OpenSCADA program description

Roman Savochenko (rom_as@oscada.org)

Maxim Lisenko (mlisenko@oscada.org)

27. May. 2010

Contents table

Introduction.

The given document is the description "open source" project of system called "OpenSCADA." OpenSCADA corresponds an open SCADA system constructed on principles of modularity, multiplatform and scalability.

As a policy of development of the given system the "open source" principles are chosen. The choice of the given policy is determined by necessity of creation of open, reliable and public SCADA system. The given policy allows to involve in development, testing, elaboration, distribution and use of a product the significant amount of developers, enthusiasts and other interested persons with minimization and distribution of financial expenses.

The OpenSCADA system is intended for acquisition, archiving, visualization of the information, delivery of operating influences, and also for other related operations characteristic for full-function SCADA systems. Owing to a high level of abstraction and modularity, the system can be used in many adjacent areas.

The OpenSCADA system can be applied:
- on industrial targets, as full-function SCADA system;
- in built in (embedded) systems, as an environment of performance, including inside PLC (programmed logic controllers);
- for construction of various models (technological, chemical, physical, electric processes);
- on personal computers, servers and clusters for acquisition, processing, representation and archiving of the information on system and its environment.

As basic (host) operational system, for development and use, the OS Linux is chosen, which is the optimum compromise in questions:
- reliability (vast majority of servers and clusters works on GNU/Linux);
- flexibility/scalability (in view of the openness and modularity allows to build decisions under any requirements);
- availability (owing to license GPL it is completely free system, and at high qualification of the user and free-of-charge);
- popularity, development, support, prevalence (the system is actively developed by set of enthusiasts, firms and official bodies from all over the world, it gets greater and greater support in the user and corporate market, it is actively implemented into the state structures of the various countries).

As far as the project is developed and realized by principles of multiplatformity, it does not make a problem to port it on other OS, that is planned in the future.

Heart of system is the modular kernel. And depending on that, what modules are connected, system can to act both in a role of various servers, and in a role of various clients, and also to combine these functions in one program. It allows to work in practice client-server architecture of SCADA system on the basis of the same components/modules, saving thus: machine memory, disk space, and also valuable time of programmers.

Server configurations of system are intended for acquisition, processing, delivery of influences, archiving, recording of the information from various sources, and also for granting of this information to clients (UI, GUI, TUI...). The modular architecture allows to expand functionality of a server without its restarting.

Client configurations can be built on the basis of various graphic libraries (GUI/TUI ToolKits), as using a kernel of the program and its modules (by addition to it the module of UI-user interface), and as the independent application, connecting the kernel of OpenSCADA as library.

The opportunity of a flexible configuration of system allows to build decisions under concrete requirements of reliability, functionality and the sizes of system.

1. Functions of the system.
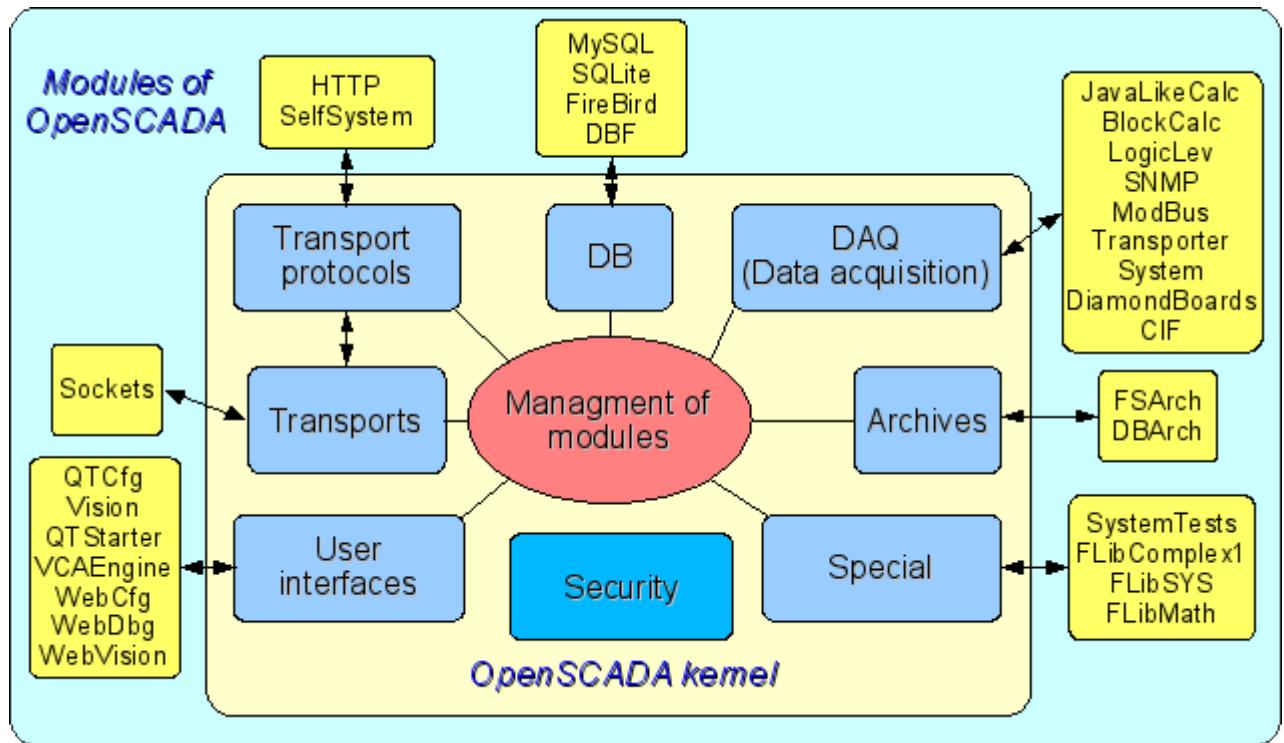
Fig. 1. The block scheme of OpenSCADA system

1.1. Modularity.

For giving flexibility and a high degree of scalability the OpenSCADA system is constructed by a modular principle. Close integration of modules with a kernel of system imposes the great responsibility on process of a writing of modules and enters an element of instability into the system, however owing to an opportunity of creation of the allocated configurations, this danger smooths out with preservation of a high degree of flexibility.

Modules of OpenSCADA system are stored in dynamic libraries. Each dynamic library can contain set of modules of various type. Filling of dynamic libraries by modules is determined by functional connectivity of modules. Dynamic libraries suppose hot replacement, that allows during functioning to update separate parts of system. The method of storage of a code of modules in dynamic libraries is the main for OpenSCADA as far as it is supported practically by all modern operational systems(OS). However it does not exclude an opportunity of development of other methods of storage of a code of modules.

On the basis of modules the following functional parts of OpenSCADA system are realized:
- databases;
- archives (messages and values);
- protocols of communication interfaces;
- communication interfaces, transports;
- sources of data and data acquisition;
- interfaces of the user (GUI, TUI, WebGUI, speech, signal...);
- the additional and special modules.

Management of modules is carried out by a subsystem "Management of modules". Functions of a subsystem are: connection, switching-off, updating of modules, and also other operations connected with modules and libraries of modules.

1.2. Subsystems.

Architecturally the OpenSCADA system shares on subsystems. Subsystems can be of two types: usual and modular. Modular subsystems possess the property of expansion by means of modules. Each modular subsystem can contain set of modular objects. For example the modular subsystem of "Database" contains modular objects of types of databases. The modular object is a root inside of the module.

In total the OpenSCADA system contains 9 subsystems from them 7 subsystems are modular. 9 subsystems of the OpenSCADA system are basic and are present at any configuration. To the list of 9 subsystems new subsystems by means of modules can be added. Subsystems of the OpenSCADA system:
- Archives (modular).
- Databases (modular).
- Safety.
- Interfaces of the user (modular).
- Management of modules.
- Data acquisition (modular).
- Transport protocols (modular).
- Special (modular).
- Transports (modular).

1.3. PLC and other sources of dynamic data. A subsystem "Data acquisition".

For support of sources of dynamic data, whether it be PLC-controllers, communication devices, virtual sources, etc., the subsystem "Data acquisition" is intended. Functions of this subsystem include granting the received data in the structured appearance and maintenance of management with these data, for example the updating of data.

The subsystem "Data acquisition" is modular and, as consequence, contains modular objects of types of sources of dynamic data. For example, for October 2007г, the OpenSCADA system supports following types of sources of data:
- Cards of data acquisition from "Diamond systems".
- Data acquisition from operational systems (OS).
- the Block calculator.
- the Calculator in Java-like language.
- the Transporter of data of a subsystem "Data acquisition" from one OpenSCADA station to another.
- Access to logic controllers by means of the protocol "ModBUS".
- Data acquisition from network devices by means of protocol SNMP.
- The source of data of a logic level of OpenSCADA system.
- Access to highly intellectual logic controllers by means of MPI protocol and communication processor CIF50PB of Hilscher GMBH firm.

Each type of a source is made in the form of the separate module which can be connected/disconnected. Each type of a source can contain separate sources (controllers).

Separately taken controller can contain the parameters of certain by the module types. For example parameters of analog type: the basic information which they gives the value of the integer or real type is. Structurally, the parameter represents the list of attributes which are contained by data. Attributes can be of four base types: symbolical string (text), integer, real and logic type.

Structures of controllers, parameters and their types are contained in the subsystem "Data acquisition", and objects of modules carry out their filling according to own specificity.

The source of dynamic data can be remotes, i.e. connected on the remote OpenSCADA system. For communication with such sources of data the transport type of controllers (Transporter) is used. Function of the given type of a source of data is reflection of sources of data of remote OpenSCADA stations on local station.

1.4. Databases. A subsystem of "Database"

For a data storage of system databases (DB) are everywhere used. With a view of systematization of access and management of databases in OpenSCADA system the subsystem "Database" is provided. For support of various DB/DBMS the subsystem is modular.

In a role of the modular objects, containing in a subsystem, type DB/DBMS acts, i.e. the module of a subsystem "Database", which practically contains realization of access to the certain type of a DB. For example modules: DBF, MySQL, SQLite.

The object of type DB/DBMS, in its turn, contains the list of objects of separated DB of the given type. And the object of a DB contains the list of objects of tables which are contained by data in the tabulated form.

Practically all the data of OpenSCADA system are stored in this or that DB. The toolkit of system allows to transfer easily the data from one type of a DB on another and as consequence provide an optimum selection of DB type under the concrete area of OpenSCADA system. Transfer of the information from one DB to another can be made by two ways. The first is a change of the address of a working DB and save of all system on it, the second is a direct copying the information between DB. Except for copying the function of direct editing of contents of tables of a DB is supported also.

For the organization of the centralized access of the allocated system to a uniform DB two ways are provided. The first is using of network DBMS, for example MySQL. The second way is using of transport type of a DB on local systems for access to one central DB (It is planned.). Function of a transport DB is transfer of queries to a DB on remote OpenSCADA system.

Data can be stored also in a configuration file of system. The mechanism of full reflection of structure of a DB on structure of a configuration file is realized. I.e. the standard configuration can be placed in a configuration file. An essence of such mechanism that by default for example at start without a DB, it is possible to describe the data of system in a configuration file. In the further, these data can be redefined in a DB. Besides for cases of impossibility of start of any DB generally, it is possible to store all data in a configuration file.

For access to databases the mechanism of registration of a DB is used. Registered DB in system are accessible to all subsystems of OpenSCADA system and can be used in their work. Owing to this mechanism it is possible to provide an allocation of data storage. For example, various libraries can be stored and extend independently, and connection of library will consist in simple registration of the necessary DB.

In the further, realization of duplication of a DB by linkage of the registered DB is planned. This mechanism will allow to increase considerably reliability of OpenSCADA system as a whole by reservation of the mechanism of a data storage. (It is planned.)

1.5. Archives. A subsystem "Archives".

Any SCADA system gives an opportunity of archiving the acquisition data, i.e. formation of history of change (dynamics) of processes. Archives, conditionally, it is possible to divide into two types: archives of messages and archives of values.

Feature of archives of messages is that the subject of archiving are, so-called, events. A characteristic attribute of event is time of occurrence of this event. Archives of messages, usually, are used for archiving messages in system, i.e. conducting logs and reports. Depending on a source, messages can be classified by various criteria. For example, it can be reports of emergencies, reports of actions of operators, reports of failures of connection, etc.

Feature of archives of values is their periodicity defined by the time interval between two adjacent values. Archives of values are applied for archiving of history of continuous processes. As far as process is continuous and it's archiving is possible only by introduction of conception of quantization of interrogation of values as differently we receive archives of the infinite sizes, in view of a continuity of the nature of process. Besides, practically, we can receive values with the period limited by sources of data. For example, qualitative enough sources of data, in the industry, data with frequency more 1kHz seldom allow to obtain. And it without taking into account sensors having even less qualitative characteristics.

For the decision of tasks of archiving data flows in OpenSCADA system the subsystem "Archives" is provided. The subsystem "Archives" allows to conduct both: archives of messages and archives of values. The subsystem "Archives" is modular. The modular object containing in a subsystem "Archives" the type of the archiver acts. The type of the archiver defines the way of a data storage, i.e. storehouse (file system, DBMS, a network, etc.). Each module of a subsystem "Archives" can realize both: archiving of messages, and archiving of values. The subsystem "Archives" can contain set of the archives served by various modules of a subsystem.

The message in OpenSCADA system is characterized: by date, by level of importance, by category and the text of the message. Date of the message specifies for the period of creation of the message. The level of importance specifies a degree of importance of the message. The category determines the address or the conditional identifier of a source of the message. Usually, the category contains a full way to a source of the message in system. The text of the message, actually, also carries meaning content of the message.

During archiving messages are passed through the filter. The filter works on a level of importance and a category of the message. The level of the message in the filter specifies that it is necessary to pass messages with specified or higher level of importance. To filtering on a category templates are used, which define what messages are applied to pass. Each archiver contains own options of the filter. Consequently it is possible to create easily various specialized archivers for archive of messages. For example archivers of messages it is possible to dedicate on:
- logs for storage of the debugging information and other working information of a server;
- various reports (the report of actions of clients, the report of infringements and exceptions, the report of events...).

The archive of values in system OpenSCADA acts as an independent component which includes the buffer processable by archivers. Key parameter of archive of value is the source of data. In a role of a source of data attributes of parameters of OpenSCADA system and also other external sources of data (a passive mode) can act. Other sources of data can be: network archivers from remote OpenSCADA systems, the environment of programming of OpenSCADA system, etc.

Key component of archiving of values of continuous processes is the buffer of values. The buffer of values is intended for intermediate storage of a file of the values received with certain periodicity (quantum of time). The buffer of values is used as for direct storage of big arrays of values in archives of values, before direct "retire" on physical carriers, and for manipulations with the staff of values, i.e. in functions of rame-accurate query of values and their placement in buffers of archives.

For the organization of the dedicated archivers, in the allocated systems it is possible to use transport type of the archiver (It is planned.). Function of transport type of the archiver is reflection of the remote central archiver on local system. As consequence, archivers of transport type carry out data transmission between local system and the archiver of the remote system, hiding from subsystems of local system the real nature of the archiver.

1.6. Communications. Subsystems "Transports" and "Transport protocols".

As far as the OpenSCADA system is pawned as is high-scaled system that support of communications should be flexible enough. For satisfaction of a high degree of flexibility, communications in OpenSCADA system are realized in subsystems "Transports" and "Transport protocols" which are modular.

The subsystem "Transports" is intended for an exchange of the not structured data between OpenSCADA system and external systems. In a role of external systems can act even remote OpenSCADA systems. Not structured data are understood as a file of symbols of the certain length. The modular object containing in a subsystem "Transports", the type of transport acts. The type of transport defines the mechanism of transfer of not structured data. For example it can be:
- sockets (TCP/UDP/UNIX);
- channels;
- shared memory.

The subsystem "Transports" includes support of input and output transports. Input transport is intended for service of external queries and sending of answers. Output transport, on the contrary, is intended for

sending messages and expectation of the answer. Consequently, input transport contains a configuration of the given station as server, and output transport contains a configuration of the remote server. The module of a subsystem "Transports" realizes support both: input and output transports.

The subsystem "Transport protocols" is intended for structurization of data received from a subsystem "Transports". As a matter of fact, the subsystem "Transport protocols" is continuation of a subsystem "Transports" and carries out functions of check of structure and integrity of the received data. So, for the indication of the protocol together with which transport should work, the special configuration field is provided. The modular object containing in a subsystem "Protocols" is the protocol. For example, transport protocols can be:
   • HTTP (Hyper Text Transfer Protocol);
   • SelfSystem (OpenSCADA the system protocol).

The full chain of connection can be written down as follows:
   • the message is transferred in transport;
   • transport transfers the message to the protocol, connected with it, by creation of new object of the protocol;
   • the protocol checks integrity of data;
   • if all data have come, transport must be informed about the termination of expectation of data and to transfer it the answer, differently to inform, that it is necessary to expect still;
   • transport, having received {confirmation, sends the answer and delete object of the protocol;
   • if confirmations are not present, the transport continues expectation of data, and in the case of their receipt transfers them to the saved object of the protocol.

Protocols for output transports are supported also. The output protocol incurs function of dialogue with transport and realization of features of the protocol. The internal side of access to the protocol is realized by data-flow way with own structure for each protocol module. Such mechanism allows to carry out transparent access to external system, by means of transport, simply specifying a name of the protocol by means of which to serve transfer.

Owing to standard API-access to transports of OpenSCADA system it is possible to change easily a way of data exchange not touching exchanging systems. For example, in the case of a local exchange it is possible to use faster transport on the basis of shared memory, and in the case of an exchange through the Internet and a local network to use TCP or UDP sockets.

1.7. Interfaces of the user. A subsystem "Interfaces of the user".

SCADA-systems as a class, assume presence of user interfaces. In OpenSCADA, for granting the user interfaces, the subsystem "The user interfaces" is provided. The user interface of OpenSCADA system is understood not only as the environment of visualization from which the end user should work, but also as everything, that concerns the user, for example:
   • environments of visualization;
   • configurators;
   • alarming and signalling devices.

The subsystem "The user interfaces" is modular. As modular object of a subsystem the concrete interface of the user actually acts. Modilarity of subsystem allows to create various interfaces of users on various GUI/TUI libraries and to use optimal of decisions in particularly taken case, for example, for environments of performance of programmed logic controllers it is possible to use configurators and visualizers on the basis of Web-technologies (WebCfg, WebUI), and in case of stationary workstations to use the same configurators and visualizers, but on the basis of libraries QT, GTK.

1.8. Security of system. A subsystem "Security".

The OpenSCADA system is the branched out system which consists of ten subsystems and can include set of modules. Consequently, granting of unlimited access by all to these resources is at least unsafe. Therefore, for differentiation of access in OpenSCADA system, the subsystem of "Security" is provided. The basic functions of a subsystem "Security" are:
- storage of registration records of users and groups of users;
- authentication of users;
- check of access rights of the user to this or that resource.

1.9. Management of libraries of modules and modules. A subsystem "Management of modules".

The OpenSCADA system is constructed by a modular principle that means presence of set of modules with which it is necessary to operate. For performance of function of management by modules of OpenSCADA system the subsystem "Management of modules" is provided. All modules, for the present moment are delivered in system by means of shared libraries (containers). Each container can contain set of modules of various type.

The subsystem "Management of modules" realizes the control over the status of containers and allows to carry out hot addition, removal and updating of containers and modules containing in them.

1.10. Unforeseen opportunities. A subsystem "Special".

Certainly, to provide all probable functions it is impossible, therefore in OpenSCADA system the subsystem "Special" is provided. The subsystem "Special" is modular and is intended for addition in OpenSCADA system unforeseen functions by modular expansion. For example, by means of a subsystem "Special" can be realized:
- tests of OpenSCADA system and its modules;
- libraries of functions of the user programming.

1.11. The user functions. Objective model and the environment of programming of system.

Any modern SCADA system should contain the mechanisms giving an opportunity to program at the user level, i.e. to contain the environment of programming. The OpenSCADA system contains such environment. By means of the environment of programming of OpenSCADA system it is possible to realize:
- Algorithms of management of technological processes.
- Large dynamic models of real time of technological, chemical, physical and other processes.
- Adaptive mechanisms of management on models.
- The user procedures of management by internal functions of system, its subsystems and modules.
- Flexible formations of structures of parameters at a level of the user, with the purpose of creation of parameters of non-standard structure and its filling on algorithm of the user.
- Auxiliary calculations.

The environment of programming of OpenSCADA system represents a complex of assets organizing the computing environment of the user. Into structure of a complex of assets are included:
- objective model of OpenSCADA system;
- modules of libraries of functions;
- computing controllers of a subsystem "Data acquisition" and other calculators.

Modules of libraries of functions give set of functions of the certain orientation expanding objective model of system. Libraries can be realized both: by the set of functions of the fixed type, and functions supposing free updating and addition.

Libraries of functions of the fixed type can be given by standard modules of system, organically

supplementing objective model. Functions of such libraries will represent the interface of access to assets of the module at a level of the user. For example, "The environment of visual data presentation" can give functions for delivery of various messages. Using these functions the user can realize interactive algorithms of communication with system.

Libraries of functions of free type give the environment of a writing of the user functions on one of programming languages. Within the limits of the module of libraries of functions mechanisms of creation of libraries of functions can be given. So, it is possible to create libraries of devices of technological processes, and in a consequence to use them by linkage. Various modules of libraries of functions can give realizations of various programming languages.

On the basis of the functions given by objective model, computing controllers are under construction. Computing controllers carry out linkage of functions with parameters of system and the mechanism of calculation.

2. SCADA systems and their structure.
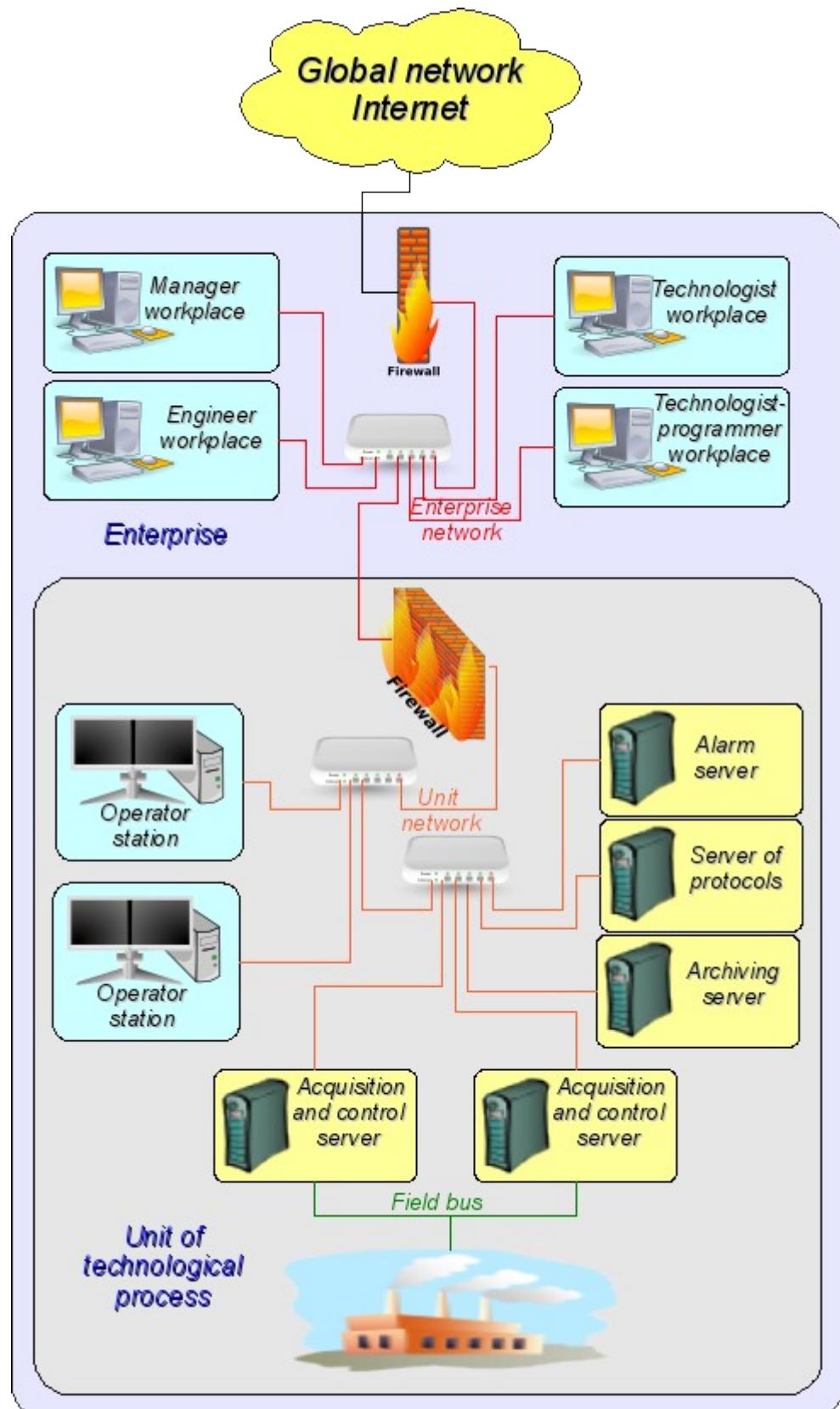
Fig. 2. SCADA-system.

SCADA (Supervisory Control And Data Acquisition), in a general view, have the allocated architecture like represented on fig. 2. Elements of SCADA systems, in sense of the software, carry out following functions:

The acquisition server: represents a task or group of tasks engaged in data acquisition from sources of data, or act in a role as a source of data. Into tasks of a server enters:
- reception and-or formation of data;
- data processing;
- service of queries about access to data;
- service of queries about updating of data.

The server of archiving: represents a task or group of tasks engaged in archiving of data. Into tasks of the server enters:
- archiving of data of SCADA-system;
- service of queries about access to contemporary records;
- import/export of archives.

The journaling server: represents a task or group of tasks engaged in archiving of messages. Into tasks of the server enters:
- archiving of messages of units of SCADA-system;
- service of queries about access to archival messages;
- import/export of archives.

The alarm server: represents a task or group of tasks carrying out functions of the server of recording concerning a narrow category of messages of the signal system.

The operator working place: represents constantly functioning GUI (Grafical User Interface) application executed in an one-monitor, multimonitor or panel mode and carrying out functions:
- granting of the user interface for the control over a condition of technological process;
- granting of an opportunity of formation of operating influences;
- granting of an opportunity of studying and the analysis of history of technological process;
- granting of toolkit for generation of the reporting documentation.

The engineer working place: represents GUI application used for configuration of SCADA system. Into tasks of the application enters:
- granting of toolkit for manipulation with system functions of system;
- granting of toolkit of a workplace of the operator;
- granting of toolkit for manipulation with architecture of SCADA system as a whole (distribution of functions between stations, creation, removal of stations...).

The chief working place: represents GUI application, as a rule, executed in an one-monitor mode and carrying out functions:
- granting of the user interface for the control over a condition of technological process;
- granting of toolkit for studying and the analysis of history of technological process as is direct from an active server, and on the basis of separate archives;
- granting of toolkit for generation of the reporting documentation.

The technologist working place: completely includes functions of a workplace of the operator plus model of technological process (without direct communication with technological process).

The work planner working place: completely includes functions of a workplace of the technologist plus toolkit for creation of models of technological processes.

3. Ways of configuration and using of OpenSCADA system.

3.1. Simple server connection.

In the elementary case the OpenSCADA system can be configured in a server mode (fig. 3.1) for acquisition and archiving of data. The given configuration allows to carry out following functions:

- interrogation of controllers;
- archiving of values of parameters;
- service of client queries about reception of various data of a server;
- granting of the configuration WEB-interface;
- the remote configuration from OpenSCADA system by means of the QT-interface or other local interface.
- secondary regulation (regulation in computing controllers);
- modeling, adjusting and supplementing calculations in computing controllers.
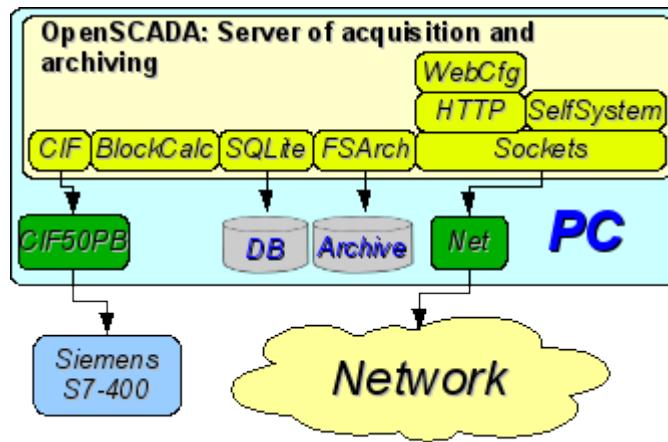
Fig. 3.1. Simple server connection.

3.2. The duplicated server connection.

For increasing of reliability and productivity the OpenSCADA system supposes plural reservation (fig. 3.2) at which controllers of one copy are reflected in other. At use of a similar configuration distribution of loading of interrogation/calculation at various stations is possible. The given configuration allows to carry out functions:

- interrogation of controllers;
- archiving of values of parameters;
- service of client queries about reception of various data of a server;
- reservation of parameters;
- reservation of archives;
- distribution of loading of interrogation on servers;
- granting of the configuration WEB-interface;
- secondary regulation (regulation in computing controllers);
- modeling, adjusting and supplementing calculations in computing controllers with an opportunity of distribution of loading on servers.
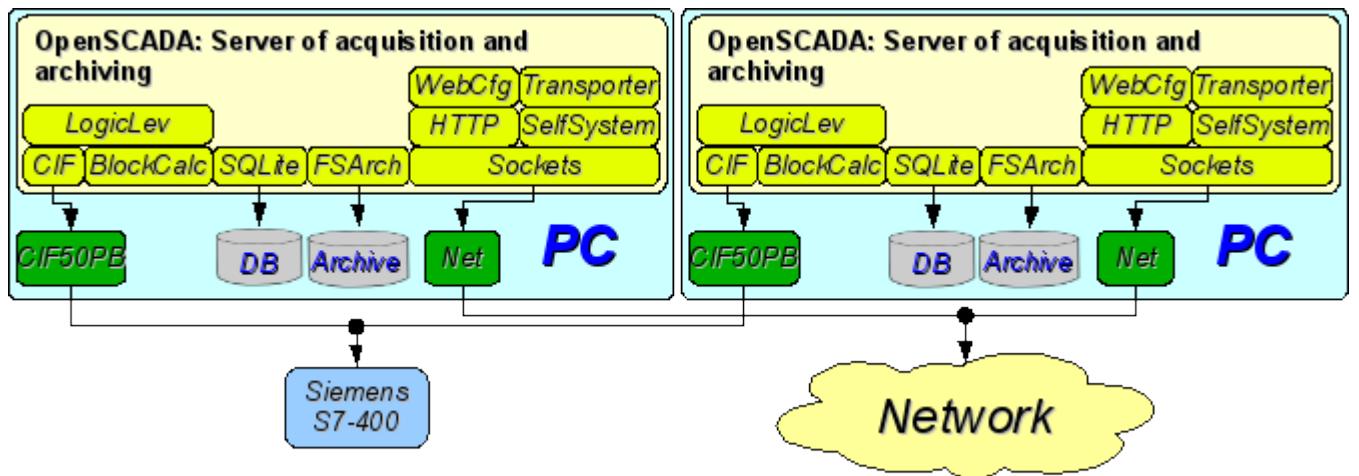
Рис. 3.2. The duplicated server connection.

3.3. The duplicated server connection on one server.

Special case of the duplicated connection is the duplicated connection within the limits of one server (fig. 3.3), that is start of several stations by one machine with a crossing of parameters. The purpose of the given configuration is increase of reliability and fault tolerance of system by reservation of software.
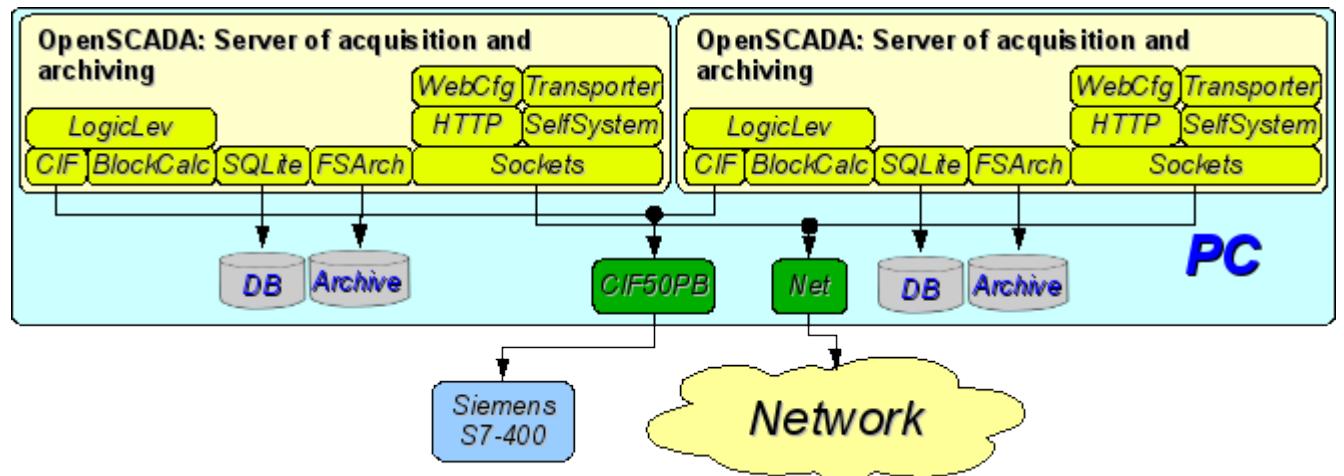
Fig. 3.3. The duplicated server connection on one server.

4.4. Client access by means of the Web-interface. A place of the manager.

For visualization of data containing on a server, the good decision is to use the user WEB-interface (fig. 3.4). The given decision allows to use a standard WEB-browser at the client side and therefore is the most flexible as it is not adhered to one platform, i.e. is multiplatform. However this decision has essential imperfections: low productivity and reliability. In this connection it is recommended to use the given method for visualization of noncritical data or data having a reserve highly reliable way of visualization. For example, the good decision will be using of this method at the heads of plants where always exists place(attendant position) with reliable way of visualization. The given configuration allows to carry out following functions:
- interrogation of a server for data acquisition of visualization and a configuration;
- visualization of data in a kind accessible to understanding;
- formation of protocols, reports;
- manipulation with parameters supposing change.
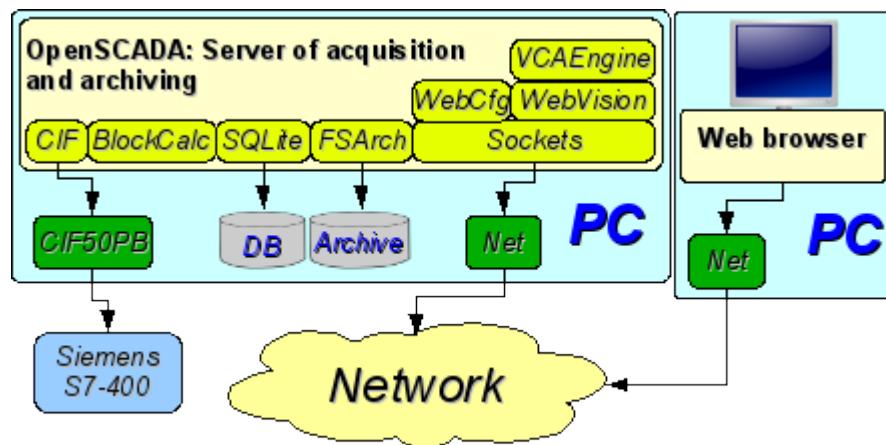
Fig. 3.4. Client access by means of the Web-interface. A place of the manager.

3.5. The automated workplace (place of the manager/operator).

For visualization of critical data, and also in case of if high quality and productivity is required, it is possible to use visualization on the basis of OpenSCADA system configured with the GUI module (fig. 3.5). The given configuration allows to carry out following functions:
- interrogation of a server for updating current values;
- visualization of the interrogated data in a kind accessible to understanding;
- formation of protocols and reports;
- manipulation with parameters supposing changes.
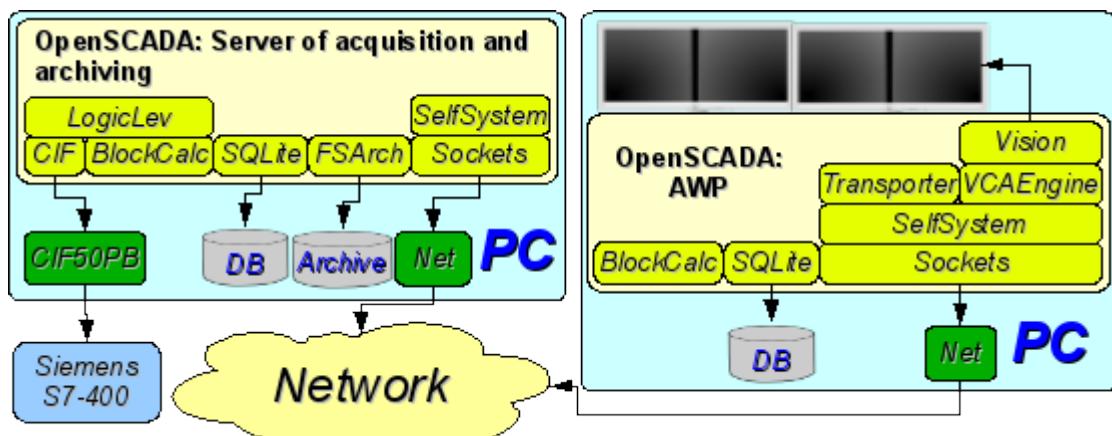


Fig. 3.5. The automated workplace (place of the manager/operator).

3.6. Automated workplace with a server of acquisition and archiving on the single machine (a place of the operator, model...).

The full-function client-server configuration on the single machine (fig. 3.6) can be used for increasing of reliability of system as a whole by start of the client and a server in different processes. The given configuration allows, without consequences for a server, to stop the client and to do with it various preventive works. It is recommended for use at stations of the operator by installation of two machines combining in itself the station of the operator and redundant server. The given configuration allows to carry out following functions:

- interrogation of controllers;
- service of client queries;
- visualization;
- delivery of operating influences;
- generation of protocols and reports;
- secondary regulation;
- modeling, adjusting and additional calculations in computing controllers;
- acquisition and visualization of the information on a personal computer, a server....
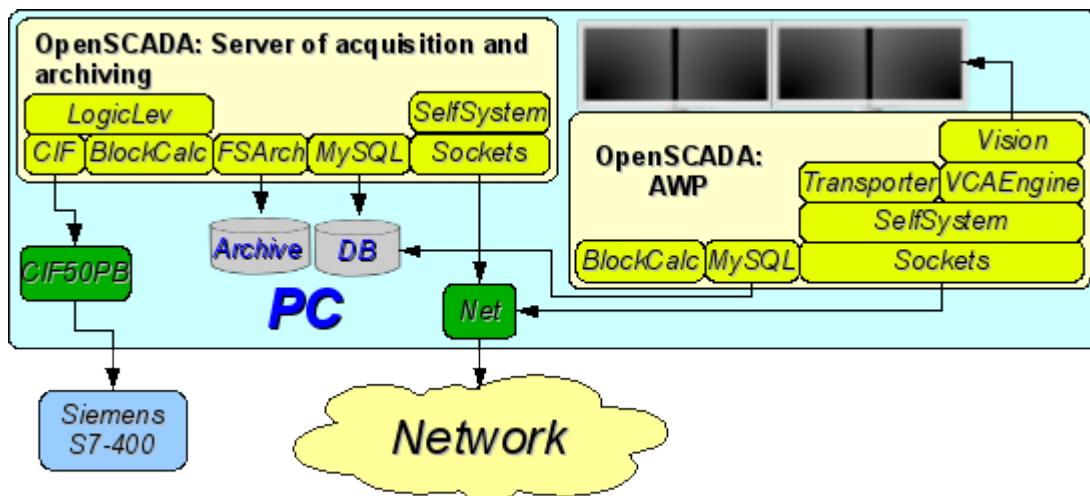
Fig. 3.6. Automated workplace with a server of acquisition and archiving on the single machine (a place of the operator, model...).

3.7. The elementary mixed connection (model, demonstration, configurator...).

The mixed connection combines functions of a server and the client (fig. 3.7). It can be used for test, demonstration functions, and also for granting models of technological processes as a unit. In this mode following functions can be carried out:

- interrogation of controllers;
- service of client inquiries;
- visualization;
- delivery of operating influences;
- generation of protocols and reports;
- secondary regulation;
- modeling, adjusting and supplementing calculations in computing controllers;
- acquisition and visualization of the current information on a personal computer, a server, model...;
- a configuration of databases, connections, etc.
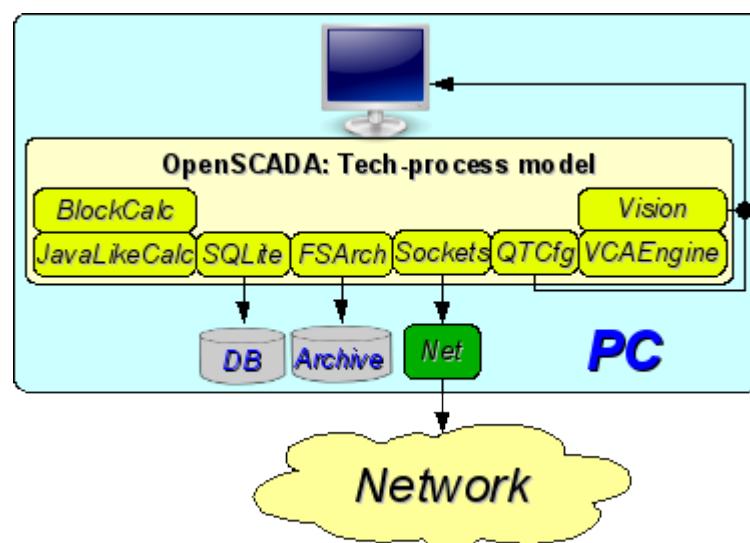
Fig. 3.7. The elementary mixed connection (model, demonstration, configurator...).

3.8. The steady, allocated configuration.

The given configuration is one of variants of steady/reliable connection (fig. 3.8). Stability is reached by distribution of functions on:
- to servers of interrogation;
- to the central server of archiving and service of client queries;
- to clients: automated workplaces and WEB-clients.
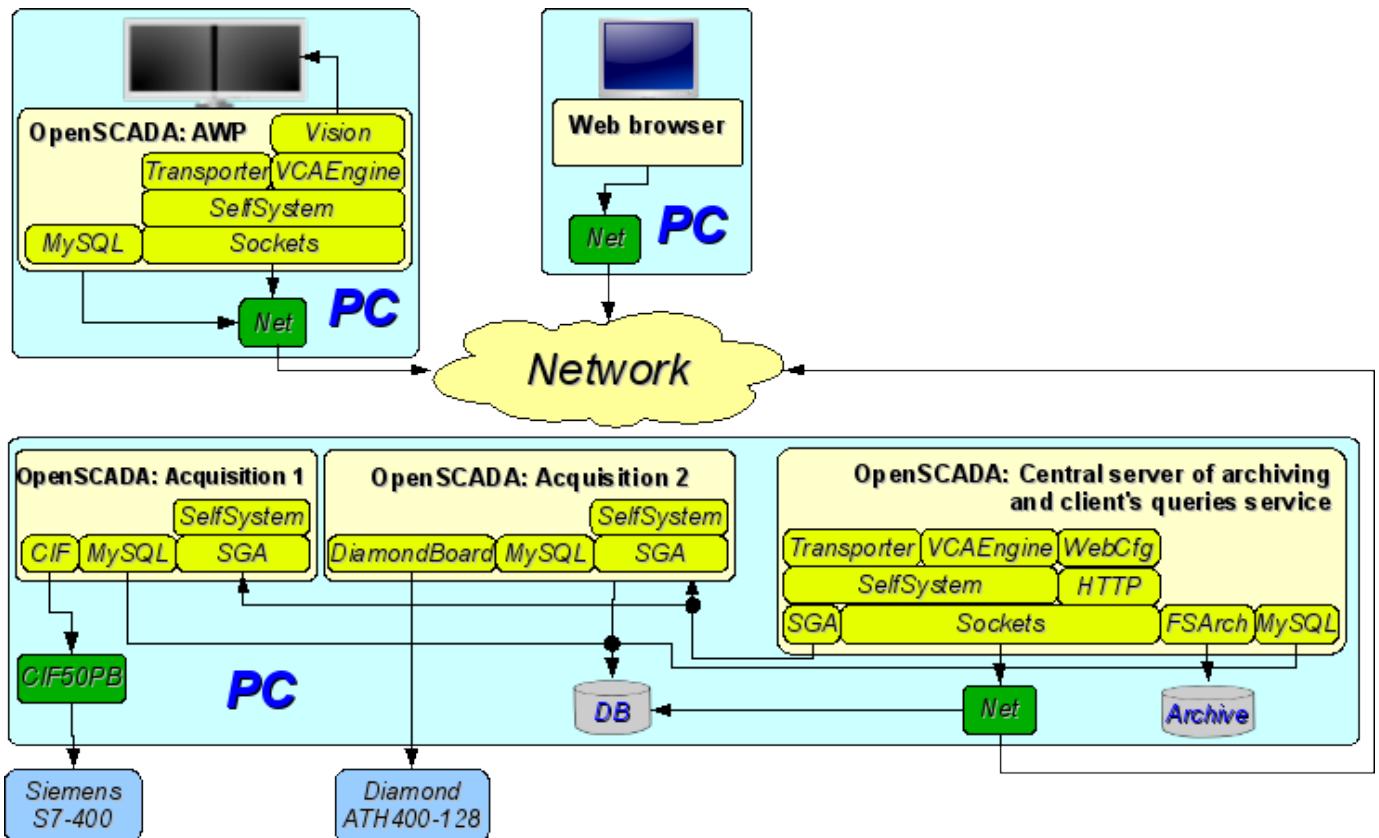
Fig. 3.8. The steady, allocated configuration.

The server of interrogation is configured on the basis of OpenSCADA system and represents the task (group of tasks) engaged with interrogation of the controller (group of controllers of the same type). The received values are accessible to the central server through any transport which support is added by connection of the corresponding module of transport. For decrease in frequency of interrogation and size of the network traffic the server of interrogation can be equipped with small archive of values. The configuration of a server of interrogation is stored in one of accessible DB.

The central server of archiving and service of client queries carries out function of the centralized acquisition and processing of parameters of servers of interrogation and their values. Access to servers of interrogation is carried out by means of one of accessible in OpenSCADA transports+protocols (for example it is SGA). For granting the uniform interface of access to parameters and controllers the module Transporter which reflects data of servers of interrogation on structure of local parameters is used.

For performance of internal calculations and the additional analysis of parameters computing controllers are used.

For versatile and deep archiving various modules of archives are used.

For access of clients to a server are used accessible for OpenSCADA network transports, for example it is Sockets, and transport protocols, for an example it is the protocol OpenSCADA "SelfSystem".

The configuration of the central server is stored in one of accessible DB (for example it is network DBMS MySQL).

For granting the user WEB-interface the module WebCfg by means of the transport protocol "HTTP" is used.

Various clients, among them automated workplaces and WEB-clients, are carried out on the separated machines in necessary quantity. The automated workplace is realized on the basis of OpenSCADA system. Its functions include interrogation of values of parameters from the central server and their visualization on the GUI interface(s). For reception of parameters in an automated workplace the module of reflection of the remote parameters Transporter, also, is used. For granting access to archives the module of archive of network type can be used. The configuration of an automated workplace can be stored in one of accessible DB (for example it is network DBMS MySQL, located on the machine of the central archiving server).

4. Configuration and adjustment of the system.

As it can be seen in the section above, OpenSCADA allows configuration for execution in various roles. Support of this possibility is provided by the developed mechanisms for configuration and storage of configuration data. This section contains a description of these mechanisms, designed to demonstrate the flexibility and diversity, thereby allowing to use OpenSCADA to 100%.

In describing the configuration mechanisms and methods of its storage in this section it will be focused the description of system-wide mechanisms. Features of the configuration of modules of subsystems of OpenSCADA are provided in their own module's documentation.

In OpenSCADA it is used the formalized approach to describing the configuration interfaces based on XML. In fact, features of the component's configuration are provided by the component itself, thereby running through the whole system, as the nervous system of the organism. In terms of OpenSCADA it is called the interface of control of OpenSCADA (Control interface). On the basis of the control interface the graphical interfaces of the user configuration are generated by means of modules of OpenSCADA. This approach has the following important advantages:
  • Scalability. You can connect only the required configuration modules or use only the remote mechanisms.
  • Excluding the need to update the configurators with the addition of new modules/functions, as well as the exclusion of "swelling" of the configurator, providing the support for all of history of now unnecessary and obsolete modules/functions.
  • Simplicity of the creation of the graphical interfaces of configuration on the different basis owing to the clear formality.
  • The possibility of dynamic configuration is available, ie configuration can be performed directly while the running of the system both locally and remotely, directly controlling the result.
  • The simple and special extensibility of the configuration interface by adding the configuration fields on the control interface's description language only in the required components.

In OpenSCADA the three configuration modules on the different basis of visualization are provided. Lets observe them and their configuration options:
  • Configuration module on the GUI library QT (http://qt.nokia.com/products) - UI.QTCfg. Provides an advanced configuration interface, allowing to operate as a local station and the remote ones in the local and global networks, including secure connection.
  • Configuration module based on the dynamic WEB-technologies (DHTML) - UI.WebCfgD. Provides an advanced configuration interface, allowing to operate as a local server's station, and the remote stations in the local and global networks, including work on the secure connection. Client connection is provided through the usual Web-browser.
  • Configuration module based on the static WEB-technologies (XHTML) - UI.WebCfg. Provides an adequate configuration interface that allows to manage the local server's station via the usual Web-browser.

Configuration values, changed in the configurators, as well as most of the data are stored in databases (DB). Given the modularity of subsystems "DB", there can be different database. Moreover, there is the possibility of storing different OpenSCADA parts in different databases of the same type and in the database of different types as well.

In addition to the database configuration information may be contained in the OpenSCADA configuration file, and passed through the command line parameter's when you call OpenSCADA. However, the saving of configuration changes in the configurators is only available in the database. Standard name of the OpenSCADA configuration file is /etc/oscada.xml. The format of the configuration file and command line parameters we'll examine in the separate section.

Further examining of the OpenSCADA configuration will be based on the interface of the configurator UI.QTCfg, but the principles of work will be fully consistent with the rest of the configurators owing to the generality in the control interface of OpenSCADA.

We will start examining with the configuration of system parameters of OpenSCADA, which is located in the three tabs at the root page of the station:

• Tab "Station" contains basic information and configuration field of the station, Fig.4a. Here are the provided fields and comments on them:

    • *ID* - contains information about the station's identifier. It is specified by the command line parameter -Station. When loading it is sought the section in the configuration file appropriate to the station identifier, and if not detected, it uses the first available one.

    • *Station* - indicates the localized station's name.

    • *Program* - contains information on the program name. Usually it is OpenSCADA or name of solution based on OpenSCADA.

    • *Version* - contains the information on the current version of the programme.

    • *Host name* - contains the information on the name of the machine that runs the station.

    • *System user* - contains the information about the user on whose behalf the program is executed in the system (OS).

    • *Operation system* - contains the information about the name and version of operation system, operation system kernel on which the program is executed.

    • *Frequency (MHZ)* - contains the information about the frequency of the CPU, which runs the program. The value of frequency is checked every 10 seconds and allows you to monitor its change, for example, by the power management mechanisms.

    • *Realtime clock resolution (msec)* - contains information about the possibility or resolution of real-time clock of the operation system. It allows you to orient with the minimum interval of time of periodic tasks, for example, for task of data acquisition.

    • *Internal charset* - contains information about the charset in which text messages are stored within the program.

    • *Config file* - contains information about the configuration file used by the program. Set by the command-line parameter -Config.

    • *Work directory* - indicates the working directory of the station. It is used in relative addressing of the objects in the file system, for example, database files. It allows the user to save the modified system data to another database. The value of this field is not stored in the database, but can be changed only in the "WorkDB" section of the configuration file.

    • *Icons directory* - indicates the directory containing the program icons. If the configuration navigation tree have no icons, then you have incorrectly entered the value of this field.

    • *Modules directory* - indicates the directory of modules for OpenSCADA. If the value of this field is incorrect, then when at start you will not see any graphical interface, but the only information in the console on the correct running of the OpenSCADA kernel.

    • *Work DB* - indicates the working database (DB), namely, the database used to store basic data of the program. Changing of this field notes all objects as modified that allows you to save or to load station's data from the specified main database.

    • *Save system at exit* - points to the need to save the changed data at finishing.

    • *Save system period* - indicates the frequency in seconds with which to save the changed station's data.

    • *Language* - indicates the language of program's messages. Changing of this field is acceptable, but leads to a change of messages' language only for the interface and dynamic messages!

    • *Text variable's base language* - is used to activate the support of multilingual text variables by specifying a non-empty basic language. The value of the basic language is selected from the list of bi-character language code, usually only the current and the base language is in the list. Further for the text variables in the non basic language in the tables of the database it will be created the separate columns. Under the text variables the all text fields of configurator, which can be translated into another language are meant. Numbers and other symbolic values are not in their number and are not translated.

    • *Messages:* - section of the parameters' group that are processing by the work and messages of the stations:

        • *Least level:* - indicates the level of messages beginning from which they are to be processed. Messages below this level will be ignored. It is necessary, for example, to exclude from processing the debug messages of level 0.

        • *To syslog* - indicates the need of sending the message to the system logger, the

mechanism of operation system for work with system messages and software. When this option is enabled the possibility appears to manage and control the OpenSCADA messages by the mechanisms of OS.

• *To stdout* - indicates the using as a standard mechanism to display the message the output to the console. Disabling of this feature will eliminate the entire output in the console, unless you specify the following parameter.

• *To stderr* - indicated the using as a standard mechanism to display the message the error output, it is also usually sent to the console.

• *To archive* - indicated the need for output of the messages in the messages' archive of OpenSCADA. This option is usually enabled and its disabling leads to the actual disabling of the archiving at the station.

• Tab "Subsystems" tab contains the list of subsystems (Fig. 4b) and allows you to jump directly to them using the context menu.

• Tab "Tasks" contains the table with opened tasks by OpenSCADA components (Fig.4c). From table you can get several information about the tasks.

• Tab "Help" tab contains the brief help for that page, Fig. 4d. In this case, it is the available command line parameters and fields of configuration file for this page.

To modify the fields of this page it may be required the super user's rights. Get these rights you can by means of including your user into the superuser's group "root", or by entering the station from the superuser "root".

We must mention another one important point: the fields of the identifiers of all OpenSCADA objects are unacceptable for direct editing, because they are keys for storage of objects' data in the database. However, to change the object's identifier you can by the command of cutting and the further pasting of the object (Cut-> Paste) in the configurator.
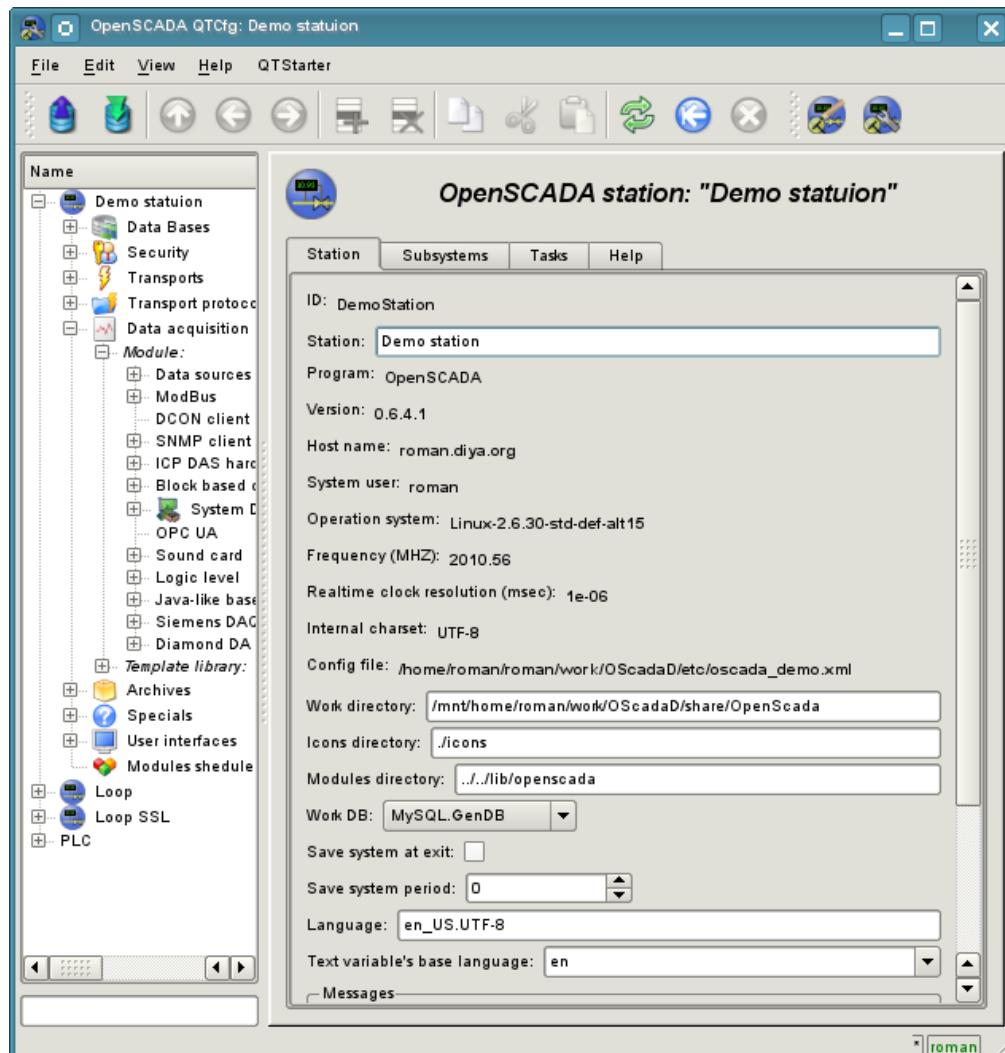
Fig. 4a. "Station" tab of the main page of the configuration of the station.
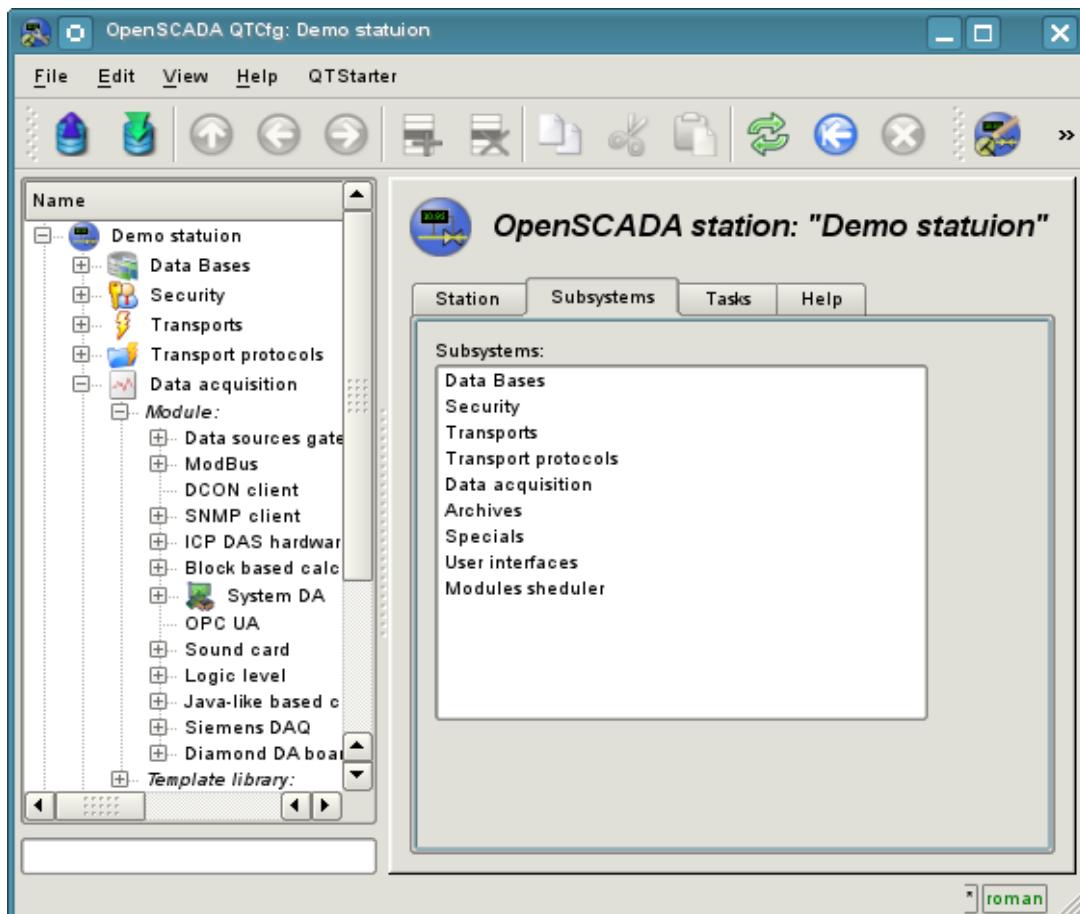
Fig 4b. "Subsystems" tab of the main page of the configuration of the station.
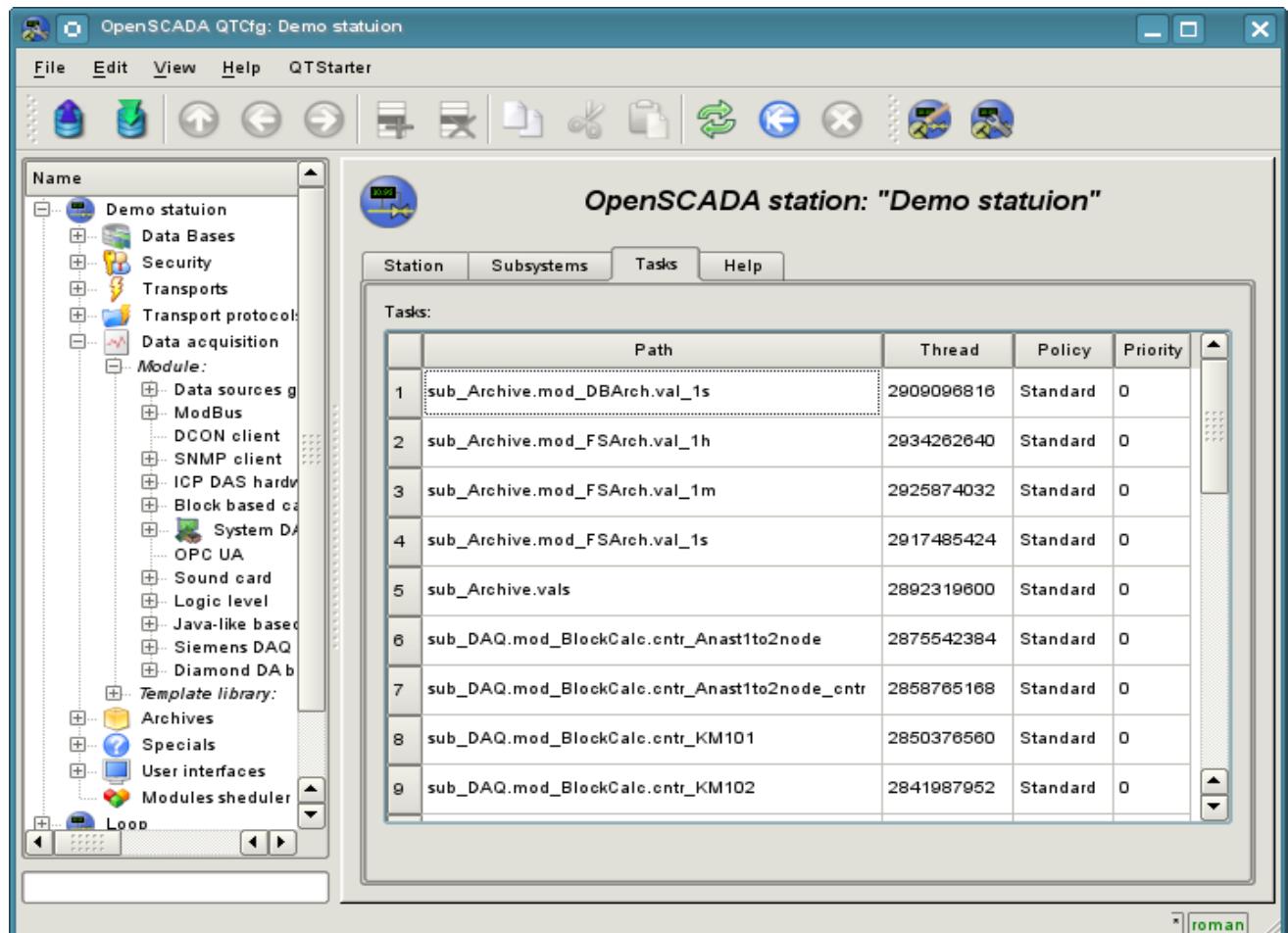
Fig 4c. "Tasks" tab of the main page of the configuration of the station.
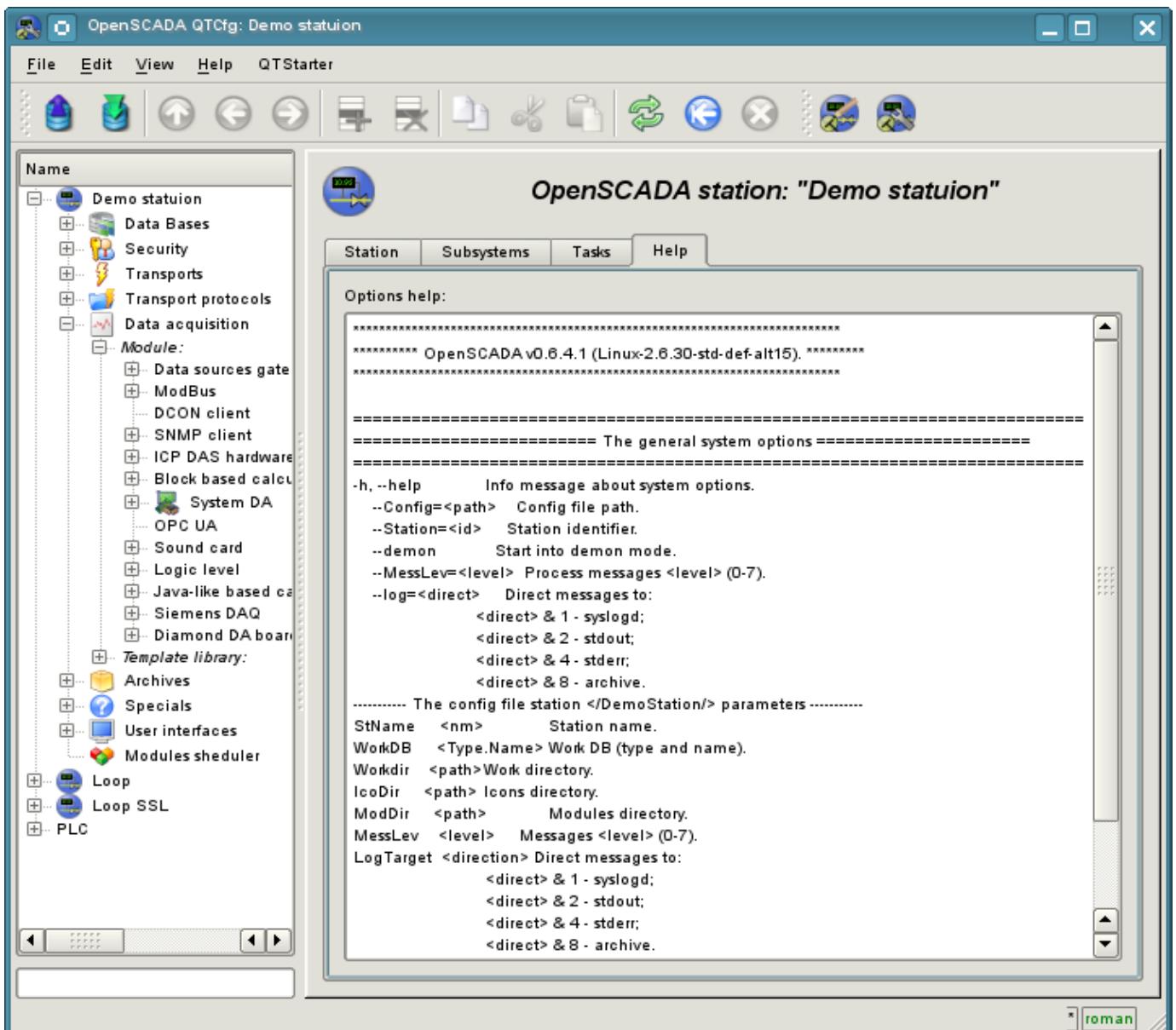
Fig. 4d. "Help" of the main page of the configuration of the station.

While examining the configuration pages of modular subsystems there will be described the general for all modules properties. However, it should be noted that each module can provide both: the additional tabs, and separate fields for the configuration of their own functioning for the pages, objects of which are inherited by modules. Information on the features and additions of modules can be found in separate documentation for each of them.

4.1. "DB" subsystem

The subsystem is the modular one and contains a hierarchy of objects depicted in Figure 4.1a. To configure the subsystem the root page of the subsystem "DB" containing the tabs "Modules" and "Help" is provided. Tab "Modules" (Fig. 4.1b) contains the list of modules in subsystem "DB", available at the station. Tab "Help" tab contains a brief help for this page.

To modify the page's fields of this subsystem it may be required the super user's rights or the inclusion of your user to the "DB" group.
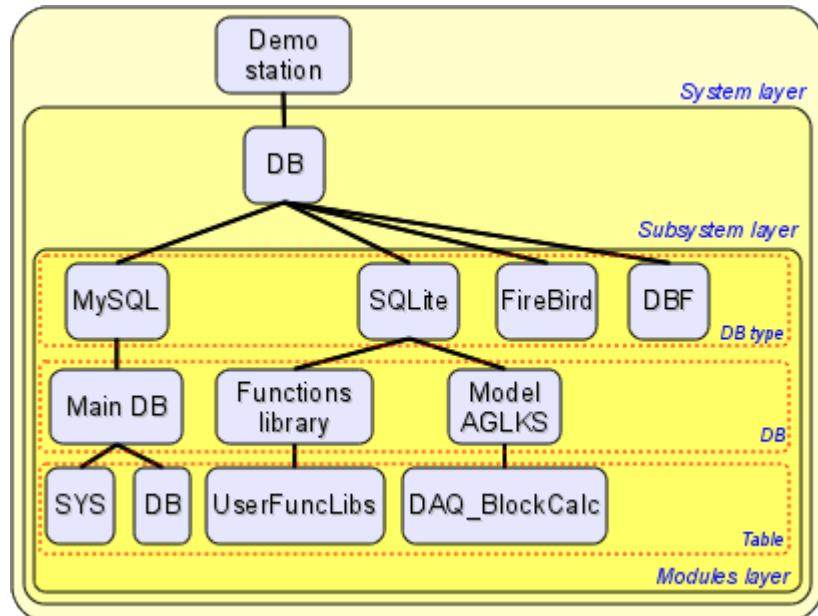
Fig. 4.1a. The hierarchical structure of "DB" subsystem.
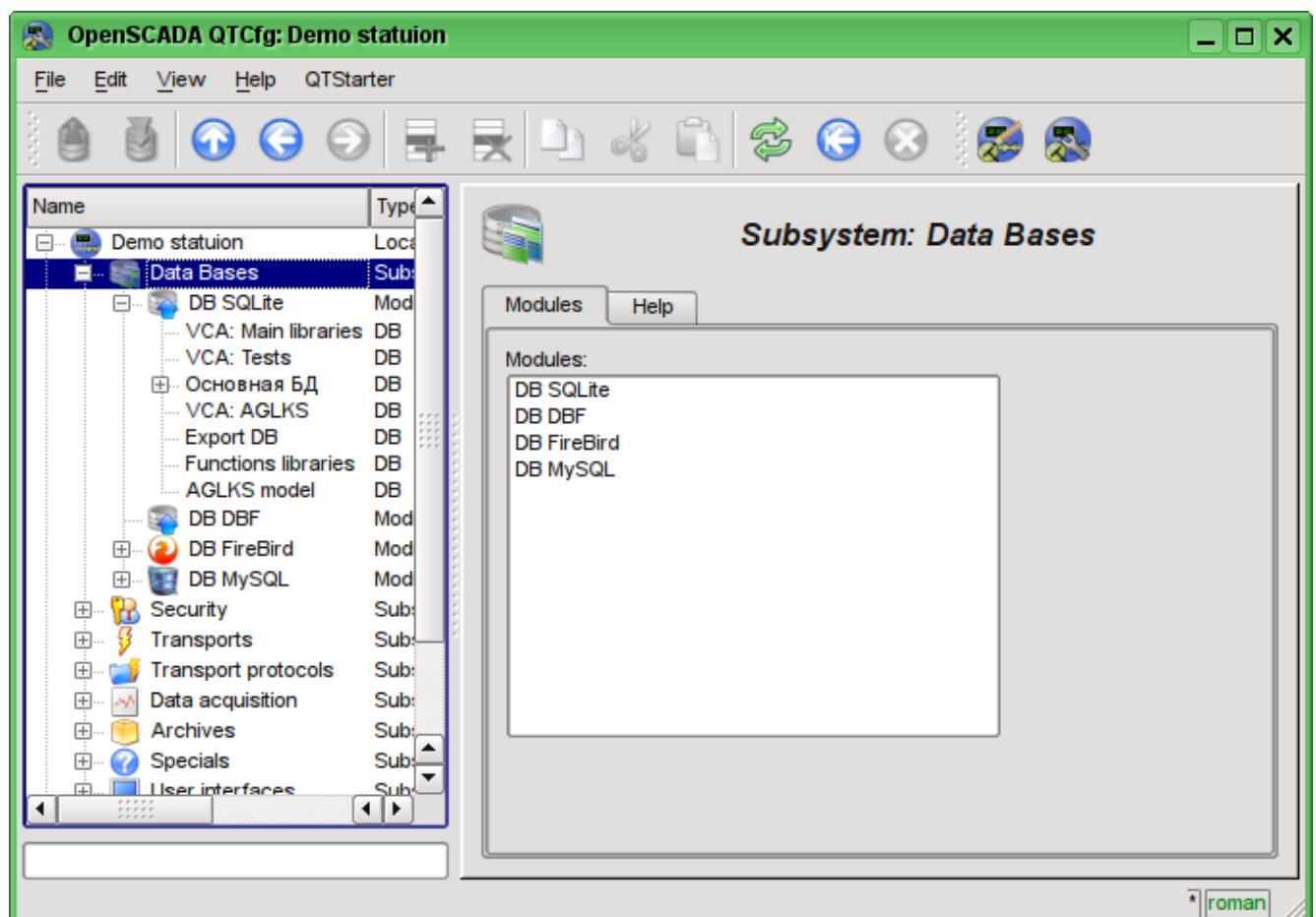
Fig. 4.1b. Tab "Modules" tab of the root page of "DB" subsystem.

Each module of the "DB" subsystem provides the configuration page with the following tabs: "DB" and "Help". "DB" tab (Fig. 4.1c) contains the list of databases registered in the module and the flag of the sign of full deleting of the database when making the delete command. In the context menu of the databases' list the user is provided with an opportunity to add, delete and move to the desired database. The "Help" tab contains information about the module of the "DB" subsystem (Fig.4.1d):

- *Module* - module's identifier.
- *Name* - module's name.
- *Type* - module's type, subsystem's identifier, which contains the module.
- *Source* - shared library - the source of the module.
- *Version* - module's version.
- *Author* - module's author.
- *Description* - module's short description.
- *License* - license agreement of module's distribution.
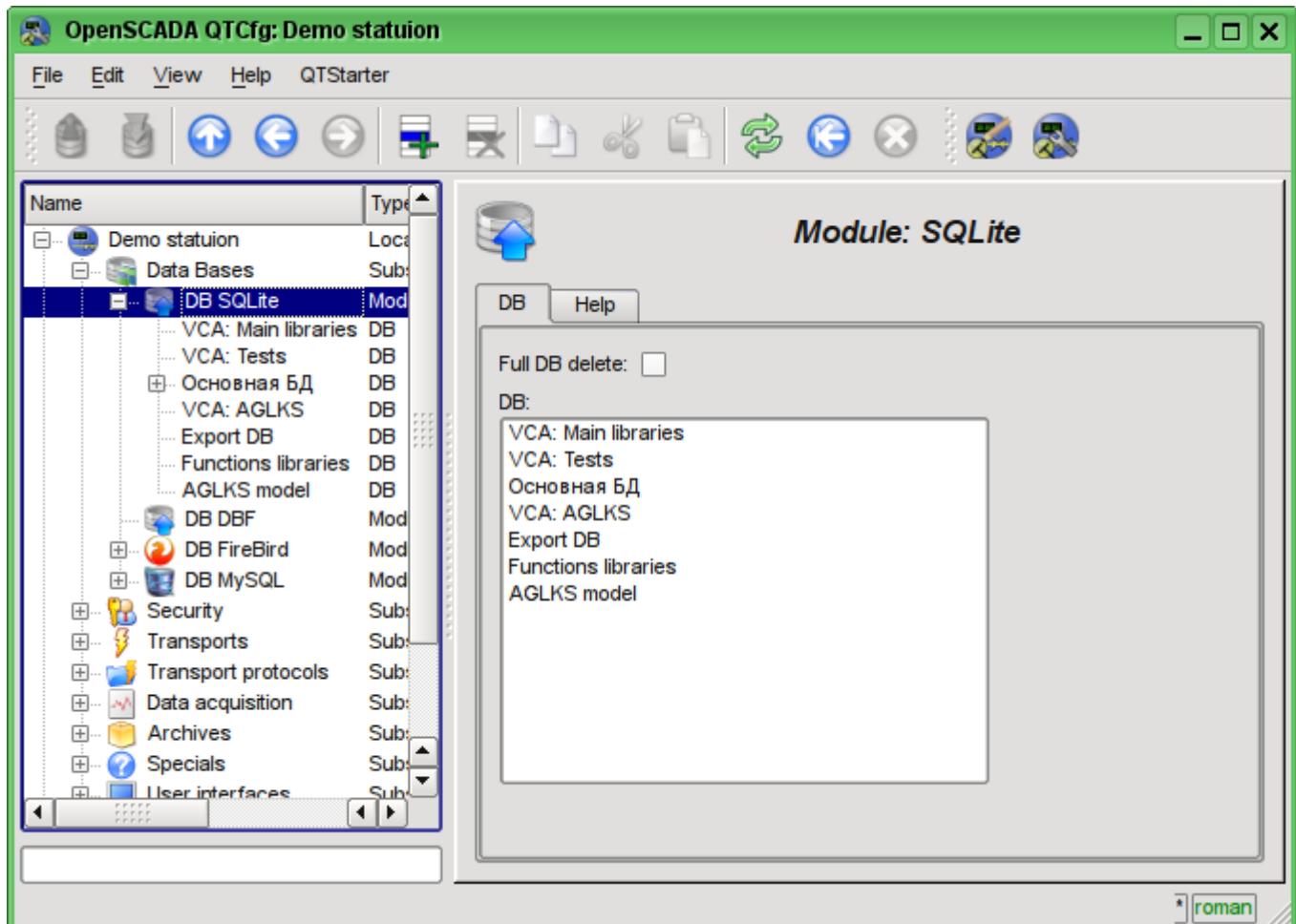
Fig. 4.1c. "DB" tab of the module of "DB" subsystem.
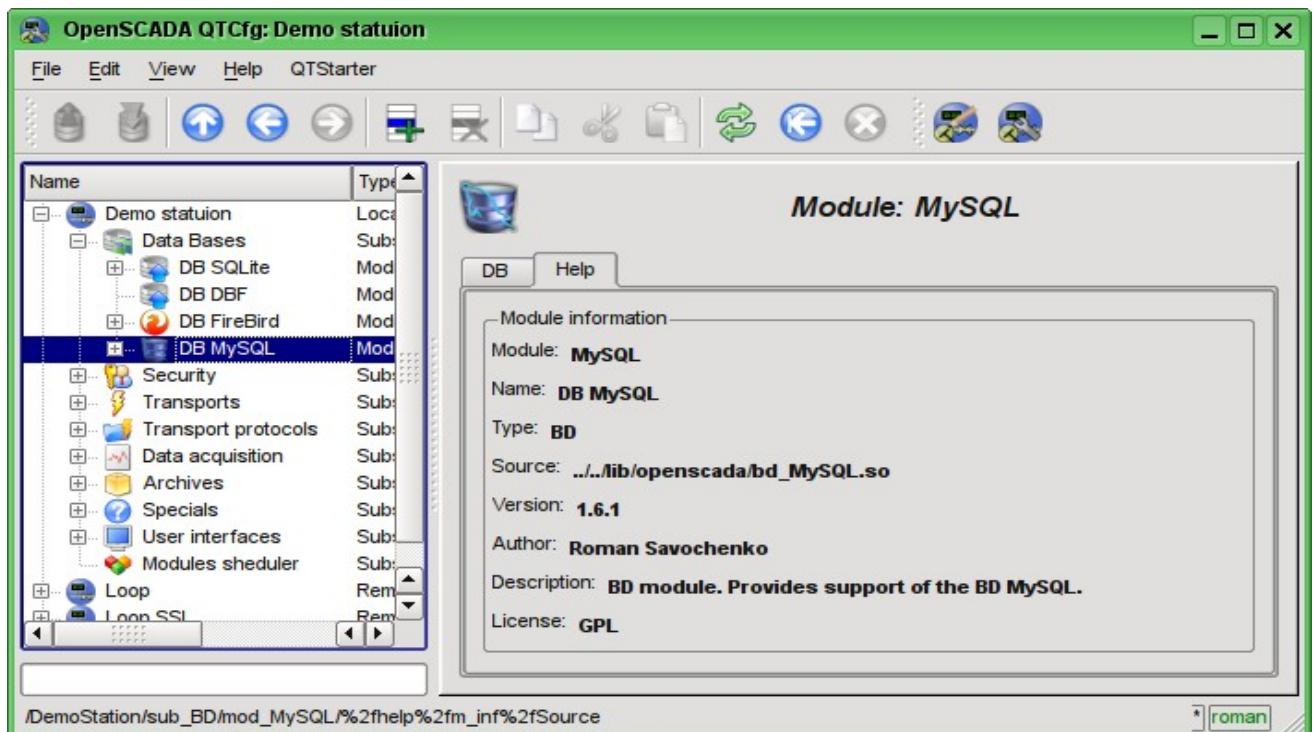
Fig. 4.1d. "Help" tab of the module of the "DB" subsystem.

Each database contains its own configuration page with the tabs "Data base" and "Tables". Besides the basic operations you can copy the contents of the DB by means of the standard function for the copying the objects in the configurator. The copying operation the DB contents involves the copying of the original database to the destination database, and the contents of the destination database is not cleared before the copy operation. Copying the contents of database is made only when the both databases are enabled, otherwise it will run a simple copy of the object of the database.

Tab "Data base" (Fig.4.1e) contains the main configuration options of the DB as follows:
- Section "State" - contains the properties which characterize the DB status:
  - *Enable* - DB status "Enable".
  - *Accessible tables* - list of tables that are in the database. Context menu of the property gives the opportunity to physically remove the tables from the database.
  - *Load system from this DB* - command to make load from this database. Can be used when transferring data in the database between stations. For example, you can save the section of one station in the export database, physically to move the DB to another station and connect it in this subsystem, and call this command.
- Section "Config" - contains the following configuration fields:
  - *ID* - contains the information on the DB identifier.
  - *Name* - specifies the DB name.
  - *Description* - short description of the DB and it's appointment.
  - *Address* - DB address in the specific for the database type (module) in the format. Format Description of the DB address recording format is usually available in the tooltip for this field.
  - *Code page* -indicates the code page, in which the text values of database are stored and provided. The value of the code page of database in conjunction with the internal code page of the station is used for clear transcoding of the text message while exchange between the station and the database.
  - *To enable* - indicates the state "Enable", in which to set the DB when start.

Tab "Tables" (Fig.4.1f) contains the list of the opened pages. In normal mode of the program operation this tab is empty, because after the completion of working with tables the program closes them. The presence of opened tables tells that the program is now working with tables or tables are opened by the user to examine their contents. In the context menu of list of opened tables you can open the table for study (the command "Add"), close the opened page (the command "Delete") and proceed to examination of the contents of the table.
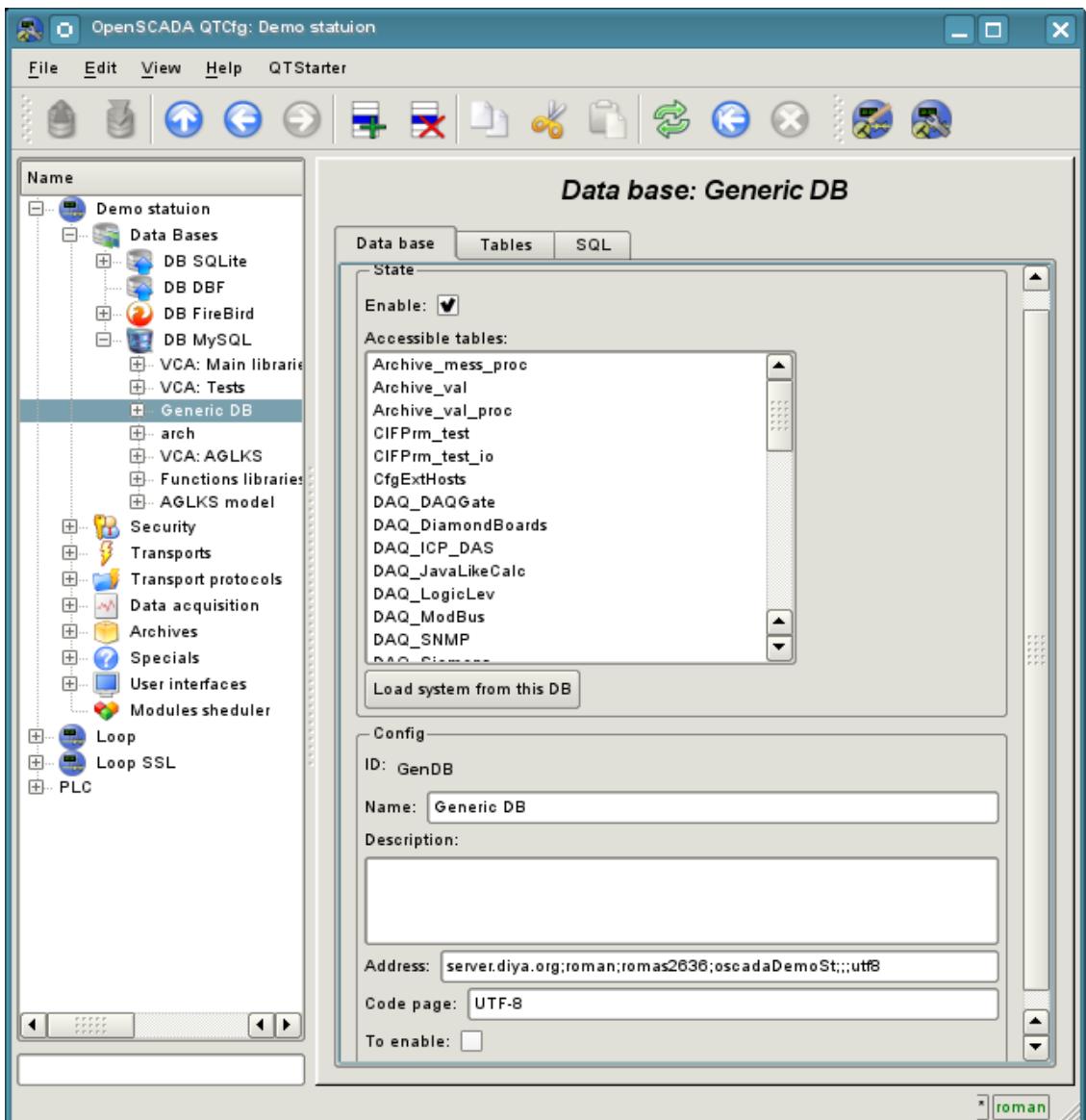
Fig. 4.1e. Tab "Data base" of the DB of module of subsystem "DB".
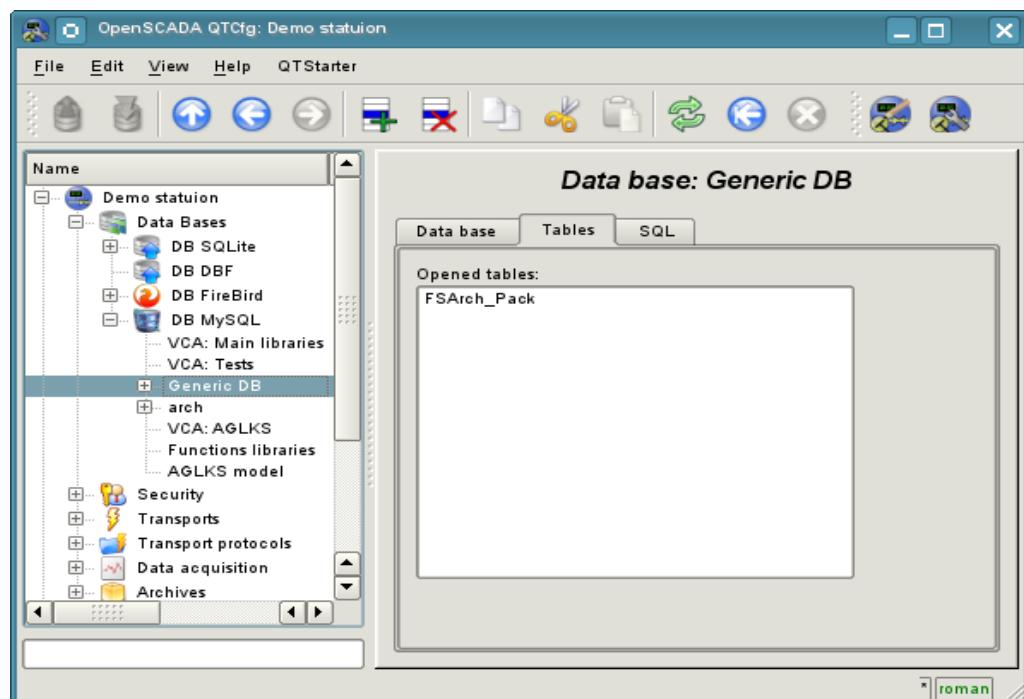
Fig. 4.1f. Tab "Tables" of the DB of module of subsystem "DB".

Page of the examination of the contents of the table contains only one tab, "Table". Tab "Table" (Figure 4.1g) contains the field of the name of the table and the table with the contents. Table of contents provides the following functions:

- table's cells content redaction;
- addition of the line;
- deleting of the line.
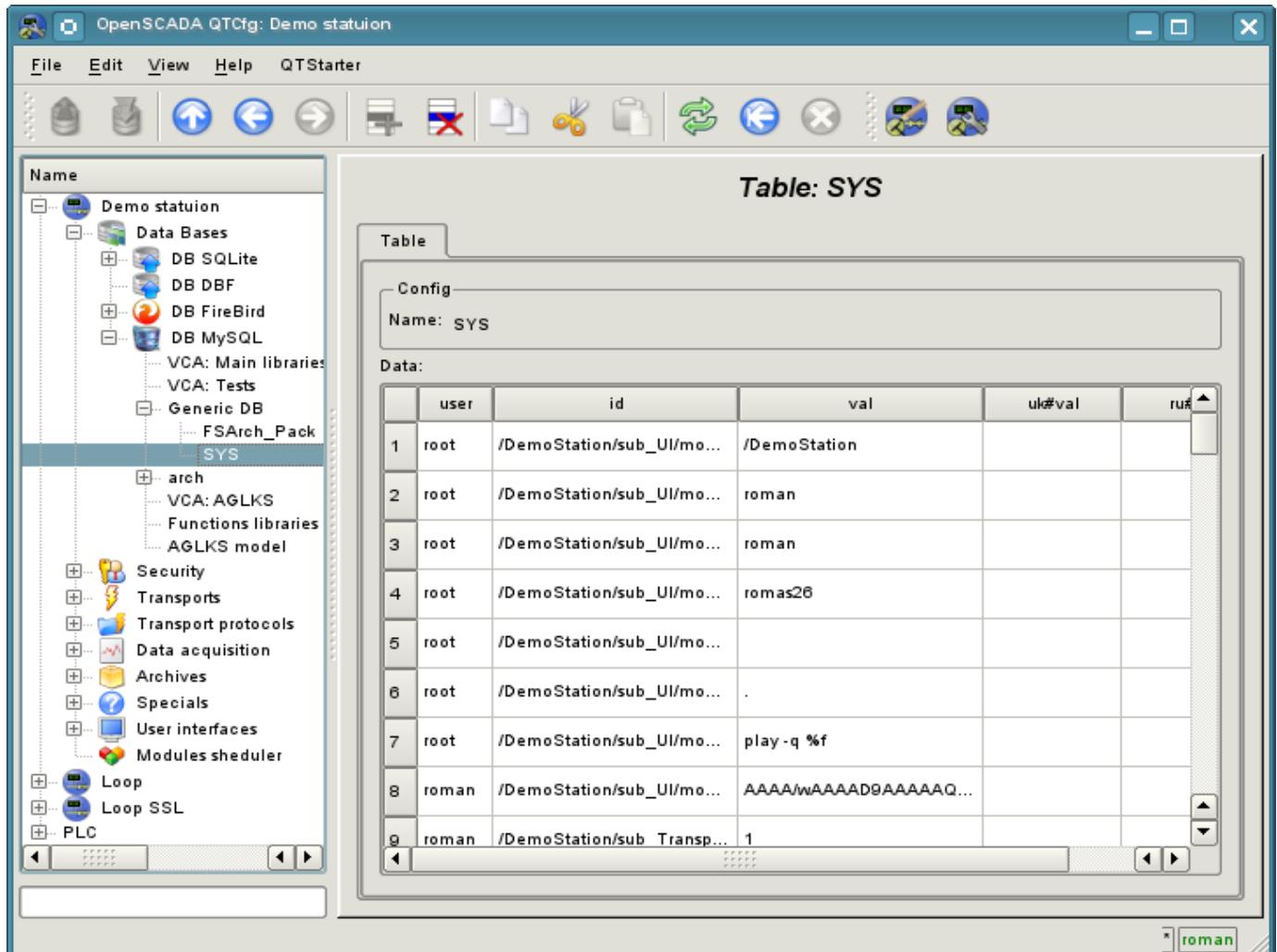
Fig. 4.1g. Tab "Table" of the DB table of the module of the subsystem "DB".

4.2. Subsystem "Security"

The subsystem is not modular one. To configure the subsystem the root page of the subsystem "Security" is provided, which contains the tab "Users and Groups" and "Help". Tab "Users and Groups" (Figure 4.2a) contains the list of users and users' groups. Users in the group "Security" and with the rights of the privileged user can add, delete the user or group of users. All other users can go to the page the user or the users' group. Tab "Help" contains the brief help for this page.
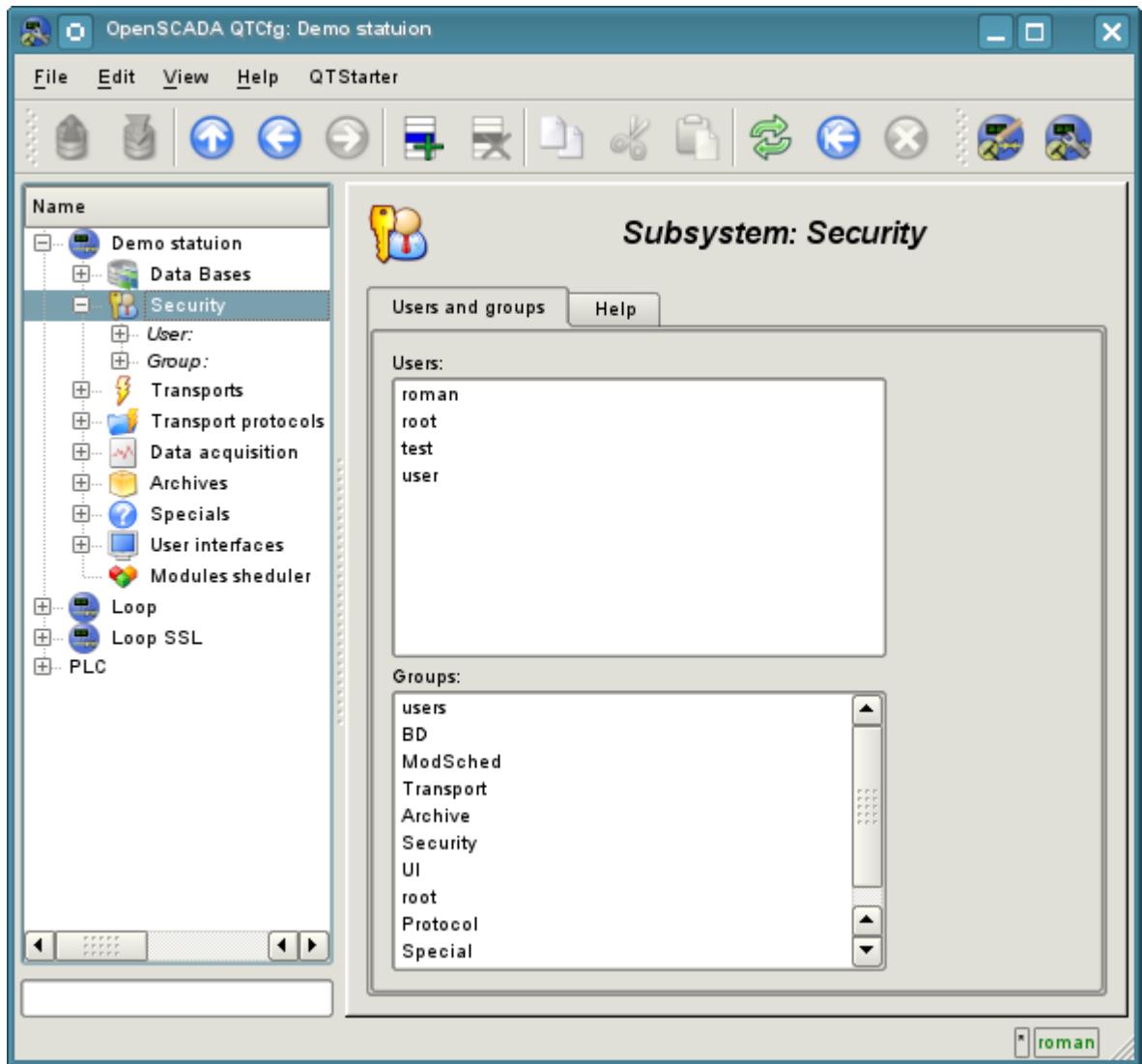
Fig. 4.2a. Tab "Users and Groups" of the root page of the subsystem "Security".

To configure the user it is provided the page containing only the tab "User" (Fig.4.2b). Tab contains the configuration data of the user's profile, which can be changed by the user itself, the user of the "Security" group or the privileged user:

- • *Name* - information about the name (identifier) of the user.
- • *Full name* - specifies the full name of the user.
- • *User picture* - specifies the user's picture. Picture can be loaded and saved.
- • *User DB* - DB address for the user's data storage.
- • *Password* - the field to change the user's password. It always displays "******".
- • *Groups* - the table with a list of user groups of the station and with the sign of identity of the user to the groups.
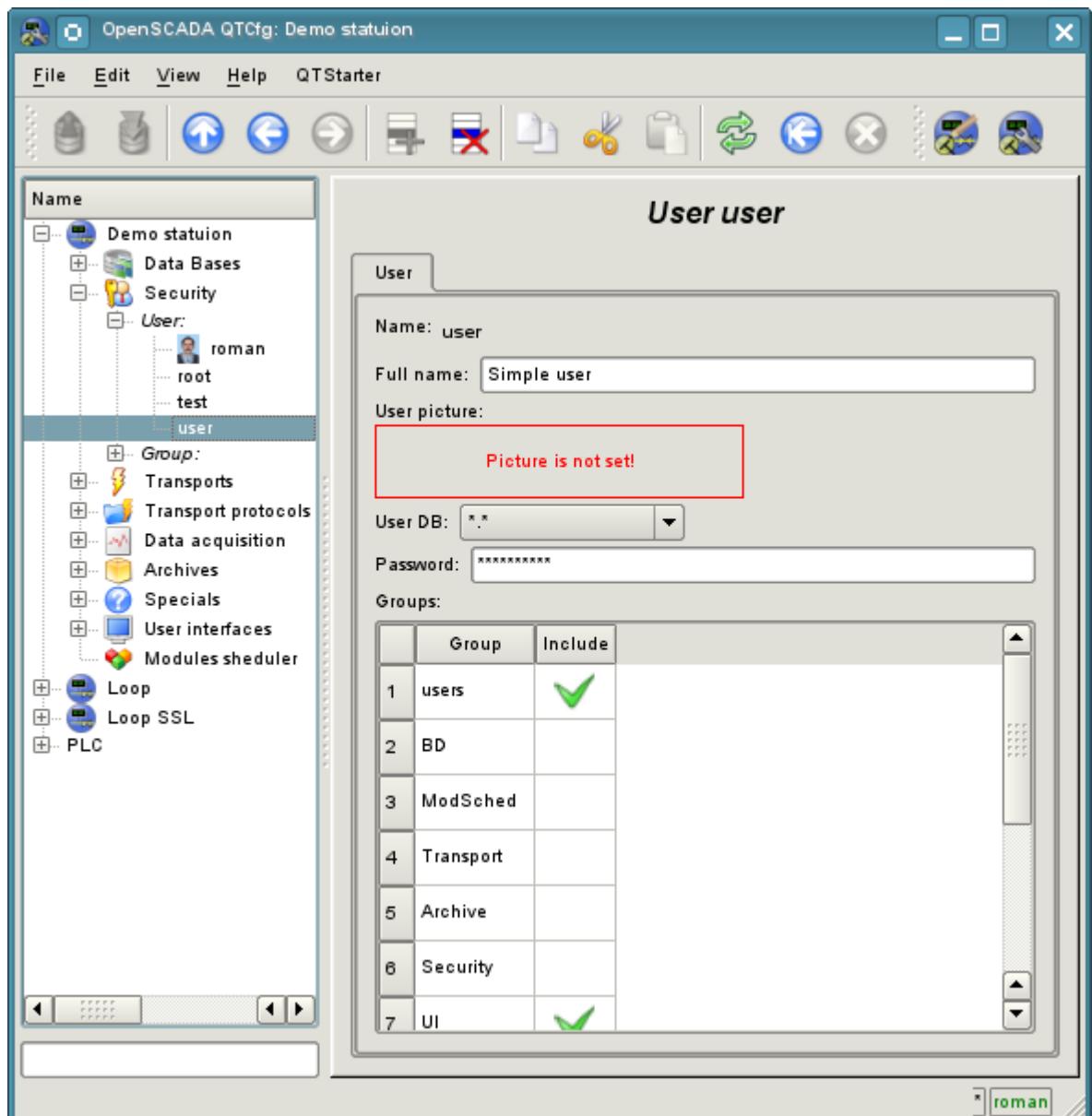
Fig. 4.2b. The tab "User" of the user's page of "Security" subsystem.

To configure the user's group it is provided the page containing only the tab "Group" (Fig.4.2c). Tab contains the configuration data of the group's profile, which can be changed only by the privileged use:
- *Name* - information about the name (identifier) of the user's group.
- *Full name* - specifies the full name of the user's group.
- *User group DB* - DB address for the user group's data storage.
- *Users* - list of users included in this group. With the context menu of the list you can add or remove the user in the group.
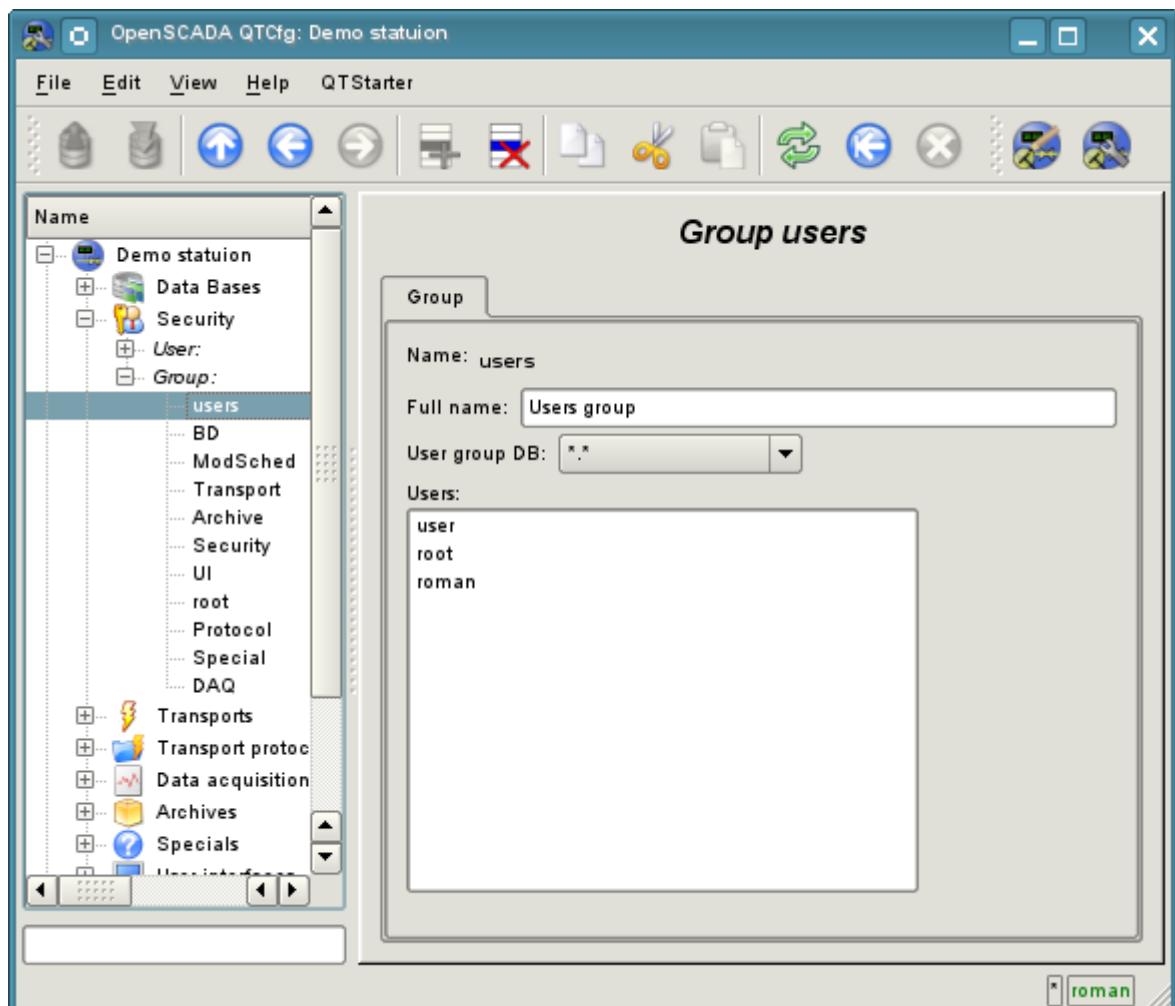
Fig. 4.2c. The tab "Group" of the user's group page of "Security" subsystem.

4.3. Subsystem "Transports"

The subsystem is the modular one and contains the hierarchy of objects shown in Figure 4.3a. To configure the subsystem it is provided the root page of the subsystem "Transports", containing the tabs "Subsystem", "Modules" and "Help".
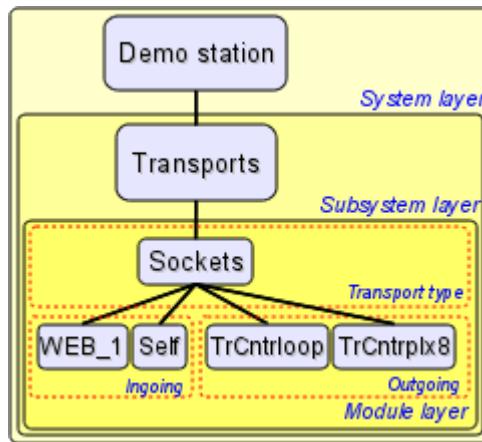

Fig. 4.3a. The hierarchical structure of subsystems "Transports".

The tab "Subsystem" (Figure 4.3b) contains the configuration table of the external stations for a given OpenSCADA. External stations can be the system's and the user's ones that is selected by the appropriate option. System's external stations are available only to the super user and are used by the components of the system purpose, for example, the mechanism of the horizontal redundancy and module DAQ.DAQGate. User's external stations are tied to the user who created them, and thus the list of user's external stations is individual for each user. User's external stations are used by the components of graphical interface, for example, UI.QTCfg, UI.WebCfgD and UI.Vision. In the table of the external stations it is possible to add and delete records about the station, as well as their modification. Each station contains the following fields:

- *Id* - identifier of the external station.
- *Name* - the name of the external host.
- *Transport* - the combobox of the subsystem's module "Transports" for the using of it when access to the external station.
- *Address* - address of the external station if the format, specific to the chosen in the previous field of the module of the subsystem "Transports".
- *User* - the name/identifier of the user of the external station on behalf of whom to perform the connection.
- *Password* - password of the user of the external station.

Tab "Modules" tab (fig. 4.1b) contains the list of modules in subsystem "Transports" and is identical for all modular subsystems. Tab "Help" contains a brief help for this page.
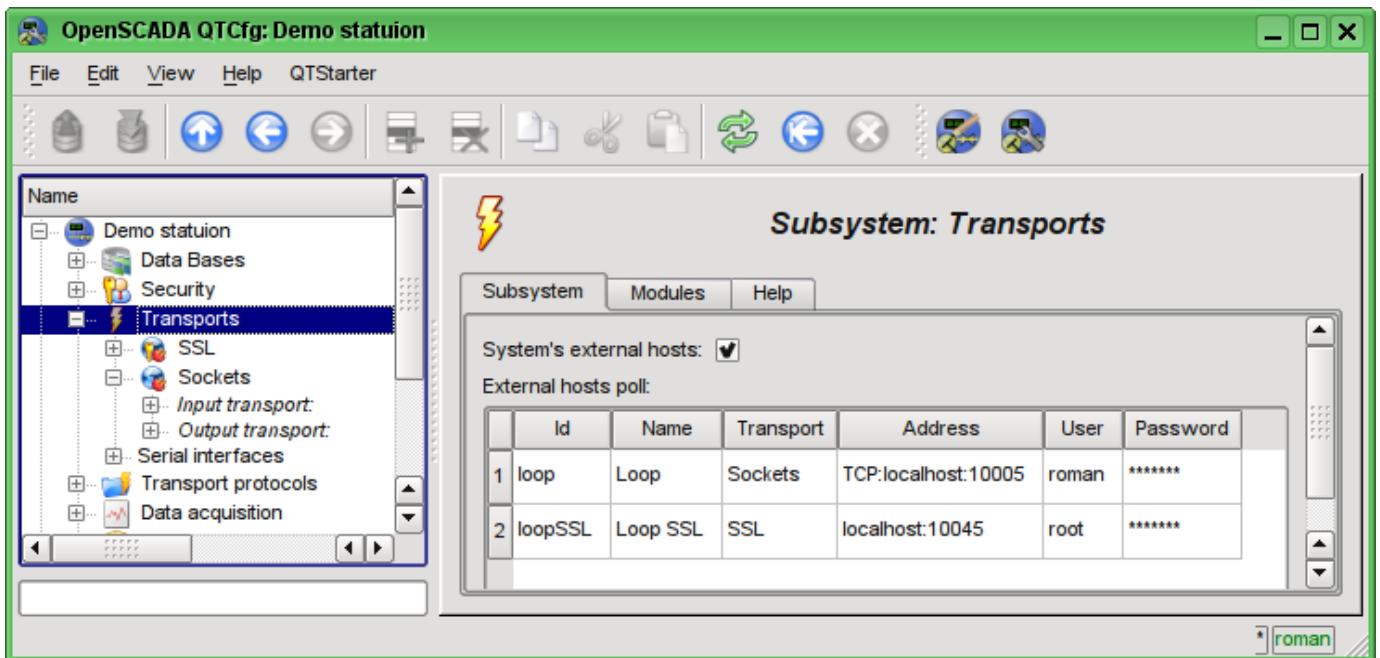
Fig. 4.3b. Tab "Subsystem" of the root page of subsystem "Transports".

Each module of the subsystem "Transports" provides the configuration page with the tabs "Transports" and "Help". The tab "Transports" (Fig.4.3c) contains the list of incoming and outgoing transports registered in the module. The context menu of lists of transports provides the user with the possibility to add, delete and move to the desired transport. On the "Help" tab it is provided the information about the module of subsystem "Transports" (Fig. 4.1d), whose structure is identical for all modules.
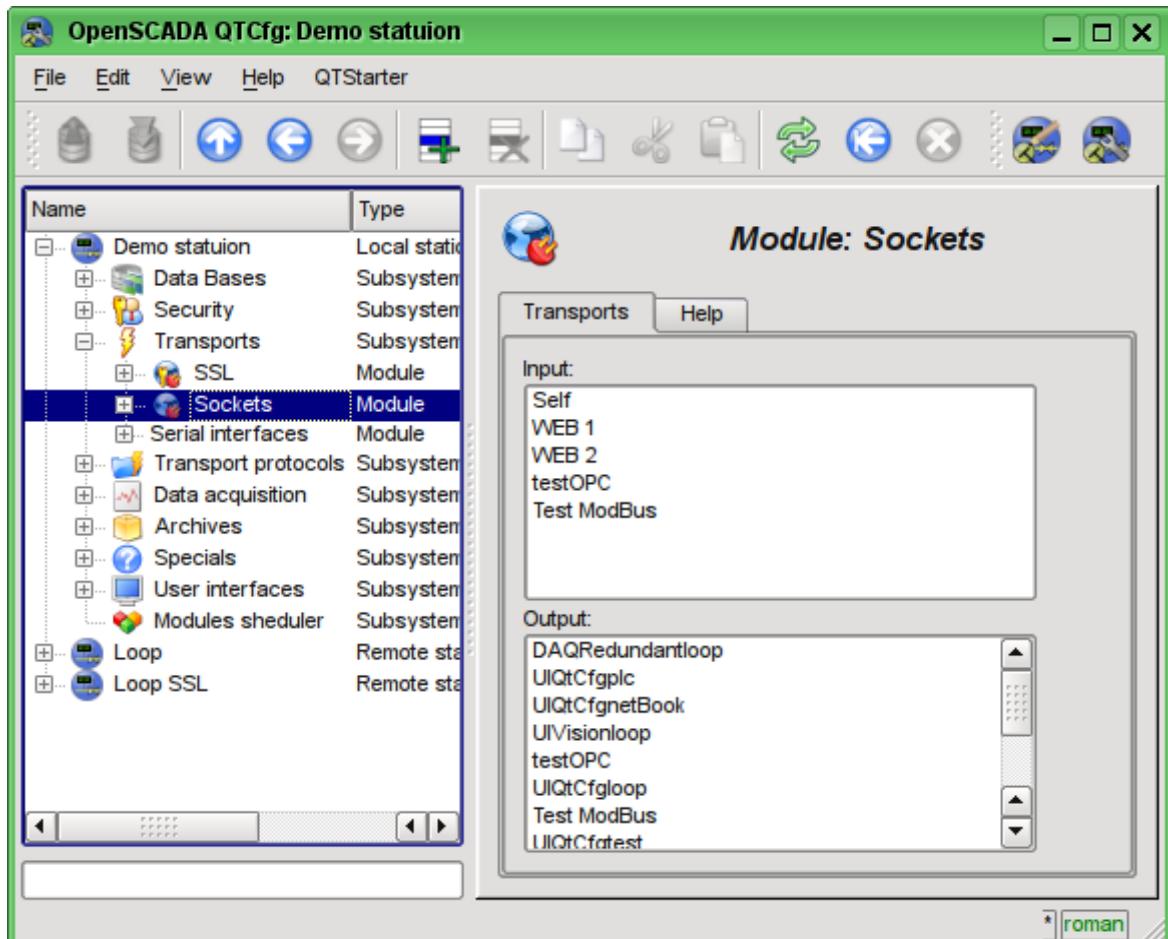

Fig. 4.3c. The tab "Transports" of the module of subsystem "Transports".

Each transport contains its own configuration page with one tab "Transport". This tab contains the basic settings of transport. Incoming transport (fig.4.3d) includes:
- Section "State" - contains the settings that characterize the state of the transport:
  - *Status* - information on the current transport's status and statistics of its work.
  - *Running* - state of the transport "Running".
  - *Transport DB* - DB address to store the transport's data.
- Section "Config" - directly contains the configuration fields:
  - *ID* - information on the transport's identifier.
  - *Name* - specifies the transport's name.
  - *Description* - brief description of the transport and its appointment.
  - *Address* - transport's address in the specific for the type of transport (module) format. Description of the record format addresses transport, as a rule, is available in the tooltip for this field.
  - *Transport protocol* - indicates the transport protocol module (subsystem "Transport protocols") that should work in conjunction with the input transport. Ie the received unstructured data this module will sent to the structuring and processing to the specified module of the transport protocol.
  - *To start* - indicates the status of "Running", in which to transfer the transport at startup.
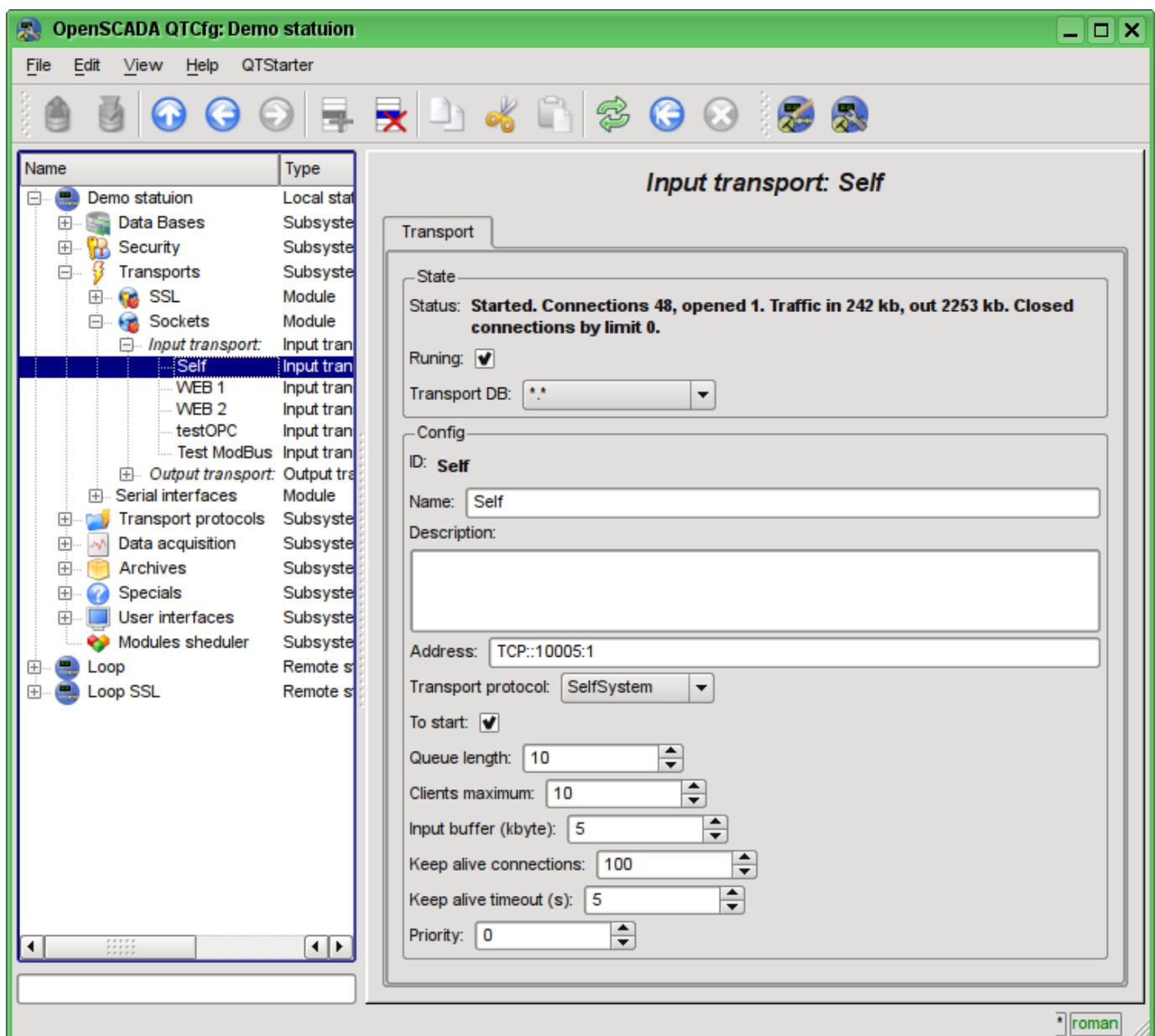
Fig. 4.3d. Tab "Transport" of the page of incoming transport of module of subsystem "Transports".

Outgoing transport (Fig. 4.3e) contains:

- Section "State" - contains the settings that characterize the state of the transport:
    - *Status* - information on the current transport's status and statistics of its work.
    - *Running* - state of the transport "Running".
    - *Transport DB* - DB address to store the transport's data.
- Section "Config" - directly contains the configuration fields:
    - *ID* - information on the transport's identifier.
    - *Name* - specifies the transport's name.
    - *Description* - brief description of the transport and its appointment.
    - *Address* - transport's address in the specific for the type of transport (module) format. Description of the record format addresses transport, as a rule, is available in the tooltip for this field.
    - *To start* - indicates the status of "Running", in which to transfer the transport at startup.
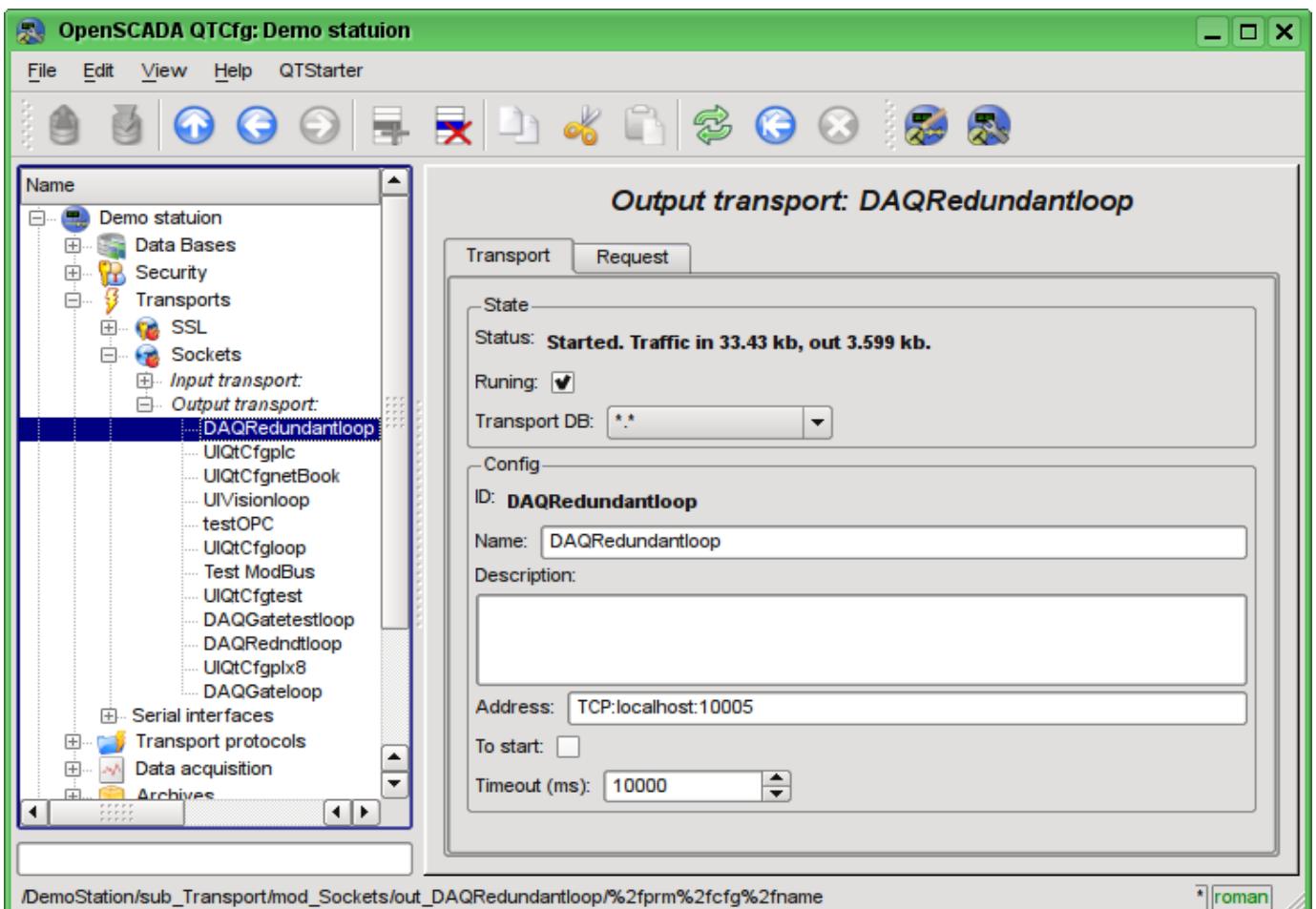
Fig. 4.3e. Tab "Transport" of the page of outgoing transport of module of subsystem "Transports".

Outgoing transport, in addition, provides the tab for forming the user request via this transport (Fig.4.3f). The tab is provided for setting communication, as well as for debugging the protocols and includes:

- *Time (ms)* - information about the time taken for request and receiving the answer.
- *Mode* - indicates the regime of data from the following list: "Text" and "Binary", in which the request will be formed and the answer will be provided. In binary mode data is recorded in pairs of numbers in hex, ie bytes, separated by spaces.
- *Send* - command to send a request.
- *Request* - contains the request in the selected mode of data representing.
- *Answer* - provides the answer in the selected mode of data representing.
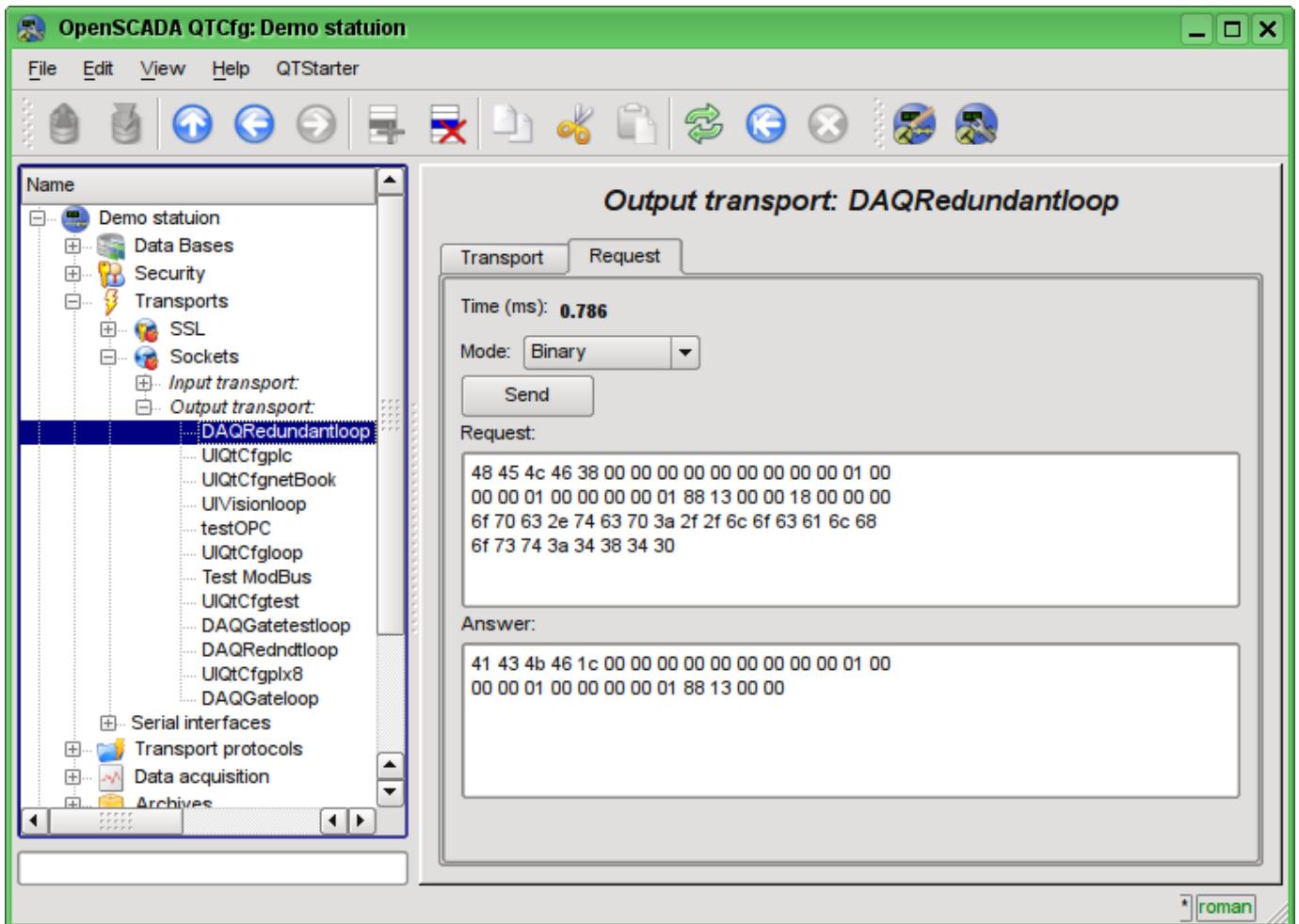
Fig. 4.3f. The tab "Request" of the page of outgoing transport of module of subsystem "Transports".

4.4. Subsystem "Transport protocols"

The subsystem is modular. To configure the subsystem the root page of the subsystem "Transport Protocols" is provided, it contains the following tabs: "Modules" and "Help". The tab "Modules" (Fig. 4.1b) contains the list of modules in subsystem "Transport Protocols" and is identical for all modular subsystems. The tab "Help" contains a brief help for this page.

Each module of subsystem "Transport Protocols" provides configuration page with the only one tab - "Help". On the tab "Help" there is the information on the module of subsystem "Transport Protocols" (Fig. 4.1d), which structure is identical for all modules.

4.5. Subsystem "Data acquisition"

The subsystem is modular and contains the hierarchy of objects depicted in Fig.4.5a. To configure the subsystem the root page of subsystem "Data acquisition" is provided, which contains the tabs "Template libraries", "Modules" and "Help".

To obtain access to modify the objects of this subsystem the user of the group "DAQ" or the rights of the privileged user are required.
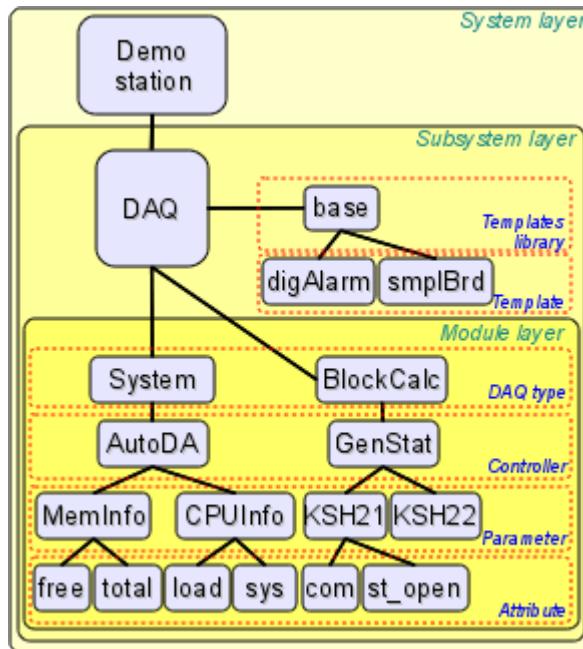

Fig. 4.5a. The hierarchical structure of subsystem "Data acquisition".

Tab "Redundancy" (Fig. 4.5b) contains the configuration of redundancy of data sources of subsystem "Data acquisition" of the station with the following settings:
- *Status* - contains information on redundancy scheme, this is usually the time spent on the execution of one cycle of the task of reserve processing.
- *Station level* - indicates the level of the station in an arrangement (0-255).
- *Redundant task period (s)* - indicates the frequency of execution of redundancy task in seconds (1-255).
- *Restore connection timeout (s)* - indicates over the which period of time to attempt to reconnect with the lost redundant station in seconds (0-255).
- *Restore data depth time (hours)* - indicates the maximum depth of archival data to restore from the archive of the remote station when start up in hours (0-12).
- *Stations* - contains the table with information about the redundant stations. Stations can be added and removed via contextual menu. Id of the added stations is to be chosen from the list of available OpenSCADA system stations. The table provides the following information about the station:
    - *ID* - ID of the system OpenSCADA station, should be changed after the addition by choosing from the list of available ones;
    - *Name* - name of the system OpenSCADA station;
    - *Live* - sign of the connection with the redundant station;
    - *Level* - level of the remote station in the redundancy scheme;
    - *Counter* - requests' counter to the redundant station or waiting time in the case of the absence of connection;
    - *Run* - the list of available controllers with the sign (+) of the local execution on the remote station.
- *Go to remote stations list configuration* - command to go to the configuration page of the remote OpenSCADA stations in the subsystem "Transports".
- *Controllers* - contains the table with the list of controllers, available for redundancy, and their current status:

- *Controller* - full controller's ID;
- *Name* - controller's name;
- *Started* - the sign of the controller's execution on the local station;
- *Redundant* - redundancy mode of the controller can be changed from the list of: "Off", "Asymmetric" and "Symmetric";
- *Preferable run* - configuration of the preferred execution at the specified station can be changed; reserved values: <High Level> - execution at the station with the highest level, <Low Level> - execution at the station with the lowest level, <Optimal> - the choice for the execution of the least loaded station.
- *Remoted* - sign indicating the execution of the controller on the remote station and the transfer of the local one to the mode of data synchronization from the remote station.
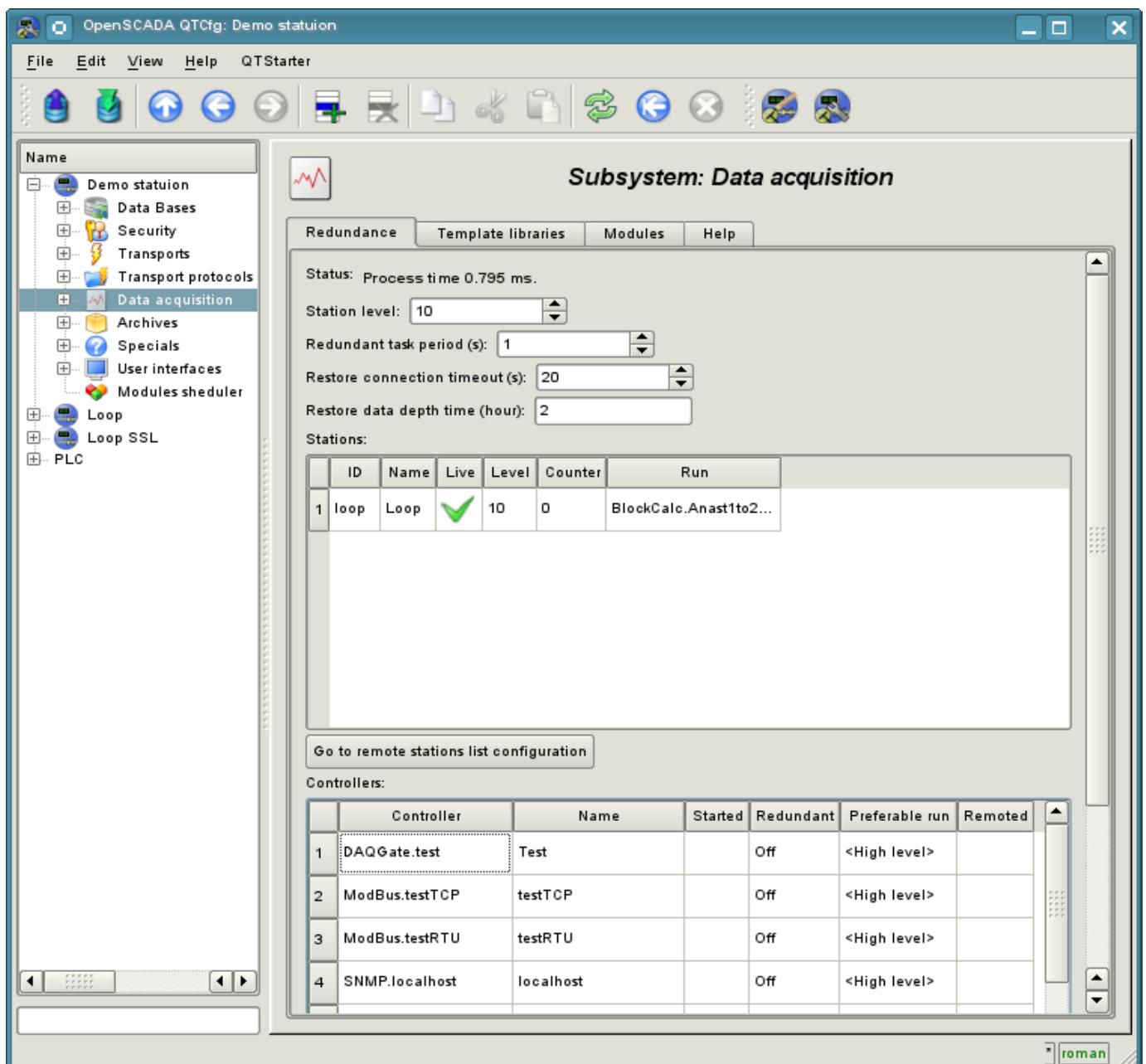
Fig. 4.5b. Tab "Redundancy" tab of subsystem "Data acquisition".

The tab "Template libraries" (Fig.4.5c) contains the list of libraries of templates for the parameters of this subsystem. In the context menu of the list of template libraries the user can add, delete and move to the desired library. The tab "Modules" (Fig. 4.1b) contains the list of modules in the subsystem "Transports" and is identical for all modular subsystems. The tab "Help" contains the brief help for this page.
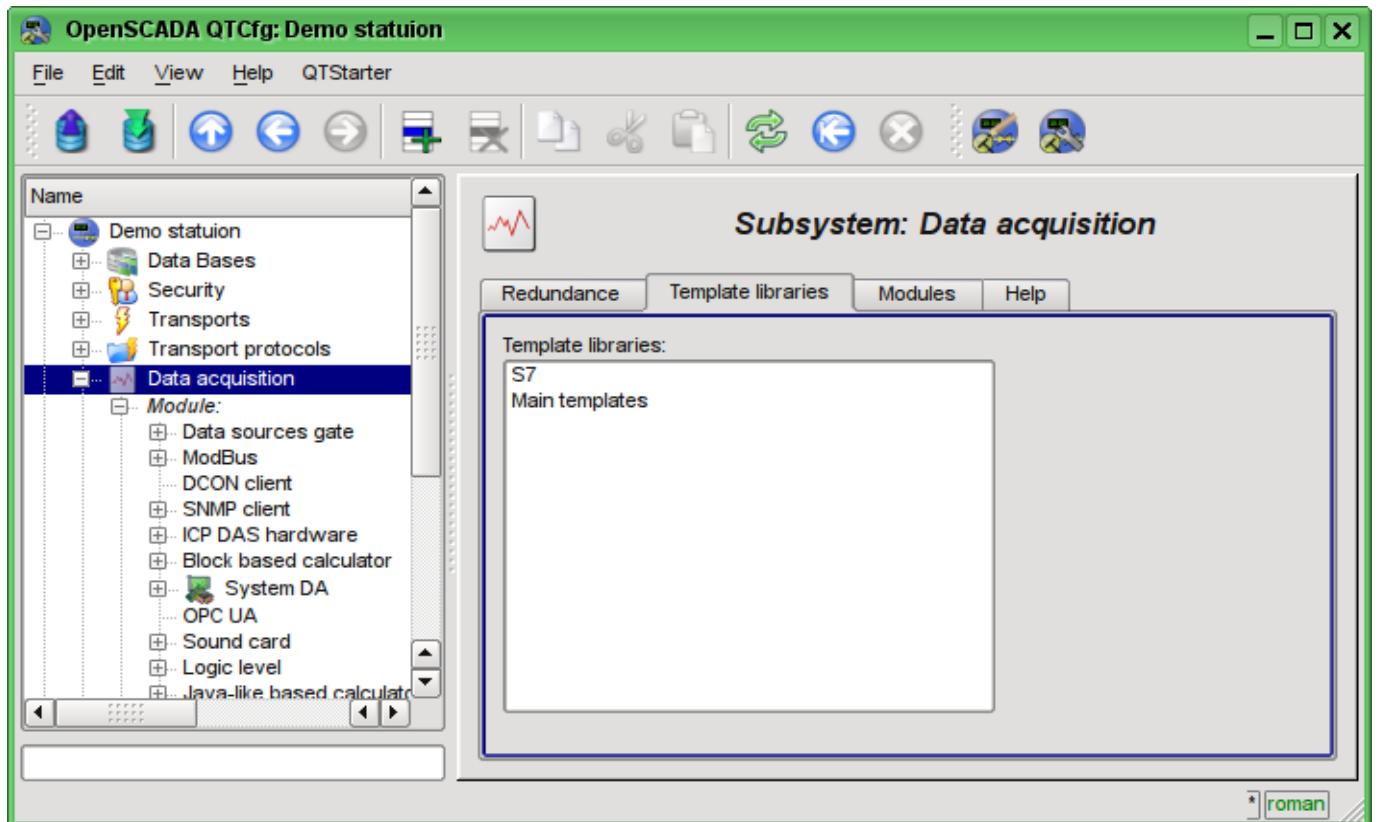
Fig. 4.5c. The tab "Template libraries" of the subsystem "Data acquisition".

Each template library of subsystem "Data acquisition" provides the configuration page with the tabs "Library" and "Parameter templates". Tab "Library" (fig. 4.5d) contains the basic settings of the library:
- Section "State" - contains properties that characterize the state of the library:
  - *Accessing* - state of library "Accessing".
  - *Library DB* - address of the database for data storage of the library and templates.
- Section "Config" - directly contains the configuration fields:
  - *ID* - information on the ID of the library.
  - *Name* - specifies the name of the library.
  - *Description* - short description of the library and its purpose.

Tab "Parameter templates" (Fig.4.5e) contains the list of templates in the library. In the context menu of the list the user can add, delete and move to the desired template.
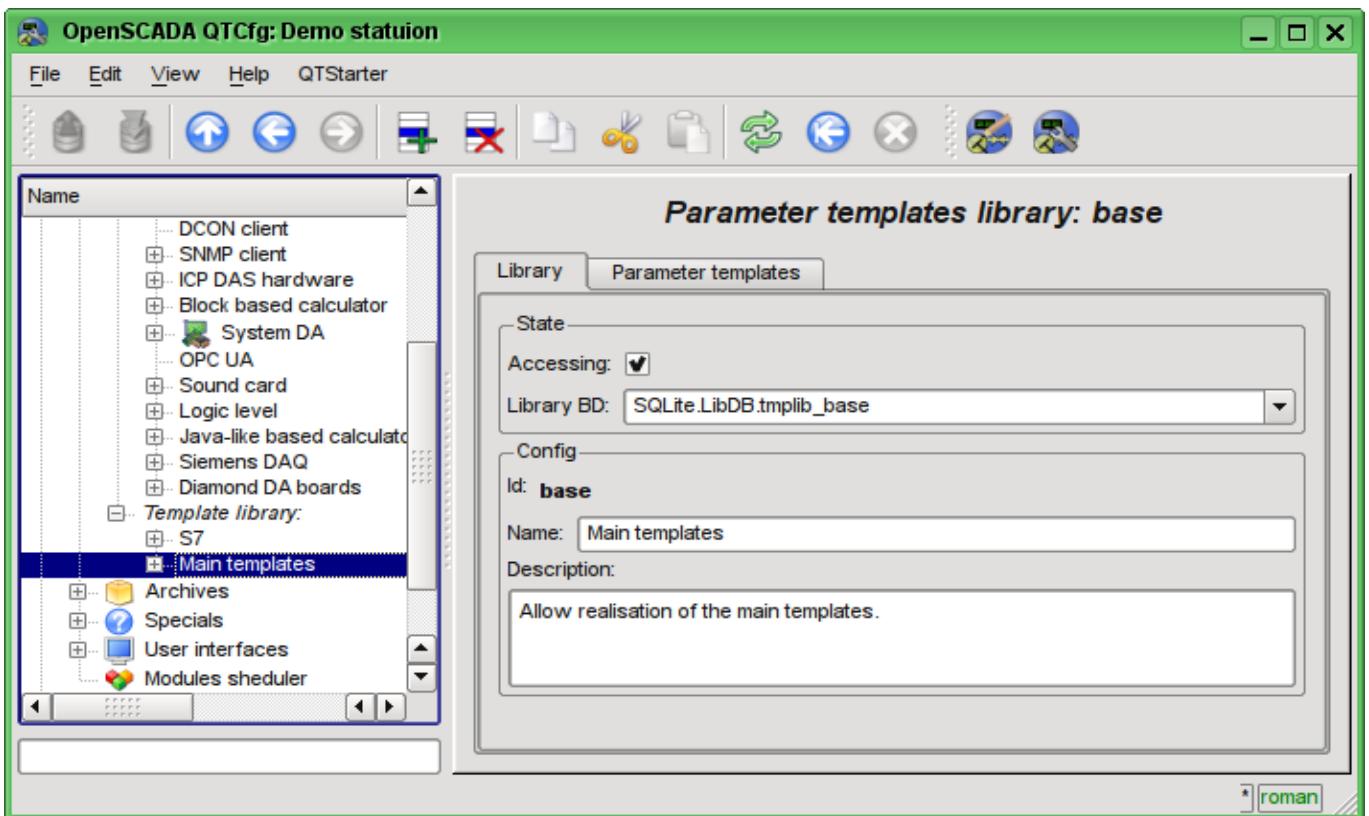
Fig. 4.5d. The main tab of configuration of template library of subsystem "Data acquisition".
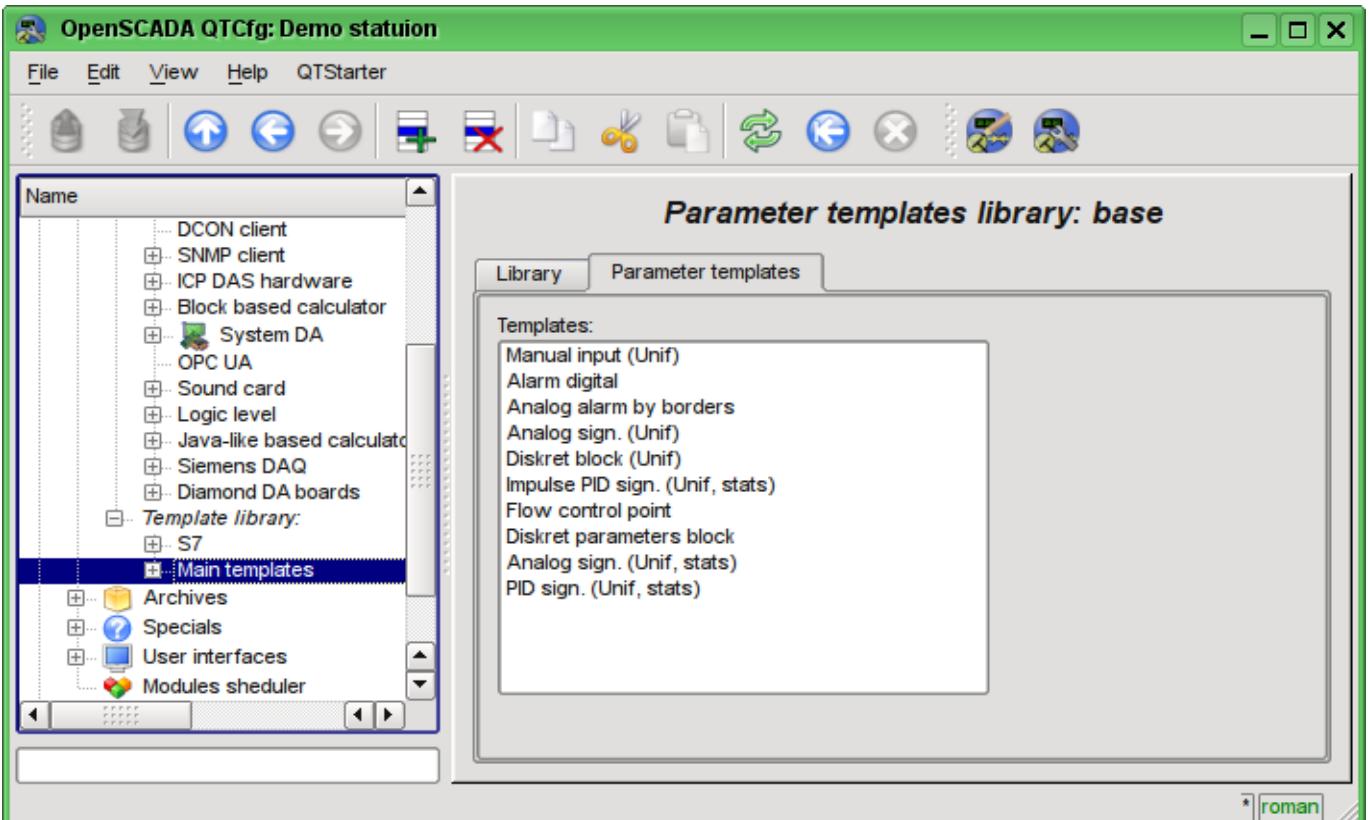
Fig. 4.5e. The tab of the list of templates in the template library of subsystem "Data acquisition".

Each template of the template library provides the configuration page with the tabs "Template" and "IO". The tab "Template" (Figure 4.5f) contains the basic settings of the template:

- Section "State" - contains properties that characterize the state of the template:
    - *Accessing* - state of template "Accessing".
    - *Used* - counter of the template's using. Allows you to determine whether the template is used and, consequently, the ability to edit the template.
- Section "Config" - directly contains the configuration fields:
    - *ID* - information on the ID of the template.
    - *Name* - specifies the name of the template.
    - *Description* - short description of the template and its purpose.
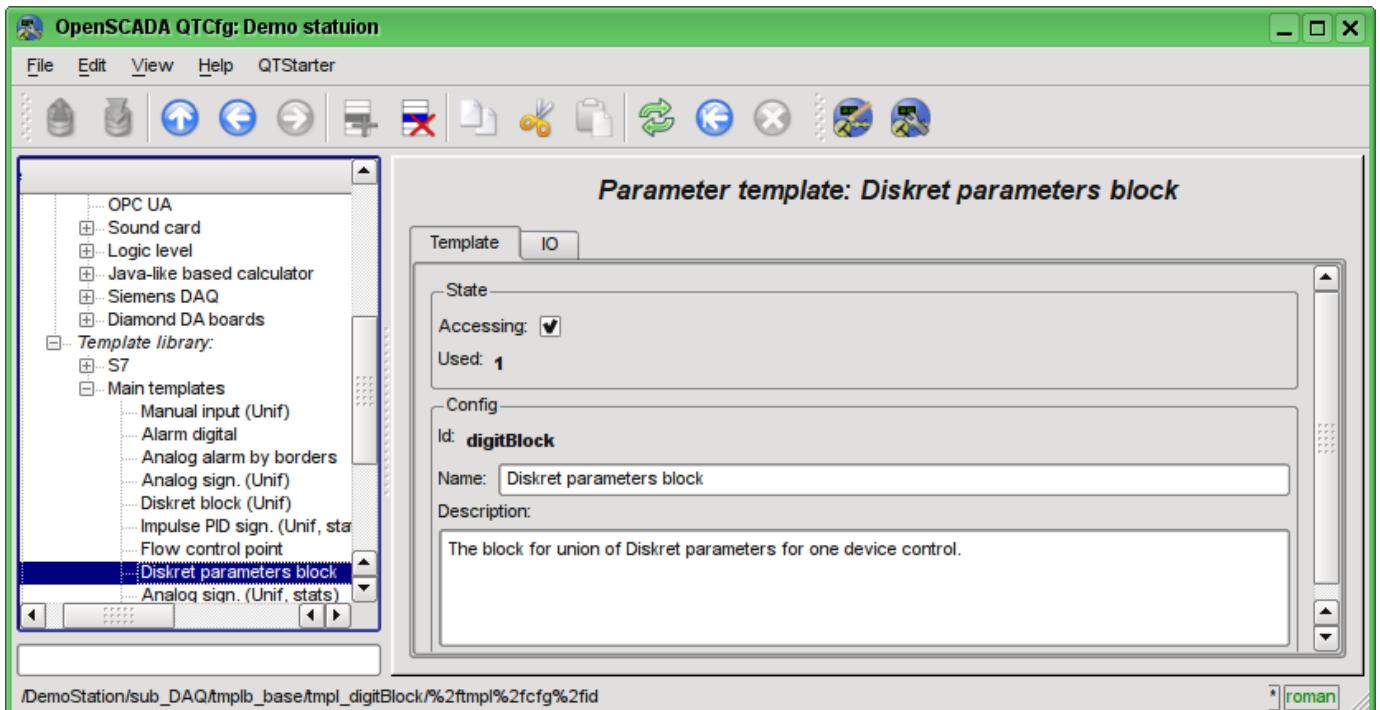
Fig. 4.5f. The main configuration tab of the parameters template of subsystem "Data acquisition".

The tab "IO" (Fig.4.5g) contains the configuration of attributes (IO) of templates and the program of template on the one of languages of the user programming of OpenSCADA, for example, DAQ.JavaLikeCalc.JavaScript. To the table of attributes of template user can, through the context menu, add, insert, delete, move up or down the record of attribute, as well as edit the attribute's fields:

- *Id* - ID of the attribute.
- *Name* - the name of the attribute.
- *Type* - select the value's type of the attribute from the following: "Real", "Integer", "Boolean", "String".
- *Mode* - select the mode of the attribute: "Input", "Output".
- *Attribute* - mode of the parameter's attribute, implemented based on a template from the list: "No attribute" ,"Read Only","Full access". For the attributes of a template, in which this field is set, it will be created an appropriate attribute in the controller's parameter of this subsystem.
- *Configure* - configuration mode of the attribute in the configuration tab of a template of the controller's parameter of this subsystem from the list: "Constant", "Public constant", "Link". In "Public constant" and "Link" modes tab in the configuration tab of the template will be added these attributes to set the constant or specify an external link of the parameter.
- *Value* -attribute's default value or template of the links to access by the link. The format of the link's template depends on the component that uses it. Usually for the module DAQ.LogicLev the link's template is written the following way: **{Parameter}|{attribute}**. Field **{Parameter}** - specifies the parameter's name as the attribute's container. Attributes with the equal value **{Parameter}** will be grouped and will be appointed only by the indication of attributes' container, and individual attributes will be associated with the attributes of the container in accordance with the field **{attribute}**.

The syntax of the language of the template's program you can see in the documentation of the module,

providing an interpreter of the chosen language. For example, a typical user programming language of OpenSCADA - DAQ.JavaLikeCalc
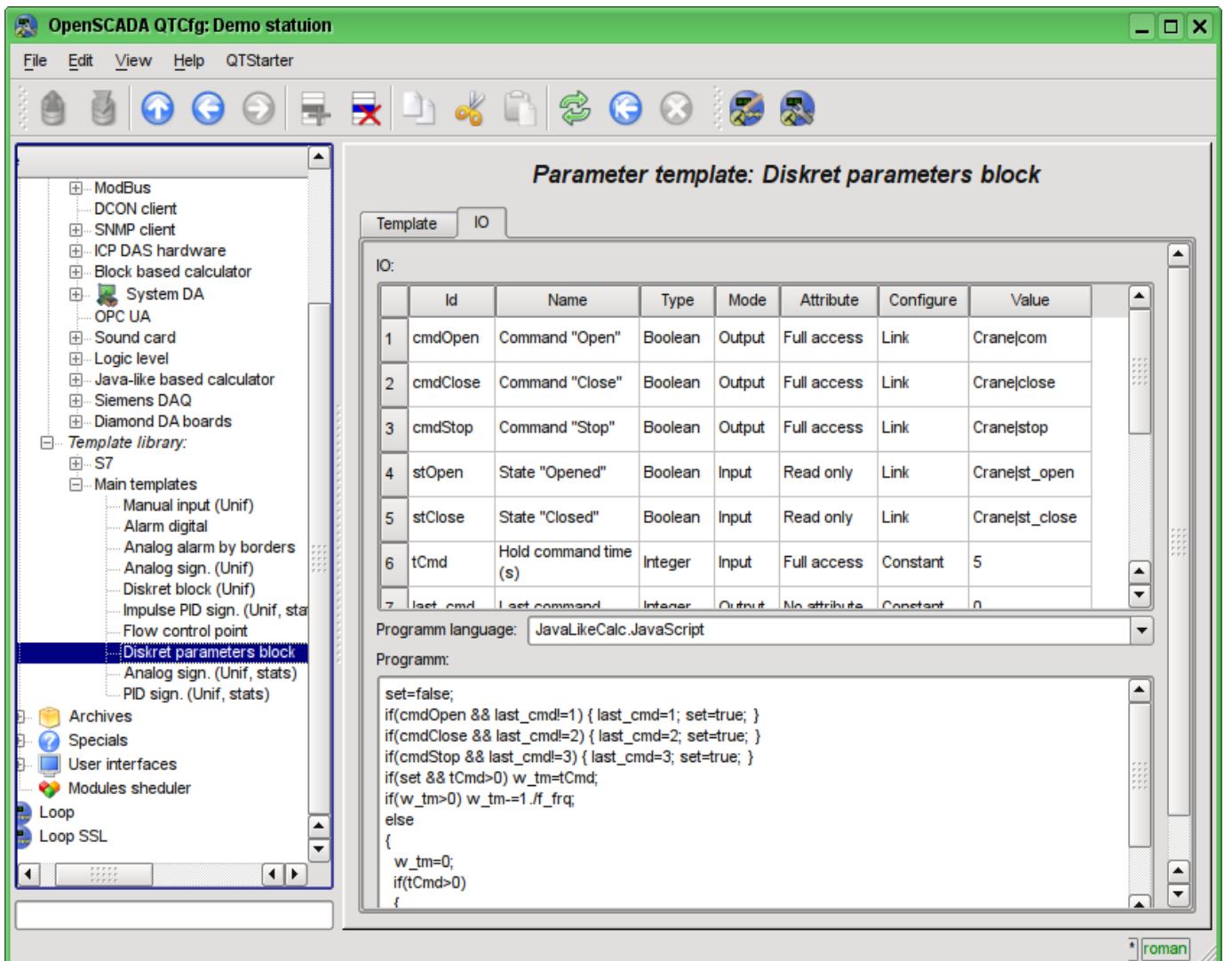
Fig. 4.5g. The configuration tab of the attributes and template's program of subsystem "Data acquisition".

Each module of the subsystem "Data acquisition" provides the configuration page with the tabs "Controllers" and "Help". The tab "Controllers" (Fig.4.5h) contains the list of controllers, registered in the module. In the context menu user can add, delete and move to the desired controller. The tab "Help" provides information about the module of the subsystem "Data acquisition" (Fig. 4.1d), which structure is identical for all modules.
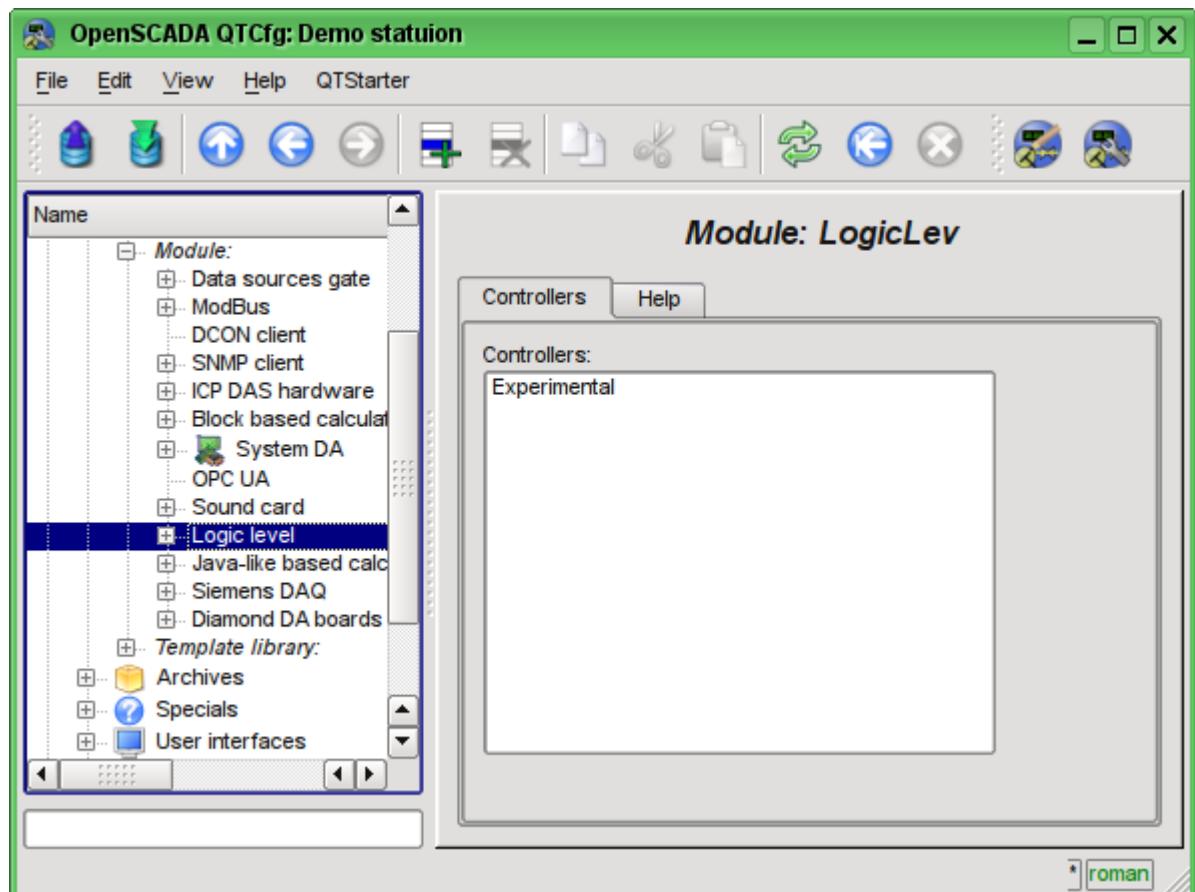
Fig. 4.5h. The tab "Controllers" of the module of the subsystem "Data acquisition".

Each controller contains its own configuration page with the tabs "Controller" and "Parameters".

The tab "Controller" (Fig.4.5i) contains the basic settings. The structure of these settings may differ slightly from one module of this subsystem to another, as you can find in the own documentation of modules. As an example, lets examine the settings of the controller in the module of the controller of logic DAQ.LogicLev:

- Section "State" - contains the properties, which characterize the state of the controller:
    - *Status* - specifies the controller's status. In our case, the controller is running and the computation time is 0.394644 milliseconds.
    - *Enable* - the state of the controller "Enable". When enabled, the controller provides the possibility of creating the parameters and their configuration.
    - *Run* - the state of the controller "Run". The running controller performs the physical data acquisition and/or includes mechanisms for access to these data.
    - *Controller DB* - the address of the database for data storage of the controller and its parameters.
- Section "Config" - directly contains the configuration fields:
    - *ID* - information on the controller's identifier.
    - *Name* - specifies the controller's name.
    - *Description* - brief description of the controller and its purpose.
    - *To enable* - indicates the status of "Enable" in which to transfer the controller at startup.
    - *To start* - indicates the status of "Run" in which to transfer the controller at startup.
    - *Redundant* - includes the controller in the scheme of the horizontal redundancy of data acquisition of the subsystem "Data acquisition" and points the redundancy mode: "Asymmetric" and "Symmetric".
    - *Preferable run* - indicates preference for the execution of the controller at the station in the redundancy scheme.
    - *Parameters table* - table name in which to save the parameters (the objects of the parameters of data acquisition) of the controller.
    - *Request data period (ms)* - periodicity of the data acquisition task. In this example it is the periodicity of the template's calculation.

- *Request task priority* - sets the priority of data acquisition of this controller. It is used when scheduling the operating system tasks. In the case of execution of the station as the superuser "root", this field includes the planning of the controller's task in real time and with the specified priority.
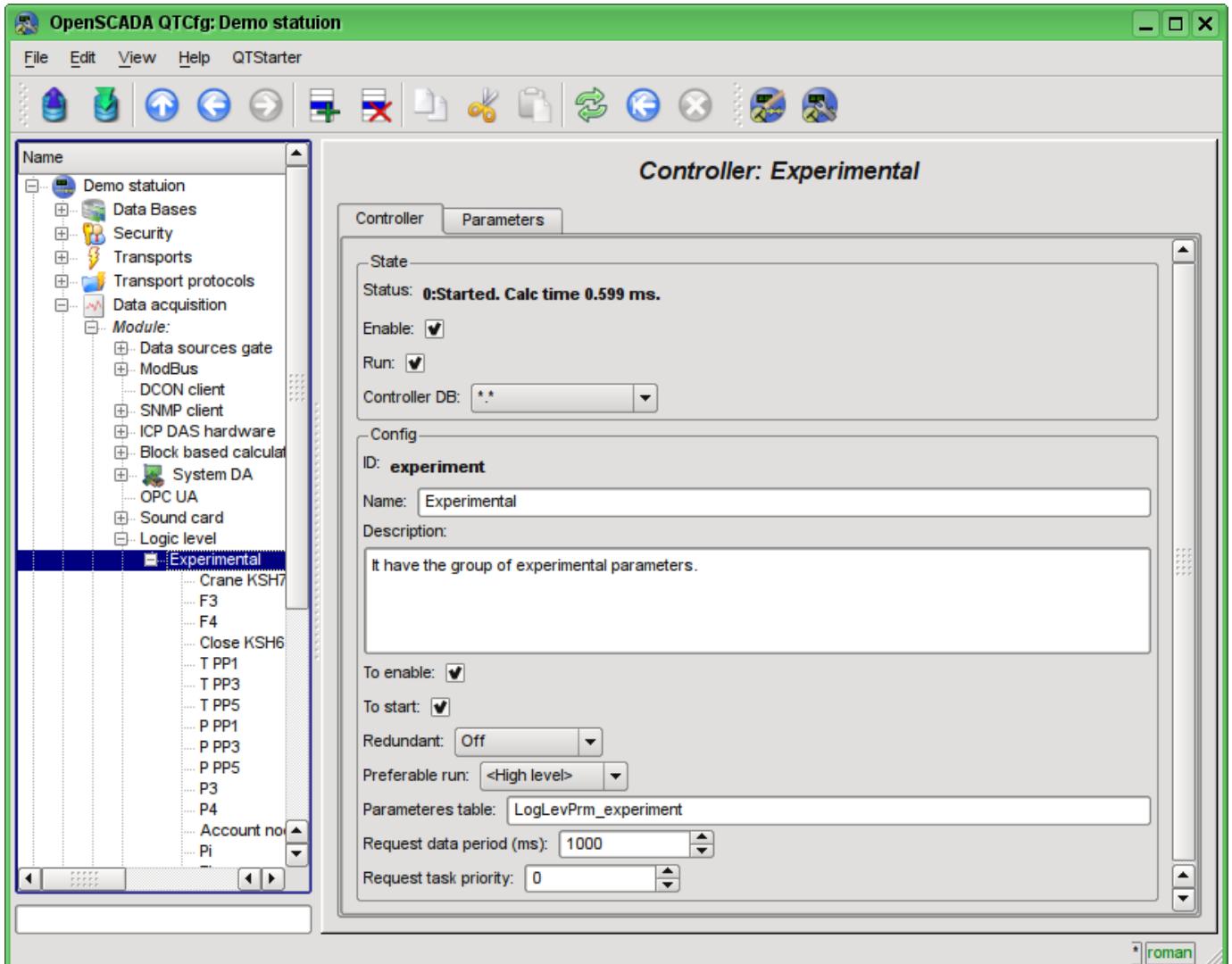
Fig. 4.5i. The main configuration tab of the controller of subsystem "Data acquisition".

"Parameters" tab (Fig.4.5j) contains a list of parameters in the controller, as well as information on the total number and the number of enabled parameters. In the context menu user can add, delete and move to the desired parameter.
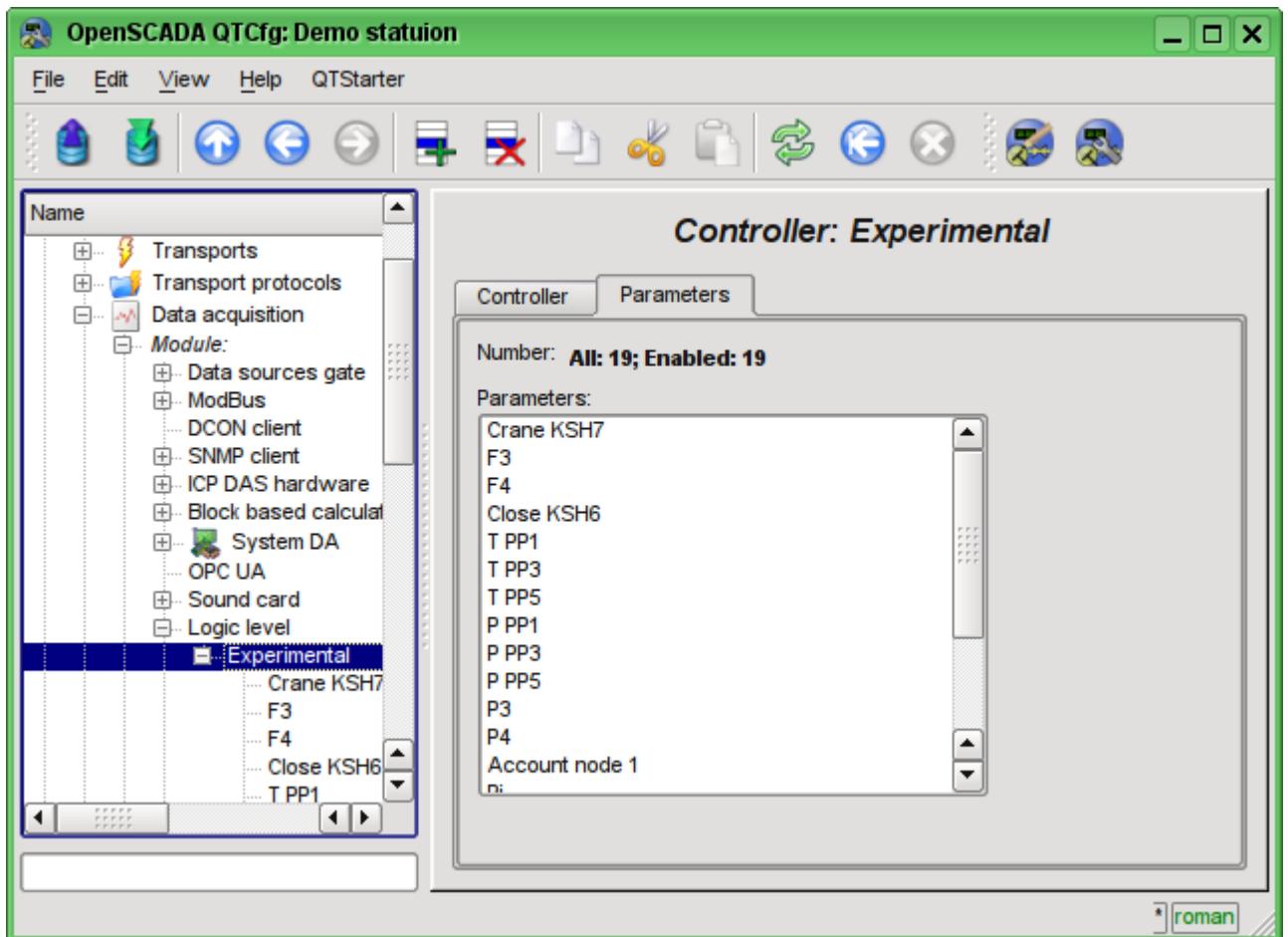
Fig. 4.5j. "Parameters" tab of the configuration page of the controller of subsystem "Data acquisition".

Parameters of the controllers of subsystem "Data acquisition" provides the configuration page with the tabs "Parameters", "Attributes", "Archiving" and "Template config". The tab "Template config" is not standard, but it is present only in the modules of subsystem "Data acquisition", which implement the mechanisms of working under the template in the context of the data source, which they are served. In this review this tab is included for logical completeness of the review of the configuration of templates of parameters of subsystem "Data acquisition" and as the final stage - using.

The tab "Parameter" (Fig.4.5k) contains the main settings:
- Section "State" - contains the properties, which characterize the state of the parameter:
    - *Type* - information on the parameter's type.
    - *Enable* - the state of the parameter "Enable". Enabled parameter is used by the controller fro data acquisition.
- Section "Config" - directly contains the configuration fields:
    - *ID* - information on the parameter's identifier.
    - *Name* - specifies the parameter's name.
    - *Description* - brief description of the parameter and its purpose.
    - *To enable* - indicates the status of "Run" in which to transfer the parameter at startup.
    - *Mode* - contains two fields: directly the mode and its configuration. In the case of the parameter of the controller of the module of this type it is the mode of working "under the template" and the address of the previously discussed template.

The tab "Attributes" (Fig.4.5l) contains the parametr's attributes and their values in accordance with the configuration of the used template and calculation of its program.

The "Archiving" tab (Fig.4.5m) contains the table with the attributes of a parameter in the columns and the archivers in rows. The user can set the archiving for the desired attribute with the required archiver

simply by changing the cell at the intersection.

The "Template config" tab (Figure 4.5n) contains the configuration fields in accordance with the template. In this example it is the group link on the external parameter. This link can be set simply by pointing the way to the parameter if the flag "Only attributes are to be shown" is not set, or to set the addresses of the attributes separately in the case if the flag is set.
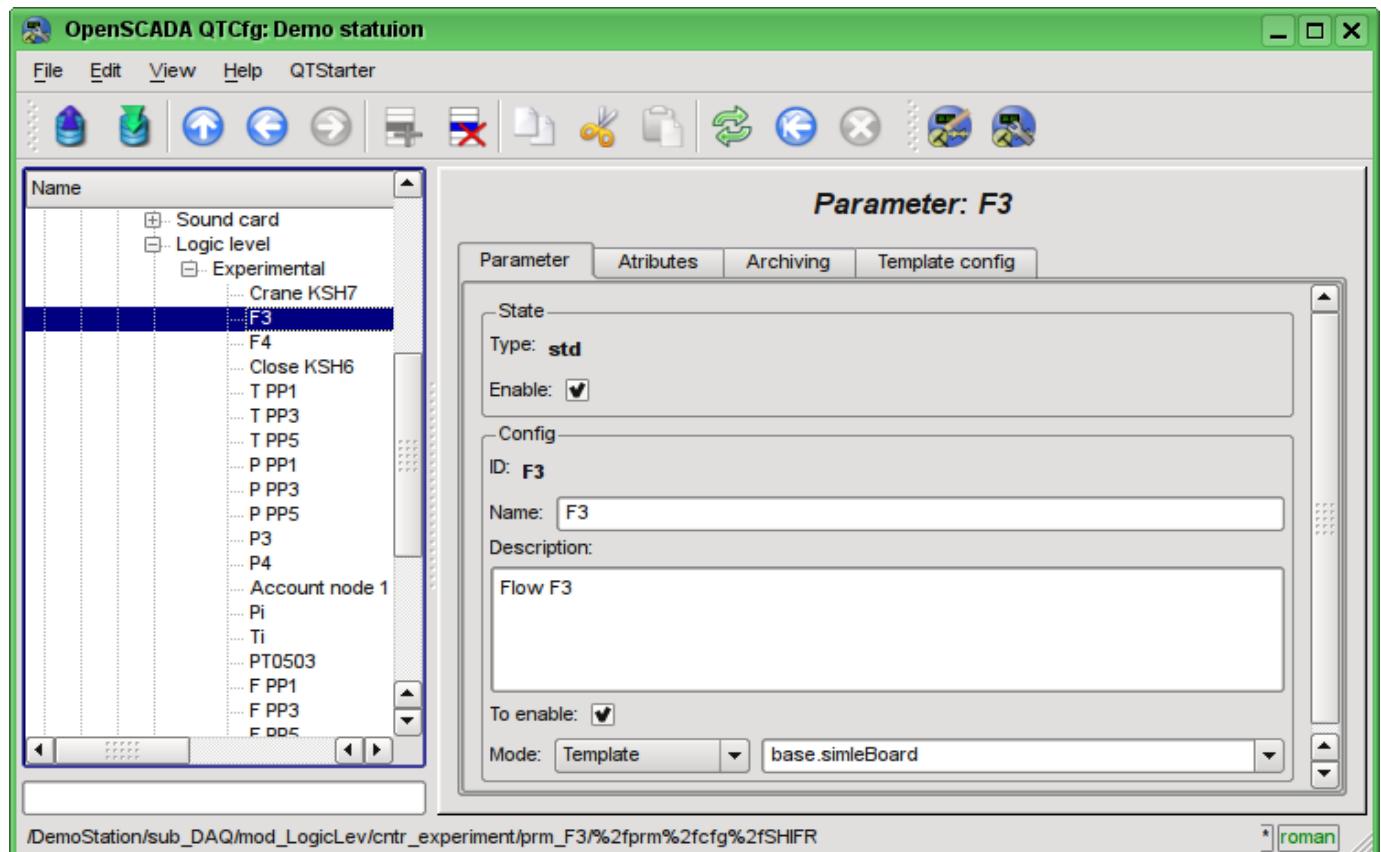
Fig. 4.5k. The main configuration tab of the parameter of the controller of subsystem "Data acquisition".
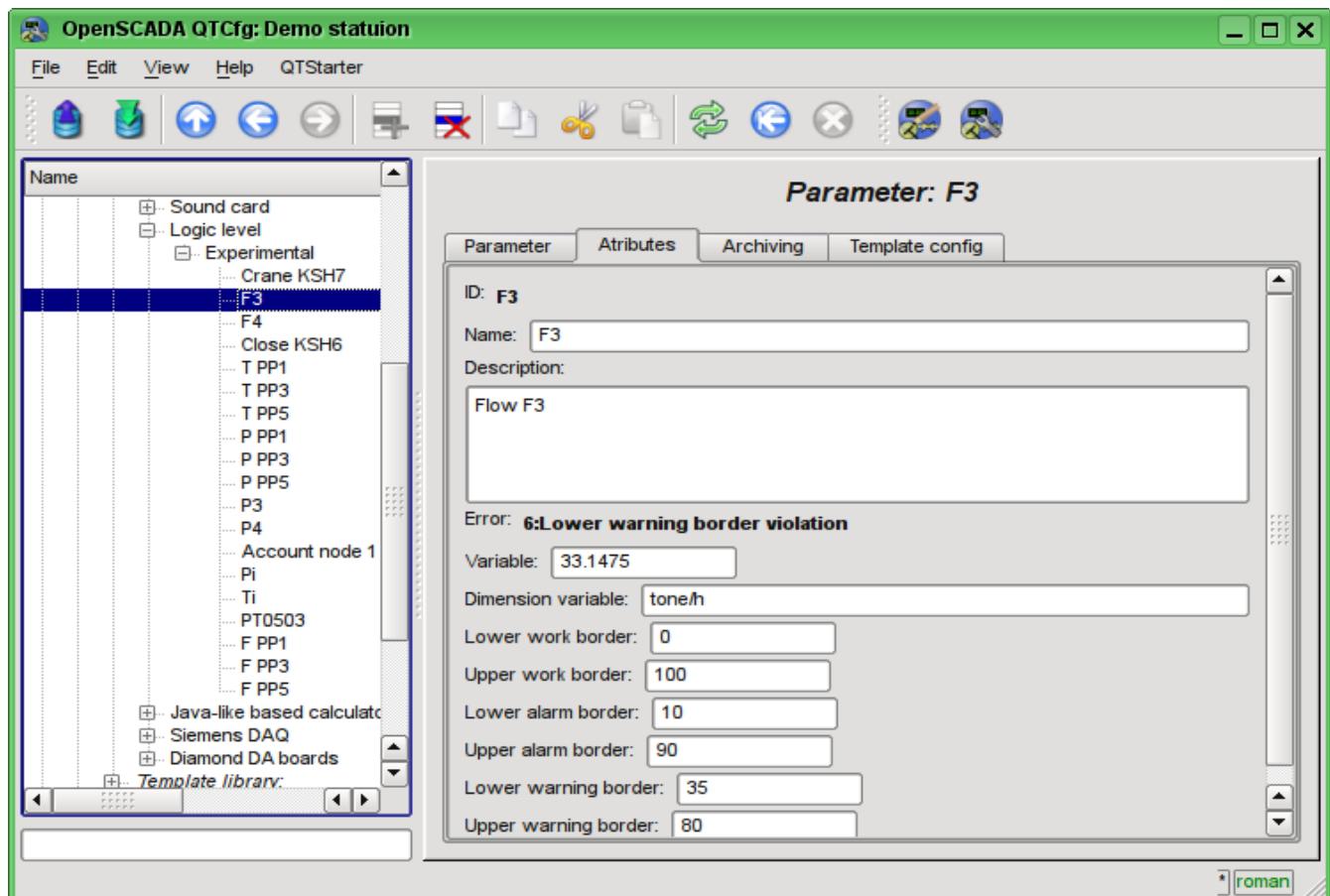
Fig. 4.5l. The "Attributes" tab of the parameter of the controller of subsystem "Data acquisition".
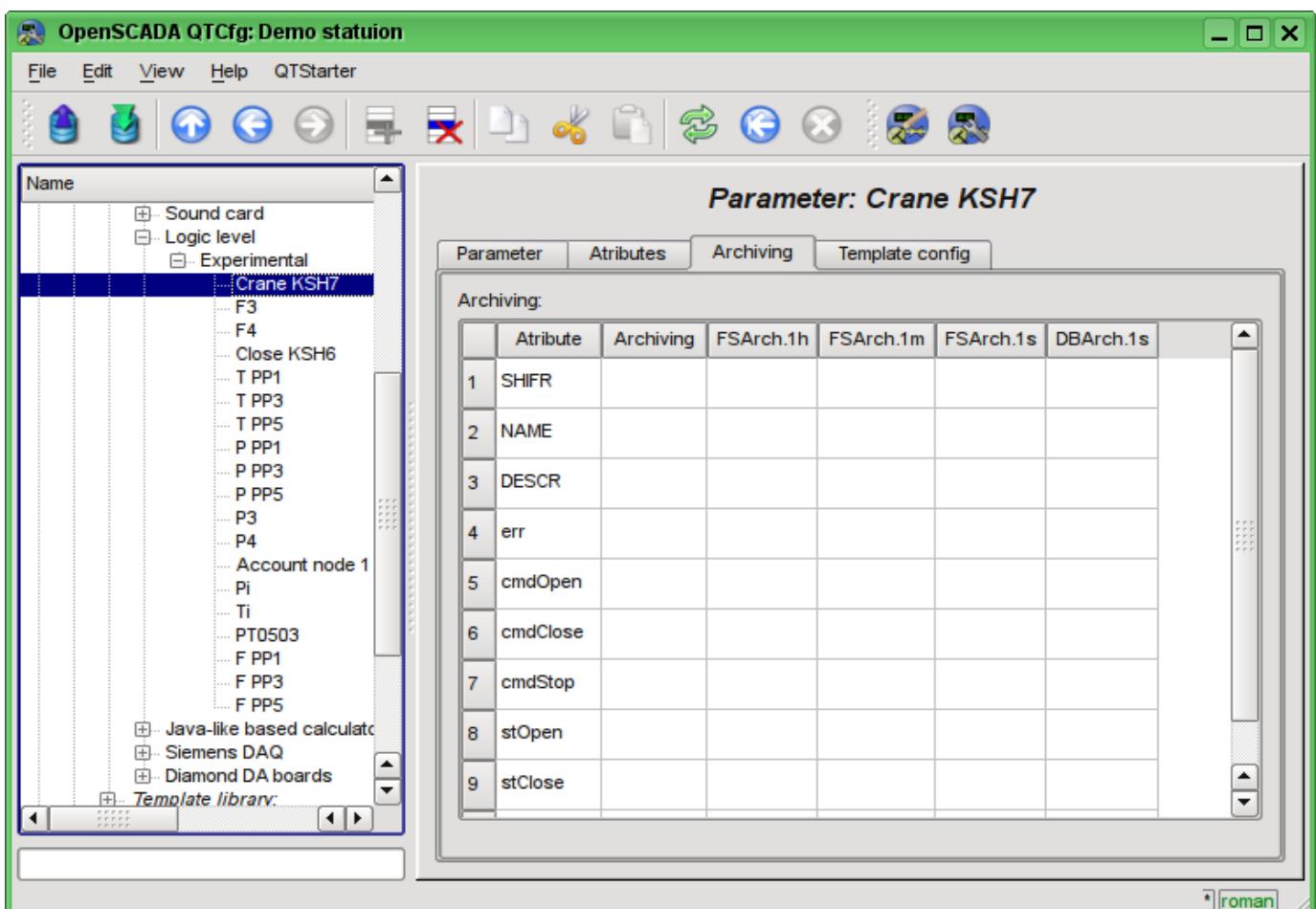
Fig. 4.5m. The "Archiving" tab of the parameter of the controller of subsystem "Data acquisition".
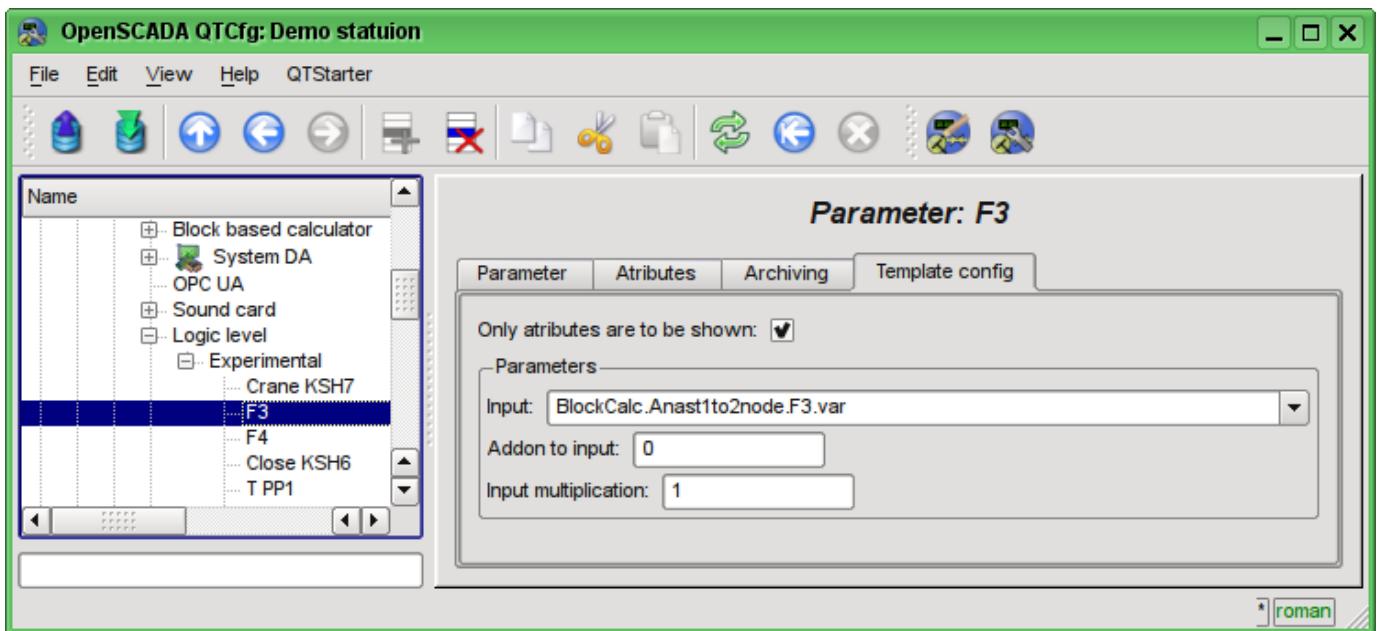
Fig. 4.5n. The "Template config" tab of the parameter of the controller of subsystem "Data acquisition".

4.6. Subsystem "Archives"

The subsystem is modular and contains the hierarchy of objects depicted in Fig.4.6a. To configure the subsystem the root page of the subsystem "Archives" is provided, it contains tabs "Messages archive", "Value archives", "Modules" and "Help".

To gain the access to modify the objects of this subsystem the user of the group "Archive" or the privileged user rights are required.
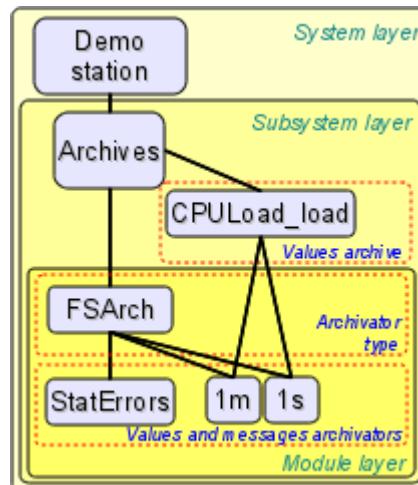
Fig. 4.6a. The hierarchical structure of subsystem "Archives"

The "Messages archive" tab (Fig.4.6b) contains the configuration of messages archive and the request form of messages from the archive.

Configuration of the messages archive is represented by the fields:
- *Maximum requested messages* - indicates the global limit on the maximum number of messages processed by the request.
- *Messages buffer size* - indicates the dimension of the area of memory reserved for the interim buffer of messages. Messages from the buffer are requested for viewing and archived with the messages archivers.
- *Archiving period (s)* - the periodicity with which the archivers select messages from the buffer for their archiving.

The messages request form contains the configuration fields of the request and the table of results. Configuration fields of the request are:
- *Time* - specifies the request time.
- *Size (s)* - specifies the size and the depth of the request in seconds.
- *Category pattern* - specifies the category of the requested messages. In the category you can specify the elements of a sample of the template, namely, the characters '*' - for any string and '?' - for any character.
- *Level* - indicates the minimum level of messages, ie request will be processed for messages with a level more than or equal to the specified one.
- *Archivator* - indicates the messages archiver, for which the request is to be processed. If the value is missing, the request will be processed for the buffers and all archivers. If <buffer> is specified, then the request will be processed only for the messages buffer.

The result table contains rows of messages with the following columns:
- *Time* - message's time.
- *Category* - message's category.
- *Level* - message's level.
- *Message* - message's text.
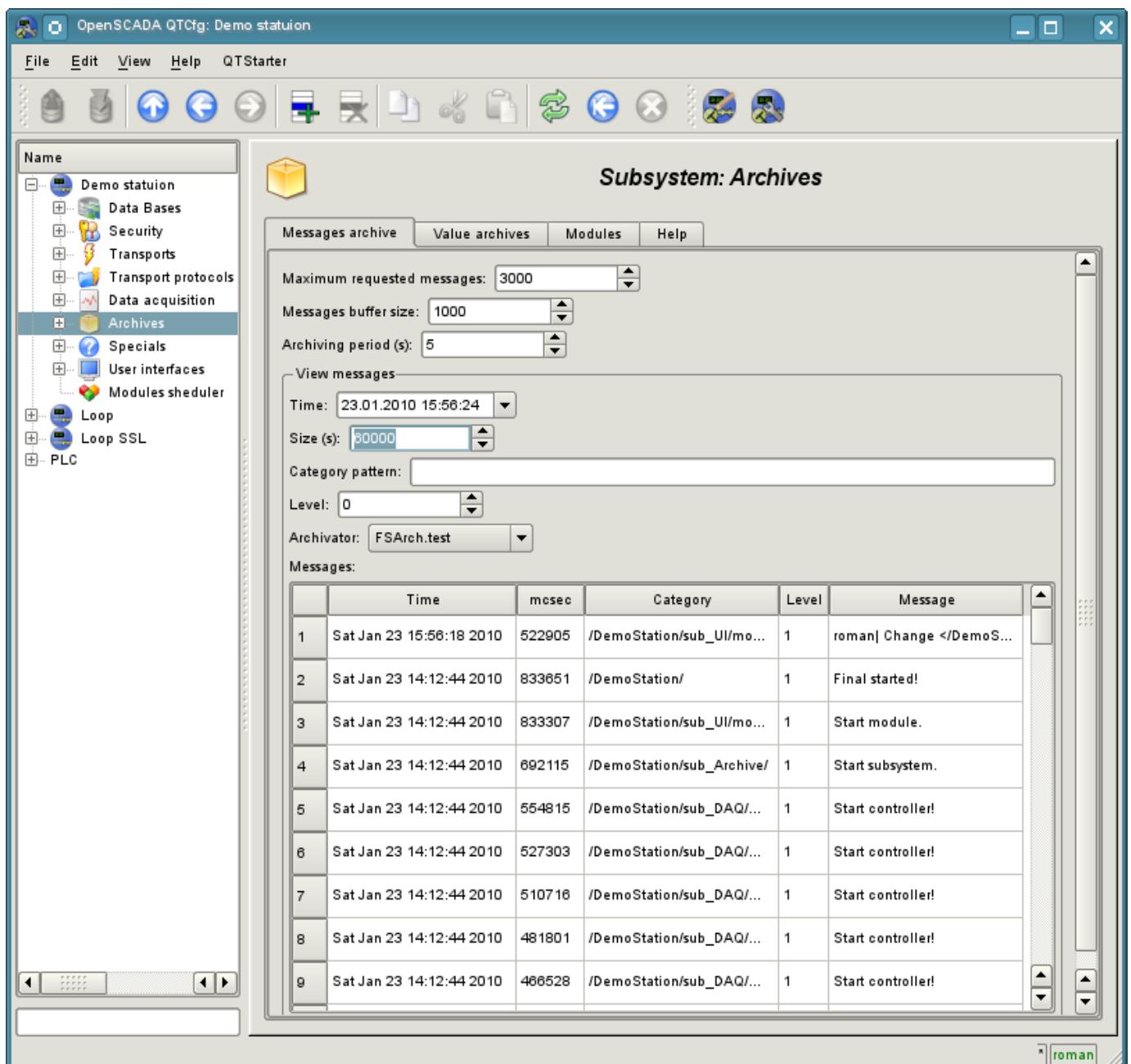
Fig. 4.6b. The "Messages archive" tab of the subsystem "Archives".

Tab "Value archives" (Fig.4.6c) contains the general configuration of value's archiving and the list of archives of values. In the context menu of the list of values the user has the opportunity to add, delete and move to the desired archive. The general configuration of archiving is represented by the fields:

- *Get data period (ms)* - indicates the periodicity of the active archiving task. In fact, the highest level of detail or the minimum period of active archives is determined by this value.
- *Get data task priority level* - sets the priority of task of active archiving. It is used when scheduling the operating system tasks. In the case of execution of the station with the rights of the superuser "root" this field includes scheduling of the archiving task in real time and with the specified priority.

The "Modules" tab (Fig. 4.1b) contains a list of modules in subsystem "Archives" and is identical for all modular subsystems. The "Help" tab contains the brief help for this page.
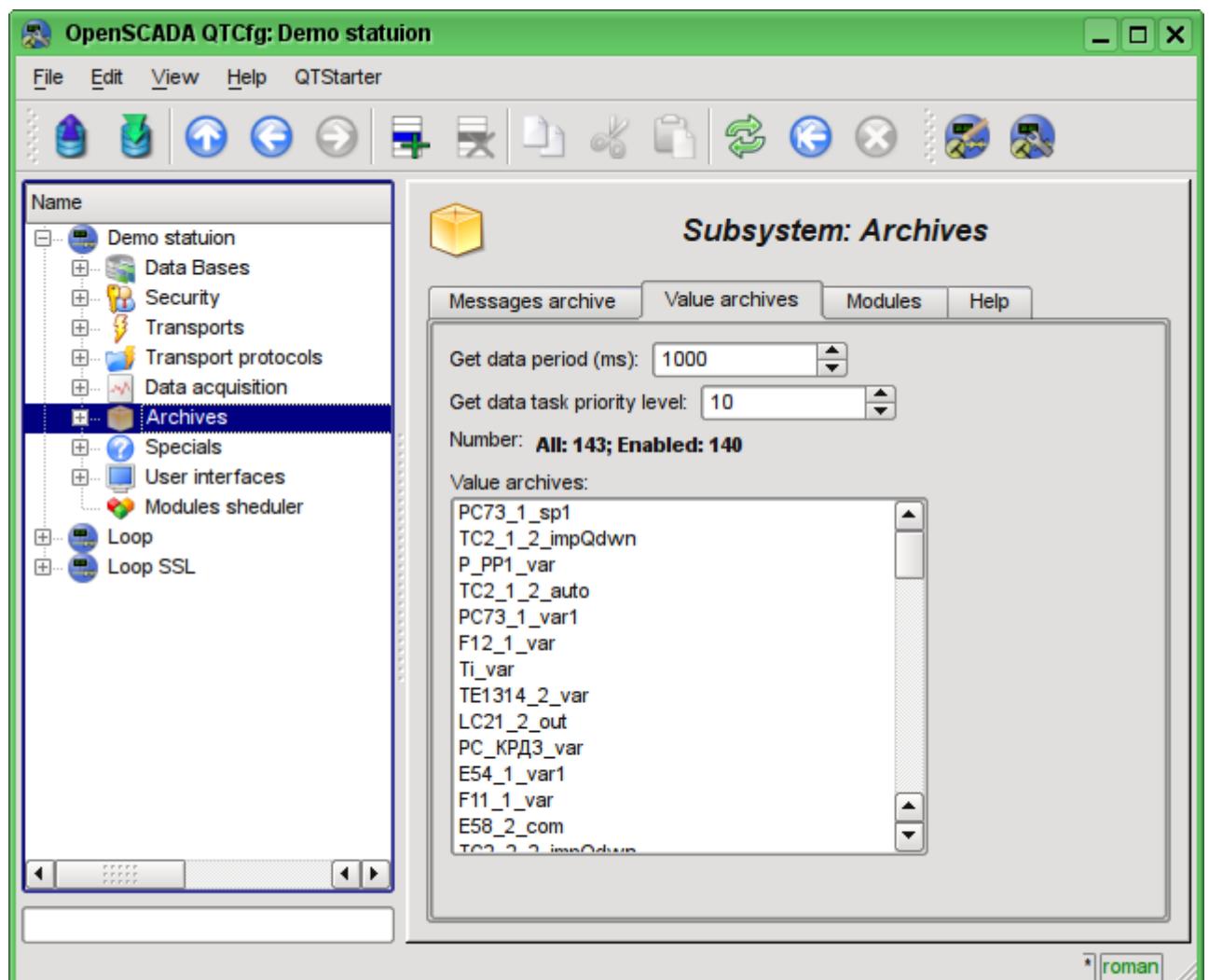
Fig. 4.6c. The "Value archives" tab of the subsystem "Archives".

Archive of values of subsystem "Archives" provides the configuration page with the tabs "Archive", "Archivators" and "Values".

Tab "Archive" (Fig.4.6d) contains the basic settings of the archive:
- Section "State" - contains the properties, which characterize the state of the archive:
  - *Running* - the state of the parameter "Running". Running archive collects data in the buffer and is served by the archivators.
  - *Archive DB* - database address for storing the archive's data.
- Section "Config" - directly contains the configuration fields:
  - *ID* - information on the archive's identifier.
  - *Name* - specifies the archive's name.
  - *Description* - brief description of the archive and its purpose.
  - *To start* - indicates the state "Running" in which to transfer the archive at startup.
  - *Value type* - indicates the type of values which are stored in the archive from the list:

"Boolean", "Integer", "Real" и "String".

•   *Source* - indicates the type and address of the source. Type of source is indicated from the list: "Passive", "Passive param. attribute" or "Active param. attribute". Passive archive does not have an associated source of values, the data to the such archive the source transfers by itself. Types with the attribute of the parameter in the address field indicate the parameter of the subsystem "Data acquisition" as the source. Passive attribute of the parameter sends data to the archive by itself with its own period of data acquisition. Active attribute of the parameter is queried by the archiving task of this subsystem.

•   *Buffer period (s)* - indicates the periodicity of values in the archive's buffer.

•   *Buffer size (items)* - indicates the dimensionality and depth of the archive's buffer. The dimensionality is usually set in terms of 60 sec of the periodicity of the archiving task with the reserve.

•   *Buffer hard time griding* - indicates the mode of the buffer. The hard grid mode involves the memory reservation for each value, but without the timestamp. This mode eliminates the possibility of packaging the adjacently-identical values, but also saves on storage of the timestamp. Otherwise, the buffer operates in the mode of storage the value and timestamp and supports the packaging of adjacently-identical values.

•   *Buffer high time resolution* - indicates the possibility of storing values at intervals up to 1 microsecond, differently the values can be stored at intervals up to 1 second.
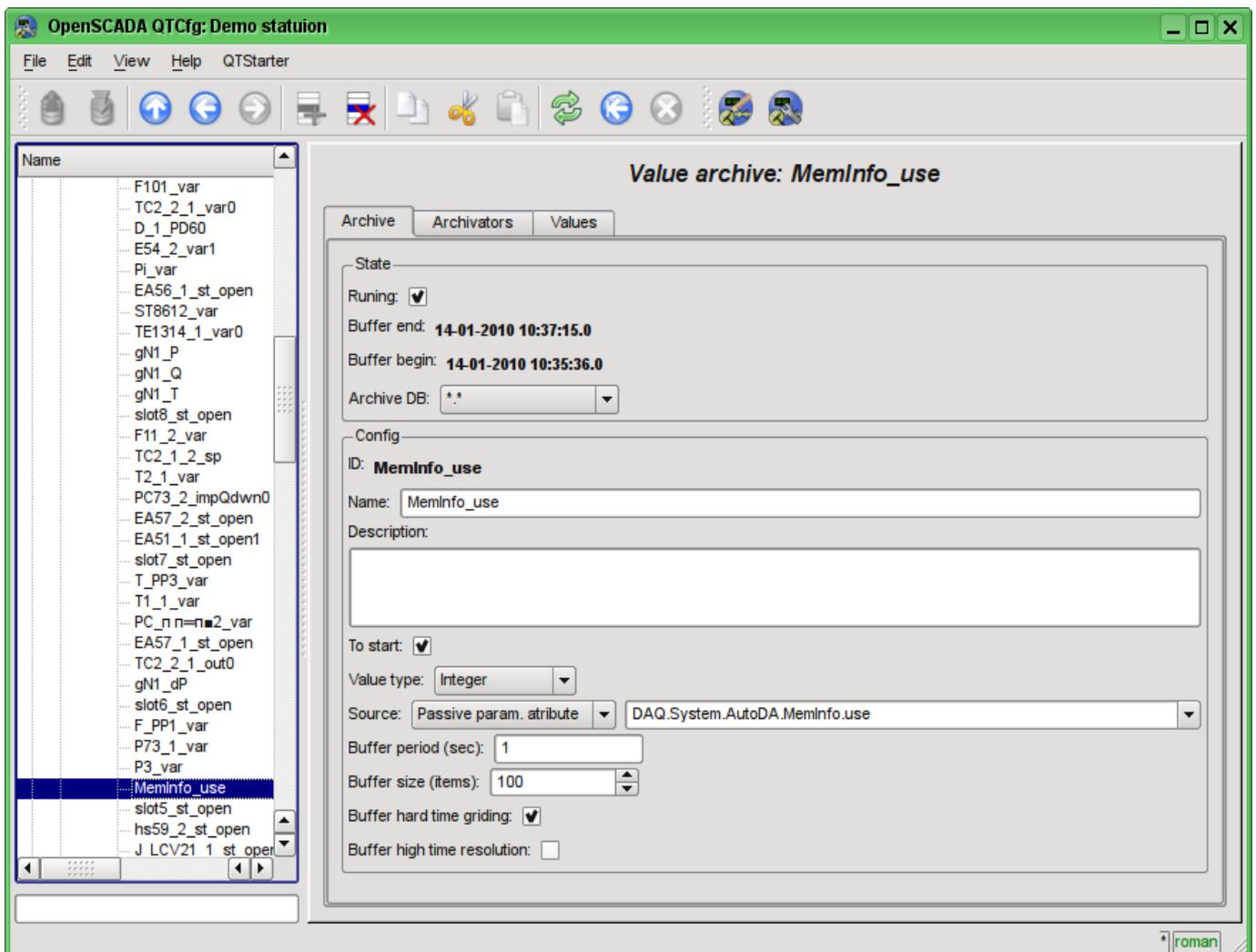
Fig. 4.6d. The main configuration tab of the values' archive of subsystem "Archives".

Tab Archivators' (Fig.4.6e) contains the table with the configuration of the processing of the archive by the available archivers. Lines are available archivers, and the columns are the following parameters:

- *Archivator* - information on the archiver's address.
- *Start* - information on the archiver's state "Started".
- *Process* - sign of the processing this archive be the archiver. The field is available for modification by the user.
- *Period (s)* - information on the periodicity of the archiver.
- *Begin* - date of the archive data beginning in the archiver.
- *End* - date of the archive data ending in the archiver.
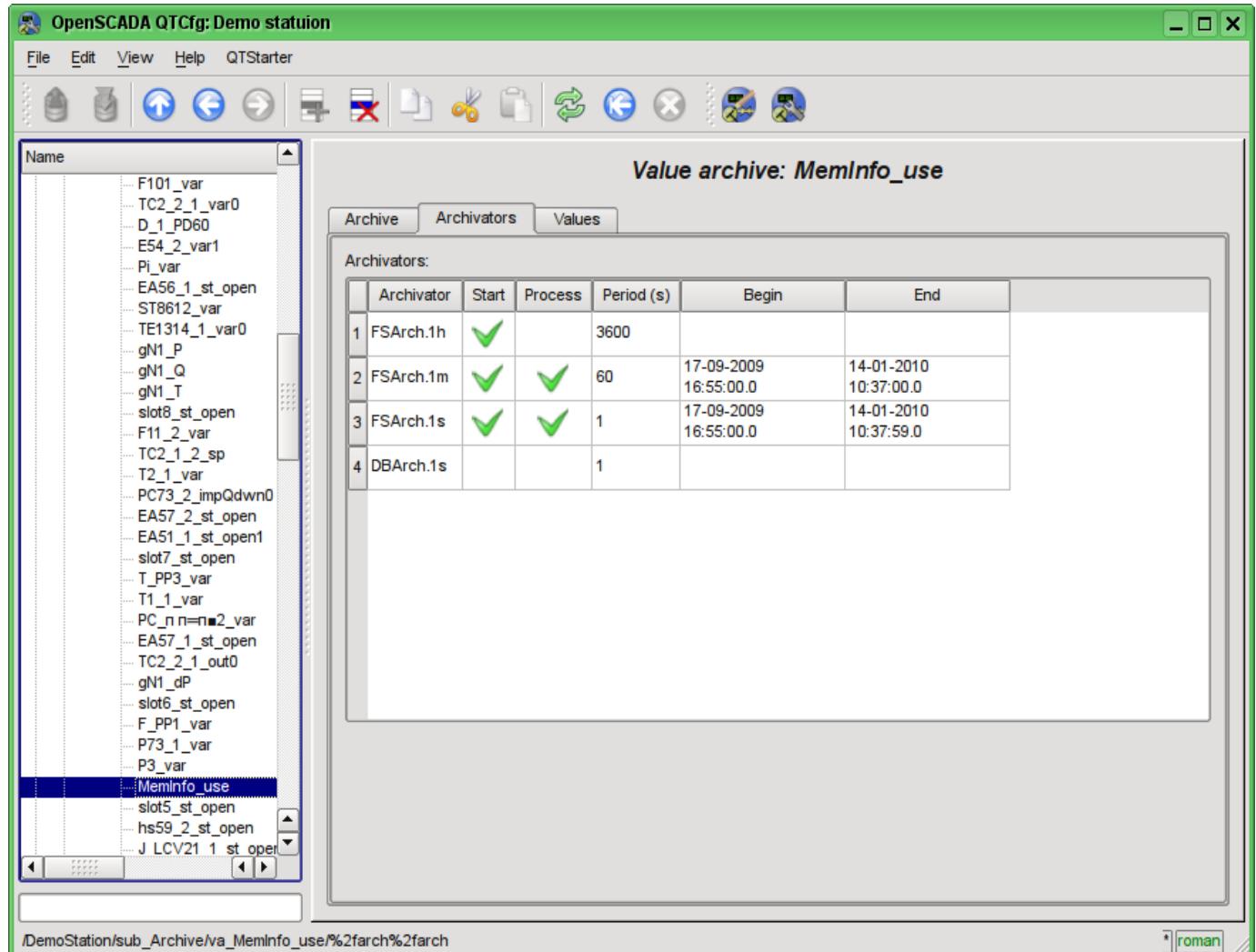
Fig. 4.6e. The "Archivators" tab of the values archive of subsystem "Archives".

Tab "Values" (Fig.4.6f) contains the values request in the archive and the result as a table of values or image of the trend. Values request contains the fields:
- *Time* - indicates the time of request. It contains two fields: the field of date + time and microseconds.
- *Size (s)* - specifies the size or depth of the request in seconds.
- *Archivator* - indicates values archiver for which the request is to be processed. If the value is missing, the request will be processed for the buffer and for all archivers. If the <buffer> is specified, then the request will be processed only for the archive's buffer.
- *Show trend* - indicates the necessity for presentation of the archive's data in the form of a graph (trend), otherwise the result is presented in a table that contains only time and value. In the case of installation of this field the schedule is formed and displayed, in addition additional configuration fields of the image settings are appeared:
  - *Picture size* - indicates the width and height of the generated image in pixels.
  - *Value scale* - indicates the lower and upper limit of the scale of value. If both values are set to 0 or equal, then the scale will be determined automatically depending on the values.
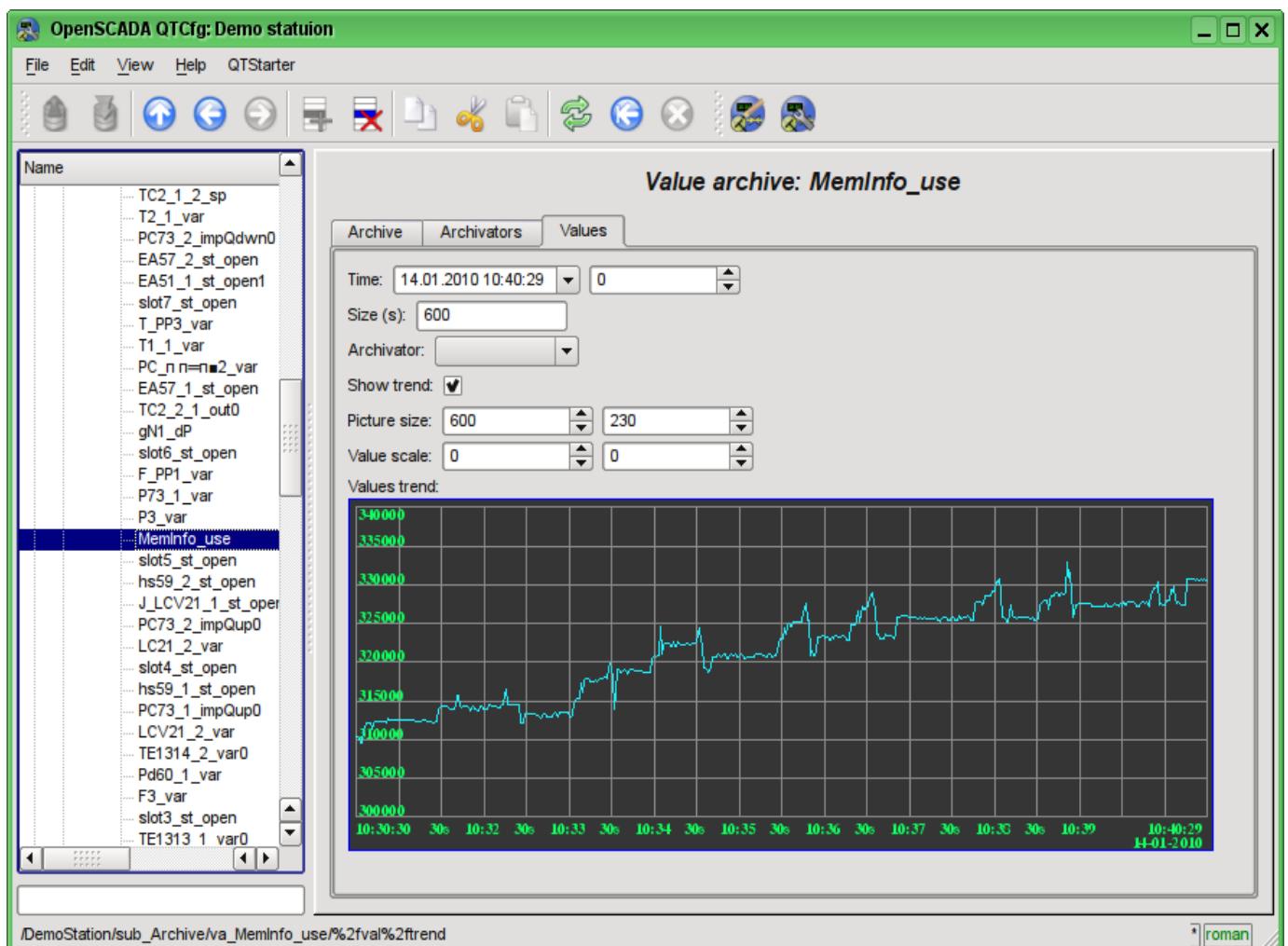
Fig. 4.6f. The "Values" tab of the values archive of subsystem "Archives".

Each module of the "Archives" subsystem provides configuration page with the tabs "Archivators" and "Help". The "Archivators" tab (Fig.4.6g) contains a list of messages and values archivers registered in the module. The context menu of the list provides user with possibility to add, delete and move to the desired controller. The "Help" tab contains information about the module of subsystem "Archives" (Fig. 4.1d), whose structure is identical for all modules.
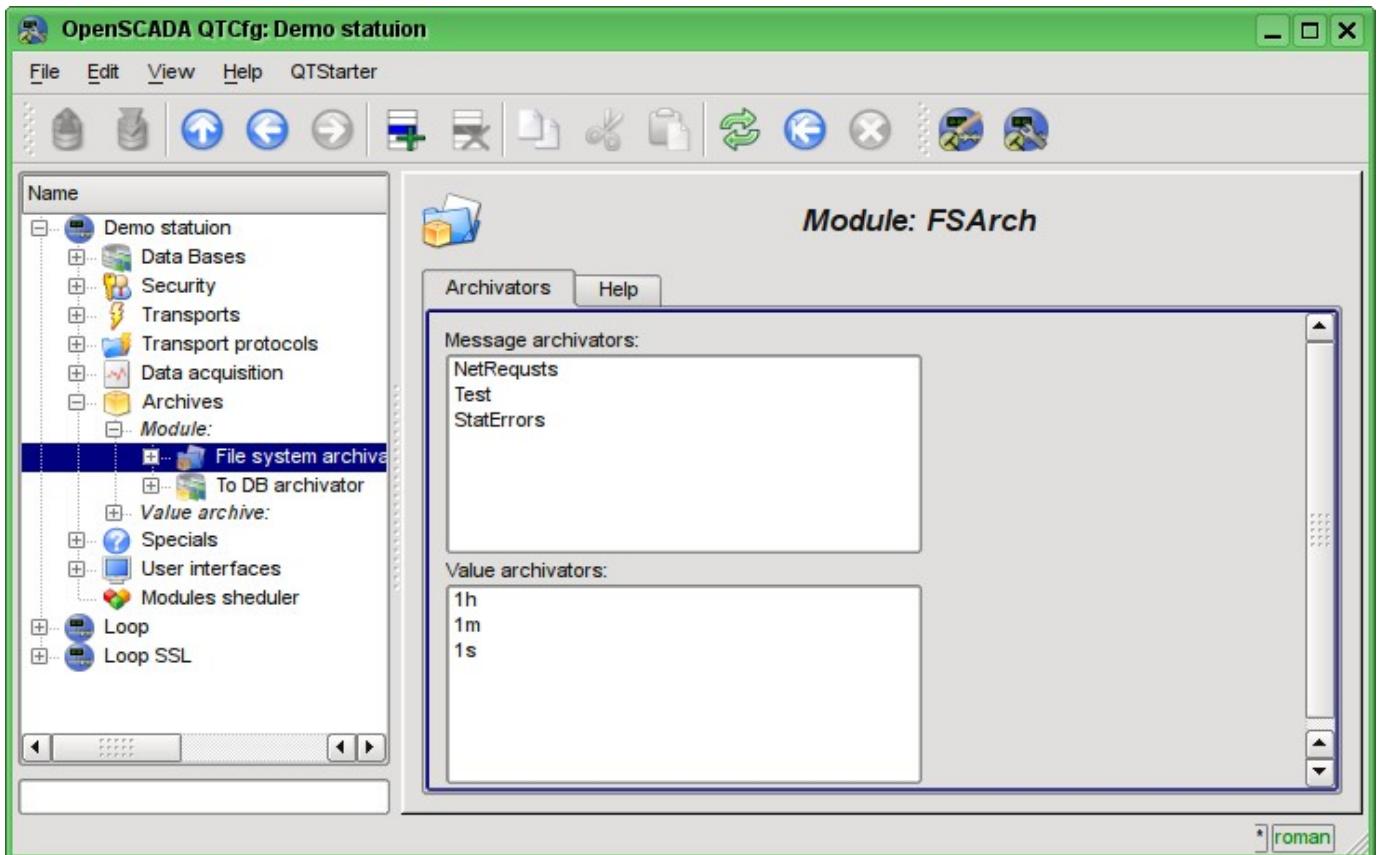
Fig. 4.6g. The "Archivators" tab of the module of subsystem "Archives".

Messages archivers contains their own configuration page with tabs "Archivator" and "Messages".

The "Archivator" tab (Fig.4.6h) contains the basic settings. The structure of these settings may differ slightly from one module of this subsystem to another as you can find in the own documentation of modules. As an example we shall examine the settings of the messages archiver from the module of the archive on the file system Arch.FSArch Settings:
• Section "State" - contains the properties, hich characterize the archivers' state:
    • *Running* - archivers' state "Running". The running archiver processes the messages archive buffer and puts his data in its repository, but also it processes requests for access to data in the repository.
    • *Archivator DB* - database address for storing the archiver's data.
    • *End* - date + time of the last data in the archiver's repository.
    • *Begin* - date + time of the first data in the archiver's repository.
    • *Archivator files size (kB)* - information about the total size of the archiver's files with the data.
    • *Archiving time (ms)* - time spent on the archiving of messages archive data.
• Section "Config" - directly contains the configuration fields:
    • *ID* - information on the archiver's identifier.
    • *Name* - indicates the archiver's name.
    • *Description* - brief description of the archiver and its purpose.
    • *Address* - address of the storage in the specific for the type of archiver (module) format. Format description usually available in the tooltip for this field. In the example it is the relative path to the storage directory.
    • *Message level* - indicates the level of archiver's messages. Messages with a level greater than or equal to the specified one are processed by the archiver.
    • *Message categories* - list of categories, separated by ';', of messages. Messages matched

with the templates of categories will be processed by the archiver. In the category you can specify the elements of a sample of the template, namely, the characters '*' - for any string and '?' - for any character.
- *To start* - indicates the status "Running", in which to transfer archiver at startup.
- Section "Additional options" - specialized section for module about the contents of which you can read in the documentation on the module.
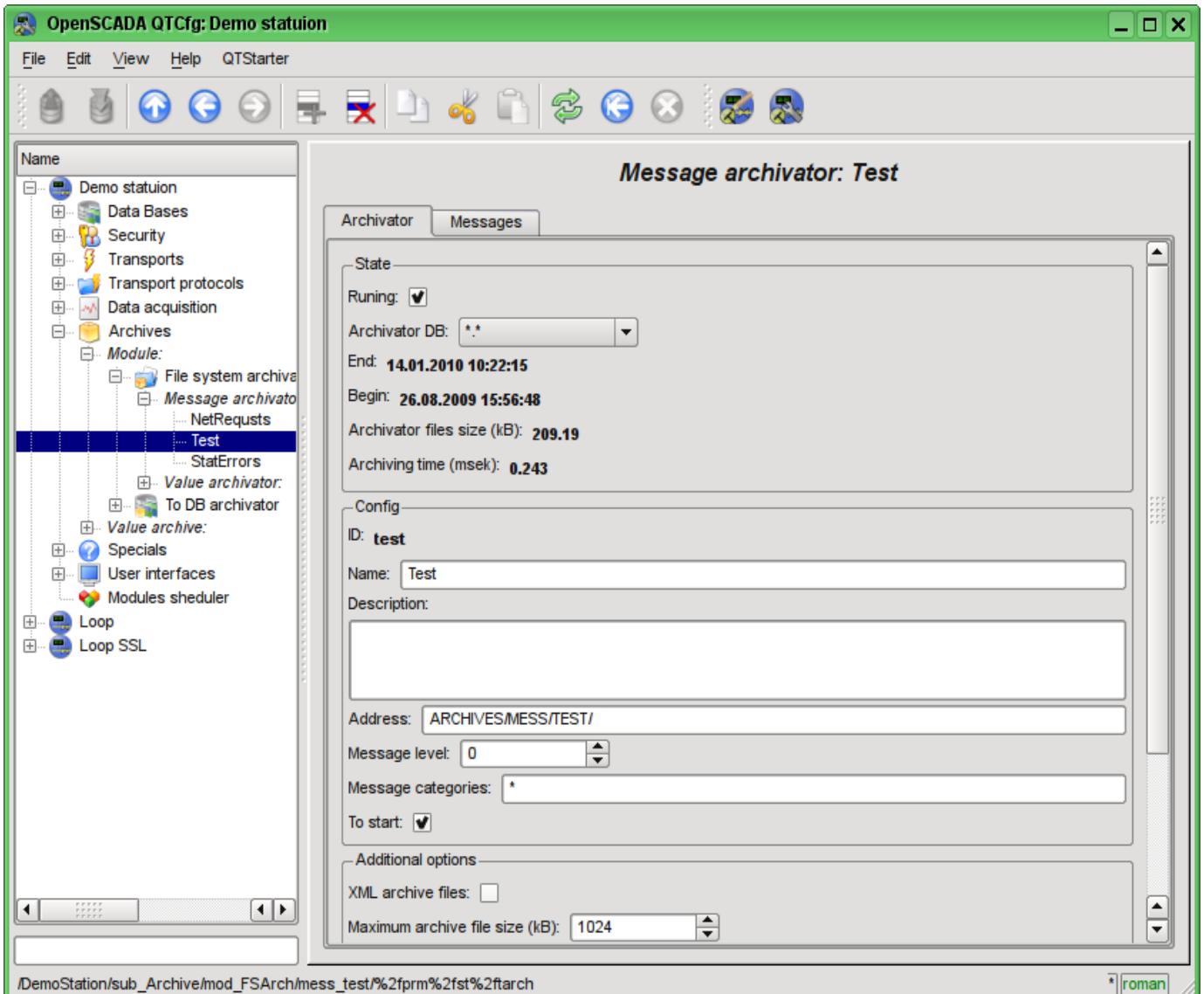
Fig. 4.6h. The main tab of the messages archiver configuration of subsystem "Archives".

The "Messages" tab (Fig.4.6i) contains the form of the messages request from the archive of the archiver:
- *Time* - indicates the time of the request.
- *Size (s)* - indicates the size and depth of the request in seconds.
- *Category pattern* - indicates the category of the requested messages. In the category you can specify the elements of a sample of the template, namely, the characters '*' - for any string and '?' - for any character.
- *Level* - indicates a minimum level of messages, ie the request will be processed for messages with the level greater or equal to the specified one.

The result table contains messages rows with the following columns:
- *Time* - message time.
- *Category* - message category.
- *Level* - message level.
- *Message* - message text.
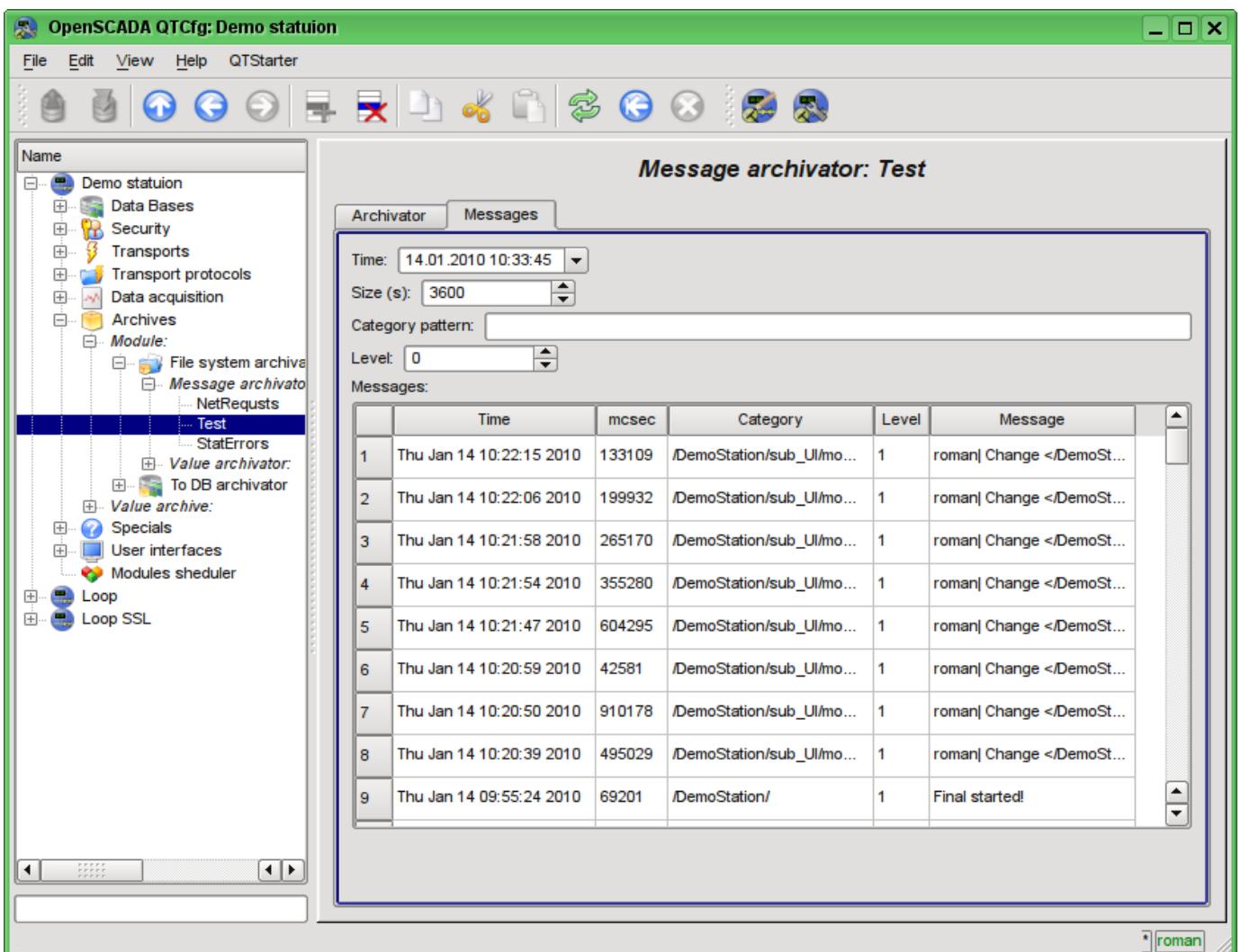
Fig. 4.6i. Tab of the messages request "Messages" of the messages archiver of subsystem "Archives".

Values archivers contains their own configuration page with tabs "Archivator" and "Archives".

The "Archivator" tab (Fig.4.6j) contains the basic settings. The structure of these settings may differ slightly from one module of this subsystem to another as you can find in the own documentation of modules. As an example we shall examine the settings of the messages archiver from the module of the archive on the file system Arch.FSArch Settings:

- Section "State" - contains the properties, hich characterize the archivers' state:
    - *Running* - archivers' state "Running". The running archiver processes the messages archive buffer and puts his data in its repository, but also it processes requests for access to data in the repository.запросы на доступ к данным в хранилище.
    - *Archiving time (ms)* -information about the time spent on archiving data of the archives buffers. Periodicity of archiving is set in the field "Period archiving" in the section "Config" of the tab.
    - *Archivator DB* - database address for storing the archiver's data.
- Section "Config" - directly contains the configuration fields:
    - *ID* - information on the archiver's identifier.
    - *Name* - indicates the archiver's name.
    - *Description* - brief description of the archiver and its purpose.
    - *Value period (s)* - indicates the periodicity of values that are contained in the archiver's repository.
    - *Period archiving (s)* - indicates the periodicity of the archives buffers data archiving task. The dimension of the archives buffers in the time expression must not be less, and preferably somewhat greater then the periodicity of the of archiving task.
    - *Address* - address of the storage in the specific for the type of archiver (module) format. Format description usually available in the tooltip for this field. In the example it is the

relative path to the storage directory.
- *To start* - indicates the status "Running", in which to transfer archiver at startup.
- Section "Additional options" - specialized section for module about the contents of which you can read in the documentation on the module.
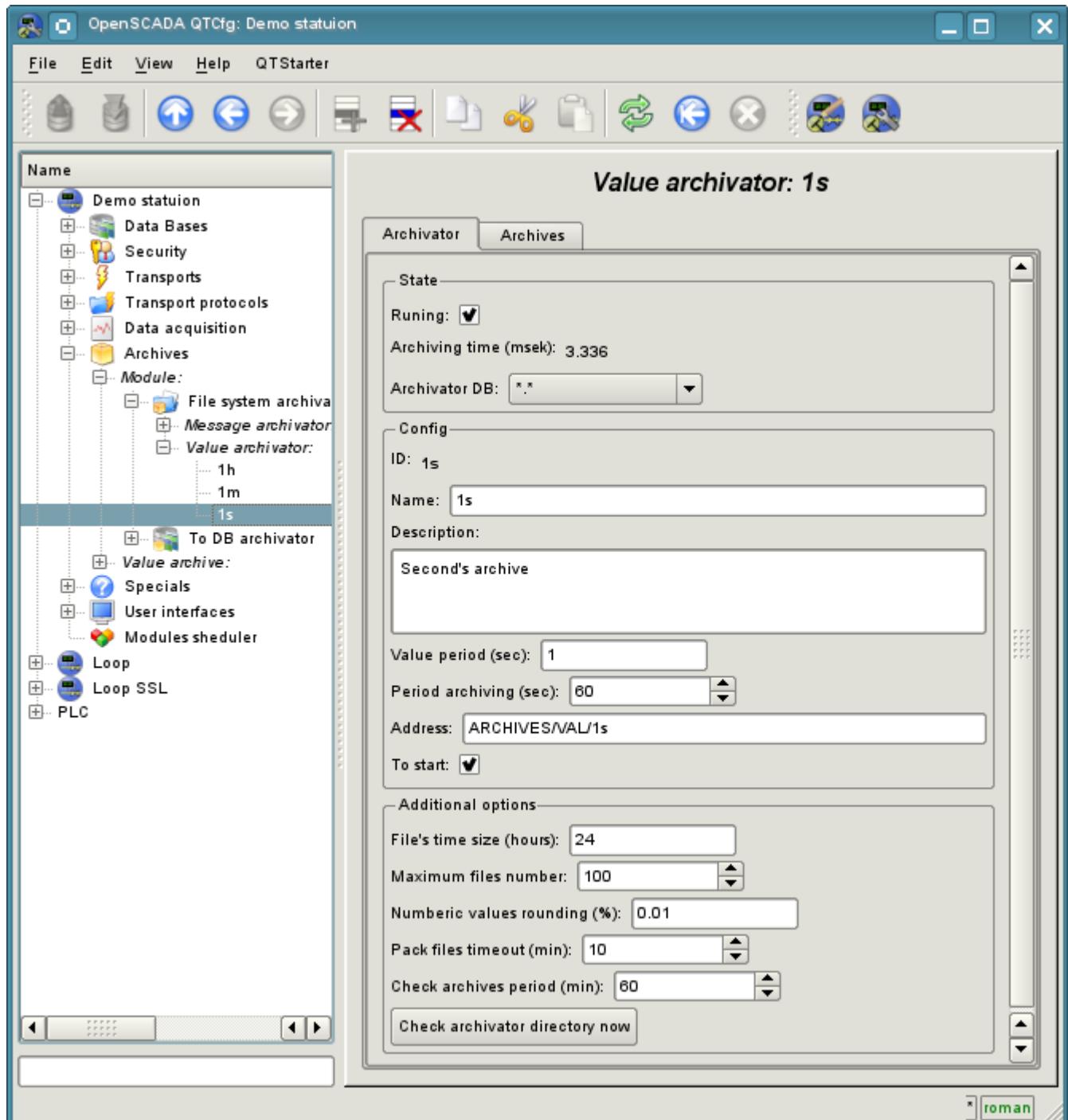
Fig. 4.6j. The main tab of the values archiver configuration of subsystem "Archives".

The "Archives" tab (Fig.4.6k) contains a table with information about the archives being processed by the archiver. In the rows the table contains archives, and in the columns - the following information:
- *Archive* - archive's name.
- *Period (s)* - archive's periodicity in seconds.
- *Buffer size* - buffer's dimension in units.
- *Files size (Mb)* - specific to the module Arch.FSArch field with information about the total size of the files of the archiver's storage for the archive.

In the case of the module Arch.FSArch in this tab you can find the form of export the archiver's data.
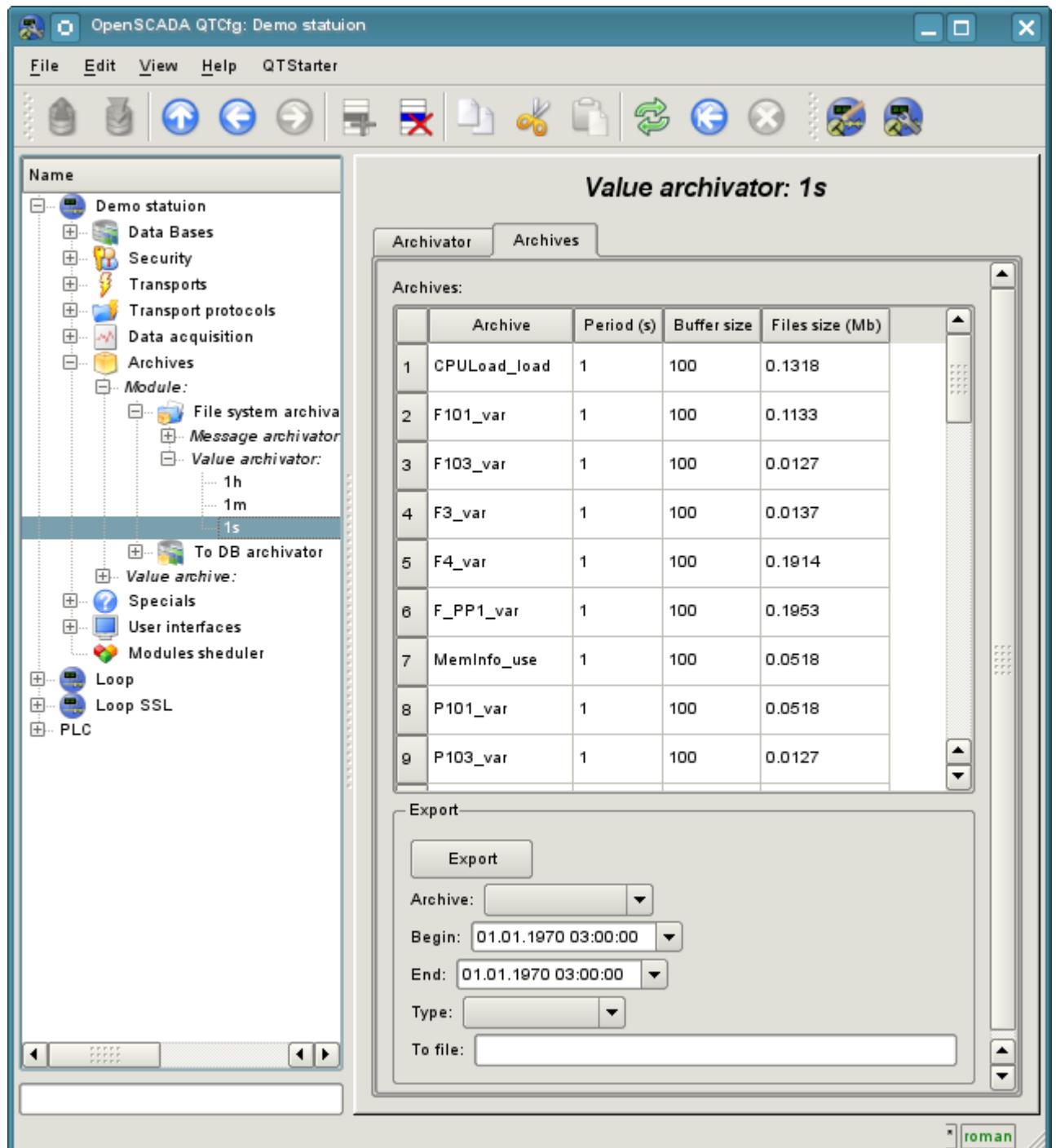
Fig. 4.6k. The "Archives" tab of the values archiver of subsystem "Archives".

4.7. Subsystem "User interfaces"

The subsystem is modular. To configure the subsystem the root page of the subsystem "User Interfaces" is provided, it contains the tabs "Modules" and "Help". The "Modules" tab (Fig. 4.1b) contains a list of modules of subsystem and it is identical for all modular subsystems. The "Help" tab contains a brief help for this page.

Each module of the subsystem "User Interfaces" provides configuration page with the tabs "User Interface" and "Help". The "User Interface" tab (Fig.4.7a) provides the parameter for monitoring the "Running" status of the module, as well as the configuration sections specialized for the modules of this subsystem. On the "Help" tab there is an information about the module of the subsystem "User Interfaces" (Fig. 4.1d), which structure is identical for all modules.
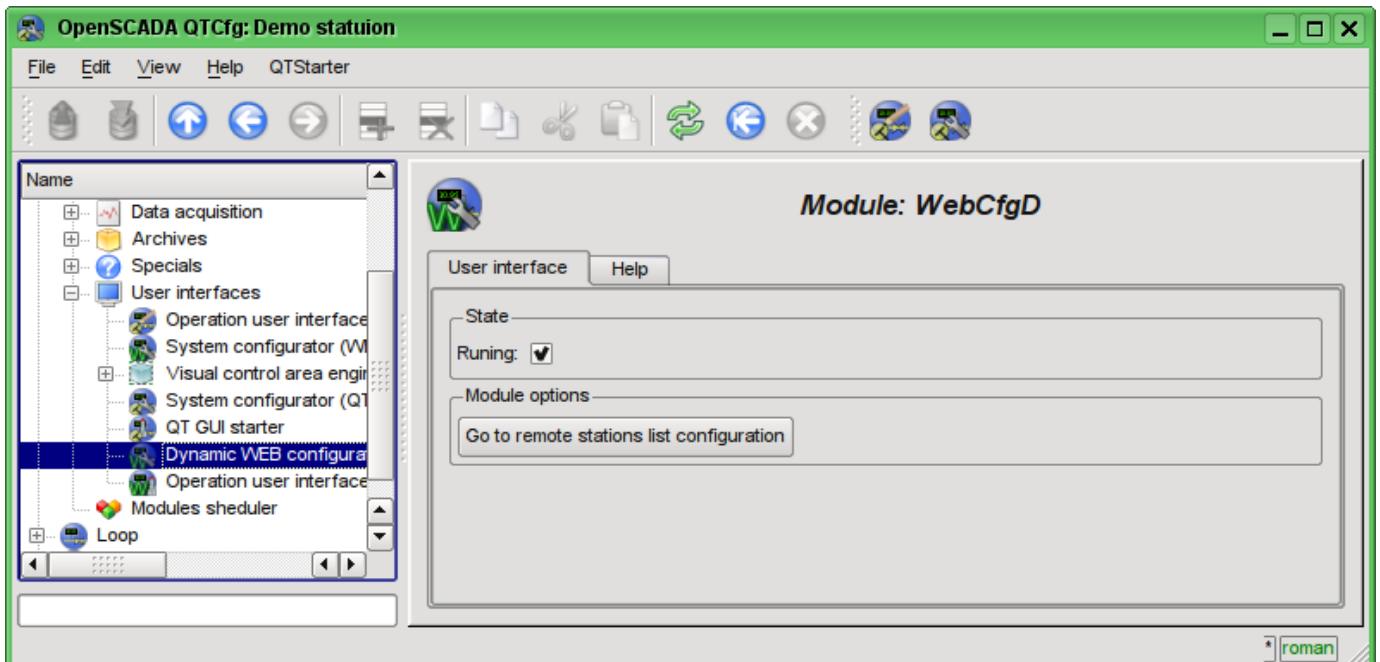
Fig. 4.7a. The "User Interface" tab of the module of subsystem "User Interfaces".

4.8. Subsystem "Specials"

The subsystem is modular. To configure the subsystem the root page of the subsystem "User Interfaces" is provided, it contains the tabs "Modules" and "Help". The "Modules" tab (Fig. 4.1b) contains a list of modules of subsystem and it is identical for all modular subsystems. The "Help" tab contains a brief help for this page.

Each module of the subsystem "Specials" provides configuration page with the tabs "Special" and "Help". The "Special" tab (Fig.4.8a) provides the parameter for monitoring the "Running" status of the module, as well as the configuration sections specialized for the modules of this subsystem. On the "Help" tab there is an information about the module of the subsystem "Specials" (Fig. 4.1d), which structure is identical for all modules.
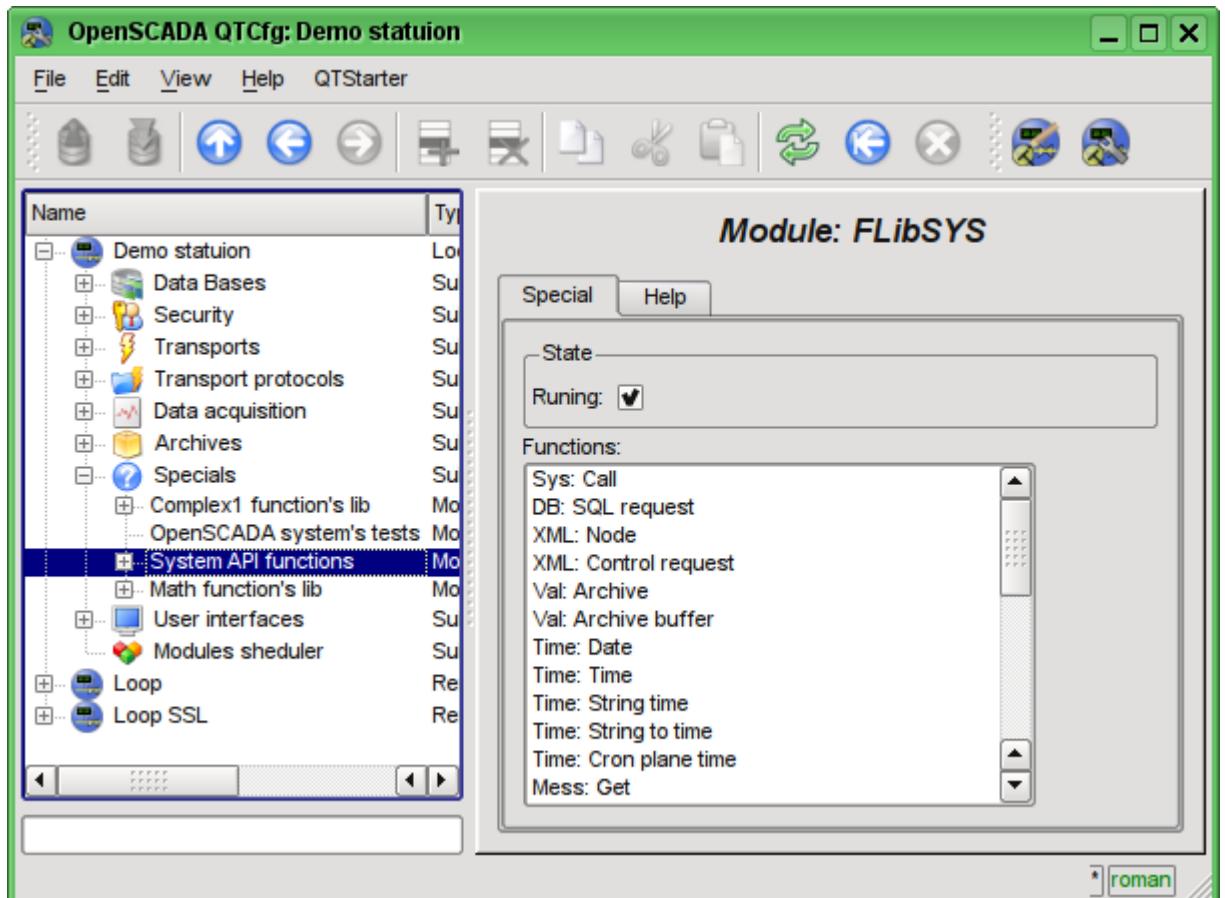
Fig. 4.8a. The "Special" tab of the module of subsystems "Specials".

4.9. Subsystem "Modules sheduler"

The subsystem is not modular. To configure the subsystem the subsystem's page "Modules sheduler" is provided, it contains tabs "Subsystem" and "Help". The "Subsystem" tab (Fig.4.9a) contains the basic settings of the subsystem. The "Help" tab contains a brief help for this page. The structure of the tab "Subsystem":

- *Path to shared libs (modules)* - information about the location of the directory with the modules of the OpenSCADA system. It is set by the parameter *<ModDir>* of the station, of the configuration file.
- *Allowed modules* - information about the list, separated by ',', of modules that are authorized for automatic connection and renewal. The value of '*' is used to resolve all the modules. It is set by the parameter *<ModAllow>* of the section of subsystem, sub_ModSched, of the station of the configuration file.
- *Denied modules* - information about the list, separated by ';' of modules that are denied for automatically connection and updating. It is set by the parameter *<ModDeny>* of the section of subsystem "sub_ModSched" of station of configuration file. List of denied modules has higher priority than allowed.
- *Check modules period (sec)* - indicates the periodicity of testing modules on the fact of their updating. Modules that are allowed for automatically connection and updating will be automatically updated.
- *Check modules now* - command to check the modules on the fact of their updating. Modules that are allowed for automatically connection and updating will be automatically updated.
- *Shared libs (modules)* - table with the list of shared libraries with the modules detected by OpenSCADA. Rows are modules, and in the columns there is an information about them:
    - *Path* - information on the full path to the shared library.
    - *Time* - information about the time the of last modification of a shared library.
    - *Modules* - information about the list of modules in a shared library.
    - *Enable* - state "Enable" of the shared library. Privileged users are provided with an opportunity to manually enable/disable the shared libraries by changing this field.
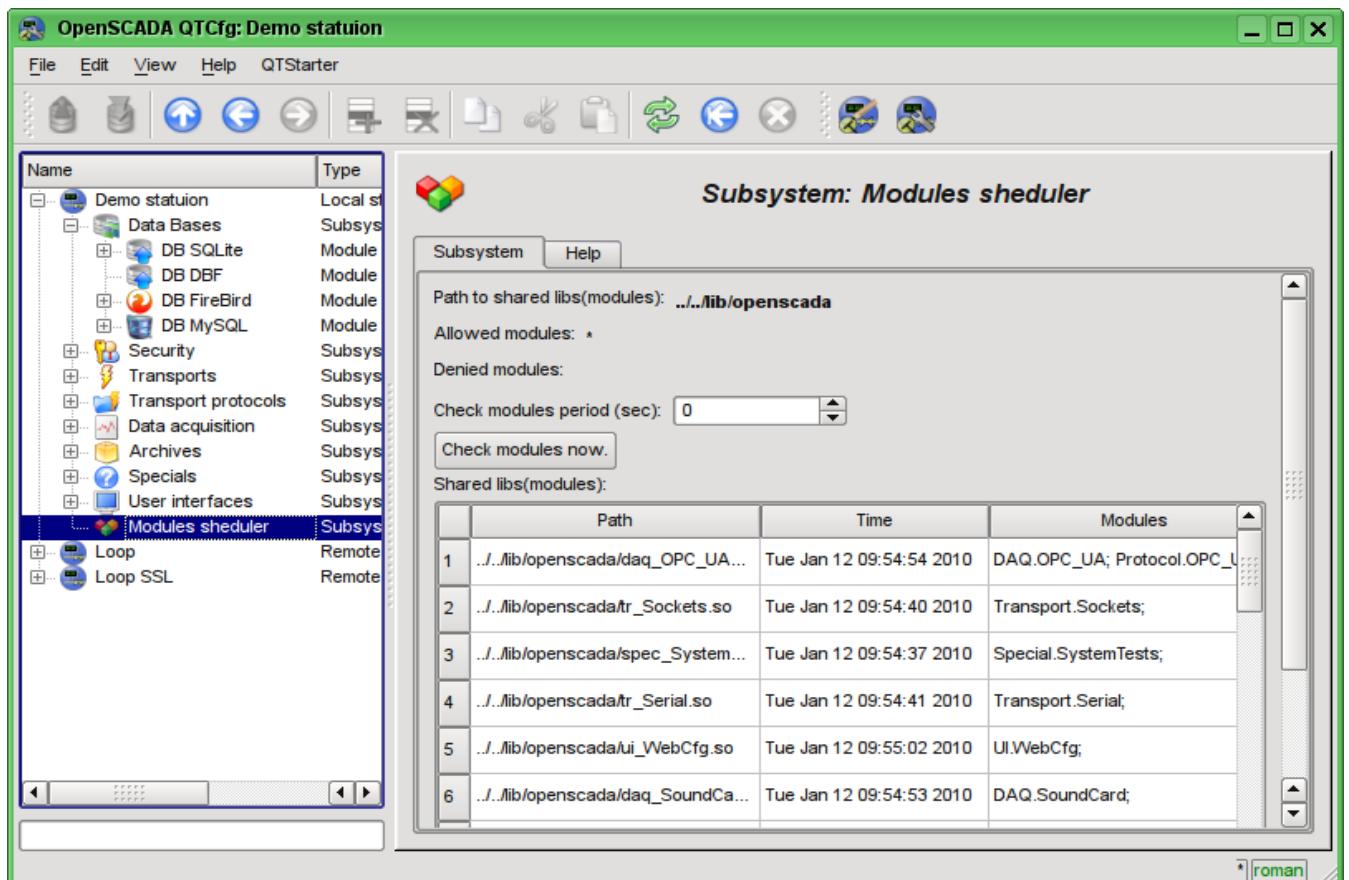
Fig. 4.9a. The main configuration tab of subsystem "Modules sheduler".

4.10. Configuration file of the OpenSCADA and parameters of command-line OpenSCADA execution.

Configuration file of the OpenSCADA system is provided to store the system and general configuration of OpenSCADA-station. Only in the configuration file and through the command-line options you can specify the part of the key system parameters of the station, so familiarity with the structure of the configuration file is necessary for professionals who make solutions based on OpenSCADA.

The configuration file of the OpenSCADA system can be called somehow, but the oscada.xml name and derived from it are accepted. The configuration file is usually indicated when you start the station by the command-line option *--Config=/home/roman/roman/work/OScadaD/etc/oscada_demo.xml*. For the convenience of the calling the startup scripts of the station are created with the correct configuration file, for example script (openscada_demo) of the demo station execution:

```
#!/bin/sh
openscada --Config=/etc/oscada_demo.xml $@
```

If the configuration file is not specified then the standard configuration file: /etc/oscada.xml is used.

Structure of the configuration file based on the extensible markup language XML. Therefore the strict adherence to the rules of XML syntax is required. An example of the configuration file of the OpenSCADA, with configuration nodes of most of the OpenASCADA components, is given below:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<OpenSCADA>
        <!-- This is the OpenSCADA configuration file. -->
        <station id="DemoStation">
                <!-- Discribe internal parameter for station. Station this only OpenSCADA programm. -->
                <prm id="StName">Demo station</prm>
                <prm id="StName_ru">Демо станция</prm>
                <prm id="StName_uk">Демо станція</prm>
                <prm id="WorkDB">SQLite.GenDB</prm>
                <prm id="Workdir">~/.openscada</prm>
                <prm id="IcoDir">./icons</prm>
                <prm id="ModDir">/usr/lib/openscada</prm>
                <prm id="LogTarget">10</prm>
                <prm id="MessLev">0</prm>
                <prm id="Lang2CodeBase">en</prm>
                <prm id="SaveAtExit">0</prm>
                <prm id="SavePeriod">0</prm>

                <node id="sub_BD">
                        <tbl id="DB">
                                <fld ID="GenDB" TYPE="SQLite" NAME="Generic DB" NAME_ru="Основная БД"
                                        NAME_uk="Основна БД" ADDR="./DEMO/DemoSt.db" CODEPAGE="UTF-8"/>
                        </tbl>
                </node>

                <node id="sub_Security">
                        <!--
                        <tbl id="Security_user">
                                <fld
                                        NAME="root"
                                        DESCR="Super user"
                                        DESCR_ru="Супер пользователь"
                                        DESCR_uk="Супер користувач"
                                        PASS="openscada"/>
                                <fld
                                        NAME="user"
                                        DESCR="System user"
                                        DESCR_ru="Системный пользователь"
                                        DESCR_uk="Системний користувач"
                                        PASS=""/>
                        </tbl>
                        <tbl id="Security_grp">
                                <fld
                                        NAME="root"
                                        DESCR="Super users groups"
                                        DESCR_ru="Группа суперпользователей"
                                        DESCR_uk="Група суперкористувачів"
                                        USERS="root;user"/>
                        </tbl>-->
                </node>

                <node id="sub_ModSched">
                        <prm id="ModAllow">*</prm>
```

```xml
            <prm id="ModDeny"></prm>
            <prm id="ChkPer">0</prm>
</node>

<node id="sub_Transport">
        <!--
        <tbl id="Transport_in">
                <fld
                        ID="WEB_1"
                        MODULE="Sockets"
                        NAME="Generic WEB interface"
                        NAME_ru="Основной WEB интерфейс"
                        NAME_uk="Основний WEB інтерфейс"
                        DESCRIPT="Generic transport for WEB interface."
                        DESCRIPT_ru="Основной транспорт для WEB интерфейса."
                        DESCRIPT_uk="Основний транспорт для WEB інтерфейсу."
                        ADDR="TCP::10002:0"
                        PROT="HTTP"
                        START="1"/>
                <fld
                        ID="WEB_2"
                        MODULE="Sockets"
                        NAME="Reserve WEB interface"
                        NAME_ru="Резервный WEB интерфейс"
                        NAME_uk="Резервний WEB інтерфейс"
                        DESCRIPT="Reserve transport for WEB interface."
                        DESCRIPT_ru="Резервный транспорт для WEB интерфейса."
                        DESCRIPT_uk="Резервний транспорт для WEB інтерфейсу."
                        ADDR="TCP::10004:0"
                        PROT="HTTP"
                        START="1"/>
        </tbl>
        <tbl id="Transport_out">
                <fld
                        ID="testModBus"
                        MODULE="Sockets"
                        NAME="Test ModBus"
                        NAME_ru="Тест ModBus"
                        NAME_uk="Тест ModBus"
                        DESCRIPT="Data exchange by protocol ModBus test."
                        DESCRIPT_ru="Тест обмена по протоколу ModBus."
                        DESCRIPT_uk="Тест обміну за протоколом ModBus."
                        ADDR="TCP:localhost:10502"
                        START="1"/>
        </tbl>-->
</node>

<node id="sub_DAQ">
        <!--
        <tbl id="tmplib">
                <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                        DESCR="" DESCR_ru="" DESCR_uk="" DB="tmplib_test2"/>
        </tbl>
        <tbl id="tmplib_test2">
                <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                        DESCR="" DESCR_ru="" DESCR_uk="" DB="test2"
                        PROGRAM="JavaLikeCalc.JavaScript&#010;cnt=5*i"/>
        </tbl>
        <tbl id="tmplib_test2_io">
                <fld TMPL_ID="test2" ID="i" NAME="I" NAME_ru="I" NAME_uk="I"
                        TYPE="4" FLAGS="160" VALUE="" POS="0"/>
                <fld TMPL_ID="test2" ID="cnt" NAME="Cnt" NAME_ru="Cnt" NAME_uk="Cnt"
                        TYPE="4" FLAGS="32" VALUE="" POS="0"/>
        </tbl>-->

        <node id="mod_LogicLev">
                <!--
                <tbl id="DAQ">
                        <fld
                                ID="test2"
                                NAME="Test 2"
                                NAME_ru="Тест 2"
                                NAME_uk="Тест 2"
                                DESCR=""
                                DESCR_ru=""
                                DESCR_uk=""
                                ENABLE="1"
                                START="1"
                                PRM_BD="test2prm"
                                PERIOD="1000"
                                PRIOR="0"/>
                </tbl>
                <tbl id="test2prm">
```

```xml
                                      <fld SHIFR="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                                          DESCR="" DESCR_ru="" DESCR_uk="" EN="1" MODE="2"
                                          PRM="test2.test2"/>
                  </tbl>-->
      </node>

      <node id="mod_System">
              <!--
              <tbl id="DAQ">
                      <fld
                              ID="DataOS"
                              NAME="Data OS"
                              NAME_ru="Данные ОС"
                              NAME_uk="Дані ОС"
                              DESCR="Data of services and subsystems OS."
                              DESCR_ru="Данные сервисов и подсистем ОС."
                              DESCR_uk="Дані сервісів та підсистем ОС."
                              ENABLE="1"
                              START="1"
                              AUTO_FILL="0"
                              PRM_BD="DataOSprm"
                              PERIOD="1000" PRIOR="0"/>
              </tbl>
              <tbl id="DataOSprm">
                      <fld SHIFR="CPU" NAME="CPU load" NAME_ru="Нагрузка CPU"
                              NAME_uk="Навантаження CPU" DESCR="" DESCR_ru="" DESCR_uk=""
                              EN="1" TYPE="CPU" SUBT="gen"/>
                      <fld SHIFR="MEM" NAME="Memory" NAME_ru="Память" NAME_uk="Пам\'ять"
                              DESCR="" DESCR_ru="" DESCR_uk="" EN="1" TYPE="MEM"/>
              </tbl> -->
      </node>

      <node id="mod_DiamondBoards">
              <!--
              <tbl id="DAQ">
                      <fld ID="Athena" NAME="Athena board" NAME_ru="Плата Athena"
                              NAME_uk="Плата Athena" DESCR="" DESCR_ru="" DESCR_uk=""
                              ENABLE="1" START="0" BOARD="25" PRM_BD_A="AthenaAnPrm"
                              PRM_BD_D="AthenaDigPrm" ADDR="640" INT="5" DIO_CFG="0"
                              ADMODE="0" ADRANGE="0" ADPOLAR="0" ADGAIN="0"
                              ADCONVRATE="1000"/>
              </tbl>
              <tbl id="AthenaAnPrm">
                      <fld SHIFR="ai0" NAME="AI 0" NAME_ru="AI 0" NAME_uk="AI 0"
                              DESCR="" DESCR_ru="" DESCR_uk=""
                              EN="0" TYPE="0" CNL="0" GAIN="0"/>
              </tbl>
              <tbl id="AthenaDigPrm">
                      <fld SHIFR="di0" NAME="DI 0" NAME_ru="DI 0" NAME_uk="DI 0"
                              DESCR="" DESCR_ru="" DESCR_uk=""
                              EN="0" TYPE="0" PORT="0" CNL="0"/>
              </tbl> -->
      </node>

      <node id="mod_BlockCalc">
              <!--
              <tbl id="DAQ">
                      <fld ID="Model" NAME="Model" NAME_ru="Модель" NAME_uk="Модель"
                              DESCR="" DESCR_ru="" DESCR_uk="" ENABLE="1" START="1"
                              PRM_BD="Model_prm" BLOCK_SH="Model_blcks"
                              PERIOD="1000" PRIOR="0" PER_DB="0" ITER="1"/>
              </tbl>
              <tbl id="Model_blcks">
                      <fld ID="Klap" NAME="Klapan" NAME_ru="Клапан" NAME_uk="Клапан"
                              DESCR="" DESCR_ru="" DESCR_uk=""
                              FUNC="DAQ.JavaLikeCalc.lib_techApp.klap" EN="1" PROC="1"/>
              </tbl>
              <tbl id="Model_blcks_io">
                      <fld BLK_ID="Klap" ID="l_kl1" TLNK="0" LNK="" VAL="50"/>
                      <fld BLK_ID="Klap" ID="l_kl2" TLNK="0" LNK="" VAL="20"/>
              </tbl>
              <tbl id="Model_prm">
                      <fld SHIFR="l_kl" NAME="Klap lev" NAME_ru="Полож. клапана"
                              NAME_uk="Полож. клапана" DESCR="" DESCR_ru="" DESCR_uk=""
                              EN="1" BLK="Klap" IO="l_kl1"/>
              </tbl> -->
      </node>

      <node id="mod_JavaLikeCalc">
              <!--
              <tbl id="DAQ">
                      <fld ID="CalcTest" NAME="Calc Test" NAME_ru="Тест вычисл."
                              NAME_uk="Тест обчисл." DESCR="" DESCR_ru="" DESCR_uk=""
```

```xml
                            ENABLE="1" START="1" PRM_BD="Cal FUNC="TemplFunc.d_alarm"
                            PERIOD="1000" PRIOR="0" PER_DB="0" ITER="1"/>
                </tbl>
                <tbl id="CalcTest_val">
                        <fld ID="in" VAL="0"/>
                        <fld ID="alrm" VAL=""/>
                        <fld ID="alrm_md" VAL="1"/>
                        <fld ID="alrm_mess" VAL="Error present."/>
                </tbl>
                <tbl id="CalcTest_prm">
                        <fld SHIFR="alrm" NAME="Alarm" NAME_ru="Авария" NAME_uk="Аварія"
                            DESCR="" DESCR_ru="" DESCR_uk="" EN="1" FLD="alrm"/>
                </tbl>
                <tbl id="lib">
                        <fld ID="TemplFunc" NAME="" NAME_ru="" NAME_uk="" DESCR="" ESCR_ru=""
                            DESCR_uk="" DB="lib_TemplFunc"/>
                </tbl>
                <tbl id="lib_TemplFunc">
                        <fld ID="d_alarm" NAME="Digit alarm" NAME_ru="Авария по дискр."
                            NAME_uk="Аварія за дискр" DESCR=""
                            FORMULA="alrm=(in==alrm_md)?&quot;1:&quot;
                                +alrm_mess:&quot;0&quot;;"/>
                </tbl>
                <tbl id="lib_TemplFunc_io">
                        <fld F_ID="d_alarm" ID="in" NAME="Input" NAME_ru="Вход" NAME_uk="Вхід"
                            TYPE="3" MODE="0" DEF="" HIDE="0" POS="0"/>
                        <fld F_ID="d_alarm" ID="alrm" NAME="Alarm" NAME_ru="Авария"
                            NAME_uk="Аварія" TYPE="0" MODE="1" DEF="" HIDE="0" POS="1"/>
                        <fld F_ID="d_alarm" ID="alrm_md" NAME="Alarm mode"
                            NAME_ru="Режим аварии" NAME_uk="Режим аварії" TYPE="3"
                            MODE="0" DEF="" HIDE="0" POS="2"/>
                        <fld F_ID="d_alarm" ID="alrm_mess" NAME="Alarm message"
                            NAME_ru="Сообщ. аварии" NAME_uk="Повід. аварії" TYPE="0"
                            MODE="0" DEF="" HIDE="0" POS="3"/>
                </tbl>-->
        </node>

        <node id="mod_Siemens">
                <!--
                <tbl id="DAQ">
                        <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                            DESCR="" DESCR_ru="" DESCR_uk="" ENABLE="1" START="1"
                            PRM_BD="test2prm" PERIOD="1000" PRIOR="0" CIF_DEV="0" ADDR="5"
                            ASINC_WR="0"/>
                </tbl>
                <tbl id="test2prm">
                        <fld SHIFR="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                            DESCR="" DESCR_ru="" DESCR_uk="" EN="1" TMPL="S7.ai_man"/>
                </tbl>-->
        </node>

        <node id="mod_SNMP">
                <!--
                <tbl id="DAQ">
                        <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                            DESCR="" DESCR_ru="" DESCR_uk="" ENABLE="1" START="1"
                            PRM_BD="test2prm" PERIOD="1000" PRIOR="0" ADDR="localhost"
                            COMM="public" PATTR_LIM="20"/>
                </tbl>
                <tbl id="test2prm">
                        <fld SHIFR="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                            DESCR="" DESCR_ru="" DESCR_uk="" EN="1" OID_LS="system"/>
                </tbl>-->
        </node>

        <node id="mod_ModBus">
                <!--
                <tbl id="DAQ">
                        <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                            DESCR="" DESCR_ru="" DESCR_uk="" ENABLE="1" START="1"
                            PRM_BD="test2prm" PERIOD="1000" PRIOR="0" TRANSP="Sockets"
                            ADDR="exlar.diya.org" NODE="1"/>
                </tbl>
                <tbl id="test2prm">
                        <fld SHIFR="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                            DESCR="" DESCR_ru="" DESCR_uk=""
                            EN="1" ATTR_LS="321:0:tst:Test"/>
                </tbl>-->
        </node>

        <node id="mod_Transporter">
                <!--
                <tbl id="DAQ">
```

```xml
                        <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                             DESCR="" DESCR_ru="" DESCR_uk="" ENABLE="1" START="1"
                             PRM_BD="test2prm" PERIOD="1000" PRIOR="0" SYNCPER="60"
                             STATIONS="loop" CNTRPRM="System.AutoDA"/>
                </tbl>-->
        </node>
</node>

<node id="sub_Archive">
        <prm id="MessBufSize">1000</prm>
        <prm id="MessPeriod">5</prm>
        <prm id="ValPeriod">1000</prm>
        <prm id="ValPriority">10</prm>
        <!--
        <tbl id="Archive_mess_proc">
                <fld
                        ID="StatErrors"
                        MODUL="FSArch"
                        NAME="Errors"
                        NAME_ru="Ошибки"
                        NAME_uk="Помилки"
                        DESCR="Local errors\' archive"
                        DESCR_ru="Архив локальных ошибок"
                        DESCR_uk="Архів локальних помилок"
                        START="1"
                        CATEG="/DemoStation*"
                        LEVEL="4"
                        ADDR="ARCHIVES/MESS/stError/"
                        FSArchMSize="300"
                        FSArchNFiles="10"
                        FSArchTmSize="30"
                        FSArchXML="1"
                        FSArchPackTm="10"
                        FSArchTm="60"/>
                <fld
                        ID="NetRequsts"
                        MODUL="FSArch"
                        NAME="Net requests"
                        NAME_ru="Сетевые запросы"
                        NAME_uk="Мережеві запити"
                        DESCR="Requests to server through transport Sockets."
                        DESCR_ru="Запросы к серверу через транспорт Sockets."
                        DESCR_uk="Запити до сервера через транспорт Sockets."
                        START="1"
                        CATEG="/DemoStation/Transport/Sockets*"
                        LEVEL="1"
                        ADDR="ARCHIVES/MESS/Net/"
                        FSArchMSize="300"
                        FSArchNFiles="10"
                        FSArchTmSize="30"
                        FSArchXML="1"
                        FSArchPackTm="10"
                        FSArchTm="60"/>
        </tbl>
        <tbl id="Archive_val_proc">
                <fld
                        ID="1h"
                        MODUL="FSArch"
                        NAME="1hour"
                        NAME_ru="1час"
                        NAME_uk="1год"
                        DESCR="Averaging for hour"
                        DESCR_ru="Усреднение за час"
                        DESCR_uk="Усереднення за годину"
                        START="1"
                        ADDR="ARCHIVES/VAL/1h/"
                        V_PER="360"
                        A_PER="60"
                        FSArchTmSize="8640"
                        FSArchNFiles="10"
                        FSArchRound="0.1"
                        FSArchPackTm="10"
                        FSArchTm="60"/>
        </tbl>
        <tbl id="Archive_val">
                <fld
                        ID="test1"
                        NAME="Test 1"
                        NAME_ru="Тест 1"
                        NAME_uk="Тест 1"
                        DESCR="Test 1"
                        DESCR_ru="Тест 1"
                        DESCR_uk="Тест 1"
```

```
                             START="1"
                             VTYPE="1"
                             BPER="1"
                             BSIZE="200"
                             BHGRD="1"
                             BHRES="0"
                             SrcMode="0"
                             Source=""
                             ArchS=""/>
              </tbl>-->
      </node>

      <node id="sub_Protocol">
      </node>

      <node id="sub_UI">
              <node id="mod_QTStarter">
                      <prm id="StartMod">QTCfg</prm>
              </node>
              <node id="mod_WebCfg">
                      <prm id="SessTimeLife">20</prm>
              </node>
              <node id="mod_VCAEngine">
                      <!--
                      <tbl id="LIB">
                              <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                                      DESCR="" DESCR_ru="" DESCR_uk="" DB_TBL="wlib_test2" ICO=""
                                      USER="root" GRP="UI" PERMIT="436"/>
                      </tbl>
                      <tbl id="wlib_test2">
                              <fld ID="test2" ICO="" PARENT="/wlb_originals/wdg_Box" PROC=""
                                      PROC_ru="" PROC_uk="" PROC_PER="-1" USER="root" GRP="UI"
                                      PERMIT="436"/>
                      </tbl> <tbl id="wlib_test2_io">
                              <fld IDW="test2" ID="name" IO_VAL="Test 2" IO_VAL_ru="Тест 2"
                                      IO_VAL_uk="Тест 2" SELF_FLG="" CFG_TMPL="" CFG_TMPL_ru=""
                                      CFG_TMPL_uk="" CFG_VAL=""/>
                              <fld IDW="test2" ID="dscr" IO_VAL="Test module 2"
                                      IO_VAL_ru="Тест модуля 2" IO_VAL_uk="Тест модуля 2"
                                      SELF_FLG="" CFG_TMPL="" CFG_TMPL_ru="" CFG_TMPL_uk=""
                                      CFG_VAL=""/>
                      </tbl>
                      <tbl id="PRJ">
                              <fld ID="test2" NAME="Test 2" NAME_ru="Тест 2" NAME_uk="Тест 2"
                                      DESCR="" DESCR_ru="" DESCR_uk="" DB_TBL="prj_test2" ICO=""
                                      USER="root" GRP="UI" PER </tbl> <tbl id="prj_test2">
                              <fld OWNER="/test2" ID="pg1" ICO="" PARENT="/wlb_originals/wdg_Box"
                                      PROC="" PROC_ru="" PROC_uk="" PROC_PER="-1" USER="root"
                                      GRP="UI" PERMIT="436" FLGS="1"/>
                              <fld OWNER="/test2/pg1" ID="pg2" ICO=""
                                      PARENT="/wlb_originals/wdg_Box" PROC="" PROC_ru="" PROC_uk=""
                                      PROC_PER="-1" USER="root" GRP="UI" PERMIT="436" FLGS="0"/>
                      </tbl>
                      <tbl id="prj_test2_incl">
                              <fld IDW="/prj_test2/pg_pg1" ID="wdg1"
                                      PARENT="/wlb_originals/wdg_Box"/>
                      </tbl>-->
              </node>
      </node>

      <node id="sub_Special">
              <node id="mod_SystemTests">
                      <prm id="PARAM" on="0" per="5" name="LogicLev.experiment.F3"/>
                      <prm id="XML" on="0" per="10" file="/etc/oscada.xml"/> <prm id="MESS" on="0"
                              per="10" categ="" arhtor="DBArch.test3"/>
                      <prm id="SOAttDet" on="0" per="20" name="../../lib/openscada/daq_LogicLev.so"
                              full="1"/>
                      <prm id="Val" on="0" per="1" name="LogicLev.experiment.F3.var" arch_len="5"
                              arch_per="1000000"/>
                      <prm id="Val" on="0" per="1" name="System.AutoDA.CPULoad.load" arch_len="10"
                              arch_per="1000000"/>
                      <prm id="BD" on="0" per="10" type="MySQL"
                              bd="server.diya.org;roman;123456;oscadaTest"
                              table="test" size="1000"/>
                      <prm id="BD" on="0" per="10" type="DBF" bd="./DATA/DBF" table="test.dbf"
                              size="1000"/>
                      <prm id="BD" on="0" per="10" type="SQLite" bd="./DATA/test.db" table="test"
                              size="1000"/>
                      <prm id="BD" on="0" per="10" type="FireBird"
                              bd="server.diya.org:/var/tmp/test.fdb;roman;123456"
                              table="test" size="1000"/>
                      <prm id="TrOut" on="0" per="1" addr="TCP:127.0.0.1:10001" type="Sockets"
                              req="time"/>
```

```
        <prm id="TrOut" on="0" per="1" addr="UDP:127.0.0.1:10001" type="Sockets"
            req="time"/>
        <prm id="TrOut" on="0" per="1" addr="UNIX:./oscada" type="Sockets"
            req="time"/>
        <prm id="TrOut" on="0" per="1" addr="UDP:127.0.0.1:daytime" type="Sockets"
            req="time"/>
        <prm id="Func" on="0" per="10"/> <prm id="SysContrLang" on="0" per="10"
            path="/Archive/FSArch/mess_StatErrors/%2fprm%2fst"/>
        <prm id="ValBuf" on="0" per="5"/> <prm id="Archive" on="0" per="30"
            arch="test1" period="1000000"/>
        <prm id="Base64Code" on="0" per="10"/>
      </node>
    </node>
  </station>
</OpenSCADA>
```

Lets examine in details the structure of the configuration file. A configuration file can contain a configuration of several stations in the sections *<station id="DemoStation"/>*. To attribute set the identifier of the station. Using one or another section of the station at startup is specified by the command-line option --*Station=DemoStation*. Section of the station directly contains parameters of the station and subsystems' sections. Configuration options of the section are written in the form *<prm id="StName">Demo station</prm>*. Where in the attribute *<id>* the ID of the attribute is specified, and in the tag's body the value of parameter "Demo station" is specified. The list of available options and their description for the station and all other sections can be obtained from the console by calling OpenSCADA with parameter --help or in the "Help" tabs of the pages of the components of the configuration files of OpenSCADA (Fig.4.10a).
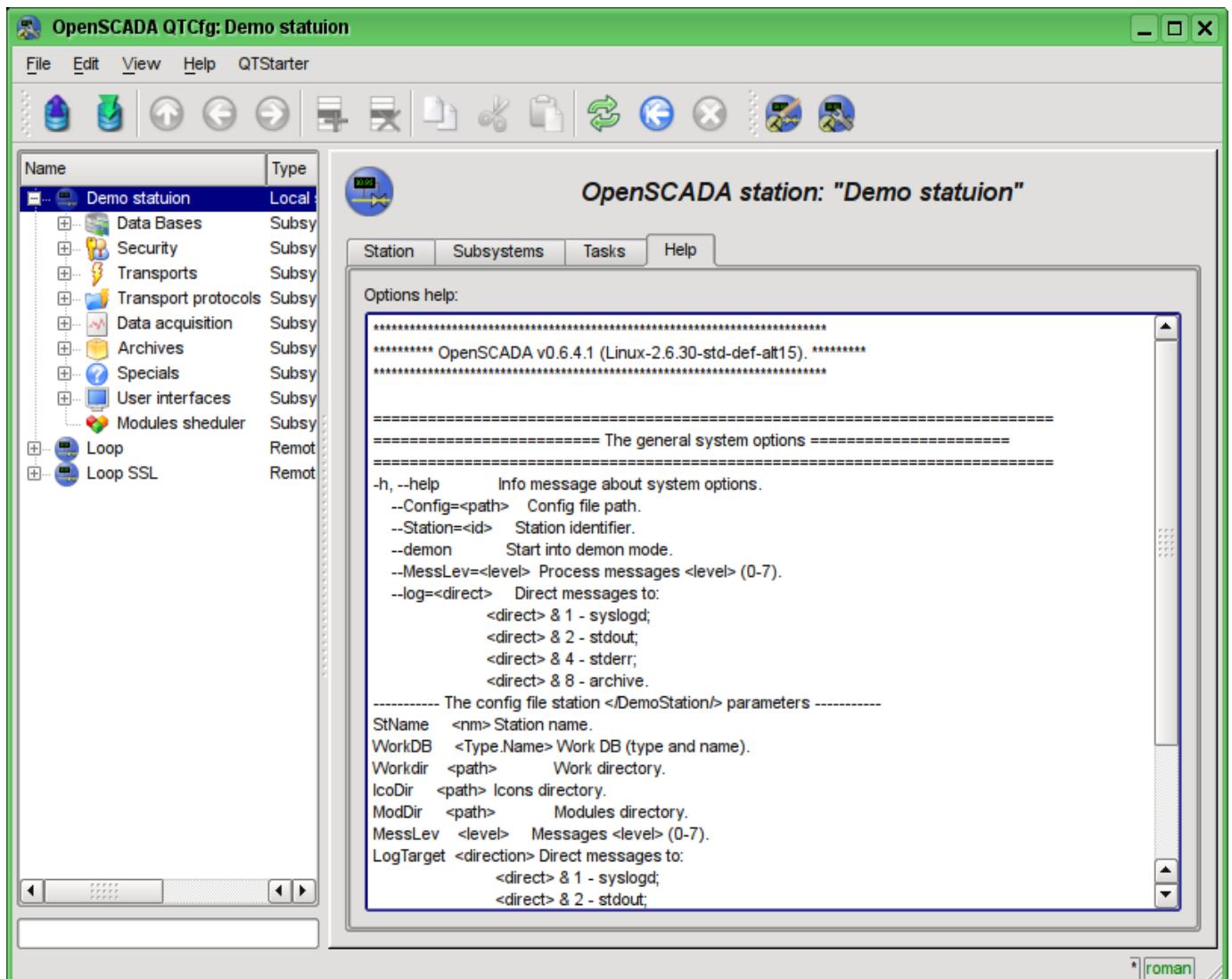

Fig. 4.10a. The "Help" tab of the OpenSCADA component.

The result of the command: **# ./openscada_demo --help**

```
****************************************************************************
********** OpenSCADA v0.6.4.1 (Linux-2.6.30-std-def-alt15). *********
****************************************************************************
============================================================================
======================= The general system options =====================
============================================================================
-h, --help          Info message about system options.
    --Config=<path>  Config file path.
    --Station=<id>   Station identifier.
    --demon          Start into demon mode.
    --MessLev=<level> Process messages <level> (0-7).
    --log=<direct>   Direct messages to:
                     <direct> & 1 - syslogd;
                     <direct> & 2 - stdout;
                     <direct> & 4 - stderr;
                     <direct> & 8 - archive.
----------- The config file station </EmptySt/> parameters -----------
StName    <nm>       Station name.
WorkDB    <Type.Name> Work DB (type and name).
Workdir   <path>     Work directory.
IcoDir    <path>     Icons directory.
ModDir    <path>     Modules directory.
MessLev   <level>    Messages <level> (0-7).
LogTarget <direction> Direct messages to:
                     <direct> & 1 - syslogd;
                     <direct> & 2 - stdout;
                     <direct> & 4 - stderr;
                     <direct> & 8 - archive.
Lang2CodeBase <lang>  Base language for variable texts translation, two symbols code.
SaveAtExit <true>    Save system at exit.
SavePeriod <sec>     Save system period.

=================== Subsystem "Module sheduler" options =================
    --ModPath=<path>  Modules <path> (/var/os/modules/).
------------ Parameters of section </DemoStation/sub_ModSched/> in config file -----------
ModPath   <path>     Path to shared libraries(modules).
ModAllow  <list>     List of shared libraries allowed for automatic loading, attaching and starting
                     (bd_DBF.so;daq_JavaLikeCalc.so). Use '*' value for allow all modules.
ModDeny   <list>     List of shared libraries deny for automatic loading, attaching and starting
                     (bd_DBF.so;daq_JavaLikeCalc.so).
ChkPer    <sec>      Period of checking at new shared libraries(modules).
========================= Subsystem "DB" options =========================

==================== Subsystem "Security" options =====================

==================== Subsystem "Transports" options ===================

============ Subsystem "Transport protocols" options ==================

================== The module <Protocol:HTTP> options ====================
---------- Parameters of the module section </DemoStation/sub_Protocol/mod_HTTP/> in config file ----------
AuthTime  <min>      Life time of the authentication, minutes (default 10).

================== Subsystem "Data acquisition" options ================
------------ Parameters of section </DemoStation/sub_DAQ/> in config file -----------
RdStLevel   <lev>    The curent station redundant level.
RdTaskPer   <s>      The redundant task call period.
RdRestConnTm <s>     Restore connection timeout to dead reserve stations.
RdRestDtTm  <hour>   Restore data archive depth from a reserve station after deadline.
RdStList    <list>   Redundant stations list, separated symbol ';' (st1;st2).

======================= Subsystem "Archives" options ==================
------------ Parameters of section </DemoStation/sub_Archive/> in config file -----------
MessBufSize <items>  Messages buffer size.
MessPeriod  <sec>    Message arhiving period.
ValPeriod   <msec>   Values arhiving period.
ValPriority <level>  Values task priority level.
MaxReqMess  <items>  Maximum request messages.
MaxReqVals  <items>  Maximum request values.

====================== Subsystem "Special" options ====================

==================== The module <Special:SystemTests> options ====================
------ Parameters of the module section </DemoStation/sub_Special/mod_SystemTests/> in config file ------
All tests main options:
  id                 test's id;
  on                 on test's flag;
  per                repeat period (sek).
*** Test's options ***
PARAM   Parameter test:
  name               paremeter full name (OperationSystem.AutoDA.CPULoad).
```

```
XML      XML parsing test:
  file                 file for XML parsing.
MESS     Messages archive test:
  arhtor archivator name (StatErrors.BaseArh);
  categ messages category pattern.
SOAttDet Attach/Detach module test:
  name                 name modul (path to module);
  full                 full attach(to start).
Val      Parameter atributes value test:
  name                 parameter attribute (OperationSystem.AutoDA.CPULoad.load);
  arch_len             archive value getting depth;
  arch_per             archive value getting period.
BD       Full database test:
  type                 type BD;
  bd                   name BD;
  table                table;
  size                 fields number.
TrOut    Output transport test:
  addr                 input transport address;
  type                 transport type;
  req                  request to a input transport.
Func Function API test;
SysContrLang System control language test:
  path                 path to language element (/Archive/BaseArh/mess_StatErrors/%2fprm%2fst).
ValBuf Value buffer tests;
Archive Value archive allocation tests:
  arch                 value archive;
  period               values period (us).
Base64Code Mime Base64 algorithm tests.

==================== Subsystem "User interfaces" options ===================
=================== The module <UI:Vision> options =======================
---------- Parameters of the module section </DemoStation/sub_UI/mod_Vision/> in config file ----------
StartUser   <user>    No password requested start user.
RunPrjs     <list>    Run projects list on the module start.
RunTimeUpdt <mode>    RunTime update mode (0 - all widgets periodic adaptive update, 1 - update only
                      changed widgets).
VCAstation  <id>      VCA station id ('.' - local).

====================== The module <UI:VCAEngine> options =======================
   --VCADBClearForce  Force clear VCA DB from data of API 1.

====================== The module <UI:QTCfg> options =======================
---------- Parameters of the module section </DemoStation/sub_UI/mod_QTCfg/> in config file ----------
StartPath   <path>    Configurator start path.
StartUser   <user>    No password requested start user.

====================== The module <UI:QTStarter> options =======================
---------- Parameters of the module section </DemoStation/sub_UI/mod_QTStarter/> in config file ----------
StartMod    <moduls>  Start modules list (sep - ';').

====================== The module <UI:WebVision> options =======================
---------- Parameters of the module section </DemoStation/sub_UI/mod_WebVision/> in config file ----------
SessTimeLife <time>   Time of the session life, minutes (default 10).
```

Sections of subsystem (*<node id="sub_DAQ" />*) contains parameters of subsystem, sections of modules and sections of tables of reflections of the data of databases in the configuration file. Sections of modules (*<node id="mod_DiamondBoards" />*) contain the individual parameters of modules and sections of tables of reflection of the data of databases in the configuration file.

Sections of the tables of reflection of the data of databases are provided for placement in the configuration file records of DB tables for the OpenSCADA components. Lets examine the table of incoming transports "Transport_in" of subsystem transports (*<node id="sub_Transport">*) from the example of configuration file above. The table contains two records with fields: ID, MODULE, NAME, DESCRIPT, ADDR, PROT, START. After booting with this section and in general without the DB in the subsystem "Transports" of the "Sockets" module you'll see two input transports. Formats of the table's structures of the main components are included in the demo configuration files. For the details of the database's structure you should read the relevant documentation of modules.

5. System-wide API of user programming.

User programming API is the tree of OpenSCADA objects, every object of which can provide own list of properties and functions. Properties and functions of objects can be used by the user in procedures on the languages of user programming of OpenSCADA. The entry point for access to the objects of system OpenSCADA from user programming language JavaLikeCalc is the reserved word "SYS" of the root OpenSCADA object. For example, to access the function of outgoing transport you should write: **SYS.Transport.Serial.out_ModBus.messIO(mess);**.

API of the objects provided by the modules is described in the own documentation of the module.

5.1. System-wide user objects.

Abstract object is an associative container of properties and functions. Properties can contain the data of four basic types and other objects. Access to the properties of an object is usually made by recording the names of properties through a point to the object *<obj.prop>*, as well as by entering the property name in brackets *<obj["prop"]>*. It is obvious that the first mechanism is static, while the second lets you to specify the name of the property through a variable. The basic definition of the object does not contain functions. Copying of an object actually makes reference to the original object. When you delete an object the reduce of the reference counter is made, and when the reference counter is equal to the zero object is removed physically.

Different components can redefine the basic object with special properties and functions. The standard extension of the object is an array "Array".

Array object

Peculiarity of the array is that it works with the properties like with the indexes, and complete their naming if senseless, and hence the mechanism of addressing is available only by the conclusion of the index in square brackets *<arr[1]>*. Array stores the properties in its own container of one-dimensional array.

Array provides the special property "length" to get the array size *<var = arr.length;>*. Also array provides the following functions:
- *string join(string sep = ","), string toString(string sep = ","), string valueOf(string sep = ",")* - Returns the string with the array elements separated by *<sep>* or the character ','.
- *Array concat(Array arr);* - Adds to the initial array the elements of the *<arr>* array. Returns the initial array with changes.
- *int push(ElTp var, ...);* - Places the element(s) *<var>* to the end of the array, as to the stack. Returns the new array size.
- *ElTp pop( );* - Deleting of the last element of the array and return of its value, as from the stack.
- *Array reverse( );* - Changing the order of the elements of the array. Returns the initial array with changes.
- *ElTp shift( );* - The shift of the array to the top. The first element is removed and its value is returned.
- *int unshift(ElTp var, ...);* - Shift element(s) *<var>* to the array. The first element to the 0, second to the 1 and so on.
- *Array slice(int beg, int end);* - Returns an array fragment from *<beg>* to *<end>*. If the value of beginning or end is negative, then the count is made from the end of the array. If the end is not specified, then the end is the end of the array.
- *Array splice(int beg, int remN, ElTp val1, ElTp val2, ...);* - Inserts, deletes or replaces the elements of the array. Returns the initial array with the changes. Firstly it is made the removing of elements from the position *<beg>* and in the quantity of *<remN>*, and then the values *<val1>* are inserted and so on, beginning from the position *<beg>*.
- *Array sort( );* - Sort array elements in lexicographical order.

XMLNodeObj object

Functions:
- *string name( )* - The name of the node, XML-tag.
- *string text( )* - The text of the node, contents of the XML-tag.
- *string attr(string id)* - The value of the node's attribute *<id>*.
- *XMLNodeObj setName(string vl)* - Setting of the node's name to *<vl>*. Returns the current node.
- *XMLNodeObj setText(string vl)* - Setting of the node's text to *<vl>*. Returns the current node.
- *XMLNodeObj setAttr(string id, string vl)* - Setting the attribute *<id>* to the value *<vl>*. Returns the current node.
- *int childSize( )* - Quantity of the embedded nodes.
- *XMLNodeObj childAdd(ElTp no = XMLNodeObj)* - Addition of the object *<no>* as the embedded one. *<no>* may be the direct object-result of the function *xmlNode()*, and the string with the name of the new tag. Returns the embedded node.
- *XMLNodeObj childIns(int id, ElTp no = XMLNodeObj)* - Insert of the object *<no>* as the embedded one to the position *<id>*. *<no>* may be the direct object-result of the function *xmlNode()*, and the string with the name of the new tag. Returns the embedded node.
- *XMLNodeObj childDel(int id)* - Deleting the embedded node from the position *<id>*. Returns the current node.
- *XMLNodeObj childGet(int id)* - Getting the embedded node in the position *<id>*.
- *string load(string str, bool file = false)* - Loading the XML from the string *<str>* or from the file with the path in *<str>* if the *<file>* "true".
- *string save(int opt, string path)* - Saving the XML tree to the string or to the file *<path>* with the formatting parameter *<opt>*. Returns the XML text or the error code. The following formatting options *<opt>* are provided:
  - 0x01 - interrupt the string before the opening tag;
  - 0x02 - interrupt the string after the opening tag;
  - 0x04 - interrupt the string after a closing tag;
  - 0x08 - interrupt the string after the text;
  - 0x10 - interrupt the string after the instruction;
  - 0x1E - interrupt the string after all.

5.2. System (SYS)

Object functions:
- *string system(string cmd, bool noPipe = false);* - calls the console commands *<cmd>* of OS returning the result by the channel. If *<noPipe>* is set the return code is returned the the execution of the programs in the background ("sleep 5 &") is possible. The function offers great opportunities to the OpenSCADA user by calling any system software, utilities and scripts, as well as by way of access to the huge volume of system data. For example the command "ls-l" returns the detailed contents of the working directory.
- *int message(string cat, int level, string mess);* - formation of the system message *<mess>* with the category *<cat>*, level *<level>*. The negative value of the level forms the alarms (Alarm).
- *int messDebug(string cat, string mess); int messInfo(string cat, string mess); int messNote(string cat, string mess); int messWarning(string cat, string mess); int messErr(string cat, string mess); int messCrit(string cat, string mess); int messAlert(string cat, string mess); int messEmerg(string cat, string mess);* -formation of the system message *<mess>* with the category *<cat>* and the appropriate level.
- *XMLNodeObj XMLNode(string name = "");* - creation of the XML node object with the name *<name>*.
- *string cntrReq(XMLNodeObj req, string stat);* - request of the control interface to the system via XML. The usual request is written as <get path="/OPath/%2felem"/>. If the station is indicated to the request to the external station is made.
- *int time(int usec);* - returns the absolute time in seconds from the epoch of 1/1/1970 and in microseconds, if *<usec>* is specified.
- *int localtime(int fullsec, int sec, int min, int hour, int mday, int month, int year, int wday, int*

yday, int isdst); - returns the full date in seconds (sec), minutes (min), hours (hour), days of the month (mday), month (month), year (year), days in the week (wday), days in the year (yday) and sign of summer time (isdst), based on the absolute time in seconds *<fullsec>* from the epoch 1.1.1970.
• *string strftime(int sec, string form = "%Y-%m-%d %H:%M:%S");* - Converts an absolute time *<sec>* to the string of the desired format *<form>*. Record of the format corresponds to the POSIX-function strftime.
• *int strptime(int str, string form = "%Y-%m-%d %H:%M:%S");* - Returns the time in seconds from the epoch of 1/1/1970, based on the string record of time *<str>*, in accordance with the specified template *<form>*. For example the template "%Y-%m-%d %H:%M:%S" corresponds with the time "2006-08-08 11:21:55". Description of the template's format can be obtained from the documentation on POSIX-function "strptime".
• *int cron(string cronreq, int base = 0);* - returns the time, planned in the format of the standard Cron *<cronreq>*, beginning from basic time *<base>* or from the current, if the basic is not specified.
• *string strFromCharCode(int char1, int char2, int char3, ...);* - String creation from symbol's codes char1, char2 ... charN.

5.3. "DB" subsystem (SYS.BD)

DB object functions (SYS.BD["TypeDB"]["DB"]):
• *Array SQLReq(string req);* - Formation of the SQL-request to the DB.

Example:
```
ListenModes=SYS.BD.SQLite.FactoryMode.SQLReq("SELECT mode from "+tablename+";");
```

5.4. Subsystem "DAQ" (SYS.DAQ)

Functions of object of atribute of controller's parameter (SYS.DAQ["Modul"]["Controller"] ["Parameter"]["Attribute"]):
• *ElTp get(int tm = 0, int utm = 0, bool sys = false);* - get attribute value at time *<tm:utm>* and system access flag *<sys>*.
• *bool set(ElTp val, int tm = 0, int utm = 0, bool sys = false);* - write value *<val>* to attribute with time label *<tm:utm>* and system access flag *<sys>*.

5.5. "Archives" subsystem (SYS.Archive)

Functions of the subsystem's object:
• *Area messGet(int btm, int etm, string cat = , int lev = 0, string arch =);* - request of the system messages for the time from *<btm>* to *<etm>* for the category *<cat>*, level *<lev>* and archiver *<arch>*.

5.6. "Transports" subsystem (SYS.Transport)

Functions of the outgoing transport object (SYS.Transport["Modul"]["OutTransp"]):
• *string messIO(string mess, real timeOut = 1000);* - sending the message *<mess>* through the transport with the waiting timeout *<timeOut>*.
• *int messIO(XMLNodeObj req, string prt);* - sending the request *<req>* to the protocol *<prt>* for the implementation of a connection session through the transport by means of protocol.