. As can be seen from the figure a sign of the package contains a length (not fixed types) or a sign of the package (fixed types) of the separately taken value. This means that to obtain the desired value of displacement it is necessary to sum up the length of previous valid values. The implementation of this operation each time and for each value is highly invoice operation. Therefore, the mechanism of caching of displacement of the values is provided. The mechanism caches displacement of values through predefined their quantity, as well as cashes the last value for which the access is made (separately for reading and writing).

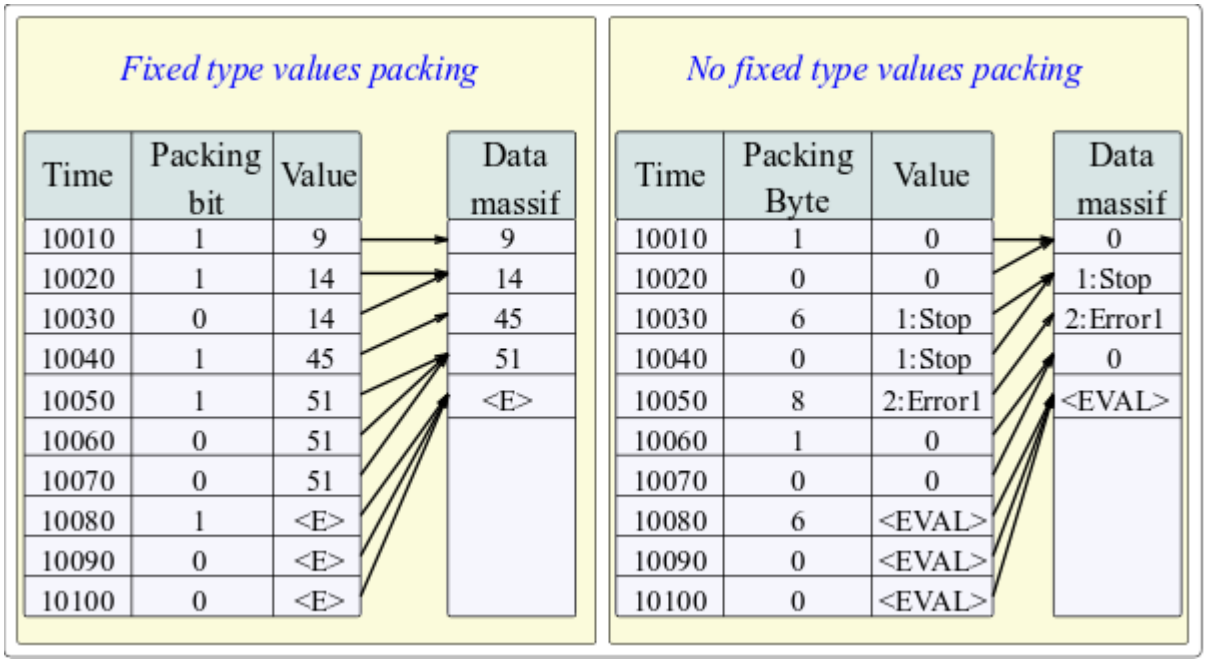

Fig. 5. The mechanism of follow packaging of values.

Changes of the values in the existing archive is also provided. However, given the necessity to implement the shifting of the tail of the archive, it is recommended to perform this operation as sparingly as possible and with as far as possible large blocks.

3. Efficiency

In the design and implementation of the module it was built mechanisms improving the process of archiving.

The first mechanism is a mechanism of block (frame-accurate or transactive) location of data in the files of the archives of values. Such an arrangement allows to achieve a maximum speed of archiving, and thus allows to archive more data streams at the same time. The experience of the practical using showed that the system of K8–3000 with a regular IDE hard drive is able to archive to 300000 data streams at a frequency of 1 second, or K5–400 system with the IDE drive (2.5”) can archive to 100 parameters with 1 millisecond intervals.

The second mechanism is the package of current values, and outdated files of archives to optimize the use of disk space. There are two packaging mechanisms: the consistent package (archives of values), and a mechanism of finish packaging of archives by means of standard packer (gzip). This approach allowed to achieve high productivity in the process of archiving of current data with the effective mechanism of consistent compression. And finish packaging by means of standard packer of obsolete archives completes the overall picture of the compact storage of large volumes of data. Statistics of practical using, in real noise signal (the worst situation), showed that the extent of consistent packaging is 10%, and the extent of the full packaging was 71%.