АННОТАЦИЯ

Настоящая часть руководства оператора содержит сведения об общесистемном API пользовательского программирования ПП «СКАДА А-СОФТ» (далее по тексту СКАДА) и предназначен для операторов, занимающихся разработкой ПО в данной системе.

СОДЕРЖАНИЕ

1 Общесистемное API пользовательского программирования

API пользовательского программирования представляет собой дерево объектов ПП «СКАДА А-СОФТ», каждый объект которого может представлять собственный перечень свойств и функций. Свойства и функции объектов могут использоваться пользователем в процедурах на языках пользовательского программирования СКАДА. Точкой входа для доступа к объектам СКАДА из языка пользовательского программирования JavaLikeCalc является зарезервированное слово "SYS" корневого объекта СКАДА. Например, для доступа к функции исходящего транспорта нужно записать: `SYS.Transport.Serial.out_ModBus.messIO(mess);`.

1.1 Общесистемные пользовательские объекты

Объект представляет собой контейнер свойств и функций. Свойства могут содержать данные базовых типов и другие объекты. Существует 4 базовых типа: нулевой, логический, числовой и строковый. Доступ к свойствам может осуществляться посредством записи имен через точку <obj.prop> или через квадратные скобки <obj["prop"]>. Объект создается посредством оператора new: <var0=newObject()>. Различные компоненты могут дополнять объект свойствами и функциями. Базовые типы также обладают свойствами и функциями.

Свойства и функции базовых типов:

- нулевой тип, функции:
  - *bool isEVal()* - возвращает true;
- логический тип, функции:
  - *bool isEVal()* - проверка на EVAL;
  - *string toString()* - представление значения в виде строки "false"или "true";
- целое и вещественное число, свойства:
  - *MAX_VALUE* - максимальное значение;
  - *MIN_VALUE* - минимальное значение;
  - *NaN*- недостоверное значение;
- целое и вещественное число, функции:
  - *bool isEVal()* - проверка на EVAL;
  - *string toExponential(int numbs=-1)* - возврат числа с плавающей точкой в виде строки с количеством значащих цифр <numbs>;

- *string toFixed(int numbs=0, int len=0, bool sign=false)* – возврат числа с фиксированной точкой в виде строки с количеством цифр после точки <numbs>, минимальной длиной <len> и знаком <sign>;

- *string toPrecision(in tprec=-1)* – возврат числа в виде строки с количеством значащих цифр <prec>;

- *string toString(int base=10, int len=-1, bool sign=false)* – возврат целого числа в виде строки с базой <base=2-36>, минимальной длиной <len> и знаком <sign>.

Строка, свойства:

- *int length* - длина строки.

Строка, функции:

- *bool is EVal()* – проверка на EVAL;
- *string charAt(int symb)* – извлекает из строки символ <symb>;
- *int charCodeAt(int symb)* – извлекает из строки код символа <symb>;
- *string concat (sting val1, string val2,...sring valN)* - возвращает строку, сформированную путем присоединения val1,..valN к исходной;
- *int indexOf(string substr, int start)* – возвращает позицию <substr> в строке начиная со <start>. Если позиция не указана, поиск идет с начала;
- *int lastIndexOf(string substr, int start)* – возвращает позицию <substr> в строке начиная со <start> с конца. Если позиция не указана, поиск идет с конца;
- *int search(string pat, string flg="")* – поиск в строке по шаблону <pat> с флагами <flg>. Возвращает позицию найденной подстроки;
- *int search (RegExp pat)* - поиск в строке по шаблону RegExp <pat> (см. описание объекта RegExp);
- *Array match(string pat, string flg="")* – поиск в строке по шаблону <pat> с флагами <flg>. Возвращает массив с найденной подстрокой и подвыражениями;
- *Array match (RegExp pat)* - поиск в строке по шаблону RegExp <pat>. Возвращает массив с найденной подстрокой и подвыражениями;
- *string slice(int beg, int end), string substr(int beg, int end)* - возврат строки, извлеченной из исходной, начиная с позиции <beg> и заканчивая <end>. Если <beg> или <end> отрицательна, поиск ведется с конца;
- *Array split(string sep, int limit)* - возврат массива элементов строки, разделенных <sep>. Количество элементов ограничено <limit>;
- *Array split(RegExp pat, int limit)* - возврат массива элементов строки, разделенных шаблоном RegExp <pat>. Количество элементов ограничено <limit>;

- *string insert(int pos, string substr)* – вставка в позицию <pos> подстроки <substr>;

- *string replace(int pos, int n, string str)* - заменяет на строку <str> <n> символов, начиная с <pos>;

- *string replace(string substr, string str)* – замена всех строк <substr> на <str>;

- *string replace(RegExp pat, string str)* – замена всех строк по шаблону <pat> на <substr>;

- *real toReal()* - преобразование строки в число типа real;

- *int toInt(int base=0)* - преобразование строки в целое число с основанием <base>. Если <base>=0, то преобразование происходит с учетом префикса строки (123 - десятичное, 0123 - восьмеричное, 0x123 - шестнадцатеричное);

- *string parse(int pos, string sep=".", int off=0)* - выделение из исходной строки элемента <pos> для разделителя элементов <sep> от смещения <off>. Результирующее смещение помещается в <off>;

- *string parsePath(int pos, int off=0)* - выделение из исходной строки элемента <pos> от смещения <off>. Результирующее смещение помещается в <off>;

- *string path2sep(string sep=".")* - преобразование пути в текущей строке в строку с разделителем <sep>.

1.1.1 *Объект "Array"*

Объект "Array" представляет собой массив, который создается командой *<var0=newArray(prm1, prm2,... prmN)>*. Массив работает со свойствами как с индексами, поэтому обращение к свойствам доступно только через квадратные скобки (var0[1]). Массив имеет набор стандартных свойств и функций.

Свойства массива:

- *length* - возвращает размер массива.

Функции массива:

- *string join(string sep=","), string toString(string sep=","), string valueOf(string sep=",")* – возвращает строку с элементами массива, разделенными <sep> или ",";

- *Array concat(Array arr)* - добавляет к исходному массиву элементы массива <arr>;

- *int push(EITp var,...)* - помещает элементы <var> в конец массива, как в стек, возвращает новый размер массива;

- *EITp pop()* - удаление последнего элемента массива и возврат его значения, как из стека;

- *Array reverse()* - изменение порядка расположения элементов массива;

- *EITp shift()* - сдвиг массива вверх, при этом первый элемент удаляется, а его значение возвращается;

- *int unshift(EITp var,...)* - задвигает элементы <var> в массив;

- *Array slice(int beg, int end)* - возвращает фрагмент массива от <beg> до <end>;

- *Array splice(int beg, int remN, EITp val1, EITp val2,...)* - вставляет, удаляет или заменяет элементы массива. Возвращает массив удаленных элементов. В первую очередь удаляются элементы с позиции <beg> и количеством <remN>, затем вставляются значения <val1> и т.д., начиная с позиции <beg>;

- *Array sort()* - сортировка элементов в лексикографическом порядке.

1.1.2 *Объект "RegExp"*

Объект работы с регулярными выражениями. При создании объекта в качестве аргументов передается строка с текстом регулярного выражения и флаги в виде строки символов:

- "g" - режим глобального поиска;
- "i" - режим регистронезависимого поиска;
- "m" - режим многострочного поиска;
- "u" - принудительное разрешение символов UTF-8;
- "p" - тестирование выражения по обычному шаблону с ключевыми символами "?", "*", "\";

Свойства объекта:

- source - исходный шаблон регулярного выражения;
- global - признак глобального поиска;
- ignoreCase - игнорировать регистр при поиске;
- multiline - признак многострочного поиска;
- UTF8 - UTF-8 - символы разрешены;
- lastindex - индекс символа за подстрокой последнего поиска, используется для продолжения сканирования при следующем вызове.

Функцииобъекта:

- Array exec(string val) - вызов поиска по строке <val>. Возвращает найденную подстроку и подвыражения в массиве. Устанавливает атрибут массива <index> в позицию найденной подстроки, атрибут <input> в значение исходной строки;

- bool test(string val) - возвращает <true>, если подстрока найдена в <val>.

Пример работы с объектом:

```
varre = newRegExp("(\\d\\d)[-/](\\d\\d)[-\](\\d\\d(:?\\d\\d)?)","");
var rez = re.exec("12/30/1969");
var month = rez[1];
var day = rez[2];
var year = rez[3];
var OK=rez.test("12/30/1969");
```

1.1.3 *Модуль FLibSys*

Специальный модуль FLibSYS предоставляет в систему СКАДА статическую библиотеку функций для работы с системой СКАДА, на уровне её системного API. Эти функции могут использоваться в среде пользовательского программирования системы СКАДА для организации неординарных алгоритмов взаимодействия.

Для адресации к функциям этой библиотеки можно использовать статический адрес вызова *"Special.FLibSYS.{Func}()"* или динамический *"SYS.Special.FLibSYS["{Func}"].call()", "SYS.Special.FLibSYS.{Func}()"*. Где *{Func}* — идентификатор функции в библиотеке.

1.1.4 *Объект XMLNodeObj*

Функции:

- *string name( )* – имя узла, XML-тега;
- *string text(bool full=false)* – текст узла, содержимое XML-тега.Установить full для получения комбинированного текста со всех включенных узлов;
- *string attr (string id)* – значение атрибута узла<id>;
- *XMLNodeObj setName(string vl)* – установка имени узла в <vl>. Возвращает текущий узел;
- *XMLNodeObj setText(string vl)* – установка текста узла в <vl>. Возвращает текущий узел;
- *XMLNodeObj setAttr(string id, string vl)* – установка атрибута <id> в значение <vl>. Возвращает текущий узел;
- *int childSize( )* – количество вложенных узлов;
- *XMLNodeObj clear (bool full = false)* – очищает узел, удалением дочерних узлов, очищает текст и атрибуты, для <full>;
- *XMLNodeObj childAdd(ElTp no = XMLNodeObj); XMLNodeObjchildAdd(string no)* – добавление объекта <no> как вложенного. <no> может быть как

непосредственнообъектом-результатом функции SYS.XMLNode(), так и строкой с именем нового тега. Возвращаетсявложенный узел;

- *XMLNodeObj childIns(int id, ElTp no = XMLNodeObj); XMLNodeObj childIns(int id, string no)* – вставка объекта <no> как вложенного в позицию <id>. <no> может быть как непосредственно объектом-результатом функции SYS.XMLNode(), так и строкой с именем нового тега. Возвращается вложенный узел;
- *XMLNodeObj childDel(int id)* – удаление вложенного узла в позиции <id>. Возвращает текущий узел;
- *XMLNodeObj childGet(int id)* – получение вложенного узла в позиции <id>;
- *XMLNodeObj childGet(string name, int num = 0)* – получает вложенный узел с именем тега <name> и порядковым номером <num>;
- *XMLNodeObj parent()* – получить родительский узел;
- *string load(string str, bool file = false, int flg = 0, string cp = "UTF-8")* – загружает XML из строки <str> или из файла с путём в <str> если <file> равно "true", с кодировкой <cp>. Где <flg> – флаги загрузки:
  - 0x01 – полная загрузка, с блоками текста и комментариями в специальных узлах;
  - 0x02 – не удалять пробелы в начале и конце текста тега;
- *string save(int flg = 0, string path = "", string cp = "UTF-8")* – сохраняет дерево XML в строку или в файл <path> с параметрами форматирования <flg> и кодировкой <cp>. Возвращает текст XML или код ошибки. Предусмотрены следующие флаги форматирования <flg>:
  - 0x01 – прерывать строку перед открывающим тегом;
  - 0x02 – прерывать строку после открывающего тега;
  - 0x04 – прерывать строку после закрывающего тега;
  - 0x08 – прерывать строку после текста;
  - 0x10 – прерывать строку после инструкции;
  - 0x1E – прерывать строку после всех;
  - 0x20 – вставлять стандартный XML-заголовок;
  - 0x40 – вставлять стандартный XHTML-заголовок;
  - 0x80 – очищать сервисные теги: <??>, <!-- -->;
  - 0x100 – не кодировать наименований тегов;
  - 0x200 – не кодировать наименований атрибутов.
- *XMLNodeObj getElementBy(string val, string attr = "id")* – получить элемент из дерева по атрибуту <attr> со значением <val>;

- *TArrayObj <XMLNodeObj> getElementsBy (string tag, string attrVal = "", string attr = "id")* – получает массив элементов из дерева по тегу <tag> (пустой для всех) и атрибуту <attr> со значением <attrVal> (пустые для пропуска).

1.2 Система (SYS)

Функции объекта:

- *string system(string cmd, bool noPipe = false);* – осуществляет вызов консольных команд <*cmd*> ОС с возвратом результата по каналу. Если <*noPipe*> установлен, то возвращается код возврата вызова и возможен запуск программ в фоне ("sleep 5 &"). Функция открывает широкие возможности пользователю СКАДА путём вызова любых системных программ, утилит и скриптов, а также получения посредством них доступа к огромному объёму системных данных. Например, команда "ls -l" вернёт детализированное содержимое рабочей директории;
- *string fileRead(string file);* – возвращает содержимое файла <*file*> в строке;
- *int fileWrite(string file, string str, bool append = false);* – записывает строку <*str*>в файл <*file*>, удаляя присутствующий файл или добавляя в него <*append*>. Возвращает количество записанных байт;
- *int fileRemove(string file);* – удаляетфайл<*file*>. Возвращает результат удаления;
- *int message(string cat, int level, string mess);* – формирование системного сообщения <*mess*>с категорией <*cat*>, уровнем <*level*>(-7...7). Отрицательное значение уровня формирует нарушения (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);* – формирование системного сообщения <mess> с категорией <cat> и соответствующим уровнем;
- *XMLNodeObj XMLNode(string name = "");* – создание объекта узла XML с именем <name>;
- *string cntrReq(XMLNodeObj req, string stat = "");* – запрос интерфейса управления к системе посредством XML. Обычный запрос записывается в виде <get path="/OPath/%2felem"/>. При указании станции осуществляется запрос к внешней станции;

Пример:

```
//получение идентификатора станции
req = SYS.XMLNode("get").setAttr("path","/%2fgen%2fid");
SYS.cntrReq(req);
idSt = req.text();
```

- *string sleep(int tm, int ntm = 0);* — приостановить поток исполнения на <*tm*> секунд и <*ntm*> наносекунд. Функция не рекомендована к использованию, особенно в процедурах пользовательского интерфейса, поскольку это приведет к блокированию интерфейса;
- *int time(int usec);* — возвращает абсолютное время в секундах от эпохи 1.1.1970 и микросекундах, если <*usec*> указан;
- *int {local time|gm time}(int fullsec, int sec, int min, int hour, int mday, int month, int year, int wday, int yday, int isdst);* — возвращает полную дату и время в секундах <sec>, минутах <min>, часах <hour>, днях месяца <mday>, месяце <month>, годе <year>, днях недели <wday>, днях в году <yday> и признак летнего времени <isdst>, исходя из абсолютного времени в секундах *fullsec* от эпохи 1.1.1970. *gmtime* возвращает время в GMT(UTC);
- *string strftime(int sec, string form = "%Y-%m-%d %H:%M:%S");* — преобразует абсолютное время <*sec*>в строку нужного формата <*form*>. Запись формата соответствует POSIX-функции strftime;
- *int strptime(string str, string form = "%Y-%m-%d %H:%M:%S");* — возвращает время в секундах от эпохи 1.1.1970, исходя из строковой записи времени <*str*>, в соответствии с указанным шаблоном <*form*>. Например, шаблону "%Y-%m-%d %H:%M:%S" соответствует время "2017-08-08 11:21:55". Описание формата шаблона можно получить из документации на POSIX-функцию "strptime";
- *int cron(string cronreq, int base = 0);* — возвращает время, спланированное в формате стандарта Cron <*cronreq*>, начиная от базового времени <*base*> или от текущего, если базовое не указано;
- *string strFromCharCode(int char1, int char2, int char3, ...);* — создание строки из кодов символов char1, char2 ... charN;
- *string strCodeConv(string src, string fromCP, string toCP);* — кодирование текста <*src*> из кодировки <*fromCP*> в <*toCP*>. Если кодировка опущена, то используется внутренняя.

1.3 Любой объект (TCntrNode) дерева СКАДА (SYS.*)

Функции объекта:

- *TArrayObj nodeList(string grp = "", string path = "");* — получение списка дочерних узлов для группы <grp> и узла по пути <path>. Если <grp> пуста, то возвращаются узлы всех групп;
- *TCntrNodeObj nodeAt(string path, string sep="");* — подключение к узлу <path> в дереве объектов СКАДА. Если указывается разделитель в <sep>, то путь обрабатывается как строка с разделителем;
- *TCntrNodeObj nodePrev();* — получить предыдущий, родительский, узел;
- *string nodePath(string sep = "", bool from_root = true);* — получение пути к текущему узлу, в дереве объектов СКАДА. Один символ разделителя указывается в <sep> для получения пути через разделитель, например, "DAQ.ModBus.PLC1.P1.var", иначе "/DAQ/ModBus/PLC1/P1/var". <from_root> указывает на необходимость формировать путь от корня и без указания идентификатора станции;
- *int messSys(int level, string mess)* — формирует системное сообщение <mess> с уровнем <level>, с путём узла в качестве категории и с читабельным путём перед сообщением.

1.4 Подсистема "Безопасность" (SYS.Security)

Функции объекта подсистемы (SYS.Security):

- *int access(string user, int mode, string owner, string group, int access)* — проверка для пользователя <user> доступа к ресурсу, который принадлежит <owner> и <group> с доступом <access> для режима <mode>:

  user — пользователь для проверки доступа;
  mode — режим доступа (4-R, 2-W, 1-X);
  owner — владелец ресурса;
  group — группа ресурса;
  access — режим доступа к ресурсу (RWXRWXRWX — 0777).

Функции объекта пользователя (SYS.Security["usr_User"]) и группы (SYS.Security["grp_Group"]):

- *ElTp cfg(string nm)* — получение значения конфигурационного поля <*nm*> объекта;
- *bool cfgSet(string nm, ElTp val)* — установка конфигурационного поля <*nm*> объекта в значение <*val*>;
- *Array groups( )* — возвращает перечень групп пользователя;
- *bool user(string nm)* — проверяет присутствие пользователя <*nm*> в данной группе.

1.5 Подсистема "БД" (SYS.BD)

Функции объекта БД (SYS.BD["TypeDB"]["DB"]):

- *ElTp cfg(string nm)* — получение значения конфигурационного поля <*nm*> объекта;
- *Bool cfgSet(string nm, ElTp val)* — установка конфигурационного поля <*nm*> объекта в значение <*val*>;
- *Array SQLReq(string req);* — формирование SQL-запроса к БД.

  Пример:

```
DBTbl=SYS.BD.MySQL.GenDB.SQLReq("SELECT * from DB;");
for(var i_rw = 0; i_rw < DBTbl.length; i_rw++)
{
var rec = "";
for( var i_fld = 0; i_fld < DBTbl[i_rw].length; i_fld++ )
rec += DBTbl[i_rw][i_fld]+"\t";
SYS.messDebug("TEST DB","Row "+i_rw+": "+rec);
//> получить значение в столбце по имени
if(i_rw) SYS.messDebug("TEST DB: ","Row "+i_rw+":
'NAME'"+DBTbl[i_rw]
["NAME"]);
}
```

Функции объекта Таблицы (SYS.BD["TypeDB"]["DB"]["Table"]):

- *XMLNodeObj fieldStruct();* – получение структуры таблицы в виде XML узла "field" с дочерними узлами-колонками `<RowIdtype="real" len="10.2" key="1" def="Defaultvalue">{Value}</RowId>`, где:
  - {RowId} – идентификатор колонки;
  - {Value} – значение колонки;
  - type – тип значения колонки: *str* – строка, *int* – целое, *real* – вещественное и *bool* – логическое;
  - len – размер значения колонки, в знаках;
  - key – признак того, что колонка является ключом, и поиск осуществляется по его значению;
  - def – значение колонки по умолчанию.
- *string fieldSeek(int row, XMLNodeObj fld);* – Запрос поля <*row*> таблицы. Если поле получено, то возвращается "1" иначе "0". В случае ошибки возвращается "0:Error";
- *string fieldGet(XMLNodeObj fld);* – Запрос значений поля. В случае ошибки возвращается "0:Error".

  Пример:

```
req = SYS.XMLNode("field");
req.childAdd("user").setAttr("type","str").setAttr("key","1").
setText("root");
```

```
req.childAdd("id").setAttr("type","str").setAttr("key","1").
setText("/Lang2CodeBase");
req.childAdd("val").setAttr("type","str");
SYS.BD.MySQL.GenDB.SYS.fieldGet(req);
SYS.messDebug("TESTDB","Value: "+req.childGet(2).text());
```

- *string fieldSet(XMLNodeObj fld);* – Установка поля. В случае ошибки возвращается "0:Error";
- *string fieldDel(XMLNodeObj fld);* – Удаление поля. В случае ошибки возвращается "0:Error".

1.6 Подсистема "Сбор данных" (SYS.DAQ)

Функции объекта подсистемы (SYS.DAQ):

- *bool funcCall(string progLang, TVarObj args, string prog);* – вызов текста функции <*prog*> с аргументами в объекте <*args*> для языка программирования <*progLang*>. Возвращает "true" при корректном вызове.

  Пример:

```
varargs = newObject();
args.y = 0;
args.x = 123;
SYS.DAQ.funcCall("JavaLikeCalc.JavaScript",args,"y=2*x;");
SYS.messDebug("TESTCalc","TESTCalcrezult: "+args.y);
```

Функции объекта контроллера (SYS.DAQ["Modul"]["Controller"]):

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <nm> объекта;
- *bool cfgSet(string nm, ElTp val)* – установка конфигурационного поля <nm> объекта в значение <val>;
- *string name()* – имя контроллера;
- *string descr()* – описание контроллера;
- *string status()* – статус контроллера;
- *bool alarmSet(string mess, int lev = -5, string prm = "")* – установка/снятие нарушения <mess> с уровнем <lev> (отрицательный для установки, иначе снятие), для параметра <prm>. Функция формирует нарушение с категорией: al{ModId}:{CntrId}[.{PrmId}], где:
  - *ModId* — идентификатор модуля;
  - *CntrId* — идентификатор контроллера;
  - *PrmId* — идентификатор параметра, из аргумента <*prm*>;
- *bool enable(bool newSt = EVAL)* – получение состояния "Включен" или изменение его назначением атрибута <*newSt*>;
- *bool start(bool newSt = EVAL)* – получение состояния "Запущен" или изменение его назначением атрибута <*newSt*>.

Функции объекта параметра контроллера (SYS.DAQ["Modul"]["Controller"]["Parameter"]):

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <*nm*> объекта;
- *ElTp cfgSet(string nm, ElTp val)* – установка конфигурационного поля <*nm*> объекта в значение <*val*>;

- *TCntrNodeObj cntr()* – возвращает объект контроллера этого параметра, независимо от вложенности.

Функции объекта атрибута параметра контроллера (SYS.DAQ["Modul"]["Controller"]["Parameter"] ["Attribute"]):

- *ElTp get(int tm = 0, int utm = 0, bool sys = false)* – запрос значения атрибута на время <tm:utm> и признаком системного доступа <sys>;
- *bool set(ElTp val, int tm = 0, int utm = 0, bool sys = false)* – запись значения <val> в атрибут с меткой времени <tm:utm> и признаком системного доступа <sys>;
- *TCntrNodeObj arch()* – получение объекта архива, связанного с этим атрибутом. В случае отсутствия связанного архива возвращается "false";
- *string descr()* – описание атрибута;
- *int time(int utm)* — время последнего значения в секундах и микросекундах в <utm>;
- *int len()* – длина поля;
- *int dec()* – разрешение для вещественного;
- *int flg()* – флаги поля
- *string def()* – значение по умолчанию;
- *string values()* – список допустимых значений или диапазон;
- *string selNames()* – список имён допустимых значений;
- *string reserve()* – резервное свойство значения.

Функции объекта библиотеки шаблона (SYS.DAQ[tmplb_Lib"]) и шаблона (SYS.DAQ[tmplb_Lib"] ["Tmpl"]) параметра контроллера:

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <nm> объекта;
- *bool cfgSet(string nm, ElTp val)* – установка конфигурационного поля <nm> объекта в значение <val>.

1.6.1 *Модуль DAQ.JavaLikeCalc*

Объект "Библиотека функций" (SYS.DAQ.JavaLikeCalc["lib_Lfunc"])

ElTp {funcID}(ElTp prm1, ...) – вызов функции библиотеки *{funcID}*. Возвращает результат вызываемой функции.

Объект "Пользовательская функция" (SYS.DAQ.JavaLikeCalc["lib_Lfunc"]["func"])

ElTp call(ElTp prm1, ...) – вызов данной функции с параметрами <*prm{N}*>. Возвращает результат вызываемой функции.

1.6.2 *Модуль DAQ.ModBus*

Объект "Контроллер" (this.cntr())

- *string messIO(string pdu)* – отправка PDU <pdu> через транспорт объекта контроллера посредством ModBus протокола. PDU результата запроса помещается вместо запроса в <pdu>, а ошибка возвращается в результате функции.

Объект «Параметр» (this)

- *bool attrAdd(string id, string name, string tp = "real", string selValsNms = "")* [для включенного параметра логического типа] – добавление атрибута <id> с именем <name> и типом <tp>. Если атрибут уже присутствует, то будут применены свойства, которые возможно изменить "на ходу": имя, режим выбора и параметры выбора.
  - id, name – идентификатор и имя нового атрибута;
  - tp – тип атрибута [boolean | integer | real | string | text | object] + режим выбора [sel | seled] + только для чтения [ro];
  - selValsNms – две строки со значениями в первой и их именами во второй, разделённые ";";
- *bool attrDel(string id)* [для включенного параметра логического типа] – удаление атрибута <id>.

1.7 Подсистема "Архивы" (SYS.Archive)

Функции объекта подсистемы:

- *Array messGet(int btm, int etm, string cat = "", int lev = 0, string arch = "");* – запрос системных сообщений за время от <btm> до <etm> для категории <cat>, уровня <lev> и архиватора <arch>. Возвращается массив объектов сообщений со свойствами:
  - *tm* – время сообщения, секунды;
  - *utm* – время сообщения, микросекунды;
  - *categ* – категория сообщения;
  - *level* – уровень сообщения;
  - *mess* – текст сообщения.
- *bool messPut(int tm, int utm, string cat, int lev, string mess);* – запись сообщения <mess> с категорией <cat>, уровнем <lev> (-7...7) и временем <tm>.<utm> в архив и/или список нарушений.

Функции объекта архиватора сообщений

(SYS.Archive["mod_Modul"]["mess_Archivator"]):

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <nm> объекта;
- *bool cfgSet(string nm, ElTp val)* – установка конфигурационного поля <nm> объекта в значение <val>;
- *bool status()* – получение статуса архиватора;
- *int end()* – поучение времени окончания данных архиватора;
- *int begin()* – поучение времени начала данных архиватора.

Функции объекта архиватора значений (SYS.Archive["val_Modul"]["val_Archivator"]):

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <nm> объекта;
- *bool cfgSet(string nm, ElTp val)* – установка конфигурационного поля <nm> объекта в значение <val>;
- *bool status()* – получение статуса архиватора «Исполнение».

Функции объекта архива (SYS.Archive["va_Archive"]):

- *ElTp cfg (string nm)* – получает значение конфигурационного поля <nm> объекта;
- *bool cfgSet (string nm, ElTp val)* – устанавливает конфигурационное поля <nm> объекта в значение <val>;

- *bool status ()* – статус архиватора "Исполнение";
- *int end (string arch = "")* – время конца данных архива для архиватора <arch>, в микросекундах;
- *int begin (string arch = "")* – время начала данных архива для архиватора <arch>, в микросекундах;
- *int period (string arch = "")* – период данных архива для архиватора <arch>, в микросекундах;
- *TArrayObj archivatorList ()* – список архиваторов, использующих данный архив как источник;
- *VarType getVal (int tm, bool up_ord = false, string arch = "")* – получает значение из архива на время <tm>, подтянув к верху <up_ord> и архиватор <arch>:
  - tm – время запрашиваемого значения, в микросекундах, установить в 0 для "end()"; этот атрибут также является выходом, соответственно реальное время полученного значения помещается сюда, если это переменная;
  - up_ord – подтягивать время запрашиваемого значения к верху сетки;
  - arch – архиватор запроса, установить в пустую строку для проверки всех архиваторов, установить в "<buffer>" для обработки только буфера.
- *bool setVal (int tm, VarType vl, string arch = "")* – устанавливает значение <vl> в архив на время <tm> и архиватор <arch>:
  - tm – время устанавливаемого значения, в микросекундах;
  - vl – значение;
  - arch – архиватор установки, установить в пустую строку для всех архиваторов, установить в "<buffer>" для обработки только буфера.

1.8 Подсистема "Транспорты" (SYS.Transport)

Функции объекта входящего транспорта (SYS.Transport["Modul"]["in_Transp"]):

- *ElTp cfg(string nm)* – получение значения конфигурационного поля <nm> объекта;
- *bool cfgSet(string nm, ElTp val)* – установка конфигурационного поля <nm> объекта в значение <val>;
- *string status()* – строка статуса транспорта;
- *string addr(string vl = "")* – адрес транспорта, устанавливает в непустое значение <vl>;
- *string writeTo(string sender, string mess)* – отправляет сообщение <mess> отправителю <sender>, как ответ;
- *TArrayObj assTrsList( )* – список связанных выходных транспортов с данным входящим.

Функции объекта исходящего транспорта (SYS.Transport["Modul"]["out_Transp"]):

- *ElTp cfg(string nm)* — получение значения конфигурационного поля <*nm*> объекта;
- *bool cfgSet(string nm, ElTp val)* — установка конфигурационного поля <*nm*> объекта в значение <*val*>;
- *string status()* — строка статуса транспорта;
- *bool start(boolvl = EVAL, inttm = 0)* — установка статуса транспорта «Запущен», включить/остановить его для значения <vl> (если оно не EVAL). Возможно установить таймаут <tm>;
- *string addr(string vl = "")* — адрес транспорта, устанавливает в непустое значение <*vl*>;
- *string timings(string vl = "")* —синхронизация транспорта, устанавливает в непустое значение <*vl*>;
- *int attempts(int vl = EVAL)* — попытка соединения транспорта, устанавливает в непустое значение <*vl*>;
- *string messIO(string mess, real timeOut = 0);* — отправка сообщения <*mess*> через транспорт с таймаутом ожидания <*timeOut*> (в секундах). В случае нулевого таймаута это время берётся из настроек исходящего транспорта.

  Пример:

```
rez=SYS.Transport.Serial.out_ttyUSB0.messIO(SYS.strFromCharC
ode(0x4B,0x00,0x37,0x40),0.2);
```

```
while(true)
{
  trez = SYS.Transport.Serial.out_ttyUSB0.messIO("");
  if( !trez.length ) break;
  rez+=trez;
  }
```

- *int messIO(XMLNodeObj req, string prt);* — отправка запроса <*req*> к протоколу <*prt*> для осуществления сеанса связи через транспорт посредством протокола.

  Пример:

```
req = SYS.XMLNode("TCP");
req.setAttr("id","test").setAttr("reqTm",500).setAttr("node",1).
setAttr("reqTry",2).setText(SYS.strFromCharCode(0x03,0x00,0x00,
0x00,0x05));
SYS.Transport.Sockets.out_testModBus.messIO(req,"ModBus");
test = Special.FLibSYS.strDec4Bin(req.text());
```

1.9 Подсистема "Пользовательские интерфейсы" (SYS.UI)

1.9.1 *Модуль UI.VCAEngine*

Объект "Сеанс" (this.ownerSess())

- *string user( )* - текущий пользователь сеанса;
- *string alrmSndPlay()* - путь виджета, для которого на данный момент воспроизводится сообщение о нарушении;
- *int alrmQuittance(int quit_tmpl, string wpath = "")* - квитирование нарушений <*wpath*> с шаблоном <*quit_tmpl*>. Если <*wpath*> пустая строка, то производится глобальное квитирование.

Объект "Виджет" (this)

- *TCntrNodeObj ownerSess( )* - получить объект сеанса данного виджета;
- *TCntrNodeObj ownerPage( )* - получить объект родительской страницы данного виджета;
- *TCntrNodeObj ownerWdg(bool base = false)* - получить родительский виджет данного виджета. При указании <*base*> будет возвращены и объекты страниц;
- *TCntrNodeObj wdgAdd(string wid, string wname, string parent)* - добавление виджета <*wid*> с именем <*wname*> на основе библиотечного виджета <*parent*>.

  Пример:

```
//Добавить новый виджет на основе виджета текстового примитива
nw = this.wdgAdd("nw", "Новый виджет", "/wlb_originals/wdg_Text");
nw.attrSet("geomX", 50).attrSet("geomY", 50);
```

- *bool wdgDel(string wid)* – удалениевиджета<wid>;
- *TCntrNodeObj wdgAt(string wid, bool byPath = false)* – подключение к дочернему виджету или глобальному посредством пути <byPath>. В случае глобального подключения можно использовать абсолютный или относительный путь к виджету. Точкой отсчёта абсолютного адреса выступает объект корня модуля "VCAEngine", а значит первым элементом абсолютного адреса является идентификатор сеанса, который опускается. Относительный адрес берёт отсчёт от текущего виджета. Специальным элементом относительного адреса является элемент вышестоящего узла "...";
- *bool attrPresent(string attr)* – проверка атрибута виджета <attr> на факт присутствия;
- *ElTp attr(string attr)* – получение значения атрибута виджета <attr>. Для отсутствующих атрибутов возвращается пустая строка;

- *TCntrNodeObj attrSet(string attr, ElTp vl)*— установка атрибута виджета <attr> в значение <vl>. Возвращается текущий объект для конкатенации функций установки;
- *string link(string attr, bool prm = false)* — получение ссылки для атрибута виджета <attr>. При установке <prm> запрашивается ссылка для группы атрибутов (параметр), представленных указанным атрибутом;
- *string linkSet(string attr, string vl, bool prm)* — установка ссылки для атрибута виджета <attr>. При установке <prm> осуществляется установка ссылка для группы атрибутов (параметр), представленных указанным атрибутом.

  Пример:

```
//Установить ссылку восьмого тренда параметром
this.linkSet("el8.name", "prm:/LogicLev/experiment/Pi", true);
```

Объект "Виджет", примитива "Документ" (this)

string getArhDoc(integer nDoc) — получить текст документа архива на глубине <*nDoc*> (0-{aSize-1}).

1.10 Подсистема "Специальные" (SYS.Special)

1.10.1 *Модуль Special.FLibSYS*

Объект "Библиотека функций" (SYS.Special.FLibSYS):

ElTp {funcID}(ElTp prm1, ...) — вызов функции библиотеки *{funcID}*. Возвращает результат вызываемой функции.

Объект "Пользовательская функция" (SYS.Special.FLibSYS["funcID"]):

ElTp call(ElTp prm1, ...) – вызов данной функции с параметрами *<prm{N}>*. Возвращает результат вызываемой функции.

1.10.2 *Модуль Special.FLibMath*

Объект "Библиотека функций" (SYS.Special.FLibMath):

ElTp {funcID}(ElTp prm1, ...) – вызов функции библиотеки *{funcID}*. Возвращает результат вызываемой функции.

Объект "Пользовательская функция" (SYS.Special.FLibMath["funcID"]):

ElTp call(ElTp prm1, ...) – вызов данной функции с параметрами *<prm{N}>*. Возвращает результат вызываемой функции.

1.10.3 *Модуль Special.FLibComplex1*

Объект "Библиотека функций" (SYS.Special.FLibComplex1):

ElTp {funcID}(ElTp prm1, ...) – вызов функции библиотеки *{funcID}*. Возвращает результат вызываемой функции.

Объект "Пользовательская функция" (SYS.Special.FLibComplex1["funcID"]):

ElTp call(ElTp prm1, ...) – вызов данной функции с параметрами *<prm{N}>*. Возвращает результат вызываемой функции.

2 Описание Java-подобного языка

2.1 Элементы языка

Ключевые слова: if, else, while, for, break, continue, return, using, true, false.

Константы:

- десятичные: цифры 0–9 (12, 111, 678);
- восьмеричные: цифры 0–7 (012, 011, 076);
- шестнадцатеричные: цифры 0–9, буквы a-f или A-F (0x12, 0XAB);
- вещественные: 345.23, 2.1e5, 3.4E-5, 3e6;
- логические: true, false;
- строковые: "hello", без переноса на другую строку, однако с поддержкой прямой конкатенации строковых констант.

Типы переменных:

- целое: -231...231, EVAL_INT(-2147483647);
- вещественное: 3.4 * 10308, EVAL_REAL(-3.3E308);
- логическое: false, true, EVAL_BOOL(2);
- строка: последовательность символов-байтов (0...255) любой длины, ограниченной объёмом памяти и хранилищем в БД;
- EVAL_STR("<EVAL>").

Встроенные константы: pi = 3.14159265, e = 2.71828182, EVAL_BOOL(2), EVAL_INT(-2147483647), EVAL_REAL(-3.3E308), EVAL_STR("<EVAL>").

Атрибуты параметров системы СКАДА (начиная с подсистемы DAQ, в виде <Тип модуля DAQ>.<Контроллер>.<Параметр>.<Атрибут>).

2.2 Операции языка

Операции, поддерживаемые языком:

Символ	Описание
()	Вызов функции
{}	Программные блоки
++	Инкремент
--	Декремент
-	Вычитание, унарный минус
+	Сложение
!	Логическое отрицание
~	Побитовое отрицание
*	Умножение
/	Деление
%	Остаток от целочисленного деления
<<	Поразрядный сдвиг влево
>>	Поразрядный сдвиг вправо
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Не равно
|	Поразрядное "или"
&	Поразрядное "и"
^	Поразрядное "исключающее или"
&&	Логическое "и"
||	Логическое "или"
?:	Условная операция (i=(i<0)?0:i;)
=	Присваивание
+=	Присваивание со сложением
-=	Присваивание с вычитанием
*=	Присваивание с умножением
/=	Присваивание с делением

2.3 Встроенные функции языка

Синтаксис функции	**Описание функции**
double max(double x, double x1)	Максимум из x и x1
double min(double x, double x1)	Минимум из x и x1
string typeof(E1tp v1)	Тип значения v1
double sin(double x)	Синус
double cos(double x)	Косинус
double tan(double x)	Тангенс
double sinh(double x)	Синус гиперболический
double cosh(double x)	Косинус гиперболический
double tanh(double x)	Тангенс гиперболический
double asin(double x)	Арксинус
double acos(double x)	Арккосинус
double atan(double x)	Арктангенс
double rand(double x)	Случайное число от 0 до x
double lg(double x)	Десятичный логарифм
double ln(double x)	Натуральный логарифм
double exp(double x)	Экспонента
double pow(double x, double x1)	x в степени x1
double sqrt(double x)	Квадратный корень
double abs(double x)	Модуль x
double sing(double x)	Знак x
double ceil(double x)	Округление до большего целого
double floor(double x)	Округление до меньшего целого

2.4 Операторы языка

- условие внутри выражения: <st_open=(pos>100)?true:false;>;
- глобальное условие if: <if (st_open>100) pos=true; else pos=false>;
- цикл while: while (<условие>) <тело_цикла>;
- цикл for: for (<пре_условие>;<анализ>;<пост_вычисление>)<тело_цикла>;
- цикл for-in: for(<переменная> in<объект>)<тело_цикла>;
- прерывание выполнения цикла: break;
- продолжение выполнения цикла с начала: continue;
- установка области видимости часто используемой библиотеки: using;
- прерывание функции и возврат результата: return;
- создание объекта: new.

2.4.1 *Условные операторы*

Языком модуля поддерживаются два типа условий. Первый – это операции условия для использования внутри выражения, второй – глобальный, основанный на условных операторах.

Условие внутри выражения строится на операциях «?» и «:». В качестве примера можно записать следующее практическое выражение <st_open=(pos>=100)?true:false;>, что читается как «Если переменная <pos> больше или равна 100, то переменной <st_open> присваивается значение true, иначе - false.

Глобальное условие строится на основе условных операторов «if» и «else». В качестве примера можно привести тоже выражение, но записанное другим способом <if(pos>100) st_open=true; else st_open=false;>. Как видно, выражение записано по-другому, но читается также.

2.4.2 *Циклы*

Поддерживаются три типа циклов: while, for и for-in. Синтаксис циклов соответствует языкам программирования: C++, Java и JavaScript.

Цикл **while** в общем записывается следующим образом:

while(<условие>) <тело цикла>;

Цикл **for** записывается следующим образом:

for(<пре-инициализ>;<условие>;<пост-вычисление>) <тело цикла>;

Цикл **for-in** записывается следующим образом:

for(<переменная>in<объект>) <тело цикла>;

Где:

<*условие*> — выражение, определяющее условие;

<*тело цикла*> — тело цикла множественного исполнения;

<*пре-инициализ*> — выражение предварительной инициализации переменных цикла;

<*пост-вычисление*> — выражение модификации параметров цикла после очередной итерации;

<*переменная*> — переменная, которая будет содержать имя свойства объекта при переборе;

<*объект*> — объект для которого осуществляется перебор свойств.

2.4.3 *Специальные символы строковых переменных*

Символ	Описание
\n	Перевод строки
\t	Символ табуляции
\b	Забой
\f	Перевод страницы
\r	Возврат каретки
\\	Символ "\"
\041	"!" восьмеричный
\x21	"!" шестнадцатиричный

2.5 Общесистемные функции

sysCall - Вызов консольных команд и утилит операционной системы.

Описание: Осуществляет вызовы консольных команд ОС. Функция открывает широкие возможности пользователю СКАДА путём вызова любых системных программ, утилит и скриптов, а также получения посредством них доступа к огромному объёму системных данных. Например, команда “ls -l” вернёт детализированное содержимое рабочей директории.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
com	Команда	Строка	Вход	

Пример:

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

UUID – код UUID.

Описание: Формирование уникального постоянного идентификатора раздела жесткого диска.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
uuid	UUID	строка	Возврат	

dbReqSQL - SQL запрос.

Описание: Формирование SQL-запроса к БД.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Объект(Массив)	Возврат	
addr	Адрес БД	Строка	Вход	
req	SQL-запрос	Строка	Вход	
trans	Транзакция	Логический	Вход	2

Для закрытия транзакции после выполнения запроса необходимо установить значение параметра trans =1. По умолчанию транзакция остается открытой.

dbImportLO – импорт большого объекта в БД.

Описание: используется для импорта большого объекта в БД.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
addr	Адрес БД	Строка	Вход	
data	Дата	Строка	полная	

dbExportLO – экспорт большого объекта в БД.

Описание: используется для экспорта большого объекта в БД.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
addr	Адрес БД	Строка	Вход	
id	ID	Строка	Вход	

dbRemoveLO –удаление большого объекта из БД.

Описание: используется для удаления большого объекта из БД.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Логический	Возврат	
addr	Адрес БД	Строка	Вход	
id	ID	Строка	Вход	

xmlNode- узел XML.

Описание: Создание объекта узла XML.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Объект(XMLNodeObj)	Возврат	
name	Имя	Строка	Вход	

Пример:

```
using Special.FLibSYS;
//Создание обьекта "get" узла XML.
req = xmlNode("get");
//Создание обьекта "get" узла XML с созданием атрибутов.
//sub_DAQ/mod_ModBus/cntr_1/prm_1 - путь согласно структуре проекта
req    =    xmlNode("get").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/
%2fprm%2fst%2fen");
```

xmlCntrReq - запрос интерфейса управления.

Описание: Запрос интерфейса управления к системе посредством XML. Обычный запрос записывается в виде <get path="/OPath/%2felem"/>. При указании станции осуществляется запрос к внешней станции.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
req	Запрос	Объект(XMLNodeObj)	Выход	
stat	Станция	Строка	Вход	

Пример:

```
usingSpecial.FLibSYS;
//Получение признака "Включен/Выключен" параметра "1" контроллера "1"
//модуля "ModBus".
//sub_DAQ/mod_ModBus/cntr_1/prm_1 - путь согласно структуре проекта
req  =  xmlNode("get").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/%2f
prm%2fst%2fen");
rez = xmlCntrReq(req);
messPut("test",0,"Значение: "+req.text());
//Установка признака "Включен" параметра "1" контроллера "1" модуля
"ModBus".
req = xmlNode("set").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/
prm_1/%2fprm%2fst%2fen").setText(1);
rez = xmlCntrReq(req);
//Установка признака "Выключен" параметра "1" контроллера "1" модуля
//"ModBus".
 req = xmlNode("set").setAttr("path","/sub_DAQ/mod_ModBus/cntr_1/prm_1/
%2fprm%2fst%2fen").setText(0);
rez = xmlCntrReq(req);
```

vArh – архив значений.

Описание: Получение объекта архива значений (VArchObj) путём подключения к архиву по его адресу.

Параметры:

ID	Имя	Тип	Режим
rez	Результат	Объект(VArchObj)	Возврат
name	Имя, адрес к атрибуту параметра с архивом (DAQ.{Module}.{Cntr}.{Prm}.{Attr}) или непосредственно к архиву значений (Archive.{ValArchive}).	Строка	Вход

2.5.1 *Объект VArchObj*

Функции:

- *begin(usec, archivator)* — Получение времени начала архива путём возврата секунд и микросекунд <usec> для архиватора<archivator>.
- *end(usec, archivator)* — Получение времени окончания архива путём возврата секунд и микросекунд <usec> для архиватора<archivator>.
- *period(usec, archivator)* — Получение периодичности архива путём возврата секунд и микросекунд <usec> для архиватора<archivator>.
- *get(sec, usec, upOrd, archivator*) — Получение значения из архива на время *<sec>:<usec>* с привязкой к верху *<upOrd>* и для архиватора *<archivator>*. Реальное время полученного значения устанавливается в *<sec>:<usec>*.
- *set(val, sec, usec)* — Запись значения *<val>* в буфер архива на время*<sec>:<usec>*.
- *copy(src, begSec, begUSec, endSec, endUSec, archivator)* — Копирование части исходного <src> архива или его буфера в текущий начиная с <begSec>:<begUSec> и заканчивая <endSec>:<endUSec> для архиватора <archivator>.
- *FFT(tm, size, archivator, tm_usec)* — Выполняет разложение в ряд Фурье с помощью FFT алгоритма. Возвращается массив амплитуд частот для окна значений из архива с временем начала *<tm>:<tm_usec>* (секунды:микросекунды), глубиной в историю архива *<size>* (секунд) и для архиватора*<archivator>*.

Пример:

```
using Special.FLibSYS;
s = 0.0;
us = 0.0;
pathAttr = "DAQ.LogicLev.cntr.prm.value";
//получение времени начала архива
s = vArch(pathAttr).begin(us,"DBArch.ArchiveChange");
time_begin = s*1.0e6 + us;
//получение времени окончания архива
s = vArch(pathAttr).end(us,"DBArch.ArchiveChange");
time_end = s*1.0e6 + us;
//пример получения значения
val = vArch(pathAttr).get(s,us,1,"DBArch.ArchiveChange");
//пример получения периода архива
s = vArch(pathAttr).period(us,"DBArch.ArchiveChange");
T = s*1.0e6 + us;
//пример установки значения
s = SYS.tmTime(0);
us = 0.0;
val = 3.14;
vArch(pathAttr).set(val, s, us);
```

vArcBuf – Буфер архива значений (vArhBuf)

Описание: Получение объекта буфера архива значений (VArchObj) для выполнения промежуточных операций над кадрами данных.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Объект(VArchObj)	Возврат	
tp	Тип значений архива (0-Boolean, 1-Integer, 4- Real, 5-String)	Целый	Вход	1
sz	Максимальный размер буфера	Целый	Вход	100
per	Периодичность буфера (в микросекундах)	Целый	Вход	1000000
hgrd	Режим «Жесткая сетка времени»	Логический	Вход	0
hres	Режим “Высокого разрешения времени (микросекунды)”	Логический	Вход	0

2.5.2 *Функции работы с астрономическим временем*

tmFStr - Строка времени.

Описание: Преобразует абсолютное время в строку нужного формата. Запись формата соответствует POSIX-функции strftime.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
val	Строка полной даты	Строка	Возврат	
sec	Секунды	Целый	Вход	0
form	Формат	Строка	Вход	%Y-%m-%d

Пример:

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

tmDate – полная дата.

Описание: Возвращает полную дату в секундах, минутах, часах и т.д, исходя из абсолютного времени в секундах от 1.1.1970.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
fullsec	Полные секунды	Целый	Вход	0
sec	Секунды	Целый	Выход	0
min	Минуты	Целый	Выход	0
hour	Часы	Целый	Выход	0
mday	День месяца	Целый	Выход	0
month	Месяц	Целый	Выход	0
year	Год	Целый	Выход	0
wday	День недели	Целый	Выход	0
yday	День в году	Целый	Выход	0
isdst	Daylight saving time	Целый	Выход	0

Пример:

```
using Special.FLibSYS; curMin=curHour=curDay=curMonth=curYear=0;
tmDate(tmTime(),0,curMin,curHour,curDay,curMonth,curYear);
messPut("test",0,"Текущая минута: "+curMin);
messPut("test",0,"Текущий час : "+curHour);
messPut("test",0,"Текущий день: "+curDay);
messPut("test",0,"Текущий месяц: "+curMonth);
messPut("test",0,"Текущий год: "+curYear);
```

tmTime - абсолютное время.

Описание: Возвращает абсолютное время в секундах от эпохи 1.1.1970 и микросекундах, если <usec> установлен в неотрицательное значение.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
sec	Секунды	Целый	Возврат	0
usec	Микросекунды	Целый	Выход	-1

tmStr2Tm – конвертация строки во время

Описание: Возвращает время в секундах от 1.1.1970, исходя из строковой записи времени, в соответствии с указанным шаблоном. Например, шаблону "%Y-%m-%d %H:%M:%S" соответствует время «2006–08–08 11:21:55».

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
sec	Секунды	Целый	Возврат	0
str	Строка даты	Строка	Вход	
form	Формат записи даты	Строка	Вход	%Y-%m-%d %H:%M:%S

tmCron - планирование времени в формате Cron.

Описание: Возвращает время, спланированное в формате стандарта Cron начиная от базового времени или от текущего, если базовое не указано.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
res	Результат	Целый	Возврат	0
str	Запись в стандарте Cron	Строка	Вход	* * * * *
base	Базовое время	Целый	Вход	0

2.5.3 *Функции работы с сообщениями*

messGet - запрос системных сообщений.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Объект(Массив)	Возврат	
btm	Время начала	Целое	Вход	
etm	Время конца	Целое	Вход	
cat	Категория	Строка	Вход	
lev	Уровень сообщения	Целый	Вход	

arch	Архиватор	Строка	Вход	

messPut - генерация сообщения.

Описание: Формирование системного сообщения.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
time	время	Строка	вход	
cat	Категория сообщения	Строка	Вход	
lev	Уровень сообщения	Целый	Вход	
mess	Текст сообщения	Строка	Вход	

Пример:

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

messPut2 - генерация сообщения с отметкой времени.

Описание: поместить собщение с отметкой времени в систему.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
time	время	Строка	вход	
cat	Категория сообщения	Строка	Вход	
lev	Уровень сообщения	Целый	Вход	
mess	Текст сообщения	Строка	Вход	

2.5.4 *Функции работы со строками*

strSize - получение размера строки.

Описание: Используется для получения размера строки.

Параметры*:*

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Целый	Возврат	
str	Строка	Строка	Вход	

strSize8 - получение строки (UTF8).

Описание: Используется для получения размера строки UTF8.

Параметры*:*

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Целый	Возврат	
str	Строка	Строка	Вход	

strSubstr – получение части строки.

Описание: Используется для получения части строки.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
str	Строка	Строка	Вход	
pos	Позиция	Целый	Вход	0
n	Количество	Целый	Вход	-1

Пример:

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

strInsert – вставка одной строки в другую.

Описание: Используется для вставки одной строки в другую.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
str	Строка	Строка	Выход	
pos	Позиция	Целый	Вход	0
ins	Вставляемая строка	Строка	Вход	

strReplace - замена части строки другой.

Описание: Используется для замены части сроки другой строкой.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
str	Строка	Строка	Выход	
pos	Позиция	Целый	Вход	0
n	Количество	Целый	Вход	-1
repl	Заменяющая строка	Строка	Вход	

strParse - разбор строки по разделителю.

Описание: Используется в разборе строки по разделителю.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
str	Строка	Строка	Вход	
lev	Уровень	Целый	Вход	
sep	Разделитель	Строка	Вход	"."
off	Смещение	Целый	Выход	

strParsePath – разбор пути на части.

Описание: Используется для разбора пути на элементы.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
path	Путь	Строка	Вход	
lev	Уровень	Целый	Вход	
off	Смещение	Целый	Выход	

strPath2Sep- путь к разделенной строке.

Описание: Используется для преобразования пути в разделенную строку.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
src	Ресурс	Строка	Вход	
sep	Сепаратор	Строка	Вход	"."

strEnc2HTML – преобразование строки в кодировку HTML.

Описание: Используется для кодирования строки в HTML.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
src	Источник	Строка	Вход	

strEnc2Bin - кодирование текста в бинарный вид.

Описание: Используется для кодирования текста в бинарный вид, из формата<00 A0 FA DE>.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию

rez	Результат	Строка	Возврат	
src	Источник	Строка	Вход	

strDec4Bin – декодирование текста из бинарного вида.

Описание: Используется для декодирования текста из бинарного вида в формат <00 A0 FA DE>.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
src	Источник	Строка	Вход	

real2str - преобразование вещественного в строку.

Описание: Используется для преобразования вещественного в строку.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
val	Значение	Вещественное	Вход	
prc	Точность	Целое	Вход	4
tp	Тип	Строка	Вход	“f”

int2str - преобразование целого в строку.

Описание: Используется для преобразования целого в строку.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Строка	Возврат	
val	Значение	Целое	Вход	
base	База, поддерживаются: 8, 10, 16	Целое	Вход	10

str2real–преобразование строки в вещественное.

Описание: Используется для преобразования строки в вещественное.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Вещественное	Возврат	
val	Значение	Строка	Вход	

str2int - преобразование строки в целое.

Описание: Используется для преобразования строки в целое.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Целое	Возврат	
val	Значение	Строка	Вход	
base	Основа	Целый	Вход	0

2.5.5 *Функции работы с вещественным*

floatSplitWord - разделение float на слова.

Описание: Разделение float (4 байтов) на слова (2 байта).

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
val	Значение	Вещественное	Вход	
w1	Слово 1	Целый	Выход	
w2	Слово 2	Целый	Выход	

floatMergeWord - объединение float из слов.

Описание: Объединение float (4 байтов) из слов (2 байта).

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Вещественное	Возврат	
w1	Слово 1	Целый	Вход	
w2	Слово 2	Целый	Вход	

doubleSplitWord – разделение на слова.

Описание: Разделение double (8 байт) на слова (2байта).

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
val	Результат	Вещественное	Возврат	
w1	Слово 1	Целый	Вход	
w2	Слово 2	Целый	Вход	
w3	Слово 3	Целый	Вход	
w4	Слово 4	Целый	Вход	

doubleMergeWord – объединение double из слов.

Описание: Объединение double (8 байт) из слов (2байта).

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Вещественное	Возврат	
w1	Слово 1	Целый	Вход	
w2	Слово 2	Целый	Вход	
w3	Слово 3	Целый	Вход	
w4	Слово 4	Целый	Вход	

CRC – объединение float из слов.

Описание: Используется для создания унифицированного кода (CRC) шириной 8-64 бит.

Параметры:

ID	Параметр	Тип	Режим	По умолчанию
rez	Результат	Целое	Возврат	
data	Дата	Строка	Вход	
poly	Polynomial (reversion)	Целый	Вход	40961
wight	Ширина	Целый	Вход	16
init	Идентификатор	Целый	Вход	-1

Перечень принятых сокращений

БД	база данных
ОЗУ	оперативное запоминающее устройство
ОС	операционная система
ПО	программное обеспечение
СВУ	система верхнего уровня
API	application programming interface (программный интерфейс приложения)
HTML	язык гипертекстовой разметки (Hypertext Mark-up Language)
SCADA	диспетчерское управление и сбор данных (Supervisory Control And Data Acquisition)

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводи тельного докум. и дата	Подп.	Дата
	Измененных	Замененных	Новых	Аннулированных					