The module <FLibSYS> of the subsystem “Specials”

Module:	FLibSYS
Name:	Library of system API functions.
Type:	Specials
Source:	spec_FLibSYS.so
Version:	1.0.0
Author:	Roman Savochenko
Translated:	Maxim Lysenko
Description:	Provides the library of system API of user programming area.
License:	GPL

Contents table

Introduction

Special module FLibSYS provides static library of functions for working with the OpenSCADA system at the level of its system API. These functions can be used in an user programming area of OpenSCADA system for the organization of not ordinary interaction algorithms.

To address the functions of the library you can use static call address "**Special.FLibSYS.{Func}()**" or dynamic "**SYS.Special.FLibSYS["{Func}"].call()**", "**SYS.Special.FLibSYS.{Func}()**". Where *{Func}* — function identifier in the library.

Below is the description of each function of the library. For each function it was evaluated the execution time. Measurements were made on the system with the following parameters: Athlon 64 3000 + (ALTLinux 4.0 (32bit)) by measuring the total execution time of the function when you call it 1000 times. Sampling was carried out of the five calculations, rounded to integer. Time is in angle brackets and is measured in microseconds.

1. System-wide functions

1.1. Calling the console commands and operating system utilities (sysCall)

Description: Call the console commands of the OS. The function offers great opportunities to the OpenSCADA user by calling any system software, utilities and scripts, as well as getting the access to the huge volume of system data by means of them. For example the command “ls-l” returns the detailed contents of the working directory.

Parameters:

ID	Name	Type	Mode	By defaults
rez	Result	String	Return	
com	Command	String	In	

Example:

```
using Special.FLibSYS;
test=sysCall("ls -l");
messPut("Example",0,"Example: "+test);
```

1.2. SQL query (dbReqSQL)

Description: Formation of the SQL-query to DB.

Parameters:

ID	Name	Type	Mode	By defaults
rez	Result	Object(Array)	Return	
addr	DB address	String	In	
req	SQL-query	String	In	

1.3. XML node (xmlNode)

Description: Creation of the XML node object.

Parameters:

ID	Name	Type	Mode	By defaults
rez	Result	Object(XMLNodeObj)	Return	
name	Name	String	In	

Example:

```
using Special.FLibSYS;
//Creating the "get"  object of the XML node.
Req = xmlNode("get");
//Creating the "get" object of the XML node with creating attributes.
//sub_DAQ/mod_ModBus/cntr_1/prm_1 - The path in accord of project structure.
Req = xmlNode("get").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/%2fprm
%2fst%2fen");
```

1.4. Request of the management interface (xmlCntrReq)

Description: Request of the management interface to the system via XML. The usual request is written in the form *<get path="/OPat/%2felem"/>*. When we indicate the station the request to the external station is made.

Parameters:

ID	Name	Type	Mode	By defaults
rez	Result	String	Return	
req	Request	Object(XMLNodeObj)	Out	
stat	Station	String	In	

Example:

```
using Special.FLibSYS;
//Geting status "Off/On" of the parameter "1" of the controller "1"
//of the module "ModBus".
//sub_DAQ/mod_ModBus/cntr_1/prm_1 -  The path in accord of project structure.
req = xmlNode("get").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/%2fprm
%2fst%2fen");
rez = xmlCntrReq(req);
messPut("test",0,"Example: "+req.text());

//Setting status "On" of the parameter "1" of the controller "1"
//of the module "ModBus".
req = xmlNode("set").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/%2fprm
%2fst%2fen").setText(1);
rez = xmlCntrReq(req);

//Setting status "Off of the parameter "1" of the controller "1"
//of the module "ModBus".
req = xmlNode("set").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/%2fprm
%2fst%2fen").setText(0);
rez = xmlCntrReq(req);
```

1.5. Values archive (vArh)

Description: Getting the object of the values archive (VArchObj) by connecting to the archive using its address.

Parameters:

ID	Name	Type	Mode	By defaults
rez	Result	Object(VArchObj)	Return	
name	Name and address to the attribute of the parameter with the archive or directly to the archive of values.	String	In	

VArchObj object

Functions:

- *begin(usec, archivator)* — Getting the start time of the archive through the return of seconds and microseconds <*usec*> for the archivator <*archivator*>.
- *end(usec, archivator)* — Getting the end time of the archive through the return of seconds and microseconds <*usec*> for the archivator <*archivator*>.
- *period(usec, archivator)* — Getting the periodicity of the archive through the return of seconds and microseconds <*usec*> for the archivator <*archivator*>.
- *get(sec, usec, upOrd, archivator)* – Getting the value from the archive at the time <*sec*>:<*usec*> linked to the top <*upOrd*> for the archivator <*archivator*>. Real time of the value obtained is set in <*sec*>:<*usec*>.
- *set(val, sec, usec)* — Writing of the value <*val*> in the archive buffer for the time <*sec*>:<*usec*>.
- *copy(src, begSec, begUSec, endSec, endUSec, archivator)* — Copying of the part of the source archive <*src*> or its buffer in the current beginning from <*begSec*>:<*begUSec*> and ending with <*endSec*>:<*endUSec*> for the archivator <*archivator*>.
- *FFT(tm, size, archivator, tm_usec)* -- Performs the Fast Fourier Transformation using the FFT algorithm. Returns an array of amplitudes of the frequencies for archive's values window for begin time <*tm*>:<*tm_usec*> (seconds:microseconds), depth to history <*size*> (seconds) and for archivator <*archivator*>.

Example:

```
using Special.FLibSYS;
val = vArh(strPath2Sep(addr)).get(time,uTime,0,archtor);
return val.isEval() ? "Empty" : real2str(val,prec);
```

1.6. Buffer of the values archive (vArhBuf)

Description: Getting the object of the buffer of the values archive (VArchObj) to perform the intermediate operations on frames of data.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	Object(VArchObj)	Return	
tp	Type of the values of the archive (0-Boolean, 1-Integer, 4-Real, 5-String)	Integer	In	1
sz	Maximum buffer size	Integer	In	100
per	periodicity of buffer (in microseconds)	Integer	In	1000000
hgrd	Mode “Hard time grid”	Boolean	In	0
hres	Mode «High time resolution (microseconds)"	Boolean	In	0

2. Functions for the astronomical time processing

2.1. Time string (tmFStr) <3047>

Description: Converts an absolute time in the string of the required format. Recording of the format corresponds to the POSIX-function strftime.

Parameters:

ID	Parameter	Type	Mode	By defaults
val	Full date string	String	Return	
sec	Seconds	Integer	In	0
form	Format	String	In	%Y-%m-%d %H:%M:%S

Example:

```
using Special.FLibSYS;
test=tmFStr(SYS.time(),"%d %m %Y");
messPut("Example",0,"tmFStr(): "+test);
```

2.2. Full Date (tmDate) <973>

Description: Returns the full date in seconds, minutes, hours, etc., based on the absolute time in seconds from the epoch of 1/1/1970.

Parameters:

ID	Parameter	Type	Mode	By defaults
fullsec	Full seconds	Integer	In	0
sec	Seconds	Integer	Out	0
min	Minutes	Integer	Out	0
hour	Hours	Integer	Out	0
mday	Day of the month	Integer	Out	0
month	Month	Integer	Out	0
year	Year	Integer	Out	0
wday	Day of the week	Integer	Out	0
yday	Day of the year	Integer	Out	0
isdst	Daylight saving time	Integer	Out	0

Example:

```
using Special.FLibSYS;
curMin=curHour=curDay=curMonth=curYear=0;
tmDate(tmTime(),0,curMin,curHour,curDay,curMonth,curYear);
messPut("test",0,"Current minute: "+curMin);
messPut("test",0,"Current hour: "+curHour);
messPut("test",0,"Current day: "+curDay);
messPut("test",0,"Current month: "+curMonth);
messPut("test",0,"Current Year: "+curYear);
```

2.3. Absolute time (tmTime) <220>

Description: Returns the absolute time in seconds from the epoch and in microseconds, if <usec> is installed in a non-negative value.

Parameters:

ID	Parameter	Type	Mode	By defaults
sec	Seconds	Integer	Return	0
usec	Microseconds	Integer	Out	-1

2.4. Conversion the time from the symbolic representation to the time in seconds from the epoch of 1/1/1970 (tmStrPTime) <2600>

Description: Returns the time in seconds from the epoch of 1/1/1970, based on the string record of time, in accordance with the specified template. For example, template "%Y-%m-%d %H:%M:%S" corresponds the time «2006–08–08 11:21:55». Description of the format of the template can be obtained from the documentation on POSIX-function “strptime”.

Parameters:

ID	Parameter	Type	Mode	By defaults
sec	Seconds	Integer	Return	0
str	Date string	String	In	
form	Date record format	String	In	%Y-%m-%d %H:%M:%S

Example:

```
using Special.FLibSYS;
curMin=curHour=curDay=curMonth=curYear=0;
tmDate(tmTime(),0,curMin,curHour,curDay,curMonth,curYear);
test = tmStrPTime(""+curYear+"-"+(curMonth+1)+"-"+curDay+" 9:0:0","%Y-%m-%d %H:
%M:%S");
messPut("Example",0,"tmStrPTime(): "+test);
```

2.5. Planning of the time in the Cron format (tmCron)

Description: Returns the time planned in the format of the Cron standard beginning from the base time of from the current time, if the base is not specified.

Parameters:

ID	Parameter	Type	Mode	By defaults
res	Result	Integer	Return	0
str	Record in the Cron standard	String	In	* * * * *
base	Base time	Integer	In	0

3. Functions of the messages processing

3.1. Messages request (messGet)

Description: Request of the system messages.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	Object(Array)	Return	
btm	Start time	Integer	In	
etm	End time	Integer	In	
cat	Category of the message	String	In	
lev	Level of the message	Integer	In	
arch	Archivator	String	In	

3.2. Generation of the message (messPut)

Description: Formation of the system message.

Parameters:

ID	Parameter	Type	Mode	By defaults
cat	Category of the message	String	In	
lev	Level of the message	Integer	In	
mess	Text of the message	String	In	

Example:

```
rnd_sq_gr11_lineClr="red";
Special.FLibSYS.messPut("Example",1,"Event: "+rnd_sq_gr12_leniClr);
```

4. Functions of the strings processing

4.1. Getting the size of the string (strSize) <114>

Descroption: It is used to get the size.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	Integer	Return	
str	String	String	In	

Example:

```
Special.FLibSYS.messPut("Example",1,"ReturnString: "+strSize("Example"));
```

4.2. Getting the part of the string (strSubstr) <413>

Description: It is used to det the part of the string.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	String	Return	
str	String	String	In	
pos	Position	Integer	In	0
n	Quantity	Integer	In	-1

Example:

```
using Special.FLibSYS;
test=strSubstr("Example", 0, strSize("Example")-1);
messPut("Example",1,"ReturnString: "+test);
```

4.3. Insert of the on string to the another (strInsert) <1200>

Description: It is used to insert of the on string to the another.

Parameters:

ID	Parameter	Type	Mode	By defaults
str	String	String	Out	
pos	Position	Integer	In	0
ins	Inserting string	String	In	

4.4. Change the part of the string with the another one (strReplace) <531>

Description: It is used to change the part of the string with the another one.

Parameters:

ID	Parameter	Type	Mode	By defaults
str	String	String	Out	
pos	Позиция	Integer	In	0
n	Quantity	Integer	In	-1
repl	Changing string	String	In	

4.5. Parsing the string on separator (strParse) <537>

Description: It is used to parse the string on separator.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	String	Return	
str	String	String	In	
lev	Level	Integer	In	
sep	Separator	String	In	"."
off	Offset	Integer	Out	

Example:

```
using Special.FLibSYS;
ExapleString="Example:123";
test=strParse(ExapleString,1,":");
messPut("Example",0,"strParse(): "+test);
```

4.6. Path parsing (strParsePath) <300>

Description: It is used for the parsing the path on the elements.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	String	Return	
path	Path	String	In	
lev	Level	Integer	In	
off	Offset	Integer	Out	

Example:

```
using Special.FLibSYS;
test=strParsePath(path,0,"/");
messPut("Example",1,"strParsePath(): "+test);
```

4.7. Path to the string with the separator (strPath2Sep)

Description: It is used to convert the path to the string with the separator.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	String	Return	
src	Source	String	In	
sep	Separator	String	In	"."

Example:

```
//Converting value "/ses_AGLKS/pg_so" of the attribute "path"
//into value "ses_AGLKS.pg_so"
using Special.FLibSYS;
test = strPath2Sep(path);
messPut("Example",0,"path: "+path);
messPut("Example",0,"strPath2Sep(): "+test);
```

4.8. Coding of the string to HTML (strEnc2HTML)

Description: It is used to code the string for using in the HTML source.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	String	Return	
src	Source	String	In	

4.9. Encode text to bin (strEnc2Bin)

Description: Use for encode text to bin, from format <00 A0 FA DE>.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	String	Return	
src	Source	String	In	

4.10. Decode text from bin (strDec4Bin)

Description: Use for decode text from bin to format <00 A0 FA DE>.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	String	Return	
src	Source	String	In	

4.11. Convert real to string (real2str)

Description: It is used to convert real to string.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	String	Return	
val	Value	Real	In	
prc	Precision	Integer	In	4
tp	Type	String	In	“f”

4.12. Convert integer to string (int2str)

Description: It is used to convert integer to string.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	String	Return	
val	Value	Integer	In	
base	Base, supported: 8, 10, 16	Integer	In	10

4.13. Convert the string to real (str2real)

Description: It is used to convert string to real.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	Real	Return	
val	Value	String	In	

4.14. Convert the to integer (str2int)

Description: It is used to convert string to integer.

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	Integer	Return	
val	Value	String	In	
base	Base	Integer	In	0

5. Functions for the real processing

5.1. Splitting the float to the words (floatSplitWord) <56>

Description: Splitting the float (4 bites) to the words (2 bites).

Parameters:

ID	Parameter	Type	Mode	By defaults
val	Value	Real	In	
w1	Word 1	Integer	Out	
w2	Word 2	Integer	Out	

5.2. Merging the float from words (floatMergeWord) <70>

Description: Merging the float (4 bites) from words (2 bites).

Parameters:

ID	Parameter	Type	Mode	By defaults
rez	Result	Real	Return	
w1	Word 1	Integer	In	
w2	Word 2	Integer	In	

6. User programming API

Some objects of the module provides functions for user's programming.

The object "Functions library" (SYS.Special.FLibMath)

- *ElTp {funcID}(ElTp prm1, ...)* — call the library function *{funcID}*. Return result of the called function.

The object "User function" (SYS.Special.FLibMath["funcID"])

- *ElTp call(ElTp prm1, ...)* — call the function with parameters *<prm{N}>*. Return result of the called function.