ENGINEERING TOMORROW

СИСТЕМА Comfort Contour Pro

Руководство по программированию в АРМ LanMon

версия 4.13

www.danfoss.ru

Содержание

О документе

Данный документ является описанием встроенных программных ресурсов АРМ LanMon: синтаксиса языков программирования, а также объектов и функций. Он предназначен для программистов, владеющих одним из языков: объектный бейсик, объектный паскаль, C++ или ява-скрипт. Этот документ не является руководством по какому-либо языку программирования. Для изучения программирования на одном из вышеназванных языков используйте соответствующую литературу.

Введение

В АРМ LanMon встроен скриптовой движок (далее скрипт) поддерживающий четыре языка программирования: C++, объектный паскаль, бейсик и ява-скрипт. Более подробно особенности скрипта в АРМ LanMon описаны в следующей главе.

Описание скриптового языка АРМ LanMon

Скриптовой язык в АРМ LanMon имеет мультиязычную архитектуру и позволяет программировать на четырех языках: C++, Pascal, Basic, Java script. Может быть выбран один из языков программирования. Специальная директива #language позволяет переключать язык по ходу выполнения программы. Это дает возможность, например, использовать функции, написанные на C++ в программе на Basic.
Скрипт имеет стандартный языковый набор:

- Переменные;
- Константы;
- Процедуры;
- Функции (с возможностью вложенности) с переменными/постоянными/умалчиваемыми параметрами;
- Стандартные операторы и объявления (включая case, try/finally/except, with);
- Стандартные типы данных: целый, дробный, логический, символьный, строковый, многомерные массивы, множество, variant;
- Классы (с методами, событиями, свойствами, индексами и свойствами по умолчанию);

АРМ LanMon имеет одну основную программу на скриптовом языке. Ее имя и частота выполнения задаются в настройках проекта. Директива #include языка C++ или USES языка Pascal позволяет включать в текст программы другие программы или модули. Основная программа АРМ LanMon содержит все процедуры, функции и обработчики событий проекта. Проект АРМ LanMon также может содержать массу выражений на скрипте (например, в динамических объектах карты).
Скрипт является компилируемым. Основная программа АРМ LanMon и все выражения на скрипте компилируется в псевдокод на этапе начала выполнения проекта, а затем псевдокод исполняется. Этим объясняется некоторая задержка при старте АРМ LanMon.
Редактор программ имеет функции пошагового исполнения программы с просмотром значений переменных после каждого шага выполнения.
В АРМ LanMon 4 встроен механизм отчетов. Каждый отчет тоже имеет свою, отдельную программу. Программа отчета «видит» все переменные и функции, описанные в основной программе.
Все выражения на скрипте также «видят» все переменные и функции, описанные в основной программе. Основное применение выражений на скрипте - в динамическом объекте карт АРМ LanMon.
Что осталось нереализованным в скрипте для любого языка:

- отсутствуют объявления типов (records, classes) в скрипте;

- нет записей (records), указателей (pointers), множеств (sets) (однако возможно использование оператора 'IN' - "a in ['a'..'c','d']");
- нет типа shortstrings;
- нет безусловного перехода (GOTO);
- нет указателей;

Реализация языка C++ имеет свою специфику:

- нет восьмеричных констант;
- нет 'break' в операторе SWITCH (SWITCH работает подобно Pascal CASE);
- операторы '++' и '--' возможны только после переменных, т.е. '++i' не будет работать; операторы '--', '++' и '=' ничего не возвращают, т.е. 'if(i++)' не будет работать;
- все идентификаторы не чувствительны к регистру;
- константа NULL это Null из Pascal- используйте nil вместо NULL;
- при инициализации массива значениями используются квадратные скобки вместо фигурных;
- нет указателей и оператора «->». Их роль выполняют ссылки на объекты и оператор «.» используется для доступа к свойствам и методам;

Синтаксис PascalScript

Далее приведено краткое описание синтаксиса:

```
Program -> [PROGRAM Ident ';']
[UsesClause]
Block '.'
UsesClause -> USES (String/,)... ';'
Block -> [DeclSection]...
CompoundStmt
DeclSection    -> ConstSection
-> VarSection
-> ProcedureDeclSection
ConstSection -> CONST (ConstantDecl)...
ConstantDecl -> Ident '=' Expression ';'
VarSection -> VAR (VarList ';')...
VarList -> Ident/','... ':' TypeIdent [InitValue]
TypeIdent      -> Ident
-> Array
Array -> ARRAY '[' ArrayDim/','... ']' OF Ident
ArrayDim -> Expression..Expression
-> Expression
InitValue -> '=' Expression
Expression -> SimpleExpression [RelOp SimpleExpression]...
SimpleExpression -> ['-'] Term [AddOp Term]...
Term -> Factor [MulOp Factor]...
Factor -> Designator
-> UnsignedNumber
-> String
-> '(' Expression ')'
-> NOT Factor
```

-> '[' SetConstructor ']'
SetConstructor -> SetNode/','...
SetNode -> Expression ['..' Expression]
RelOp -> '>'
-> '<'
-> '<='
-> '>='
-> '<>'
-> '='
-> IN
-> IS
AddOp-> '+'
-> '-'
-> OR
-> XOR
MulOp-> '*'
-> '/'
-> DIV
-> MOD
-> AND
-> SHL
-> SHR
Designator -> ['@'] Ident ['.' Ident | '[' ExprList ']' | '(' ExprList')']...
ExprList -> Expression/','...
Statement -> [SimpleStatement | StructStmt]
StmtList -> Statement/';'...
SimpleStatement -> Designator
-> Designator ':=' Expression
-> BREAK | CONTINUE | EXIT
StructStmt -> CompoundStmt
-> ConditionalStmt
-> LoopStmt
-> TryStmt
-> WithStmt
CompoundStmt -> BEGIN StmtList END
ConditionalStmt -> IfStmt
-> CaseStmt
IfStmt -> IF Expression THEN Statement [ELSE Statement]
CaseStmt -> CASE Expression OF CaseSelector/';'... [ELSE Statement]
[';'] END
CaseSelector -> SetConstructor ':' Statement
LoopStmt -> RepeatStmt
-> WhileStmt
-> ForStmt
RepeatStmt -> REPEAT StmtList UNTIL Expression
WhileStmt -> WHILE Expression DO Statement
ForStmt -> FOR Ident ':=' Expression ToDownto Expression DO Statement

ToDownto -> (TO | DOWNTO)
TryStmt -> TRY StmtList (FINALLY | EXCEPT) StmtList END
WithStmt -> WITH (Designator/,..) DO Statement
ProcedureDeclSection -> ProcedureDecl
-> FunctionDecl
ProcedureDecl -> ProcedureHeading ';'
Block ';'
ProcedureHeading -> PROCEDURE Ident [FormalParameters]
FunctionDecl -> FunctionHeading ';'
Block ';'
FunctionHeading -> FUNCTION Ident [FormalParameters] ':' Ident
FormalParameters -> '(' FormalParam/';'... ')'
FormalParm -> [VAR | CONST] VarList

Синтаксис C++Script

Далее приведено краткое описание синтаксиса:
Program -> [UsesClause]
[DeclSection]...
CompoundStmt
UsesClause -> '#' INCLUDE (String/,)...
DeclSection -> ConstSection
-> ProcedureDeclSection
-> VarStmt ';'
ConstSection -> '#' DEFINE ConstantDecl
ConstantDecl -> Ident Expression
VarStmt -> Ident Ident [Array] [InitValue] /','...
ArrayDef -> '[' ArrayDim/','... ']'
ArrayDim -> Expression
InitValue -> '=' Expression
Expression -> SimpleExpression [RelOp SimpleExpression]...
SimpleExpression -> ['-'] Term [AddOp Term]...
Term -> Factor [MulOp Factor]...
Factor -> Designator
-> UnsignedNumber
-> String
-> '(' Expression ')'
-> '!' Factor
-> '[' SetConstructor ']'
-> NewOperator
SetConstructor -> SetNode/','...
SetNode -> Expression ['..' Expression]
NewOperator -> NEW Designator
RelOp -> '>'
-> '<'
-> '<='

-> '>='
-> '!='
-> '=='
-> IN
-> IS
AddOp-> '+'
-> '-'
-> '||'
-> '^'
MulOp-> '*'
-> '/'
-> '%'
-> '&&'
-> '<<'
-> '>>'
Designator -> ['&'] Ident ['.' Ident | '[' ExprList ']' | '(' ExprList')']...
ExprList -> Expression/','...
Statement -> [SimpleStatement ';' | StructStmt | EmptyStmt]
EmptyStmt -> ';'
StmtList -> (Statement...)
SimpleStatement -> DeleteStmt
-> AssignStmt
-> VarStmt
-> CallStmt
-> ReturnStmt
-> (BREAK | CONTINUE | EXIT)
DeleteStmt -> DELETE Designator
AssignStmt -> Designator ['+'|'-'|'*'|'/']'=' Expression
CallStmt -> Designator ['+"+'|'-"-']
ReturnStmt -> RETURN [Expression]
StructStmt -> CompoundStmt
-> ConditionalStmt
-> LoopStmt
-> TryStmt
CompoundStmt -> '{' [StmtList] '}'
ConditionalStmt -> IfStmt
-> CaseStmt
IfStmt -> IF '(' Expression ')' Statement [ELSE Statement]
CaseStmt -> SWITCH '(' Expression ')' '{' (CaseSelector)... [DEFAULT
':' Statement] '}'
CaseSelector -> CASE SetConstructor ':' Statement
LoopStmt -> RepeatStmt
-> WhileStmt
-> ForStmt
RepeatStmt -> DO Statement [';'] WHILE '(' Expression ')' ';'
WhileStmt -> WHILE '(' Expression ')' Statement
ForStmt -> FOR '(' ForStmtItem ';' Expression ';' ForStmtItem ')'

```
Statement
ForStmtItem   -> AssignStmt
-> VarStmt
-> CallStmt
-> Empty
TryStmt -> TRY CompoundStmt (FINALLY | EXCEPT) CompoundStmt
FunctionDecl          -> FunctionHeading CompoundStmt
FunctionHeading       -> Ident Ident [FormalParameters]
FormalParameters      -> '(' [FormalParam/';'...] ')'
FormalParam           -> TypeIdent (['&'] Ident [InitValue]/',')...
```

Синтаксис JScript

Далее приведено краткое описание синтаксиса:

```
Program -> Statements
Statements -> Statement...
Block -> '{' Statements '}'
ImportStmt -> IMPORT (String/,)...
VarStmt -> VAR (VarDecl/',')...
VarDecl -> Ident [Array] [InitValue]
Array -> '[' (ArrayDim/',')... ']'
ArrayDim -> Expression
InitValue -> '=' Expression
Expression -> SimpleExpression [RelOp SimpleExpression]...
SimpleExpression -> ['-'] Term [AddOp Term]...
Term -> Factor [MulOp Factor]...
Factor  -> Designator
-> UnsignedNumber
-> String
-> '(' Expression ')'
-> '!' Factor
-> NewOperator
-> '<' FRString '>'
SetConstructor -> SetNode/','...
SetNode -> Expression ['..' Expression]
NewOperator -> NEW Designator
RelOp  -> '>'
-> '<'
-> '<='
-> '>='
-> '!='
-> '=='
-> IN
-> IS
AddOp-> '+'
-> '-'
```

-> '||'
-> '^'
MulOp-> '*'
-> '/'
-> '%'
-> '&&'
-> '<<'
-> '>>'
Designator -> ['&'] Ident ['.' Ident | '[' ExprList ']' |
'(' [ExprList] ')']...
ExprList -> Expression/','...
Statement -> (AssignStmt | CallStmt | BreakStmt | ContinueStmt |
DeleteStmt | DoWhileStmt | ForStmt | FunctionStmt |
IfStmt | ImportStmt | ReturnStmt | SwitchStmt |
VarStmt | WhileStmt | WithStmt | Block) [';']
BreakStmt -> BREAK
ContinueStmt -> CONTINUE
DeleteStmt -> DELETE Designator
AssignStmt -> Designator ['+'|'-'|'*'|'/']'=' Expression
CallStmt -> Designator ['+"+'|'-"-']
ReturnStmt -> RETURN [Expression]
IfStmt -> IF '(' Expression ')' Statement [ELSE Statement]
SwitchStmt -> SWITCH '(' Expression ')' '{' (CaseSelector)... [DEFAULT
':' Statement] '}'
CaseSelector -> CASE SetConstructor ':' Statement
DoWhileStmt -> DO Statement [';'] WHILE '(' Expression ')' ';'
WhileStmt -> WHILE '(' Expression ')' Statement
ForStmt -> FOR '(' ForStmtItem ';' Expression ';' ForStmtItem ')'
Statement
ForStmtItem -> AssignStmt
-> CallStmt
-> VarStmt
-> Empty
TryStmt -> TRY CompoundStmt (FINALLY | EXCEPT) CompoundStmt
FunctionStmt -> FunctionHeading Block
FunctionHeading -> FUNCTION Ident FormalParameters
FormalParameters -> '(' [FormalParam/','...] ')'
FormalParam -> ['&'] Ident
WithStmt -> WITH '(' Designator ')' Statement

Синтаксис BasicScript

Далее приведено краткое описание синтаксиса:
Program -> Statements
Statements -> (EOL | StatementList EOL)...
StatementList -> Statement/':'...

```
ImportStmt -> IMPORTS (String/,)...
DimStmt -> DIM (VarDecl/',')...
VarDecl -> Ident [Array] [AsClause] [InitValue]
AsClause -> AS Ident
Array -> '[' ArrayDim/',... ']'
ArrayDim -> Expression
InitValue -> '=' Expression
Expression -> SimpleExpression [RelOp SimpleExpression]...
SimpleExpression -> ['-'] Term [AddOp Term]...
Term -> Factor [MulOp Factor]...
Factor -> Designator
-> UnsignedNumber
-> String
-> '(' Expression ')'
-> NOT Factor
-> NewOperator
-> '<' FRString '>'
SetConstructor -> SetNode/','...
SetNode -> Expression ['..' Expression]
NewOperator -> NEW Designator
RelOp -> '>'
-> '<'
-> '<='
-> '>='
-> '<>'
-> '='
-> IN
-> IS
AddOp-> '+'
-> '-'
-> '&'
-> OR
-> XOR
MulOp-> '*'
-> '/'
-> '\'
-> MOD
-> AND
Designator -> [ADDRESSOF] Ident ['.' Ident | '[' ExprList ']' |
'(' [ExprList] ')']...
ExprList -> Expression/','...
Statement       -> BreakStmt
-> CaseStmt
-> ContinueStmt
-> DeleteStmt
-> DimStmt
-> DoStmt
```

-> ExitStmt
-> ForStmt
-> FuncStmt
-> IfStmt
-> ImportStmt
-> ProcStmt
-> ReturnStmt
-> SetStmt
-> TryStmt
-> WhileStmt
-> WithStmt
-> AssignStmt
-> CallStmt
BreakStmt -> BREAK
ContinueStmt -> CONTINUE
ExitStmt -> EXIT
DeleteStmt -> DELETE Designator
SetStmt -> SET AssignStmt
AssignStmt -> Designator ['+'|'-'|'*'|'/']'=' Expression
CallStmt -> Designator ['+"+'|'-"-']
ReturnStmt -> RETURN [Expression]
IfStmt -> IF Expression THEN ThenStmt
ThenStmt -> EOL [Statements] [ElseIfStmt | ElseStmt] END IF
-> StatementList
ElseIfStmt -> ELSEIF Expression THEN
(EOL [Statements] [ElseIfStmt | ElseStmt] | Statement)
ElseStmt -> ELSE (EOL [Statements] | Statement)
CaseStmt -> SELECT CASE Expression EOL
(CaseSelector...) [CASE ELSE ':' Statements] END SELECT
CaseSelector -> CASE SetConstructor ':' Statements
DoStmt -> DO [Statements] LOOP (UNTIL | WHILE) Expression
WhileStmt -> WHILE Expression [Statements] WEND
ForStmt -> FOR Ident '=' Expression TO Expression [STEP Expression] EOL
[Statements] NEXT
TryStmt -> TRY Statements (FINALLY | CATCH) [Statements] END TRY
WithStmt -> WITH Designator EOL Statements END WITH
ProcStmt -> SUB Ident [FormalParameters] EOL [Statements] END SUB
FuncStmt -> FUNCTION Ident [FormalParameters] [AsClause] EOL
[Statements] END FUNCTION
FormalParameters -> '(' (FormalParam/',')... ')'
FormalParm -> [BYREF | BYVAL] VarList

Структура программы на скрипте

В APM LanMon должна быть одна программа на скрипте. Она должна храниться в файле в поддиректории .\PROGRAM\ текущего проекта. Имя файла программы задается в настройках проекта APM LanMon на вкладке «Программа».

На этой же вкладке задается и частота выполнения основной процедуры скрипта. Основная процедура скрипта выполняется постоянно только в режиме выполнения проекта. Обычно программу называют main.*, давая ей расширение в зависимости от языка программирования. Основная программа может включать другие модули и программы с помощью специальной директивы (#include для C++, uses для Pascal и т.д.). Основная программа может быть произвольного размера и содержать все функции и обработчики событий, определенные программистом.

Структура программы на PascalScript

```
#language PascalScript { опционально }

program MyProgram; { опционально }
{ раздел uses должен быть перед любыми другими разделами }
{uses 'unit1.pas', 'unit2.pas';}
{ раздел var }
var
i, j: Integer;
{ раздел const }
const
pi = 3.14159;
{ процедуры и функции }
procedure p1;
var
  i: Integer;
  { вложенная процедура }
  procedure p2;
  begin
  end;
begin
end;

{ Это основной исполняемый модуль скрипта }
begin
end.
```

Структура программы на C++Script

```
#language C++Script          // опционально

// раздел include - должен быть перед любым другим разделом, но после директивы #language C++Script
// синтаксис директивы отличается от языка C++
//#include "unit1.cpp", "unit2.cpp"

// раздел констант должен идти до описания переменных и функций
#DEFINE pi 3.14159
```

```
// раздел переменных - может быть в любом месте
int i, j = 0;

// функции
void p1()
{
}

// главная исполняемая функция main()
{
  p1();
}
```

Структура программы на JScript

```
#language JScript      // опционально

// раздел import - должен быть перед любым другим разделом
//import "unit1.js", "unit2.js"

// раздел переменных - может быть в любом месте
var i, j = 0;

// функции
function p1()
{
  //
}

// главная исполняемая функция.
p1();
for (i = 0; i < 10; i++) j++;
```

Структура программы на BasicScript

```
#language BasicScript          ' опционально

' раздел imports - должен быть перед любым другим разделом
'imports "unit1.vb", "unit2.vb"

' раздел переменных - может быть в любом месте
dim i, j = 0

' функции
sub p1()
  ' Добавте сюда ваш код
end sub
```

```
' главная исполняемая функция.
print( "Hello !" )
for i = 0 to 10
  p1()
next
```

Типы данных

Скрипты в APM LanMon работают с типом Variant и основаны на нём. Тем не менее, можно использовать следующие предопределённые типы в скриптах:

Целочисленные четыре байта со знаком (-2 147 483 648 … +2 147 483 647)

Byte
Word
Integer
Longint
Cardinal
TColor

Логические

Boolean

С плавающей точкой восьмибайтовые

Real
Single
Double
Extended
Currency
TDate - дата
TTime - время
TDateTime – дата и время

Строковый

Char
String

Вариантный тип

Variant
Pointer

Массив

Array

Соответствие некоторых типов C++Script стандартным типам:

Наименование типа C++Script	Наименование стандартного типа
int	Integer
long	Integer
void	Integer
float	Extended
double	Extended

JScript не имеет описаний типов - все типы являются Variant.

BasicScript может использовать описание типов (например: **dim i as Integer**), а может опускать тип или даже объявление переменной. В этом случае она считается типом Variant.

Помимо встроенных типов, вы можете использовать перечислимые типы, объявленные в вашем приложении или в дополнительных модулях.

Приведение типов

Не все из типов данных скрипта могут быть неявно приведены один к другому. Вы не можете привести Extended или String к Integer. Только один тип Variant - может быть присвоен любому типу и получить значение от любого типа. При попытке недопустимого приведения типа будет сгенерировано исключение. Если исключение не будет поймано в блоке try {} except {}, то выполнение программы будет прервано с ошибкой.
Есть специальные функции, которые производят преобразования типа из любого значения в указанный тип без генерации исключений:

Double AsFloat(Variant v);
Integer AsInteger(Variant v);
String AsString(Variant v);
Variant AsVariant(Variant v);
Boolean AsBool(Variant v);

Эти очень удобны при использовании в выражениях на скрипте.

Классы

Вы не можете объявить собственный класс объекта в программе на скрипте, но вы можете использовать классы, которые АРМ LanMon экспортирует в скрипт. Например, TChannel – это класс канала LanMon. АРМ LanMon экспортирует класс TChannel в скрипт. Скрипт уже «знает» описание этого класса, все его свойства и методы.
Далее следует описание объектов АРМ LanMon и классов их описывающих. Описание методов и свойств приводится на языке Pascal.

Объект «Вывод звука»

Предоставляет доступ к звуковой подсистеме АРМ LanMon. Это системный объект он доступен по имени: Sound: Class Tsound

Свойство/Метод	Значение
property Busy: Boolean	false-сейчас тишина; True-сейчас что-либо проигрывается Только для чтения.
procedure Stop	Очередь проигрывания очищается и проигрывание останавливается.
function Play(File: String): Boolean	Поставить в очередь на проигрывание звуковой файл File. Причем если файл задан без пути – он будет искаться в поддиректории для звуков в папке проекта .\Wav.

Объект «Канал тип 1»

Канал тип 1 представлен классом TChannel. Доступ к каналу возможен через список каналов тип 1. Динамически создать объект TChannel нельзя.

Свойство/Метод	Значение
property A1: Integer *property A2: Integer* *property A3: Integer* *property A4: Integer*	Адрес канала LanMon, ассоциированного с данным объектом. Адрес состоит из 4х цифр, каждая в диапазоне от 1 до 65535. Для чтения и записи.
property Addr: String	Адрес канала в виде текстовой строки. Элементы адреса разделены символом «.» (точка). Только для чтения.
property AddrText: String	Синоним Addr
property NAME: String	Название канала уровня A4. Только для чтения.
property DTYPE: Integer	Тип данных канала (тип значения свойства VALUE). Расшифровку смотрите в приложении «Расшифровка значений типа канала (свойство DTYPE)».Только для чтения.
property DT: TDateTime	Метка времени: время последнего изменения свойств STATE, VALUE или Values. Для чтения и записи.
property TIME: TDateTime	Синоним для DT
property Quality: Integer	Качество (состояние) канала LanMon. Расшифровку смотрите в приложении «Расшифровка значений состояния канала тип 1 (свойства Quality, STATE)». Свойства VALUE и Values имеют смысл только когда STATE равен нулю (т.е. состояние канала ОК). Для чтения и записи.
property STATE: Integer	Синоним для Quality
property VALUE: Variant	Значение канала. Тип значения зависит от типа канала и от значения Quality. Для чтения и записи.
property ValCount: Integer	Количество значений для данного типа канала. Некоторые типы каналов несут более одного значения. Например DTYPE 24 имеет пять значений. Для большинства типов каналов равно 1. Только для чтения.
index property Values(p0: Integer): Variant	Массив значений канала: от Values[0] до Values[ValCount-1]. Примечание Values[0] и VALUE это одно и то же. Для чтения и записи.
property Owner: Integer	Идентификатор источника последнего значения и/или качества канала (ID учетной записи сервера LanMon или драйвера). Только для чтения.

Свойство/Метод	Значение
property ID: Integer	Синоним для Owner
property SYSM: Integer	Подсистема, к которой относится данный канал. Расшифровку смотрите в приложении «Расшифровка кода системы (свойство SYSM)». Только для чтения.
property PARAM: Integer	Единица измерения значения, которое несет данный канал. Расшифровку смотрите в приложении « Расшифровка единицы измерения значения канала (свойство PARAM)». Только для чтения.
function AddrIs(A1,A2,A3,A4: Word): Boolean	Функция осуществляет сравнения адреса канала с указанной маской каналов A1.A2.A3.A4 Значение «0» означает любой адрес. Функция возвращает значение **true** если адрес канала соответствует маске и **false** в противном случае.
function Format(mask: String): String	Сформатировать параметры канала тип 1 в текстовую строку. Возвращает полученную строку. mask - маска форматирования. Может содержать произвольный текст и следующие подстановки: **%DATE** – дата изменения значения канала в формате DD:MM **%TIME** – время изменения значения канала в формате HH:MM:SS **%DT** – дата и время в формате, заданном в системе **%DT(формат) -** дата и время в заданном формате «формат». Описание формата смотрите в приложении «Маска форматирования для функции FormatDateTime» **%IDT** – дата и время интеллектуальном формате: Если сегодня, то выводится только время в формате, заданном в системе. Если не старше 7 дней, то выводится название дня недели + время в формате, заданном в системе. В противном случае выводятся дата и время в формате, заданном в системе. **%ADDR** – текстовый адрес канала **%A1** – текстовое наименование канала уровня A1 **%A2** – текстовое наименование канала уровня A2 **%A3** – текстовое наименование канала уровня A3 **%A4** – текстовое наименование канала уровня A4 **%COMMENTS** – полное текстовое наименование канала **%STATE** – текстовое описание состояния канала **%VALUEn(%mask)** – значение канала. n – номер значения с нуля (только для массивов). %mask – маска форматирования значения канала. Описание формата маски смотрите в приложении «Маска форматирования для функций Format и sprintf». **%NL** – вставка символов новой строки
function SendControl(v: Variant): Boolean	Послать команду управления драйверам оборудования и на сервер LanMon. Функция получает значение для отправки. Если не

Свойство/Метод	Значение
	удалось отправить команду на сервер и драйверам возвращает false.
function SendState(STATE: Word; v: Variant): Boolean	Послать состояние канала на сервер LanMon подобно тому, как это делает опросчик. Функция получает новые состояние и значение канала. При ошибке возвращает false.
function SendStateLocal(STATE: Word; v: Variant): Boolean	Послать состояние канала самому себе. Подобно тому, как будто бы оно пришло от сервера или от драйвера оборудования. Функция получает новые состояние и значение канала. При ошибке возвращает значение false. Ошибка может возникнуть в двух случаях: - такого канала нет в дереве - произошла ошибка преобразования типов параметра v (например, канал целочисленный, а в функцию передана строка)

Объект «Список каналов тип 1»

Список каналов тип 1 представлен классом TRTV. Определена глобальная переменная RTV: class TRTV, которая служит для доступа к списку каналов.

Свойство/Метод	Значение
index property Items(p0: Integer): TCannel	Ссылка на объекты канал TChannel. Только для чтения. Канал доступен по индексу.
property Count: Integer	Количество объектов в свойстве Items[]. Максимальный индекс Items[Count-1]. Только для чтения.
function Get(A1,A2,A3,A4: Word): TChannel	Получение ссылки на канал по его адресу. Если такого канала нет в списке, возвращается nil.
function Get2(addr: String): TChannel	Получение ссылки на канал по его адресу. Адрес закодирован в строке в формате «A1.A2.A3.A4». Если такого канала нет в списке, возвращается nil.

Объект «Канал тип 2»

Канал тип 2 представлен классом TChannel2. Доступ к каналу возможен через список каналов тип 2.

Свойство/Метод	Значение
property Active: Boolean	Активность канала. Если значение False, то данные по каналу не идут. Только для чтения. Для установки или снятия активности канала следует использовать функцию *RegisterChannelActive2*.
property Addr: String	Текстовый адрес канала. Только для чтения.
property ID: String	То же что и Addr.
property Number: Integer	Уникальный номер канала с 1. Только для чтения.
property Type: Integer	Тип данных значения канала. Только для чтения.
property TypeText: String	Текстовое описание типа данных канала. Только для чтения.
property DT: TDateTime	Метка времени значения канала: время последнего изменения свойств Quality, Value. Только для чтения.

Свойство/Метод	Значение
property Quality: Integer	Качество (состояние) канала. Расшифровку смотрите в приложении «Расшифровка значений состояния канала тип 2 (свойство TChannel2:Quality)» Свойства Value, Values имеют смысл только когда Quality равен нулю (т.е. состояние канала ОК). Только для чтения.
property QualityText: String	Текстовая расшифровка значения Quality. Только для чтения.
property Unit: String	Единицы измерения (атрибут «Единицы» id 100). Только для чтения.
property Value: Variant	Значение канала. Тип значения определяется Type. Актуально только при Quality = 0. Для установки значения канала следует использовать функцию *RegisterChannelValue2*.
property ValueCount: Integer	Кол-во значений канала. Больше 1 для типов «массив». Только для чтения.
index property Values(p0: Integer): Variant	Массив значений канала. Индекс от 0 до ValueCount-1.
property ValuesText: String	Текстовое представление значения канала. Только для чтения.
property FormatString: String	Маска форматирования значения канала измерения (атрибут «Формат» id 5002). Описание формата маски смотрите в приложении «Маска форматирования для функций Format и sprintf». Только для чтения.
property Prefix: String	Префикс описания канала. Только для чтения.
property Comments: String	Текстовое описание канала. Только для чтения.
property FullComments: String	Prefix + Comments
property Signification: Integer	Назначение (физический смысл) канала. Описание смотрите в приложении «Расшифровка назначения канала тип 2 (свойство TChannel2:Signification)». Только для чтения.
property SignificationText: String	Текстовая расшифровка Signification. Только для чтения.
property Creator: Integer	Идентификатор создателя канала. Значение меньше нуля означает идентификатор драйвера (ID). Ноль – сам сервер LanMon. Значение больше нуля означает идентификатор одной из учетных записей сервера LanMon (ID). Только для чтения.
property Owner: Integer	Идентификатор источника последнего значения и/или качества канала. Расшифровка значений как у свойства Creator. Только для чтения.
property WriteEnable: Boolean	True – запись значения в контроллер для изменения уставок или управления работой разрешена. Только для чтения. Для отправки команды управления драйверам или на сервер LanMon следует использовать функцию *RegisterChannelValue2*.
property EUType: Integer	Интерпретация значений канала. 0 – Нет информации о характере данных в канале.

Свойство/Метод	Значение
	1 – Аналоговая величина — атрибут EUInfo1 присутсвует и содержит массив из 2х значений типа (VT_VECTOR | VT_R8) соответствующих нижней и верхней границе диапазона измерения. Минимальное и максимальное значения, которые могут быть выданы аппаратурой. Используется, например, при построении графиков. 2 – Перечисление - атрибут EUInfo2 присутсвует и содержит массив строк (VT_VECTOR | VT_STRING). (Пример: атрибут EUInfo2 содержит список строк “Открыто”, “Закрыто”, “Сломано”, который соответствует последовательным значениям канала (0, 1, 2 …). 3 – Ссылка на набор состояний внутри APM LanMon. EUInfo3 присутсвует и содержит номер набора состояний APM LanMon, тип значения VT_I4. Только для чтения.
property EUInfo1: Variant	см. описание EUType
property EUInfo2: Variant	см. описание EUType
property EUInfo3: Integer	см. описание EUType
property IsFromDriver: Boolean	True – источник последнего значения канала драйвер. Только для чтения.
property IsFromNetwork: Boolean	True – источник последнего значения канала одна из учетных записей сервера LanMon. Только для чтения.
property IsFromServer: Boolean	True – источник последнего значения канала сервер LanMon. Только для чтения.
property WatchLimits: Boolean	True - включить аналоговый аларм для данного канала с параметрами: HiHiLimit, HiLimit, Deadband, LoLimit, LoLoLimit, AlarmMessage, SoundFile. Только для чтения.
property HiHiLimit: Variant	см. описание WatchLimits
property HiLimit: Variant	см. описание WatchLimits
property Deadband: Variant	см. описание WatchLimits
property LoLimit: Variant	см. описание WatchLimits
property LoLoLimit: Variant	см. описание WatchLimits
property AlarmMessage: String	см. описание WatchLimits
property SoundFile: String	см. описание WatchLimits
property AttributeCount: Integer	Кол-во атрибутов канала. Только для чтения.
index property Attributes(p0: Integer): TChannelAttribute	Получить ссылку на объект – атрибут канала тип 2 по индексу. Индекс от 0 до AttributeCount-1. Удалять объект после получения ссылки не требуется. Описание класса TChannelAttribute см. ниже. Только для чтения.

Свойство/Метод	Значение
	Для установки значения атрибута следует использовать функцию *RegisterAttributeValue2*.
index property AttributeByID(AttrID: Word): TChannelAttribute	Получить ссылку на объект – атрибут канала тип 2 по идентификатору атрибута. Удалять объект после получения ссылки не требуется. Только для чтения.
index property AttributeValueByID(AttrID: Word): Variant	Получить значение атрибута канала тип 2 по идентификатору атрибута. Тип значения определяется описанием атрибута с данным идентификатором. Только для чтения.
function Format(mask: String): String	Сформатировать параметры канала тип 2 в текстовую строку. Возвращает полученную строку. mask - маска форматирования. Может содержать произвольный текст и следующие подстановки: **%DATE** – дата изменения значения канала в формате DD:MM **%TIME** – время изменения значения канала в формате HH:MM:SS **%DT** – дата и время в формате, заданном в системе **%DT(формат) -** дата и время в заданном формате «формат». Описание формата смотрите в приложении «Маска форматирования для функции FormatDateTime» **%ADDR** – текстовый адрес канала **%COMMENTS** – полное текстовое наименование канала **%STATE** – текстовое описание состояния канала **%VALUEn(%mask)** – значение канала. n – номер значения с нуля (только для массивов). %mask – маска форматирования значения канала. Описание формата маски смотрите в приложении «Маска форматирования для функций Format и sprintf». **%UNIT** – единицы измерения **%ATTR(AttrID)** – значение атрибута с идентификатором AttrID **%MASK** – признак маскирования одного из алармов, привязанных к данному каналу **%NL** – вставка символов новой строки

Объект «Атрибут канала тип 2»

Атрибут канала тип 2 представлен классом TChannelAttribute. Доступ к атрибутам возможен через свойства Attributes и AttributeByID объекта TChannel2. У канала может быть от 0 до нескольких десятков атрибутов. Применяемый атрибут должен быть предварительно определен на сервере LanMon (задается идентификатор, тип данных и т.д.). У одного канала некоторые атрибуты могут создаваться драйвером, а некоторые оператором.

Свойство/Метод	Значение
property ID: Integer	Числовой идентификатор атрибута (больше ноля). Только для чтения.
property Type: Integer	Тип данных значения атрибута. Только для чтения.
property TypeText: String	Текстовое описание типа данных атрибута. Только для чтения.

Свойство/Метод	Значение
property Value: Variant	Значение атрибута. Тип значения определяется Type. Только для чтения.
property ValueText: String	Текстовое представление значения атрибута Value. Только для чтения.
property Owner: Integer	Идентификатор источника последнего значения атрибута. Значение меньше ноля означает идентификатор драйвера (ID). Ноль – сервер LanMon. Значение больше ноля означает одну из учетных записей сервера LanMon. Только для чтения.
property LastChange: TDateTime	Дата время последнего изменения значения атрибута. Только для чтения.
property LastChangeText: String	Текстовое представление LastChange. Только для чтения.

Объект «Список каналов тип 2»

Список каналов тип 2 представлен классом TChannelList2. Определена глобальная переменная ChannelList2: class TChannelList2, которая служит для доступа к списку каналов.

Свойство/Метод	Значение
property Count: Integer	Количество каналов тип 2 (объектов в свойстве ChannelByIndex []). Только для чтения.
index property ChannelByIndex (p0: Integer): TCannel2	Получение канала тип 2 по индексу от 0 до Count-1 (экземпляра объекта типа TChannel2). После получения канала его нужно удалить оператором delete. Только для чтения.
index property ChannelByAddr (p0: String): TCannel2	Получение канала тип 2 (экземпляра объекта типа TChannel2) по адресу. После получения канала его нужно удалить оператором delete. Только для чтения.
index property ChannelByID (p0: String): TCannel2	Получение канала тип 2 (экземпляра объекта типа TChannel2) по адресу. После получения канала его нужно удалить оператором delete. Только для чтения. То-же, что и ChannelByAddr.
index property ChannelByNumber (p0: Cardinal): TCannel2	Получение канала тип 2 (экземпляра объекта типа TChannel2) по номеру. После получения канала его нужно удалить оператором delete. Только для чтения.
property ActiveCount: Integer	Количество активных каналов тип 2. Только для чтения.
function SetUserValue(ID: String; Number: Cardinal; Value: String): Boolean	Установить значение поля TChannel2::UserValue для канала с адресом ID или номером Number.

Пример работы со списком каналов тип 2 на C++ скрипте (фрагмент кода):

```
// делаем активными все неактивные каналы
try
{
  TChannel2 p;
  int count = ChannelList2.Count;
  for(int i=0; i<count; i++)
```

```
  {
   // Создание нового объекта p – ссылки на канал тип 2
   p = ChannelList2.ChannelByIndex[i];
   if( ! p.Active )
     RegisterChannelActive2(p.ID, p.Number, true);
   // Удаление объекта p после использование обязательно !
   delete p;
  }
 }
 except
 {
  LMProtokol("ERROR (" + ExceptionClassName + ") Message: " + ExceptionMessage);
 }
```

Объект «Графический объект карты»

Базовый класс для всех графических объектов карты *TMonControl*. К каждому объекту карты можно привязать канал тип 1 или тип 2. Поэтому часть свойств у объекта карты совпадает со свойствами классов *TChannel, TChannel2*. Доступ к объектам карты возможен через список объектов карты (класс *TMap*) или через создание именованного объекта карты (через окно свойств объекта). Класс *TMonControl* имеет следующие свойства и методы:

Свойство/Метод	Значение
property A1: Integer *property A2: Integer* *property A3: Integer* *property A4: Integer*	Адрес канала тип 1, привязанного к объекту. Адрес состоит из 4х цифр, каждая в диапазоне от 1 до 65535. Только для чтения. Если канал не назначен все эти свойства будут равны нулю.
property Addr: String	Адрес канала тип 1 или тип 2, привязанный к объекту. Только для чтения.
property AddrType: Integer	Тип адрес канала, привязанный к объекту: 0-адрес не задан 1-«тип 1» 2-«тип 2». Только для чтения.
property Comments: String	Канал тип 1: текстовое описание. Канал тип 2: текстовое описание. Только для чтения.
property NAME: String	Канал тип 1: название канала уровня A4. Канал тип 2: текстовое описание. Только для чтения.
property DTYPE: Integer	Тип значения для канала тип 1. Расшифровку смотрите в приложении «Расшифровка значений типа канала (свойство DTYPE)». Только для чтения.
property TIME: Extended	Дата и время последнего изменения значения или качества канала. Время представлено типом TDateTime (double). Для чтения и записи.
property DT: Extended	Синоним TIME.
property Quality: Integer	Качество (состояние) канала, привязанного к объекту. Расшифровку смотрите в приложении «Расшифровка значений состояния канала тип 1 (свойства Quality, STATE)». Свойства

Свойство/Метод	Значение
	VALUE и Values имеют смысл только когда Quality равен нулю (т.е. состояние канала ОК). Для чтения и записи.
property State: Integer	Синоним Quality.
property QualityText: String	Текстовое описание текущего качества канала, привязанного к объекту карты.
property VALUE: Variant	Значение канала тип 1 или тип 2, привязанного к объекту карты. Тип значения зависит от типа канала и от текущего значения Quality. Для чтения и записи.
property ValCount: Integer	Количество значений для тип данных «массив». Для большинства типов каналов равно 1. Только для чтения.
index property Values(p0: Integer): Variant	Массив значений канала тип 1 или тип 2, привязанного к объекту карты: от Values[0] до Values[ValCount-1]. Примечание Values[0] и VALUE это одно и то же. Для чтения и записи.
property Signification: Integer	Назначение (физический смысл) канала тип 2. Описание смотрите в приложении «Расшифровка назначения канала тип 2 (свойство TChannel2:Signification)». Только для чтения.
property SignificationText: String	Текстовая расшифровка Signification канала тип 2. Только для чтения.
property ValueOwner: Integer	Идентификатор источника последнего значения и/или качества канала тип 1 или тип 2, привязанного к объекту карты (ID учетной записи сервера LanMon или драйвера). То-же что и TChannel::Owner и TChannel2::Owner. Только для чтения.
property SYSM: Integer	Подсистема, к которой относится данный канал тип 1. Только для чтения.
property Text: String	Заголовок (текстовая подпись) объекта карты.
property LMHINT: String	Всплывающая подсказка объекта карты.
property Type: Integer	Тип графического объекта карты: 1 TMonStd 2 TMonText 3 TMonShape 4 TMonImage 8 TMonGraph 9 TMonGauge 11 TBasText 12 TMonList 13 TFRText 14 TMonActiveX 15 TMonRegion 16 TMonActionButton 17 TMonButton 18 TMonOnOff 19 TMonMaps 20 TMonMapFlag 21 TMonTrackBar 22 TMonGraph2

Свойство/Метод	Значение
property x: Integer *property y: Integer*	Координаты центра объекта на карте. Для чтения и записи. Изменяя можно двигать объект.
property Width: Integer *property Height: Integer*	Ширина и высота объекта. Для чтения и записи. Изменяя можно менять размеры объекта.
property Sound: String	Имя звукового файла, назначенного данному объекту.
property Visible: Boolean	Объект виден сейчас ? Можно управлять видимостью объекта. Для чтения и записи.
property Masked: Boolean	Значение ***true*** означает, что алармы для канала, привязанного к данному объекту карты замаскированы. Замаскированные алармы не выдают тревожные срабатывания (оповещения оператору). Маскирование отображается появлением значка «СТОП» на объекте карты. Для чтения и записи.
property Map: Class TMap	Ссылка на объект карты, на которой данный объект располагается. Только для чтения.
property MapIndex: Integer	Индекс карты в списке, на которой данный объект располагается. Только для чтения. Для доступа к объекту карты используется выражение *Maps.Items[<ссылка на объект карты>.MapIndex]*
property Canvas: Class TCanvas	Используется для пользовательского рисования (Custom Draw).
procedure Invalidate	Принудительная перерисовка, объекта карты. Применяется после использования свойства Canvas для рисования.
procedure SendToBack	Помещение объекта на задний план, под остальные объекты.
procedure BringToFront	Помещение объекта на передний план, поверх остальных объектов.

Объект «Набор картинок»

Объект представлен классом TMonStd. Базовый класс для этого объекта TMonControl. Это означает что все свойства и методы TMonControl доступны в полном объеме. Класс имеет следующие свойства и методы:

Свойство/Метод	Значение
property LibraryIndex: integer	Номер библиотеки картинок из которой нужно брать картинку. Используется для отображения картинки в ручном режиме (из программы на скрипте). Для чтения и записи.
property ImageIndex: integer	Индекс картинки в библиотеке картинок с нуля. Меняя индекс картинки можно менять отображаемую картинку объекта. Для чтения и записи.
property ImageCount: integer	Количество картинок для отображения. Если установлено значение больше 1, то производится автоматическая смена отображаемой картинки от ImageIndex до ImageIndex+ ImageCount. Смена картинки производится через каждые 200 мсек.

Объект «Картинка»

Объект представлен классом TMonImage. Базовый класс для этого объекта TMonControl. Класс имеет следующие свойства и методы:

Свойство/Метод	Значение
property BitmapFile: String	Имя файла картинки BMP. Если путь не задан, то берется файл из поддиректории .\BMP\ проекта. Для чтения и записи.
property Bitmap: Class TBitmap	Указатель на текущую отображаемую картинку. Только для чтения. Типовое использование – самостоятельная прорисовка содержимого картинки из программы.

Объект «Текст»

Объект представлен классом TMonText. Базовый класс для этого объекта TMonControl. Это означает что все свойства и методы TMonControl доступны в полном объеме. Класс имеет следующие свойства и методы:

Свойство/Метод	Значение
property Text: String	Текст для вывода в объекте. Возможно применение подстановок. Для чтения и записи.
property Caption: String	То же что и свойство Text.
property Color: TColor	Цвет заливки области текста.

Объект «Фигура»

Объект представлен классом TMonShape. Базовый класс для этого объекта TMonControl. Класс имеет следующие свойства и методы:

Свойство/Метод	Значение
property ShapeType: Integer	Тип фигуры 1-эллипс 2-прямоугольник. Для чтения и записи.
property Color: TColor	Цвет заливки области фигуры.

Объект «Прогресс бар» (Полоса заливки)

Объект представлен классом TMonGauge. Базовый класс для этого объекта TMonControl. Класс имеет следующие свойства и методы:

Свойство/Метод	Значение
property MinValue: Extended	(Тип Extended в Pascal это аналог double в C++) Минимальное значение (соответствует отсутствию заливки). Сохраняется в файле карты. Для чтения и записи.
property MaxValue: Extended	Максимальное значение (соответствует полностью залитому прогресс бару). Сохраняется в файле карты. Для чтения и записи.
property CurValue: Extended	Текущее значение заливки (должно располагаться между минимальным и максимальным значениями). Для чтения и записи.
property ForeColor: Integer	Цвет заливки. Сохраняется в файле карты. Для чтения и записи.

property BackColor: Integer	Цвет фона прогресс бара. Сохраняется в файле карты. Для чтения и записи.
property Text: String	Текст для вывода. Возможно применение подстановок. Для чтения и записи.

Объект «Список»

Объект представлен классом *TMonList*. Базовый класс для этого объекта *TMonControl*. Все свойства и методы базового класса наследуются. *TMonList* имеет следующие свойства и методы:

Свойство/Метод	Значение
property ItemsCount: Integer	Количество строк в настоящее время с списке.
property ItemIndex: Integer	Номер текущей выделенной строки в списке с нуля. Если ни одна строка не выделена, то значение свойства: -1
property List : TListBox	Ссылка на объект списка.

Объект «Область»

Объект представлен классом *TMonRegion*. Базовый класс *TMonControl*. Все свойства и методы базового класса наследуются. *TMonRegion* имеет следующие свойства и методы:

Свойство/Метод	Значение
property Caption: String	Текст, отображаемый в области. Для чтения и записи.
property Text: String	То же.
property FrameColor: TColor	Цвет окантовки. Для чтения и записи.
property FillColor: TColor	Цвет заливки. Для чтения и записи.
property VisualSelect: bool	Выделять объект при наведении курсора мыши? Для чтения и записи.

Объект «Выключатель»

Объект представлен классом *TMonOnOff*. Базовый класс *TMonControl*. Все свойства и методы базового класса наследуются. *TMonOnOff*имеет следующие свойства и методы:

Свойство/Метод	Значение
property On: bool	True – включен, False - выключен

Объект «Ползунок»

Объект представлен классом *TMonTrackBar*. Базовый класс *TMonControl*. Все свойства и методы базового класса наследуются. *TMonTrackBar* имеет следующие свойства и методы:

Свойство/Метод	Значение
property CurValue: Extended	Текущее положение указателя.

Объект «ActiveX»

Объект представлен классом *TMonActiveX*. Базовый класс для этого объекта *TMonControl*. Все свойства и методы базового класса наследуются. *TMonActiveX* имеет следующие свойства и методы:

Свойство/Метод	Значение
function Call(Name:String; Type:String; Params: Variant): Variant	Выполнить функцию или вызвать свойство объекта ActiveX. *Name* – Имя функции или свойства. *Type* – Строка, обозначающая тип вызываемого объекта. Может быть одним из следующих: "**FUNC**" – *Name* это имя функции "**PROPERTYGET**" - *Name* это имя свойства, возвращающего значение "**PROPERTYPUT**" - *Name* это имя свойства, принимающего значение *Params* – массив входных параметров для вызова функции или свойства Возвращаемое значение функции ***Call(…)*** это то, что возвращает вызванная функция объекта или свойство ***PROPERTYGET***.
procedure OnActiveXEvent(Sender: TMonActiveX; MessageType: Integer; var Params: Variant)	Обработчик событий от объекта ActiveX. Параметры: *Sender* - ссылка на объект карты TMonActiveX, от которого пришло событие; *MessageType* - идентификатор события (можно посмотреть в списке функций для данного ActiveX. В скрипте определены константы идентификаторов типовых событий: DISPID_CLICK DISPID_DBLCLICK DISPID_KEYDOWN DISPID_KEYPRESS DISPID_KEYUP DISPID_MOUSEDOWN DISPID_MOUSEMOVE DISPID_MOUSEUP DISPID_READYSTATECHANGE *Params* - массив параметров события. Используется и для выдачи параметров из обработчика.

В проекте «Пример работы с CamControl» иллюстрируется вызов функций ActiveX объекта и обработка событий.

Объект «Карта»

Карта представлена классом *TMap*. Чтобы карта стала доступна в скрипте, в параметрах карты задайте имя объекта карты. Все свойства и методы карты будут доступны через это имя. Альтернативный способ доступа к объекту карты – через список карт *Maps* (класс *TMaps*). Динамически создать карту (объект *TMap*) нельзя.

Свойство/Метод	Значение
property Addr: String	Адрес канала тип 1 или тип 2, привязанный к объекту. Только для чтения.
property Ohrana: boolean	Состояние охраны карты. Для чтения и записи.
property Name: String	Название карты, отображаемое в ее заголовке. Для чтения и записи.
property File: String	Файл карты. Только для чтения.
property Bitmap: String	Имя файла с подложкой карты в формате BMP. Для чтения и записи. Если путь не задан – подразумевается, что подложка находится в поддиректории.\BMP\ данного проекта.
property Sound: String	WAV файл, назначенный данной карте в параметрах карты. Для чтения и записи. Если путь не задан – подразумевается, что файл находится в поддиректории.\WAV\ данного проекта.
property Left: Integer *property Top: Integer* *property Width: Integer* *property Height: Integer*	Координаты карты на экране. Для чтения и записи. Координата 0,0 это левый верхний угол экрана.
property MapIndex: Integer	Индекс карты в массиве карт списка карт (Maps.Items[MapIndex]). Только для чтения.
property Tag: Integer	Может использоваться программистом произвольным образом. Для чтения и записи.
index property Items[p0: Integer]: TMonControl	Список всех графических объектов карты (базовый класс TMonControl). Только для чтения.
property Count: Integer	Количество объектов в свойстве Items[]. Максимальный индекс Items[Count-1]. Только для чтения.
procedure Show	Сделать карту видимой и выдвинуть ее на первый план. И, при необходимости, развернуть из панели задач.
procedure Hide	Сделать карту невидимой.
property Visible: Boolean	Карта видна сейчас? Примечание: карта может быть видна (Visible=true), но перекрыта другой картой или свернута в панель задач. Вызов процедура Show развернет карту и выдвинет ее на первый план в любом случае. Для чтения и записи.

Объект «Список карт»

Все карты в АРМ LanMon принадлежат общему глобальному списку. Список представлен классом TMaps. Для доступа к списку карт существует глобальная переменная Maps: Class Tmaps.

Свойство/Метод	Значение
index property Items[p0: Integer]: TMap	Ссылка на объекты карт TMap. Только для чтения. Карта доступна по индексу.
property Count: Integer	Количество объектов в свойстве Items[]. Максимальный индекс Items[Count-1]. Только для чтения.

Объект «Группа каналов»

Группа каналов содержит список каналов. Они используются, например, для создания охранных зон. Группы создаются в редакторе групп. В программе на скрипте доступен ряд свойств группы каналов. Доступ к группе каналов из программы возможен только через список групп каналов. Группа описывается классом TGroup.

Свойство/Метод	Значение
property Name: String	Название группы, заданное при ее создании в редакторе групп каналов. Только для чтения.
property Neisprav: boolean	Есть хоть один неисправный канал в группе ? Канал считается неисправным, когда его качество STATE>2. Только для чтения.
property Ohrana: boolean	Состояние охраны группы каналов. Для чтения и записи. После запуска проекта все группы каналов сняты с охраны.
property State: Integer	Состояние группы: 0-все спокойно: нет срабатываний, нет тревог 1-есть срабатывание хотя бы по одному каналу в группе 2-есть тревога хоть по одному каналу в группе Срабатывание и тревога определяются по дискретному аларму, назначенному группе в редакторе групп. Только для чтения.
property Tag: Integer	Пользовательское свойство. Может использоваться произвольным образом. Для чтения и записи.

Объект «Список групп каналов»

Все группы каналов принадлежат общему глобальному списку. Список представлен классом TGroups. Для доступа к списку групп каналов существует глобальная переменная Groups.

Свойство/Метод	Значение
index property Items(index: integer): TGroup	Ссылка на объекты группа каналов TGroup. Группа каналов доступна по индексу. Только для чтения. Index может изменяться от 0 до Count-1
property Count: Integer	Текущее количество групп каналов. Только для чтения.
function GetByAddr(A1,A2,A3,A4: Word): TGroup	Получить ссылку на группу каналов по адресу канала, входящего в группу. Если группа с указанным каналом не найдена будет возвращено значение nil. Пример: *TGroup p;* *p = Groups.GetByAddr(7,1,1,11);* *if(p)* * print("Группа найдена: " + p.Name);* *else* * print("Группа с каналом 7.1.1.11 НЕ найдена !");*
function GetByGroupAddr(A1,A2,A3,A4 : Word): TGroup	Получить ссылку на группу каналов по адресу канала, привязанного к группе. Если группа с указанным каналом не найдена будет возвращено значение nil.

Объект «Абонент IP телефонии»

Объект представлен классом ***TIPAbonent***. Класс имеет следующие свойства и методы:

Свойство/Метод	Значение
property Name: String	Имя абонента. Для чтения и записи.
property FullAddress: String	Адрес абонента в формате H.323: [alias@][transport$]host[:port] . Для чтения и записи.
property Tag1: String	Дополнительный параметр 1. Для свободного использования программистом. Для чтения и записи.
property Tag2: String	Дополнительный параметр 2. Для свободного использования программистом. Для чтения и записи.

Объект «Список абонентов IP телефонии»

Объект представлен классом ***TIPAbonentList***. Ссылка на экземпляр этого класса ***IPAL*** всегда доступна из программы на скрипте. Класс имеет следующие свойства и методы:

Свойство/Метод	Значение
property Count: Integer	Количество абонентов в списке. Только для чтения.
property CurIndex: Integer	Номер текущего абонента в списке от 0 до (Count-1). Значение -1 говорит о том, что ни один абонент не выбран в качестве текущего. При входящем звонке генерируется событие с идентификатором hm323Ring. Если удаленный абонент найден в списке абонентов, он становится текущим. Таким образом, можно понимать от кого идет звонок. Для чтения и записи.
index property Items(index: Integer): TIPAbonent	Ссылка на абонента IP телефонии класс TIPAbonent по индексу от 0 до (Count-1). Только для чтения.
function Add(p: TIPAbonent): Boolean	Функция добавления абонента в список. Возвращаемое значение ***true*** если абонент успешно добавлен в список и ***false*** в противном случае.
procedure Delete(index: Integer)	Процедура удаления абонента из списка. Получает один параметр – номер абонента с нуля.
procedure Clear	Процедура удаления всех абонентов из списка.
function Save: Boolean	Функция сохранения списка абонентов на диске. Возвращаемое значение ***true*** если список абонентов сохранен успешно и ***false*** если при сохранении произошла ошибка.

В следующем фрагмента программы на C++ скрипте производится вывод списка абонентов IP телефонии в окно отладки программы:

```
print( IPAL.CurIndex );
for(int i=0; i<IPAL.Count; i++)
{
  print( sprintf("Name=\"%s\" Addr=%s Tag1=%s Tag2=%s",
        IPAL.Items[i].Name,
```

```
        IPAL.Items[i].FullAddress,
        IPAL.Items[i].Tag1,
        IPAL.Items[i].Tag2
        ) );
}
```

Объект «Запись архива звукозаписей»

Архив звукозаписей состоит из записей. Каждая запись представлена классом TAudioArchiveRec. Ссылка на этот класс передается в качестве параметра обработчику нажатия специальной кнопки в окне просмотра звукозаписей *OnAudioArchiveButton(TAudioArchiveRec)*.

Свойство/Метод	Значение
property Name: String	название объекта с кем бал разговор
property ChannelName: String	название переговорного канала
property File: String	полное имя звукового файла
property Start: TDateTime	дата и время начала разговора (тип TDateTime)
property Length: Integer	длительность разговора в секундах
property Format: String	текстовое описание формата звукового файла

Предопределенные объекты

В скрипте определено несколько ссылок на глобальные объекты АРМ LanMon:

- Application: Class T Application – Объект для управления программой АРМ LanMon.
- Groups: Class TGroups – список групп каналов. Группа представлена классом TGroup;
- IPAL: Class TIPAbonentList – список абонентов IP телефонии. Абонент представлен классом TIPAbonent;
- Maps: Class TMaps – список карт. Карта представлена классом TMap;
- RTV: Class TRTV – список каналов. Канал представлен классом TChannel;
- Sound: Class Tsound – класс для проигрывания звуковых файлов;
- SystemADOConnection: Class TADOConnection – основное подключение к базе данных через механизм ADO. Используется для доступа к SQL серверу. По умолчанию используется всеми элементами доступа к данным в отчете. Параметры подключения настраиваются в настройках проекта;
- SystemADOConnection1: Class TADOConnection – первое дополнительное подключение к базе данных через механизм ADO. Параметры подключения настраиваются в настройках проекта;
- SystemADOConnection2: Class TADOConnection – второе дополнительное подключение к базе данных через механизм ADO. Параметры подключения настраиваются в настройках проекта;
- SystemReport: Class TfrxReport – Объект для управления генератором отчетов.

Использование этих переменных иллюстрируется в примерах программ из комплекта поставки АРМ LanMon, а также в демонстрационных проектах.

Функции общего назначения

В скрипте можно использовать богатый набор стандартных функций.

Преобразование типов

Описание	Комментарии
function IntToStr(i: Integer): String	Перевод целого в строку
function FloatToStr(e: Extended): String	Перевод числа с плавающей запятой в строку
function DateToStr(e: Extended): String	Перевод даты в строку
function TimeToStr(e: Extended): String	Перевод времени в строку
function DateTimeToStr(e: Extended): String	Перевод даты и времени в строку
function VarToStr(v: Variant): String	Перевод variant в строку
function StrToInt(s: String): Integer	Перевод строки в целое
function StrToFloat(s: String): Extended	Перевод строки в число с плавающей запятой
function StrToDate(s: String): Extended	Перевод строки в дату
function StrToTime(s: String): Extended	Перевод строки во время
function StrToDateTime(s: String): Extended	Перевод строки в дату и время

Форматирование

Описание	Комментарии
function Format(Fmt: String; Args: array): String	Форматирование по маске Fmt массива значений Args. Описание формата маски смотрите в приложении «Маска форматирования для функций Format и sprintf».
function FormatFloat(Fmt: String; Value: Extended): String	Форматирование числа с плавающей запятой
function FormatDateTime(Fmt: String; DateTime: TDateTime): String	Форматирование даты и времени DateTime по маске Fmt. Описание формата маски смотрите в приложении «Маска форматирования для функции FormatDateTime».
function FormatMaskText(EditMask: string; Value: string): string	Форматирование строки Value по маске EditMask.
function sprintf(Fmt: String; a1: Variant=0; a2: Variant=0; a3: Variant=0; a4: Variant=0; a5: Variant=0; a6: Variant=0; a7: Variant=0; a8: Variant=0; a9: Variant=0; a10: Variant=0;): String	sprintf форматирует параметры a1...a10 по маске Fmt и возвращает полученную строку. Может быть задано от одного до десяти параметров для форматирования. Описание формата маски смотрите в приложении «Маска форматирования для функций Format и sprintf». *Пример:* print(sprintf("%.2u часов %.2u минут", 10, 6)); Выведет на печать строку: «10 часов 06 минут»
function match(str,pattern,fmt:String;var rv:Variant):Integer	Функция для проверки соответствия текста **str** регулярному выражению **pattern** и выделения найденных фрагментов. **str** - обрабатываемый текст **pattern** - регулярное выражение **fmt** - форматирование найденных подстрок. Может содержать символы выбора параметра '_' или символ параметра 's' rv - массив из 10 возвращаемых значений: rv[0] - первое, rv[9] последнее

Описание	Комментарии
	Функция возвращает 0 если соответствие не найдено и 1 в противном случае. Пример использования функции смотрите в файле \program\samples\LanMon\c++\match.cpp.

Дата/время

Описание	Комментарии
function EncodeDate(Year, Month, Day: Word): TDateTime	Перевод года, месяца и дня в формат даты
procedure DecodeDate(Date: TDateTime; var Year, Month, Day: Word)	Перевод даты в года, месяц и день
function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime	Перевод часов, минут и секунд в формат времени
procedure DecodeTime(Time: TDateTime; var Hour, Min, Sec, MSec: Word)	Перевод времени в часы, минуты и секунды
function Date: TDateTime	Текущая дата
function Time: TDateTime	Текущее время
function Now: TDateTime	Текущие дата и время
function DayOfWeek(aDate: DateTime): Integer	День недели
function IsLeapYear(Year: Word): Boolean	Високосный год
function DaysInMonth(nYear, nMonth: Integer): Integer	Дней в месяце
function GetTickCount: Cardinal	Получение текущего счетчика миллисекундных тиков в системе. Счетчик тиков сбрасывается в ноль при переполнении Cardinal и начинает отсчет заново

Строковые функции

Описание	Комментарии
function ExtractSubString(s: String; number: Integer; delim: String): String	Извлечь из строки s подстроку с номером Number. Number – номер подстроки с единицы. Подстроки разделяются delim. Эта функция удобна для извлечения отдельных слов из строки. Пример на C++ скрипт: *print(ExtractSubString("Hello, world !", 2, " "));* Выводит на печать: "world"
function Length(s: String): Integer	Длина строки
function Copy(s: String; from, count: Integer): String	Возвращает подстроку из строки с заданной позиции заданной длины
function Pos(substr, s: String): Integer	Позиция подстроки в строке
procedure Delete(var s: String; from, count: Integer)	Удаляет подстроку из строки с заданной позиции заданной длины
procedure DeleteString(var s: String; from, count: Integer)	Удаляет подстроку из строки с заданной позиции заданной длины

Описание	Комментарии
procedure Insert(s: String; var s2: String; pos: Integer)	Добавляет первую строку ко второй строке
function Uppercase(s: String): String	Перевод строки в верхний регистр
function Lowercase(s: String): String	Перевод строки в нижний регистр
function Trim(s: String): String	Удаляет окружающие пробелы из строки
function NameCase(s: String): String	Перевод первого символав верхний регистр
function CompareText(s, s1: String): Integer	Сравнение строк
function Chr(i: Integer): Char	Возвращает символ с заданным номером
function Ord(ch: Char): Integer	Возвращает номер заданного символа
procedure SetLength(var S: String; L: Integer)	Устанавливает длину строки

Математические функции

Описание	Комментарии
function Round(e: Extended): Integer	Округление до ближайшего
function Trunc(e: Extended): Integer	Округление до меньшего
function Int(e: Extended): Integer	Возвращает целую часть
function Frac(X: Extended): Extended	Возвращает дробную часть
function Sqrt(e: Extended): Extended	Возвращает квадратный корень
function Abs(e: Extended): Extended	Возвращает модуль числа
function Sin(e: Extended): Extended	Синус
function Cos(e: Extended): Extended	Косинус
function ArcTan(X: Extended): Extended	Арктангенс
function Tan(X: Extended): Extended	Тангенс
function Exp(X: Extended): Extended	Экспонента
function Ln(X: Extended): Extended	Натуральный логарифм
function Pi: Extended	Число Пи

Другие

Описание	Комментарии
procedure Inc(var i: Integer; incr: Integer = 1)	Инкремент
procedure Dec(var i: Integer; decr: Integer = 1)	Декремент
procedure RaiseException(Param: String)	Генерация исключения
procedure ShowMessage(Msg: Variant)	Вывод сообщения
procedure Randomize	Инициализация генератора псевдослучайных чисел
function Random: Extended	Генерация псевдослучайного числа
function ValidInt(cInt: String): Boolean	Проверка правильности целого в строке
function ValidFloat(cFlt: String): Boolean	Проверка правильности цисла с плавающей запятой в строке
function ValidDate(cDate: String): Boolean	Проверка правильности даты в строке

Описание	Комментарии
function CreateOleObject(ClassName: String): Variant	Создание OLE-объекта
function VarArrayCreate(Bounds: Array; Typ: Integer): Variant	Создание динамического массива

Как видите, некоторые функции и процедуры содержат параметры по умолчанию. Вы можете вызывать их так:
Inc(a);
Inc(b, 2);

Функции специфичные для APM LanMon

Описание функции приводится в синтаксисе языка паскаль.

Функции специального преобразования типов

Описание	Комментарии
function AsString(v: Variant): String	Преобразует аргумент в выражение типа string
function AsInteger(v: Variant): Integer	Преобразует аргумент в выражение типа integer
function AsFloat(v: Variant): Extended	Преобразует аргумент в выражение типа Extended (double в C++Script)
function AsVariant(v: Variant): Variant	Преобразует аргумент в выражение типа Variant
function AsBool(v: Variant): Boolean	Преобразование аргумента любого типа в Boolean. Исключения не формируются. Аргумент текстовая строка «true», «yes», «да» преобразуются в значение true независимо от регистра.
function StrToUtf8(s: String): String	Преобразование строки в кодировке Windows 1251 в Unicode строку в кодировке UTF8. Byte Order Mark (байты EF BB BF) в начале строки не добавляются.

Особенностью этих функций преобразования типов является то, что при невозможности приведения типа они не генерируют исключений и возвращают нулевое значение. В связи с этим их удобно использовать в выражениях на скрипте.

Функции, специфичные для APM LanMon

Описание	Комментарии
function cq(ChannelAddress: String): Integer	Получение качества (состояния) канала тип 1 или тип 2. В качестве аргумента указывается адрес канала тип 1 или тип 2 в текстовом виде. При ошибках формируются исключения. Функция очень удобна в выражениях на скрипте. Пример: *// Получение качества канала тип 1 адрес 1.2.3.4* *try {*

Описание	Комментарии
	int i = cq("1.2.3.4"); *} except* *{* *LMProtokol("ERROR (" + ExceptionClassName + ") Message: " + ExceptionMessage);* *}*
function cv(ChannelAddress: String): Variant	Получение значения канала тип 1 или тип 2. В качестве аргумента указывается адрес канала тип 1 или тип 2 в текстовом виде. Тип возвращаемого значения зависит от типа значения канала. При ошибках формируются исключения. Функция очень удобна в выражениях на скрипте. Пример: *// Получение значения канала тип 1 адрес 1.2.3.4* *try {* *Variant v = cv("1.2.3.4");* *} except* *{* *LMProtokol("ERROR (" + ExceptionClassName + ") Message: " + ExceptionMessage);* *}*
function LMIIF(Expr: Boolean; TrueValue, FalseValue: Variant): Variant	Если выражение *Expr* истинно, то функция *LMIIF* возвращает значение выражения *TrueValue*. Иначе – возвращает значение выражения *FalseValue*. Функция очень удобна в выражениях на скрипте.
function RGB(Red, Green, Blue: Byte): TColor	Формирование цвета RGB из компонентов. Каждый компонент цвета лежит в диапазоне от 0 до 255.
function LMWorkDir: String	Возвращает папку с текущим проектом АРМ LanMon.
procedure LMProtokol(Str: String; File: String='')	Запись текстовой строки *Str* и метки времени в конец текстового файла *File*. Если *File* не задан или задана пустая строка – файлом считается *lanmon.log* в директории проекта. Пример записи сформатированной строки в файл lanmon.log на C++Script: *// "\x9" – это символ табуляции* *LMProtokol(sprintf("INFO\x9%s\x9%d", "hello !", 123));*
function LMExec(AppName: String; ShowWindow: Word=1; Wait: Boolean=False; var ExitCode: Integer=Nil): Boolean	Запуск внешней программы AppName или просмотр документа внешней программой. При запуске программы AppName может содержать аргументы командной строки. ShowWindow:

Описание	Комментарии
	1 – показать в нормальном размере 2 – показать минимизированное окно 3 – показать распахнутое окно Wait – ждать завершения вызванного процесса. Если задать True – работа АРМ LanMon будет заморожена до завершения вызванного приложения. Можно ставить True ТОЛЬКО если вызываемая программа выполняется не более одной секунды и ТОЛЬКО если вам нужен ExitCode. ExitCode – значение возвращаемое вызванной программой. Актуально только если Wait=True. Возвращает True если внешняя программа запушена успешно и False в противном случае. Примеры на C++: *// просмотр документа manager.pdf* *LMExec("c:\\program files\\LanMon 3\\DOC\\manager.pdf");* *// просмотр текстового файла в блокноте, распахнутом на весь экран* *LMExec(LMWorkDir() + "state.txt", 3);* *// Вызов внешней программы с параметрами* *LMExec("d:\\myprog.exe param1 param2");*
function GetDesktopWindow: Integer	Получить идентификатор (HANDLE) окна рабочего стола. Используется для вызова ShellExecute.
procedure LMShowMessage(Str: Variant; Color: TColor=clInfoBk; Left: Integer=-1; Top: Integer=-1)	Показ немодального окна с текстовым сообщением в центре экрана. Str – строка для отображения. Может включать символы перевода строки. Все остальные параметры не являются обязательными и имеют значения по умолчанию. Color – цвет фона окна. Left, Top – координаты левого верхнего угла окна. Если в параметре Str задана пустая строка – окно будет закрыто.
procedure LMShowMessage2(Text: String; Caption: String='Ошибка'; Modal: Boolean=False; Error: Boolean=True; Position: TPosition= poOwnerFormCenter)	Показ немодального или модального окна с текстовым сообщением в указанной позиции экрана. **Text** – строка для отображения. Может включать символы перевода строки. Все остальные параметры не являются обязательными и имеют значения по умолчанию. **Caption** – заголовок окна. Modal – признак модальности окна. **Error** – определяет надо ли отображать иконку «STOP» в окне ?

Описание	Комментарии
	Position – определяет позицию на экране для отображения окна. Возможные значения приведены в описании функции **QuestionYesNo**.
procedure Print(Str: Variant)	Печать строки Str в окне отладочной печати редактора программ.
function LMSendControl(A1, A2, A3, A4, STATE, VAL: Integer): Integer	Послать команду управления драйверам оборудования или на сервер LanMon. Функция получает адрес канала, его состояние и значение. При ошибке возвращает нулевое значение. Пример: *LMSendControl(1,2,3,4,0,1);*
function LMSendControl2(Addr: String; Quality: WORD; Value: Variant): Boolean	Послать команду управления драйверам оборудования или на сервер LanMon. *Addr* – адрес канала тип 1 или тип 2 *Quality* – качество (состояние) канала *Value* – значение При ошибке возвращает false и пишет сообщение в протокол событий проекта. Для каналов тип 2 эта функция аналогична вызову *RegisterChannelValue2* с параметром *Action = 4*. Пример: *LMSendControl2("1.2.3.4",0,1);*
function LMSendState(A1, A2, A3, A4, STATE: Integer; VAL: Variant): Integer	Послать состояние канала на сервер LanMon. Функция получает адрес канала, его состояние и значение. При ошибке возвращает ноль. Пример: *LMSendState(1,2,3,5, 0, 1.5);*
function LMSendState2(Addr: String; Quality: WORD; Value: Variant): Boolean	Послать состояние канала на сервер LanMon. Метка времени устанавливается в текущее время. *Addr* – адрес канала тип 1 или тип 2 *Quality* – качество (состояние) канала *Value* – значение При ошибке возвращает false и пишет сообщение в протокол событий проекта. Пример: *LMSendState2("1.2.3.5", 0, 1.5);*
function LMSendStateLocal(A1, A2, A3, A4, STATE: Integer; VAL: Variant): Boolean	Зарегистрировать новое состояние канала локально. Функция получает адрес канала, его состояние и значение. При ошибке возвращает значение false. Ошибка может возникнуть в двух случаях: - такого канала нет в дереве каналов - произошла ошибка преобразования типов параметра VAL (например, канал целочисленный, а в функцию передана строка)

Описание	Комментарии
	Пример: *LMSendStateLocal(1,2,3,4,0,1);*
function LMSendStateLocal2(Addr: String; Quality: WORD; Value: Variant): Boolean	Зарегистрировать новое состояние канала локально. *Addr* – адрес канала тип 1 или тип 2 *Quality* – качество (состояние) канала *Value* – значение При ошибке возвращает false и пишет сообщение в протокол событий проекта. Пример (регистрируем новое значение строкового канала тип 2): *LMSendStateLocal2("режим_сушки", 0, "ручной");*
function LMSendChannel(c: TChannel; Action: Integer=3): Boolean	Послать канал на сервер/самому себе. Action задает куда мы посылаем значение канала. Если Action=1 – отправка канала на сервер LanMon. Если Action=2 – отправка канала самому себе. Если Action=3 – отправка и туда и туда (значение по умолчанию).
function RegisterChannelValue2(Addr: String; Number: Cardinal; DT: TDateTime; Quality: WORD; Value: Variant; Action: Integer=3): Boolean	Зарегистрировать новое значение канала тип 2 или отправить команду управления на сервер LanMon. Для идентификации канала надо указать адрес канала Addr или номер Number, а также метку времени DT, качество Quality, значение Value, действие Action (1-отправить всем клиентам сервера LanMon, 2-отправить только себе, 4-отправить команду управления). При ошибке функция возвращает false. Пример: *// Отправить команду управления для включения насоса* *// Канал тип 2 указываем по адресу* *RegisterChannelValue2("nasos2", 0, Now(), 0, 1, 4);*
function RegisterAttributeValue2(Addr: String; Number: Cardinal; AttrID: WORD; Value: Variant): Boolean	Зарегистрировать новое значение атрибута канала 2. Для идентификации канала надо указать адрес канала Addr или номер Number, а также цифровой идентификатор атрибута и его значение. При ошибке функция возвращает false.
function RegisterChannelActive2(Addr: String; Number: Cardinal=0; Activ: Boolean=True): Boolean	Установить или снять статус активности канала 2. Для идентификации канала надо указать адрес канала Addr или номер Number, а также состояние активности канала. При ошибке функция возвращает false.
function LMServerStatus(type: Integer): Variant	Если type=0 функция возвращает статус подключения к серверу LanMon в числовом виде. Если type=1 функция возвращает статус подключения к серверу LanMon в виде текстовой строки.

Описание	Комментарии
	Возможные возвращаемые значения: *0-"ОТКЛЮЧЕН !"* *1-"Подключение..."* *2-"Регистрация..."* *3-"Запись маски..."* *4-"Получение A1..."* *5-"Получение A2..."* *6-"Получение A3..."* *7-"Получение A4..."* *8-"Получение каналов..."* *9-"ОК"* *10-"ОТКЛЮЧЕН ! (РЕЗЕРВНЫЙ)"* *11-"Подключение... (РЕЗЕРВНЫЙ)"* *12-"Регистрация... (РЕЗЕРВНЫЙ)"* *13-"Запись маски... (РЕЗЕРВНЫЙ)"* *14-"Получение A1... (РЕЗЕРВНЫЙ)"* *15-"Получение A2... (РЕЗЕРВНЫЙ)"* *16-"Получение A3... (РЕЗЕРВНЫЙ)"* *17-"Получение A4... (РЕЗЕРВНЫЙ)"* *18-"Получение каналов... (РЕЗЕРВНЫЙ)"* *19-"ОК (РЕЗЕРВНЫЙ)"* Примечание: Возвращаемое значение 1 означает НАЧАЛО подключения к серверу, а 2 успешное окончание и т.д.
procedure LMViewAnalogAlarm	Показать на экране окно аналоговых алармов.
procedure LMViewAlarm	Открыть окно просмотра алармов.
procedure LMExit	В режиме отладки проекта: завершение отладки. В режиме выполнения проекта: завершение работы APM LanMon.
function LMFormatString(A1,A2,A3,A4: Integer; MASK: String): String	Сформатировать параметры канала тип 1 в текстовую строку. Возвращает полученную строку. Первые 4 аргумента – адрес канала тип 1. MASK - маска форматирования. Может содержать произвольный текст и следующие подстановки: **%DATE** – дата изменения значения канала в формате DD:MM **%TIME** – время изменения значения канала в формате HH:MM:SS **%DT** – дата и время в формате, заданном в системе **%DT(формат)** - дата и время в заданном формате «формат». Описание формата смотрите в приложении «Маска форматирования для функции FormatDateTime» **%IDT** – дата и время интеллектуальном формате:

Описание	Комментарии
	Если сегодня, то выводится только время в формате, заданном в системе. Если не старше 7 дней, то выводится название дня недели + время в формате, заданном в системе. В противном случае выводятся дата и время в формате, заданном в системе. **%ADDR** – текстовый адрес канала **%A1** – текстовое наименование канала уровня A1 **%A2** – текстовое наименование канала уровня A2 **%A3** – текстовое наименование канала уровня A3 **%A4** – текстовое наименование канала уровня A4 **%COMMENTS** – полное текстовое наименование канала **%STATE** – текстовое описание состояния канала **%VALUEn(%mask)** – значение канала. n – номер значения с нуля (только для массивов). %mask – маска форматирования значения канала. Описание формата маски смотрите в приложении «Маска форматирования для функций Format и sprintf». **%NL** – вставка символов новой строки Пример: *LMFormatString(1,2,3,4,"%DATE %TIME%NL%A3 %A4 %STATE")*
function LMFormatString2(Addr, Mask:: String): String	Сформатировать параметры канала тип 1 или тип 2 в текстовую строку. Возвращает полученную строку. Addr – адрес канала тип 1 или тип 2. Mask - маска форматирования. Может содержать произвольный текст и следующие подстановки: **%DATE** – дата изменения значения канала в формате DD:MM **%TIME** – время изменения значения канала в формате HH:MM:SS **%DT** – дата и время в формате, заданном в системе **%DT(формат)** - дата и время в заданном формате «формат». Описание формата смотрите в приложении «Маска форматирования для функции FormatDateTime» **%IDT** – дата и время интеллектуальном формате: Если сегодня, то выводится только время в формате, заданном в системе. Если не старше 7 дней, то выводится название дня недели + время в формате, заданном в системе. В противном случае выводятся дата и время в формате, заданном в системе. **%ADDR** – текстовый адрес канала **%COMMENTS** – полное текстовое наименование канала

Описание	Комментарии
	%UNIT – единицы измерения (только для канала тип 2) **%ATTR(идентификатор атрибута (ID))** – значение указанного атрибута (только для канала тип 2) **%STATE** – текстовое описание состояния канала (для канала тип 2 включает единицы измерения) **%VALUEn(%mask)** – значение канала. n – номер значения с нуля (только для массивов). %mask – маска форматирования значения канала. Описание формата маски смотрите в приложении «Маска форматирования для функций Format и sprintf». **%NL** – вставка символов новой строки Пример: *LMFormatString2("1.2.3.4","%DT(dd mmm hh:nn:ss)%NL%COMMENTS %VALUE %UNIT")*
function AAState(Alarm: Variant; ResultType: Integer): Variant	Получить состояние аналогового аларма. Alarm - название аларма или его номер. ResultType – тип возвращаемого значения. Если ResultType равен нулю, функция возвращает целочисленное состояние аларма: -1-указанный аларм не найден 0-ок 1-повышение 2-недопустимое повышение 3-понижение 4-недопустимое понижение 5-неисправность канала Если ResultType равен единице, функция возвращает текстовую расшифровку состояния аларма.
procedure Sleep(Milliseconds: int)	Пауза в выполнении программы на указанное количество миллисекунд.
procedure LMCreateDigitalAlarm(Message: String; SoundFile1,SoundFile2: String; PlayCount: Integer; A1,A2,A3,A4: Word)	Выдать строку Message в окно тревожных сообщений и проиграть 2 звуковых файла. **Message** – текстовое сообщение для оператора **SoundFile1**, **SoundFile2** – два звуковых файла в формате wav. Проигрываются последовательно. Если указана пустая строка – проигрывание не производится. **PlayCount** – Количество проигрываний звуковых файлов. **A1,A2,A3,A4** – Адрес канала, который будет показан на картах при выборе этого сообщения в окне

Описание	Комментарии
	тревожных сообщений. Если указаны нули – привязка к адресу канала не производится. Пример: *LMCreateDigitalAlarm("Авария дом 12Б !","Дом.wav","12Б.wav",1,0,0,0,0);*
procedure LMRegisterAlarmMessage(Message: String; SoundFile1,SoundFile2,SoundFile3: String; PlayCount,PlayPause: Integer; Addr: String) *procedure LMCreateDigitalAlarm2(Message: String; SoundFile1,SoundFile2,SoundFile3: String; PlayCount, PlayPause: Integer; Addr: String)*	Процедуры ***LMCreateDigitalAlarm2*** является синонимом для ***LMRegisterAlarmMessage***. Выдать строку Message в окно тревожных сообщений и проиграть 3 звуковых файла. **Message** – текстовое сообщение для оператора **SoundFile1**, **SoundFile2**, **SoundFile3** – три звуковых файла в формате wav. Проигрываются последовательно. Если указана пустая строка – проигрывание не производится. **PlayCount** – Количество проигрываний звуковых файлов. **PlayPause** – Пауза перед повторным проигрыванием в секундах. **Addr** – Адрес канала тип 1 или тип 2, который будет показан на картах при выборе этого сообщения в окне тревожных сообщений. Если указана пустая строка – привязка к адресу канала не производится. Пример: *LMCreateDigitalAlarm2("Авария дом 12Б !","Дом.wav","12Б.wav","",1000,"1.4.1.12");*
procedure LMConfirmAlarmMessage(action: Integer) *procedure LMConfirmDigitalAlarm(action: Integer)*	Процедуры ***LMConfirmDigitalAlarm*** является синонимом для ***LMConfirmAlarmMessage***. Подтвердить тревогу, выданную в окно тревожных сообщений. Параметр **action**: 1 - имитируем нажатие кнопки "Закрыть" в окне алармов 2 - имитируем нажатие кнопки "Очистить" в окне алармов 3 - имитируем нажатие кнопки "Стоп звук" в окне алармов Пример: // Закрыть окно тревожных сообщений *LMConfirmAlarmMessage (1);*
function SaveParam(Name: String; Value: String): Boolean	Сохранение параметра Name со значением Value в файле vars.ini в директории проекта.

Описание	Комментарии
	Пример: *SaveParam(“param1”, “12,6”);*
function LoadParam(Name: String; var Value: String): Boolean	Чтение параметра Name из файла vars.ini в директории проекта. Значение параметра возвращается в параметре Value. Функция возвращает true если параметр с именем Name найден и false в противном случае. Пример: *String Value;* *if(! LoadParam("param1", Value))* *LMProtokol("Параметр param1 не был сохранен вызовом функции SaveParam!");* *else* *LMProtokol("Значение параметра param1 = " + Value);*
function IsMasked(AlarmID: Integer; Addr: String): Boolean	Определяет состояние маскирования аларма с идентификатором **AlarmID** для адреса канала **Addr** (тип 1 или 2). Если **AlarmID** равен нулю, то определяет хоть 1 замаскированный аларм для адреса канала **Addr**. Замаскированный аларм при срабатывании не выдает окно тревожных сообщений. Идентификатор аларма (номер аларма) можно посмотреть в редакторе алармов или в окне мониторинга алармов.
function ChangeMask(AlarmID: Integer; Addr: String): Boolean	Изменяет (инвертирует) состояние маскирования аларма с идентификатором **AlarmID** для адреса канала **Addr** (тип 1 или 2). Замаскированный аларм при срабатывании не выдает окно тревожных сообщений. Возвращает **true** если произведено изменение состояния аларма и **false** в противном случае. Идентификатор аларма (номер аларма) можно посмотреть в редакторе алармов или в окне мониторинга алармов.
function GetAccountLogin(id: Integer): String	Получить логин учетной записи сервера для указанного идентификатора учетной записи id. В качестве параметра обычно используются свойства каналов: TChannel::Owner, TChannel2::Owner, TChannel2::Creator
function GetAccountComments(id: Integer): String	Получить примечание к учетной записи сервера для указанного идентификатора учетной записи id. В качестве параметра обычно используются свойства

Описание	Комментарии
	каналов: TChannel::Owner, TChannel2::Owner, TChannel2::Creator
procedure ViewChannelHistory(A1, A2, A3, A4: Word; From, To: TDateTime)	Показать историю по каналу тип 1. A1.A2.A3.A4 – Адрес канала типа 1. From – дата и время начала отображаемого периода. To - дата и время окончания отображаемого периода.
procedure ViewChannel2History(Addr: String; From, To: TDateTime)	Показать историю по каналу тип 2. Addr – Адрес канала типа 2. From – дата и время начала отображаемого периода. To - дата и время окончания отображаемого периода.
procedure EnableGenerator(Vis: Boolean)	Разрешить/запретить окно генератора для графических объектов карт (пункт контекстного меню)
procedure EnableHistory(Vis: Boolean)	Разрешить/запретить окно истории для графических объектов карт (пункт контекстного меню)

Функции для работы с операторами

Текущий оператор задается глобальным объектом скрипта **OperatorInfo**, тип объекта **TOperatorInfo**. Доступ к параметрам текущего оператора выполняется через свойства объекта **OperatorInfo**:

Свойство	Описание
property UserID: Integer	Уникальный идентификатор (номер) текущего оператора. Только для чтения.
property FIO: String	Фамилия Имя Отчество текущего оператора. Причем имя и отчество сокращены до первых букв. Только для чтения.
property LevelSec: Integer	Права доступа текущего оператора (0-супервизор, 1-администратор, 2-общие). Только для чтения.
property LevelSecStr: String	Строковое наименование прав доступа текущего оператора (“супервизор”, “администратор”, “общие”). Только для чтения.

Следующие функции могут быть вызваны из программы на скрипте:

Функция	Описание
procedure LMOperatorChange	Вызов окна регистрации оператора для смены текущего оператора. Пример на C++ скрипте: *LMOperatorChange();*
procedure LMOperatorsEdit	Вызов окна редактирования операторов. Обычно окно редактирования операторов используется для смены пароля оператора в режиме выполнения проекта.

Функция	Описание
	Пример на C++ скрипте: *LMOperatorsEdit();*

После успешной регистрации нового оператора вызывается обработчик скрипта *OnOperatorRegister*. В качестве параметра он получает ссылку на объект с параметрами нового оператора. Пример:

```
// Это обработчик вызывается при регистрации нового оператора
void OnOperatorRegister(TOperatorInfo NewOperator)
{
  // Запись строки в файл lanmon.log в папке проекта
  LMProtokol("Зарегистрирован новый оператор " + NewOperator.FIO +
            " [права " + NewOperator.LevelSecStr + "]"+
            "; вместо " + OperatorInfo.FIO +
            " [права " + OperatorInfo.LevelSecStr + "]"+
            );
}
```

Функции для работы с журналом событий

Функция	Описание
procedure LMShowEvents(Address: String=''; Status: Integer=-1; Signification: Integer=-1; FilterAnd: Boolean=True)	Показ окна журнала событий с заданием фильтров. *Address* – регулярное выражение для поля «**Фильтр по тексту**». *Status* – содержимое поля «**Фильтр по статусу**» (возможные значения смотри ниже). *Signification* – содержимое поля «**Фильтр по назначению**» (возможные значения смотри ниже). *FilterAnd* – true - объединить фильтры по логике «И»; false - объединить фильтры по логике «ИЛИ». Примеры на C++ скрипте: *// Показ окна журнала событий со всеми событиями – фильтрация выключена* *LMShowEvents();* *// Показ окна журнала событий – включена фильтрация: показывать только события с адресом SYSTEM* *LMShowEvents("^SYSTEM$");*

Функция	Описание
	// Показ окна журнала событий – включена фильтрация: показывать только события со статусом тревога датчиков охранной сигнализации *LMShowEvents(“”, 2, 5);*
procedure LMRegisterEvent(DT: TDateTime; Address, Comments, State: String; Status, Signification: Integer; Addstatus: Integer=0)	Регистрация нового события в журнале. DT - содержимое колонки «**Дата Время**» Address - содержимое колонки «**Адрес**» Comments - содержимое колонки «**Наименование**» State – содержимое колонки «**Состояние**» Status – статус (расшифровку смотри ниже) Signification – назначение (расшифровку смотри ниже) Addstatus – 0 снят с охраны, 1 взят под охрану - определено только если Signification = 5 Пример на C++ скрипте: *// Добавление нового события в журнал* *// Текущая дата время, статус «Внимание», назначение «Не определено»* *LMRegisterEvent(Now(),“SYSTEM”,“Режим проветривания”, “Включен оператором”, 3, 0);*

Возможные значения статуса Status (функции *LMShowEvents* и *LMRegisterEvent*, объект *TEventLogItem)*:

Значение	Описание
0	Норма
1	Срабатывание
2	Тревога
3	Внимание
4	Служебное
5	Неисправность

Возможные значения назначения Signification (функции *LMShowEvents* и *LMRegisterEvent*, объект *TEventLogItem)*:

Значение	Описание
0	Не определено.
1	Интегратор учета в импульсах. Например, интегратор БРК-К или БТС.
2	Интегратор учета в физических единицах. Например, интегратор газа счетчика Омега в [М3].
3	Канал управления чем-либо. Например, управление клапаном счетчика Омега или канал управления БИУ.

Значение	Описание
4	Канал контроля чего-либо. Например, сухой контакт БИУ-Р или фаза БИУ.
5	Охранный извещатель или шлейф.
6	Пожарный извещатель или шлейф.
7	Состояние охраны зоны.
8	Сигнализатор загазованности.
9	Датчик температуры.
10	Извещатель пожарный ручной.
11	Датчик затопления
12	Лифт

Запись журнала событий определяется объектом скрипта типа *TEventLogItem*:

Свойство	Описание
property DT: TDateTime	Дата и время события. Только для чтения.
property Address: String	Адрес источника события. Только для чтения.
property Comments: String	Наименование объекта события. Только для чтения.
property State: String	Состояние объекта события. Только для чтения.
property Status: Integer	Статус. Только для чтения.
property Signification: Integer	Назначение. Только для чтения.
property Addstatus: Integer	0 снят с охраны, 1 взят под охрану (определено только если Signification = 5) . Только для чтения.
property Owner: short	Идентификатор источника события: учетная запись сервера LanMon, конкретный драйвер или сервер LanMon. Только для чтения.
property UserID: Integer	Идентификатор оператора, в смену которого данное событие было зарегистрировано в журнале. Только для чтения.
property ArmID: short	Идентификатор АРМ, который зарегистрировал данное событие (учетная запись сервера LanMon). Только для чтения.

При нажатии клавиши в окне журнала событий вызывается обработчик скрипта *OnJurnalKeyDown*. В качестве параметра он получает ссылку на объект записи журнала событий *TEventLogItem*. Пример на C++ скрипт:

```
// Это обработчик события на нажатие клавиши в журнале событий
// При нажатии клавиши ENTER производим запись строки в протокол работы
// проекта lanmon.log
void OnEventLogKeyDown(TEventLogItem Sender, Word Key, TShiftState Shift)
{
  if( Key == VK_RETURN )
  LMProtokol("Нажатие клавиши ENTER в журнале событий:" +
    " DT=" + DateTimeToStr(Sender.DT) +
    " Address=" + Sender.Address +
    " Comments=" + Sender.Comments +
    " State=" + Sender.State +
```

```
    " Status=" + IntToStr(Sender.Status) +
    " Addstatus=" + IntToStr(Sender.Addstatus) +
    " Signification=" + IntToStr(Sender.Signification) +
    " Owner=" + IntToStr(Sender.Owner) +
    " UserID=" + IntToStr(Sender.UserID) +
    " ArmID=" + IntToStr(Sender.ArmID));
}
```

При двойном щелчке мышью на записи в окне журнала событий вызывается обработчик скрипта *OnJurnalDblClick*. В качестве параметра он получает ссылку на объект записи журнала событий *TEventLogItem*. Пример на C++ скрипт:

```
// Это обработчик двойного щелчка мышью на записи в журнале событий
// Производим запись строки в протокол работы проекта lanmon.log
void OnEventLogDblClick(TEventLogItem Sender)
{
  LMProtokol("Двойной щелчок мышью в журнале событий:" +
    " DT=" + DateTimeToStr(Sender.DT) +
    " Address=" + Sender.Address +
    " Comments=" + Sender.Comments +
    " State=" + Sender.State +
    " Status=" + IntToStr(Sender.Status) +
    " Addstatus=" + IntToStr(Sender.Addstatus) +
    " Signification=" + IntToStr(Sender.Signification) +
    " Owner=" + IntToStr(Sender.Owner) +
    " UserID=" + IntToStr(Sender.UserID) +
    " ArmID=" + IntToStr(Sender.ArmID));
}
```

Обработчики событий

Описание	Комментарии
procedure OnServerStatusChange(Status: Integer; Main: Boolean)	Если данная функция-событие определено в тексте программы, то она будет вызываться при изменении состояния подключения к серверу LanMon. **Status:** 0 - произошло отключение от сервера 2 - произошло подключение к серверу по TCP/IP 4 - успешная регистрация на сервере 9 - начало нормальной работы с сервером **Main:** true – идет работа с основным сервером false – идет работа с резервным сервером

Функция для выполнения команд проводника ShellExecute

Функция *ShellExecute* предназначена для выполнения команд проводника Windows. Она позволяет открывать и печатать документы, открывать папки, запускать проводник, запускать Интернет браузер, запускать почтовый клиент.

Описание:
function ShellExecute(hwnd: Integer; Operation, File, Parameters, Directory: String; ShowCmd: Integer): Integer

Параметры:
hwnd – идентификатор (HANDLE) родительского окна. Это окно будет владельцем всех окон-сообщений об ошибках.
Operation – строка согласно таблице:

Operation	Описание
"open"	Функция открывает файл, указанный в *File*. Файл может быть исполняемой программой, документом или папкой.
"print"	Функция печатает файл, указанный в *File*. Файл должен быть документом. Если указанный файл является исполняемой программой – функция запускает программу.
"explore"	Функция открывает файл, указанный в *File* в проводнике.

File – имя файла исполняемой программы, документа, папки на диске, url.
Parameters – параметры для запуска программы. Если в параметре *File* задан документ, то *Parameters* должен быть пустой строкой.
Directory – директория по умолчанию для запускаемого процесса.
ShowCmd – константа согласно таблице:

ShowCmd	Описание
SW_HIDE	Спрятать окно и активировать другое окно.
SW_MAXIMIZE	Распахнуть окно.
SW_MINIMIZE	Свернуть окно и активизировать следующее окно верхнего уровня в Z порядке.
SW_RESTORE	Активировать и показать окно. Если окно свернуто или распахнуто Windows восстановит его оригинальный размер и координаты. Приложение должно указать этот флаг для восстановления свернутого окна.
SW_SHOW	Активировать окно и показать его в текущем размере и позиции.
SW_SHOWDEFAULT	Sets the show state based on the SW_ flag specified in the STARTUPINFO structure passed to the CreateProcess function by the program that started the application. An application should call ShowWindow with this flag to set the initial show state of its main window.
SW_SHOWMAXIMIZED	Активировать окно и показать его распахнутым.
SW_SHOWMINIMIZED	Активировать окно и показать его свернутым.

ShowCmd	Описание
SW_SHOWMINNOACTIVE	Показать окно как свернутое. Текущее активное окно останется активным.
SW_SHOWNA	Показать окно в состоянии по умолчанию. Текущее активное окно останется активным.
SW_SHOWNOACTIVATE	Показать окно в последнем состоянии. Текущее активное окно останется активным.
SW_SHOWNORMAL	Активировать и показать окно. Если окно свернуто или распахнуто Windows восстановит его оригинальный размер и координаты. Приложение должно указать этот флаг для показа окна впервые.

Возвращаемое значение:
Если при выполнении функции произошла ошибка, то возвращается код ошибки - число меньше или равно 32. В противном случае функция выполнена успешно.

Примеры использования функции на языке C++:

```
 // Открыть папку "c:\"
int error = ShellExecute(GetDesktopWindow(), "open", "c:\\", "", "", SW_SHOWNORMAL);
if( error > 32 )
  print("OK");
else
  print( "Ошибка: " + IntToStr(error) );

// Открыть проводник в папке "c:\"
ShellExecute(GetDesktopWindow(), "explore", "c:\\", "", "", SW_SHOWNORMAL);

// Написать письмо с адресом назначения "lanmon@mnppsaturn.ru"
ShellExecute(GetDesktopWindow(), "open", "mailto:lanmon@mnppsaturn.ru", "", "", SW_SHOWNORMAL);

// Открыть Интернет браузер по умолчанию со ссылкой "http://www.mnppsaturn.ru"
ShellExecute(GetDesktopWindow(), "open", "http://www.mnppsaturn.ru", "", "", SW_SHOWNORMAL);

// Запустить командный процессор "cmd.exe"
ShellExecute(GetDesktopWindow(), "open", "cmd.exe", "", "", SW_SHOWNORMAL);

// Запустить команду ping для указанного адреса
ShellExecute(GetDesktopWindow(), "open", "ping.exe", "192.168.1.1", "", SW_SHOWNORMAL);
```

Текст примера находится в проектах примеров в файле *"..\program\samples\LanMon\c++\shell.cpp"*

Функции для работы с файлами

Описание	Комментарии
function addfile(File, Str: String): Integer	Добавить строку в текстовый файл File. Первый аргумент – имя файла. Второй аргумент – строка.

Описание	Комментарии
	После записи строки в файл автоматически добавляются символы перевода строки. Возврат: 0 – ошибка открытия или записи файла 1- все хорошо
function DeleteFile(FileMask: String): Boolean	Удаление указанного файла или файлов по маске *FileMask*. Если удаление прошло успешно – возвращается *true*.
function LMGetCSV(File: String; Row,Col: Integer): String	Прочитать одно значение из файла в формате CVS (текстовый файл с колонками, разделенными запятыми). Аргументы: имя файла, номер строки, номер колонки. Возвращает найденное значение типа string или строку «0» при ошибке. Пример использования смотрите в файле GetCSV.bas
function FileExists(FileName: String): Boolean	Проверка существования указанного файла. Если файл FileName существует функция возвращает true.
function FileCopy(ExistingFileName, NewFileName: String): Boolean	Копирование файла ExistingFileName в NewFileName. Если файл NewFileName существует – он будет перезаписан. Функция возвращает true если копирование прошло успешно и false в противном случае.
function SelectDirectory(Caption: String; Root: String; var Directory: string): Boolean	Показ диалога выбора папки. Заголовок окна задается параметром *Caption*. Начальная папка задается параметром Root. Выбранная в диалоге папка помещается в параметр *Directory*. При подтверждении выбора папки функция возвращает *true* и *false* в противном случае.
function IncludeTrailingBackslash(s: String): String	Если имя папки *s* не заканчивается символом ‘\’ – функция добавляет его и возвращает полученную строку.
function ChangeFileExt(FileName, Extension: String): String	Функция изменяет расширение имени файла *FileName* на *Extension* и возвращает имя файла с новым расширением.
function ExtractFileDir(FileName: String): String	Функция извлекает из полного имени файла *FileName* папку (полный путь к файлу) и возвращает его.
function ExtractFileName(FileName: String): String	Функция извлекает из полного имени файла *FileName* имя файла и возвращает его.
function ExtractFileDrive(FileName: String): String	Функция извлекает из полного имени файла *FileName* имя диска и возвращает его.
function ExtractFileExt(FileName: String): String	Функция извлекает из имени файла *FileName* расширение и возвращает его.

Описание	Комментарии
function FileGetAttr(FileName: String): Integer	Функция возвращает атрибуты файла *FileName*. Все поддерживаемые атрибуты приведены в таблице ***«Атрибуты файла»***.
function FileSetAttr(FileName: String; Attr: Integer): Integer	Функция устанавливает атрибуты файла *FileName*. Если операция прошла успешна функция возвращает ноль. В противном случае возвращается код ошибки.
function FileIsReadOnly(FileName: String): Boolean	Функция проверяет: у файла *FileName* установлен атрибут «только для чтения».

Атрибуты файла

Обозначение	Описание
faReadOnly	Только для чтения
faHidden	Скрытый
faSysFile	Системный
faVolumeID	Метка тома
faDirectory	Директория (папка)
faArchive	Архивный
faAnyFile	Любой файл или директория. Этот атрибут является суммой всех вышеперечисленных.

Функции для работы с генератором отчетов

Описание	Комментарии
procedure frSetDefaultConnection(ADOConnection: TADOConnection)	Устанавливает подключение для отчетов по умолчанию. Если в отчете не задано свое подключение, то по умолчанию, будет использоваться подключение ***SystemADOConnection***. Эта процедура позволяет переопределить соединение, используемое по умолчанию. Пример: *frSetDefaultConnection(SystemADOConnection1);* Примечание: функция ***frLoad()*** восстанавливает использование ***SystemADOConnection*** в качестве соединения по умолчанию.
function frLoad(File: String): Integer	Загружается в память шаблон отчета. Шаблоны отчетов хранятся в файлах с расширением «fr3» в поддиректории проекта «.\FR\». Если задано краткое имя файла шаблона отчета, оно будет искаться в поддиректории проекта «.\FR\». Функция ***frLoad()*** восстанавливает использование ***SystemADOConnection*** в качестве соединения по умолчанию.

Описание	Комментарии
	Функция возвращает ноль в случае ошибки и любое положительное число при успешной загрузке отчета.
function frSave(File: String): Integer	Сохраняет шаблон отчета из памяти в файл. Если задано краткое имя файла шаблона отчета, оно будет сохранено в поддиректорию .\FR Функция возвращает ноль в случае ошибки.
function frRun(Clear: Integer): Integer	Запускает шаблон отчета из памяти на выполнение. После выполнения отчета показывает его в окне предварительного просмотра. Шаблон отчета должен быть предварительно загружен функцией frLoad(). Если аргумент равен единице – ранее построенный отчет очищается. Если нулю – новый отчет будет добавлен к ранее построенному. Функция возвращает ноль в случае ошибки.
function frPrepare(Clear: Integer): Integer	Запускает шаблон отчета из памяти на выполнение. Выполнение программы блокируется на время выполнения отчета. Для показа выполненного отчета в окне предварительного просмотра вызовите функцию frShowPrepared(). Шаблон отчета должен быть предварительно загружен функцией frLoad(). Если аргумент равен единице – ранее построенный отчет очищается. Если нулю – новый отчет будет добавлен к ранее построенному. Функция возвращает ноль в случае ошибки.
function frShowPrepared	Показывает текущий отчет, ранее подготовленный функциями frPrepare, frAsyncPrepare или frRun. Функция возвращает ноль в случае ошибки. Ошибка может возникнуть если фоновое выполнение отчета, запущенное функцией frAsyncPrepare, еще не завершилось.
function frDesign	Вызывает редактор шаблонов отчетов fr3 в режиме выполнения проекта. Функция возвращает ноль в случае ошибки. Ошибка может возникнуть если фоновое выполнение отчета еще не завершилось.
function frPrint(ShowDialog: Integer): Integer	Печать отчета, ранее подготовленного функциями frPrepare или frRun. Если аргумент равен единице – показывает окно выбора принтера. Если нулю – печать осуществляется на принтере по умолчанию без запроса пользователя.
function frExport(Format: Integer; File: String=''): Integer	Производит экспорт текущего отчета в файл одного из форматов. Отчет должен быть ранее подготовлен

Описание	Комментарии
	функциями *frPrepare* или *frRun*. Первый параметр функции *frExport* – это формат экспорта. Может принимать одно из следующих –значений: 0 – TXT 1 – HTML 2 – XLS 3 – XML 4 – RTF 5 – BMP 6 – JPEG 7 – TIFF 8 – PDF 9 – GIF 10 – простой текст 11 – CSV 12 – Отправить отчет по электронной почте 13 – ODS (Open Office) 14 – ODT (Open Office) Если второй аргумент функции – “имя файла” – не задан, то выдается диалоговое окно параметров экспорта. Если файл задан – производится экспорт без запросов пользователя. Функция возвращает ноль в случае ошибки.

Функции фонового выполнения отчета в генераторе отчетов

Описание	Комментарии
function frAsyncPrepare(ClearLastReport: Boolean; OnReport: String='OnReportNotify'): Boolean	Запускает шаблон отчета из памяти на выполнение. Шаблон отчета должен быть предварительно загружен функцией frLoad(). Управление возвращается незамедлительно и начинается фоновое выполнение отчета. После выполнения отчета вызывается событие OnReportNotify. Для показа выполненного отчета в окне предварительного просмотра вызывайте функцию frShowPrepared(). Если параметр ClearLastReport равен True – ранее построенный отчет очищается. Если False – новый отчет будет добавлен к ранее построенному. Параметр OnReport задает название обработчика события, которое вызывается по окончанию выполнения отчета (по умолчанию 'OnReportNotify').

Описание	Комментарии
	Функция возвращает False если отчет уже выполняется.
function frBusy: Boolean	Возвращает True если выполняется фоновый отчет запущенный функцией frAsyncPrepare и False в противном случае.
procedure frBreak	Прерывает фоновое выполнение отчета, запущенное функцией frAsyncPrepare.
procedure OnReportNotify(RetVal: Integer)	После завершения выполнения фонового отчета, запущенного функцией frAsyncPrepare, вызывается событие OnReportNotify. RetVal – код результата выполнения отчета: 0-при выполнении отчета произошла ошибка 1-отчет успешно выполнен 2-выполнение отчета прервано вызовом frBreak Для добавление в текст программы события OnReportNotify, в редакторе программ в меню «События» выберите пункт «Выполнение фонового отчета завершено». Типовой код обработчика этого события – вызов frShowPrepared для показа готового отчета в окне предварительного просмотра. В функции frAsyncPrepare может быть задано другое имя для обработчика события по выполнении отчета.

ВНИМАНИЕ: При фоновом выполнении отчета обращение к глобальной переменной объекта отчета SystemReport запрещено !

Функции для работы со встроенным клиентом IP телефонии H323

Все функции IP телефонии работают асинхронно. Например, функция посылки вызова H323Call() возвращает управление немедленно, но процедура дозвона и установления соединения может занять несколько секунд.

Описание	Комментарии
function H323State: Integer	Получить текущее состояние встроенного IP телефона. Возвращает значение типа integer: 0 – трубка на крючке 1 – Трубка снята, идет разговор 2 – Идет набор номера (ожидание поднятия трубки на удаленной стороне) 3 – Идут входящие звонки
function H323Call(Abonent: Variant): Integer	Послать вызов (позвонить). Если задан аргумент типа integer – это порядковый номер абонента из записной книжки, начиная с нуля. Если задан аргумент типа string – это узел или IP адрес

Описание	Комментарии
	вызываемого абонента. Данная функция выполняется, только если трубка на крючке. При успешном вызове возвращает ненулевое значение. Возвращает ноль при ошибке. Для получения кода ошибки используйте функцию H323LastError().
function H323Handup: Integer	Положить трубку (завершение разговора). Аргументов не имеет. Возвращает всегда ноль.
function H323Answer: Integer	Снять трубку когда идут входящие звонки: H323State()=3. Аргументов не имеет. Возвращает всегда ноль.
function H323LastError(type: Integer): Variant	Получение последней ошибки клиента IP телефонии. Имеет 1 аргумент: 0 – дать числовой код ошибки 1 – дать строковое описание ошибки Список ошибок приведен в таблице ниже.
function H323ShowWindow(Show: Integer): Integer	Показать/скрыть окно клиента H323 с записной книжкой. Имеет один аргумент: 0 – скрыть окно 1 – показать окно Возвращает всегда ноль.
function H323Direction(transmit: Boolean): Boolean	Управление направлением звука при разговоре с блоком сети СОС-95 (БГС, УИР-РЦ, БДК-М) через шлюз H323 sos95gw. Дело в том, после установления соединения с переговорным блоком СОС-95, связь работает в режиме полудуплекса с автопереключением направления передачи от голоса диспетчера. Функция H323Direction позволяет принудительно переключить направление звука. Если transmit=true блок СОС-95 слушает диспетчера. Если transmit=false диспетчер слушает блок. Возвращает true если направление переключено успешно. Типовое применение данной функции – переключения направления связи для проигрывания звука с компьютера диспетчера в переговорный блок СОС-95. **Примечания:** - Эта функция работает только при разговоре с блоком СОС-95 - Эту функцию надо вызывать после установления соединения с блоком СОС-95 - После вызова этой функции режим автопереключения направления звука от голоса диспетчера выключается - Данная функция подключается к шлюзу sos95gw и использует команды SETATX / SETARX. Поэтому, для ее успешной работы, нужны версия шлюза 1.3.0 или старше.
function H323LoadMixerProfile(Profile : Integer=0): Boolean	Микшер в АРМ LanMon имеет два профиля настроек: основные и альтернативные. Основные настройки используются для разговора, а альтернативные для других

Описание	Комментарии
	целей. Например, для проигрывания звука в переговорное устройство в лифте. Функция H323LoadMixerProfile выполняет загрузку выбранного профиля настроек микшера. Profile=0 - основные настройки. Profile=1 – альтернативные настройки. Возвращает true если настройки загружены успешно. **Примечания:** - После старта АРМ LanMon всегда загружены основные настройки.
function H323TransferCall(Abonent: Variant): Boolean	Переадресация текущего звонка указанному абоненту. Если задан аргумент типа integer – это порядковый номер абонента из записной книжки, начиная с нуля. Если задан аргумент типа string – это узел или IP адрес вызываемого абонента. Данная функция выполняется, только если разговор уже идет. Функция возвращает true если идет разговор и false в противном случае.
procedure H323WindowSelectView(View: Integer=0)	Выбор вкладки в окне разговора. View – номер вкладки с нуля: 0 – основная вкладка со списком абонентов 1 - дополнительная вкладка с объектом ActiveX
procedure ShowAudioArchive(bShow: Boolean=True)	Показать или закрыть окно просмотра архива звукозаписей.
function H323SetTransferParty(Party: String): Boolean	Установить адрес для переадресации входящего вызова при состоянии «занято». Может быть вызвана только в обработчике *OnH323* с типом события *hm323Ring2* для переадресации вызова.

Коды ошибок H323LastError()

Текстовое описание ошибки	Код ошибки
ОК	0
Неверно указан адресат !	1
Удаленный абонент повесил трубку...	2
Удаленный абонент прервал звонок...	3
Удаленный абонент отказался от разговора…	4
Соединение перегружено	5
Удаленный абонент не ответил на звонок...	6
Сбой соединения	7
Не найден общий кодек	8
Мы не приняли входящий звонок	9
Входящий звонок отвергнут	10
Привратник не нашел указанного абонента	11
Звонок прерван ! (Не хватает полосы канала)	12
Удаленный абонент недостижим	13
Удаленный абонент сейчас отключен	14
На удаленной стороне нет телефона	15

Текстовое описание ошибки	Код ошибки
Ошибка звонка !	16

Обработчик событий IP телефонии

Для упрощения работы с асинхронными функциями IP телефонии, в программе можно определить обработчик событий. Для вставки обработчика в текст программы, в редакторе программ выберите в меню «События / события от клиента IP телефонии». При этом в текст программы будет вставлена процедура-обработчик:

```
void OnH323(int Type, String Info1, String Info2)
{
}
```

где, Type – тип события, Info1, Info2 – дополнительная информация о событии.

Возможные типы событий приведены в следующей таблице:

Type	Info1	Info2	Описание
hm323OK			Успешное завершение звонка
hm323Error	Текстовое описание ошибки	Код ошибки	Произошла ошибка.
hm323Busy			При звонке: абонент занят
hm323Stat	Статистика по разговору		Выбран пункт меню «Получить статистику разговора»
hm323Call	Имя абонента	Адрес абонента	Произошло успешное установление голосового соединения при исходящем или входящем звонке
hm323Ring	Имя абонента	Адрес абонента	Получен входящий вызов (приходит один раз)
hm323End			Разговор завершен
hm323Forwarded	Алиас того, кому переадресовали вызов	Адрес того, кому переадресовали вызов	Звонок был успешно переадресован.
hm323AppInfo	Информация об удаленной стороне		Информация об удаленном приложении, с которым мы разговариваем. Приходит после hm323Call
hm323CallCmd	Номер абонента в списке		Оператор нажал кнопку «Позвонить» в окне разговора
hm323Ring2	Имя абонента	Адрес абонента	Получен входящий вызов, когда идет разговор (приходит один раз). Если в этом обработчике вызывать H323SetTransferParty() с указанием адреса, то будет произведено перенаправление вызова.

Функции по работе с трендами (устарело)

Эти функции оставлены для обеспечения совместимости старых проектов.

Описание	Комментарии
procedure TrendClear	Удалить все графики в окне вывода графиков и сбросить назначенное увеличение (Zoom). Удаляются тренды, добавленные вызовами функции *TrendAssign* Рекомендуется использовать эту процедуру перед отображением новых графиков.
function TrendAssign(Trend: Variant; GraphIndex: Integer; ArcIndex: Integer=1): Integer	Отобразить тренд на один из 10 возможных графиков в окне просмотра графиков. ***Trend*** – название тренда. Если Trend равен пустой строке «» - это означает: спрятать указанный график. ***GraphIndex*** - индекс графика, на который отображаем тренд от 0 до 9. ***ArcIndex*** - опциональный, аргумент. Это номер архивного файла тренда от 1 до 99. Если третий аргумент отсутствует – он считается равным единице, т.е. берутся текущие значения тренда. При ошибке функция возвращает ноль.
procedure TrendShowWindow(Show: Integer)	Показать/скрыть окно вывода графиков. Аргумент Show: 0-скрыть 1-показать
procedure TrendSetCaption(Caption: String)	Установить строку заголовка окна графиков.
function TrendLoadParam(File: String): Integer	Загрузить файл с настройками параметров окна просмотра графиков. Файл настроек имеет расширение .grf и должен сохраняться в поддиректорию проекта \TEMPLATE\. Аргумент функции – имя файла .grf. Если полный путь не задан – подразумевается поддиректорию проекта \TEMPLATE\. Создание файла настроек выполняется из окна настройки параметров графиков в режиме редактирования проекта.
function TrendCreateFromSQL(Trend: String; Number: Integer; Sql: String; Params: Variant; ADOConnection: TADOConnection): Integer	Создание трендовой переменной из данных, накопленных в SQL сервере. Выполняется указанный SQL запрос и используются первые три колонки его результата. Первая колонки интерпретируется как время, вторая как значение, третья как состояние канала (качество). Функция создает файл тренда в поддиректории проекта \TREND\ Имя файла совпадает с именем трендовой

Описание	Комментарии
	переменной. Далее можно использовать функции ***TrendAssign*** и ***TrendShowWindow*** для показа тренда. Параметры: ***Trend*** – имя трендовой переменной. Должно содержать только символы, разрешенные в именах файлов. ***Number*** – номер периода накопления (1 – текущий период). ***Sql*** – текст SQL запроса к базе данных. Параметры запроса предваряются двоеточием. ***Params*** – массив параметров запроса. Должен быть массивом Variant даже если надо передать одно значение ! Если параметры не используются можно передать ноль - 0. ***ADOConnection*** – ссылка на подключение к базе данных через ADO. Можно указать одно из системных подключений ***SystemADOConnection***, ***SystemADOConnection1***, ***SystemADOConnection2***. Системные подключения настраиваются в настройках проекта. Если указать ***nil*** то будет ***использоваться подключение SystemADOConnection***. Функция возвращает число строк в полученном результате выполнения запроса >= 0. Если возвращаемое значение меньше нуля – это код ошибки: -1 "Трендовая переменная не задана !"; -2 "Неверный номер периода накопления ! (должен быть от 1 до 99)"; -3 "SQL запрос не задан !"; -4 "Недостаточно параметров для SQL запроса !"; -5 "Ошибка подключения к SQL серверу ! (Проверьте настройки ADO в параметрах программы.)"; -6 "Ошибка создания файла тренда !"; -7 "Произошло неизвестное исключение !"; -8 "Произошло исключение ! Описание ошибки записано в lanmon.log"; -9 “В полученном результате SQL запроса меньше 2-х колонок !”
function TrendCreateFromLocal(Trend: String; Number: Integer; A1,A2,A3,A4: Word; Start, End: TDateTime): Integer	Создание трендовой переменной из данных, накопленных локально при работе АРМ в поддиректории .\LOG\ проекта. Выборка данных производится по каналу с адресом A1.A2.A3.A4

Описание	Комментарии
	Функция создает файл тренда в поддиректории проекта \TREND\ Имя файла совпадает с именем трендовой переменной. Далее можно использовать функцию ***TrendAssign*** и ***TrendShowWindow*** для показа тренда. Параметры: ***Trend*** – имя трендовой переменной. Должно содержать только символы, разрешенные в именах файлов. ***Number*** – номер периода накопления (1 – текущий период). ***A1,A2,A3,A4*** – адрес канала, для которого надо запросить данные. ***Start,End*** – Интервал времени для запроса данных. Функция возвращает число найденных записей в полученном результате выполнения запроса >= 0. Если возвращаемое значение меньше нуля – это код ошибки: -1 "Трендовая переменная не задана !"; -2 "Неверный номер периода накопления ! (должен быть от 1 до 99)"; -3 "Задан неверный адрес канала"; -4 "Начальное время интервала больше конечного"; -5 "Ошибка создания файла тренда !"; -6 "Произошла неизвестная ошибка !"; -7 "Произошло исключение ! Смотрите протокол lanmon.log";
procedure TrendLine(Trend: String; Value: Extended; Number: Integer)	Создание тренда – линии из двух точек. Значения точек по оси времени берутся минимальное и максимальное из графиков, уже назначенных функцией ***TrendAssign.*** ***Trend*** – имя трендовой переменной. Должно содержать только символы, разрешенные в именах файлов. ***Number*** – номер периода накопления (1-99 единица означает текущий период накопления). ***Value*** – значение обоих точек по вертикальной оси. После создания тренда его надо отобразить в один из графиков функцией ***TrendAssign.*** Данная процедура используется для рисования горизонтальных границ в окне графиков.
procedure TrendGetMinMax(var MinValue: Extended; var MaxValue: Extended)	Получение минимального и максимального значений по всем графикам, отображенным функцией ***TrendAssign***.

Описание	Комментарии
void OnTrendWindowButtonClick(int Button, int Info)	Обработчик события по нажатию клавиш «Предыдущий период накопления» и «Следующий период накопления» в окне просмотра трендов. ***Button*** – номер нажатой кнопки. 1 - «Предыдущий период накопления». 2 - «Следующий период накопления». ***Info*** – Для кнопок №1 и №2 - номер требуемого периода накопления. Данный обработчик предназначен для просмотра динамически генерируемых трендов. Например, для трендов, созданных функцией ***TrendCreateFromSQL***. При нажатии кнопки «Предыдущий период накопления» нужный тренд может быть динамические сгенерирован. Обработчик вызывается ТОЛЬКО есть тренд назначен функцией ***TrendAssign***. Для добавления обработчика данного события в код программы в редакторе программ выберите пункт меню «События / Нажатие клавиши в окне просмотра графиков».
function TrendGetLastError: String	Получения текстового описания ошибки, произошедшей при последнем вызове одной из функции Trend… Если ошибки не было – возвращается пустая строка.

Функции по работе с графиками

Функция	Описание
procedure Trend2Clear	Удалить все графики из окна графиков. Пример на C++ скрипте: *Trend2Clear();*
function Trend2Add(Address: String; Expr: String=''; Mask: String=''): Boolean	Добавить график в окно графиков для канала с адресом Address. Для Address можно указывать адреса каналов тип 1 и 2. В параметре Expr можно указать выражение для пересчета всех значений канала. В параметре Mask можно задать маску форматирования значения. Временной интервал может быть предварительно установлен функцией *Trend2SetPeriod*. Возвращаемое значение: true – успешно, false – ошибка. Пример на C++ скрипте:

Функция	Описание
	/ Добавить график по каналу тип 2 с адресом 1100150_bkd_A0_Usos в список графиков Все значения канала умножить на 2. Изменить маску форматирования значения на "%.1f" */* *Trend2Add("1100150_bkd_A0_Usos", "value * 2.0", "%.1f");* */* Добавить график по каналу тип 1 с адресом 1.1.4.2 в список графиков */* *Trend2Add("1.1.4.2");*
function Trend2SetLimits(Address: String, LoLimit, HiLimit: Extended): Boolean	Добавить график в окно графиков для канала с адресом Address и с указанием верхней HiLimit и нижней LoLimit границ по оси Y. Если такой график уже есть - производится установка его цвета. Для Address можно указывать адреса каналов тип 1 и 2. Возвращаемое значение: true – успешно, false – ошибка. Пример на C++ скрипте: *Trend2SetLimits("1100150_bkd_A0_Usos", -1, 12);*
function Trend2SetColor(Address: String, Color: Integer): Boolean	Добавить график в окно графиков для канала с адресом Address и с указанием цвета. Если такой график уже есть - производится установка его цвета. Для Address можно указывать адреса каналов тип 1 и 2. Для Color можно указывать константы цвета с префиксом *cl* или указать цвет в кодировке *RGB (*см. функцию *RGB())*. Возвращаемое значение: true – успешно, false – ошибка. Пример на C++ скрипте: *// Установить голубой цвет графика* *Trend2SetColor("1100150_bkd_A0_Usos", clBlue);*
function Trend2AddLine(Value: Extended): Boolean	Добавить график – горизонтальную линию со значением Value.
procedure Trend2SetPeriod(From, To: TDateTime)	Установить период построения графиков. Пример на C++ скрипте: *// Установить временной интервал графиков…* *TDateTime t = Now();* *// Сегодня с ноля часов по настоящий момент* *Trend2SetPeriod(int(t), t);* *// За вчерашние сутки* *Trend2SetPeriod(int(t)-1, int(t));*
procedure Trend2ShowWindow(Show: Boolean=True)	Показать или скрыть окно графиков.

Функция	Описание
	Пример на C++ скрипте: *// Показать окно графиков* *Trend2ShowWindow(true);* *// Скрыть окно графиков* *Trend2ShowWindow(false);*
function Trend2LoadParam(File: String): Boolean	Загрузить перечень графиков, интервал времени и содержимое вкладки «**Настройки**» из файла **ini**. Если полный путь не задан, то файл ищется в папке проекта **template**. Пример на C++ скрипте: *// Загрузить настройки из файла \template\my.ini* *Trend2LoadParam ("my.ini");*

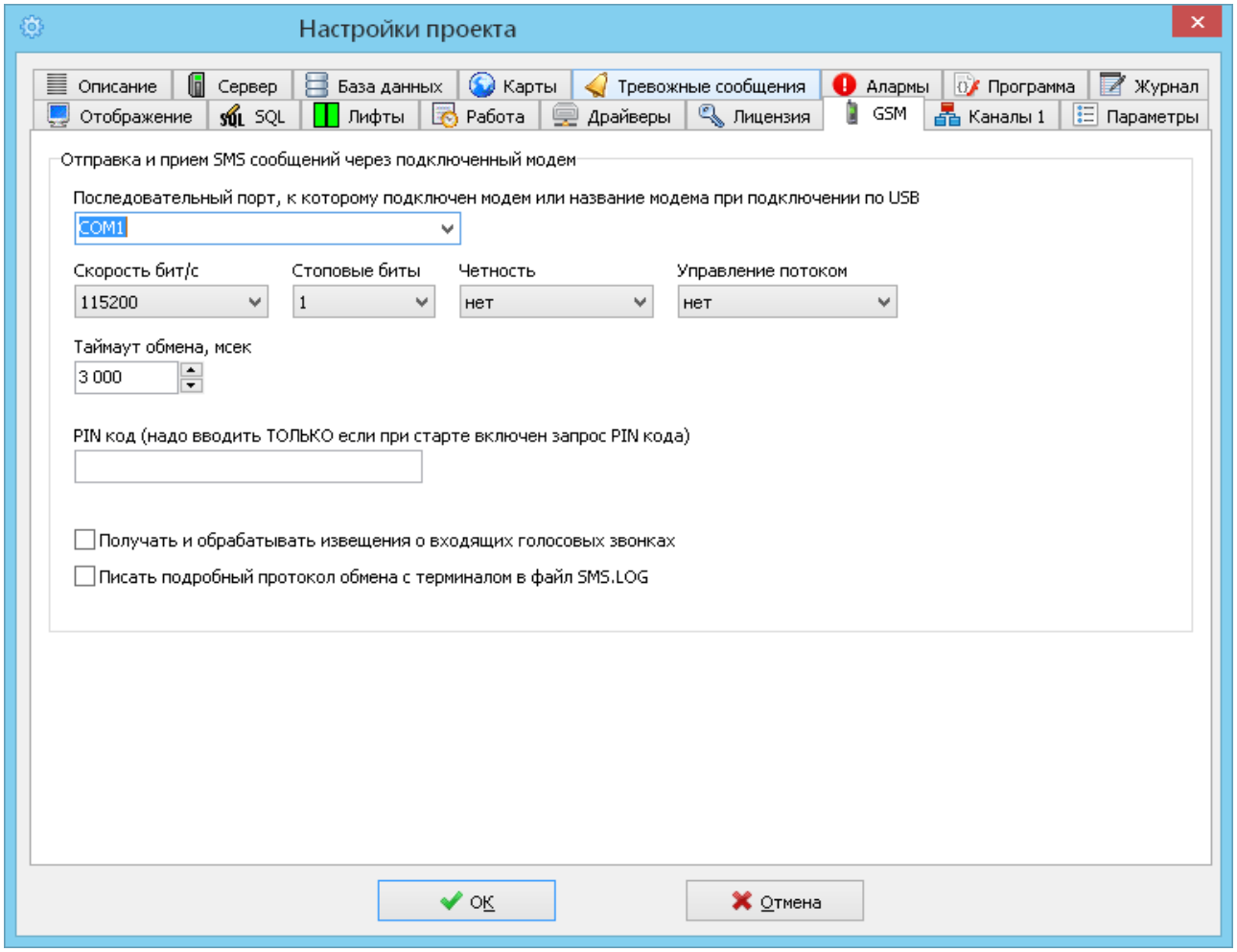
Функции для работы с драйверами устройств

Описание	Комментарии
procedure DriverShow(Show: Boolean=True)	Показать или скрыть окно драйверов оборудования. В режиме выполнения проекта функции настройки, добавления и удаления каналов недоступны.
function DriverChannelInfo(A1,A2,A3,A4: Word; Delim: Char='\x9'): String	Получить дополнительную информацию о канале от драйверов оборудования. В качестве входного параметра надо задать адрес канала. Информация возвращается в виде текстовой строки. Можно задать разделитель для полей (по умолчанию это символ табуляции). Если драйверы оборудования не используются, функция возвращает пустую строку. Пример на C++Script: *String s = DriverChannelInfo(1,1,2,5);* *if(Length(s) > 0) LMShowMessage(s);* *else LMShowMessage("Информация об этом канале недоступна.");*
function DriverCount(type: Integer=0): Integer	Получить количество драйверов в зависимости от параметра. Возможные значения параметра type: 0 – возвращается общее (загруженное на данный момент) кол-во драйверов 1 – кол-во драйверов, прошедших начальную инициализацию (DevOpen выполнена). Таким драйверам уже можно посылать команды управления и т.п. 2 – кол-во драйверов, успешно работающих (статус «ОК»)

Функции модуля приема и отправки SMS сообщений в сети GSM

Модуль GSM предназначен для приема и посылки SMS сообщений и регистрации входящих голосовых вызовов. Модуль работает только с Siemens совместимым мобильным телефоном или мобильным терминалом, подключенным к компьютеру по последовательному порту RS232 или по USB с эмуляцией последовательного интерфейса. Модуль GSM управляется функциями, доступными из программы на скрипте.

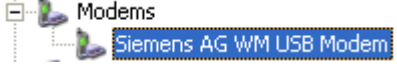
Модуль GSM настраивается в настройках проекта на вкладке «**SMS**»:



Прежде всего, надо указать последовательный COM порт компьютера (RS-232), к которому подключен мобильный терминал. При использовании подключения модема через USB необходимо указать связанный с ним виртуальный COM порт.

Для подключения GSM терминала по USB:

1. установите драйвер USB, поставляемый в комплекте с терминалом;
2. подключите терминал к компьютеру при помощи USB шнура;
3. убедитесь, что в списке системных устройств, в разделе «модемы» появился наш терминал;



4. в поле последовательного порта укажите название модема «**Siemens AG WM USB Modem**»

В поле «**PIN код**» укажите PIN код для используемой SIM карты. Если на SIM карте запрос PIN кода отключен, то это поле можно оставить пустым.

Чтобы включить регистрацию входящих голосовых вызовов установите галочку «Получать и обрабатывать извещения о входящих звонках».
Работа модуля GSM протоколируется в отдельном файле **sms.log** Этот файл находится в директории проекта.
Рекомендуется использовать мобильный терминал – Siemens ES75 с подключением к компьютеру по интерфейсу RS-232 или USB.
В комплект поставки АРМ LanMon входит проект «**Пример работы с сообщениями SMS**», который демонстрирует работу с модулем.

Для приема и отправки SMS сообщений из кода скрипта используются объекты типа TSMS. Объект TSMS имеет следующие свойства и методы:

Описание	Комментарии
property String Telephone	Номер телефона куда отправлять SMS или с которого пришло SMS сообщение. Номер необходимо указывать полностью, включая код страны и код сети. Например: +7-903-123-45-67 или 79031234567 Все символы кроме цифр игнорируются.
property String Text	Текст сообщения для отправки или принятое сообщение. Допускается кириллица. Если в сообщении нет русских букв – оно отправляется в семибитной кодировке, содержащей только латинский алфавит. Максимальная длина такого сообщения 160 символов. При наличии русских букв сообщение отправляется в кодировке юникоде (2 байта на символ). В этом случае, максимальная длина сообщения составляет 70 символов.
property bool Flash	True – посылать автоматически всплывающее сообщение. False – посылать обычное SMS сообщение. Данное свойство действует только на посылку сообщения. Данное свойство действует, только если сообщение (свойство Text) не содержит русских букв.
property bool Transliterate	Если данное свойство имеет значение true, то при посылке русские буквы в сообщении (свойство Text) перекодируются в латиницу по правилам ГОСТ. Максимальная длина такого сообщения 160 символов.

Для получения зарегистрированных входящих звонков используются объекты типа TRing. Объект TRing имеет следующие свойства и методы:

Описание	Комментарии
property String Telephone	Номер телефона, с которого поступил голосовой вызов. Например: «89031234567»
property TDateTime DateTime	Дата и время регистрации входящего звонка
property Boolean International	True – Если номер телефона в поле Telephone представлен в интернациональном формате (номер начинается с кода страны, например с +7) False – Если номер телефона в поле Telephone представлен в местном формате.
property Integer Count	Количество поступивших извещений RING до отбоя звонка. Извещение RING поступает, примерно, каждые 5 секунд.

Модуль управляется через вызов следующих функций скрипта:

Описание	Комментарии
function SMSSend(p: TSMS): Boolean	Отправка SMS сообщения. Объект SMS сообщения (класс TSMS) должен быть предварительно создан. Функция возвращает значение *true* если мобильный терминал подключен к указанному в настройках порту, отвечает на команды и сообщение поставлено в очередь на отправку. Возвращает *false* в противном случае. Фрагмент кода для отправки SMS сообщения на C++ скрипт: *TSMS p = new TSMS;* *p.Telephone = "+7-903-123-45-67";* *p.Text = "Hello, world";* *p.Flash = false;* *p.Transliterate = false;* *if(SMSSend(p))* * print("Сообщение поставлено в очередь на отправку.");* *else* * print("Ошибка отправки сообщения !");* *delete p;*
function SMSCount: Integer	Модуль автоматически принимает входящие SMS сообщения и сохраняет их во внутреннем буфере. Функция *SMSCount* позволяет получить количество принятых сообщений. Если функция вернула ноль – ни одного сообщения не принято. Для приема SMS сообщений программа должна периодически вызывать эту функцию и если результат больше нуля – принять сообщение функцией *SMSReceive*.
function SMSSendCount: Integer	Функция *SMSSendCount* позволяет получить количество сообщений, поставленных в очередь на отправку функцией SMSSend, но еще не отправленных в эфир.

Описание	Комментарии
function SMSReceive(p: TSMS): Boolean	Прием SMS сообщения. Объект SMS сообщения (класс TSMS) должен быть предварительно создан. Пример кода для приема SMS сообщения на C++ скрипт: *TSMS p = new TSMS;* *if(SMSReceive(p))* *{* *print("Принято новое SMS сообщение !");* *print("С номера: " + p.Telephone);* *print("Текст: " + p.Text);* *}* *delete p;*
function RingCount: Integer	Модуль автоматически регистрирует факт поступления входящего звонка и сохраняет все параметры звонка во внутреннем буфере. Функция *RingCount* позволяет получить количество зарегистрированных звонков. Если функция вернула ноль – ни одного звонка не поступало. Для получения информации о звонках, программа должна периодически вызывать эту функцию и если результат больше нуля – вызвать функцией *RingReceive*.
function RingReceive(p: TRing): Boolean	Получение информации о входящем звонке. Объект звонка (класс TRing) должен быть предварительно создан. Пример кода для приема звонка на C++ скрипт: *TRing p = new TRing;* *if(RingReceive(p))* *{* *print("Принят входящий звонок !");* *print("С номера: " + p.Telephone);* *}* *delete p;*
function GSMState: Integer	Получение текущего состояния модуля GSM. Функция может возвратить одно из следующих значений: 0-модуль не инициализирован (не используется) 1-ожидание открытия порта RS232 для обмена с терминалом 2-ожидание инициализации терминала 3-работа в штатном режиме (прием и отправка SMS) 4-модуль GSM завершил свою работу
function ATCmd(cmd: String; var answer: String; timeout: Integer; waiting: String='OK'): Boolean	Выполнить AT команду GSM терминала **cmd**. Ответ ожидается в течении **timeout** мсек и помещается в **answer**. Ожидаемый ответ на команду указывается в параметре **waiting** (по умолчанию 'OK'). Функция возвращает **true** если AT команда выполнена успешно и **false** в противном случае.

Для работы модуля приема SMS сообщений должна быть приобретена отдельная лицензия. Эта лицензия может храниться в локальном USB ключе или на сервера LanMon (в этом случае сервер должен быть версии 3.32 или старше).

Функции отправки сообщений по электронной почте

АРМ LanMon начиная с версии 4.13 может отправлять сообщения по электронной почте, используя скриптовые функции. Для этого в программе на скрипте доступен специальный объект типа TEMAIL, описывающий отправляемое письмо.

Описание	Комментарии
property String FromName	Имя отправителя письма, которое увидит получатель Например: Объект ВЗУ-78 Может быть пустой строкой (допускается не заполнять)
property String FromAddress	Адрес электронной почты отправителя (От кого письмо) Например: user2016@yandex.ru
property String EMailAddresses	Адрес электронной почты получателя или получателей (Кому письмо) Например: User@mail.ru или несколько адресов user@mail.ru; user@gmail.com
property String Subject	Тема письма Например: Автоматическое сообщение об аварии Может быть пустой строкой (допускается не заполнять)
property String Text	Текст сообщения для отправки Например: Авария насоса 5 !
property String Host	Адрес почтового SMTP сервера Например: smtp.yandex.ru
property Integer Port	Порт почтового сервера Например: 465
property String Username	Логин доступа к почтовому серверу Например: user2016
property String Password	Пароль доступа к почтовому серверу Например: mypassword
property Integer ConnectTimeout	Таймаут отправки почтового сообщения в миллисекундах Например: 30000

Описание	Комментарии
	Допускается не задавать – будет использовано значение по умолчанию
property Boolean UseSSL	Использовать SSL шифрование Например: True
property Integer Debug	Разрешение вывода отладочных сообщений в файл протокола: «C:\ProgramData\LanMon4\email.log» Например: 0 Допускается не задавать – отладочные сообщения запрещены. Возможные значение: 0 – не выполнять запись сообщений в файл 1 – общая информация о письме и результат –сообщение почтового сервера 2 – то же что и 1 с добавлением информации о почтовом сервере 3 - то же что и 2 с добавлением текста письма

Работа с электронной почтой выполняется при помощи следующих функций скрипта:

Описание	Комментарии
procedure SendMail(p: TEMAIL;f: TStringList)	Отправка email сообщения. Объект p сообщения (класс TEMAIL) должен быть предварительно создан и заполнен данными письма. После вызова функции SendMail объект p следует удалить. Второй параметр f – это набор строк, содержащий имена файлов, которые следует отправить по электронной почте (приложения к письму). Используйте nil если вложения не требуются
function SendMailResult: String	Возвращает результат отправки email сообщения в виде текстовой строки

Примеры отправки почтового сообщения на языке C++Script

Пример отправки сообщения через почтовый сервер с SSL шифрованием без приложения файлов:

```
{
	//Создать письмо
	TEMAIL p=new TEMAIL;
	//От кого
	p.FromName="";
	p.FromAddress="user2016@yandex.ru";
	p.Subject="Автоматическое сообщение с объекта 35";
	//Кому
	p.EMailAddresses="user@mail.ru";
	p.Text="Авария насоса 4 !";
	//Настройка сервера перед отправкой
```

```
        p.Host="smtp.yandex.ru";
        //465 – порт с SSL шифрованием
        p.Port=465;
        p.Username="user2016";
        p.Password="mypassword";
        //Использовать SSL
        p.UseSSL=True;
        //Отладка
        p.Debug=1;
        //Отправить письмо без вложений
        SendMail(p,nil);
        //Удалить письмо
        delete p;
}
```

Пример отправки сообщения через почтовый сервер без SSL шифрования и без приложения файлов:

```
{
        //Создать письмо
        TEMAIL p=new TEMAIL;
        //От кого
        p.FromName="";
        p.FromAddress="user2016@yandex.ru";
        p.Subject="Автоматическое сообщение с объекта 35";
        //Кому
        p.EMailAddresses="user@mail.ru";
        p.Text="Авария насоса 4 !";
        //Настройка сервера перед отправкой
        p.Host="smtp.yandex.ru";
        //25 – порт без SSL шифрования
        p.Port=25;
        p.Username="user2016";
        p.Password="mypassword";
        //Не использовать SSL
        p.UseSSL=False;
        //Отладка
        p.Debug=1;
        //Отправить письмо без вложений
        SendMail(p,nil);
        //Удалить письмо
        delete p;
}
```

Пример отправки сообщения через почтовый сервер с SSL шифрованием и приложением файлов:

```
{
        //Создать список посылаемых файлов ATTACHMENTS
        TStringList att=new TStringList;
        //Добавим 2 файла
        att.Add("D:\\Tmp\\ScreenShot.png");
```

```
	att.Add("D:\\Tmp\\Export.xls");
	//Создать письмо
	TEMAIL p=new TEMAIL;
	//От кого
	p.FromName="";
	p.FromAddress="user2016@yandex.ru";
	p.Subject="Автоматическое сообщение с объекта 35";
	//Кому
	p.EMailAddresses="user@mail.ru";
	p.Text="Авария насоса 4 !";
	//Настройка сервера перед отправкой
	p.Host="smtp.yandex.ru";
	//465 – порт с SSL шифрованием
	p.Port=465;
	p.Username="user2016";
	p.Password="mypassword";
	//Использовать SSL
	p.UseSSL=True;
	//Отправить письмо с вложениями
	SendMail(p,att);
	//Удалить письмо
	delete p;
	//Удалить список файлов
	delete att;
	//Пример вызова функции получения строки результата отправки почты
	Print(SendMailResult());
}
```

Пример отправки почтового сообщения на языке PascalScript

Пример отправки сообщения через почтовый сервер с SSL шифрованием и приложением файлов:

```
{ Это основная процедура скрипта main }
var
	att: TStringList;
	p:   TEMAIL;
begin
	att := TStringList.Create;
	{Список посылаемых файлов}
	att.Add('D:\Tmp\04_Измеритель.png');
	att.Add('D:\Tmp\Export.xls');
	{Письмо}
	p:= TEMAIL.Create;
	p.FromName:='АРМ-35';
	p.FromAddress:='user2016@yandex.ru';
	p.Subject:='Автоматическое сообщение с объекта 35';
	{кому}
```

```
        p.EMailAddresses:='user@mail.ru';
        p.Text:='Авария насоса 4 !';
        {Настройка сервера перед отправкой}
        p.Host:='smtp.yandex.ru';
        p.Port:=465;
        p.Username:='user2016';
        p.Password:='mypassword';
        {Использовать SSL}
        p.UseSSL:=True;
        {Отладка}
        p.Debug:=1;
        {Отправить письмо и заданные файлы}
        SendMail(p,att);
        {Удалить письмо}
        p.Free;
        {Удалить список файлов}
        att.Free;
        {Пример вызова функции получения строки результата отправки почты}
        Print(SendMailResult);
end.
```

Пример отправки почтового сообщения на языке BasicScript

Пример отправки сообщения через почтовый сервер с SSL шифрованием и приложением файлов:

```
dim att= new TStringList
dim p= new TEMAIL
'Список посылаемых файлов
att.Add("D:\Tmp\04_Измеритель.png")
att.Add("D:\Tmp\Export.xls")
'Письмо
p.FromName="АРМ-35"
p.FromAddress="user2016@yandex.ru"
p.Subject="Автоматическое сообщение с объекта 35"
'кому
p.EMailAddresses="user@mail.ru"
p.Text="Авария насоса 4 !"
'Настройка сервера перед отправкой
p.Host="smtp.yandex.ru"
p.Port=465
p.Username="user2016"
p.Password="mypassword"
'Использовать SSL
p.UseSSL=True
'Отладка
p.Debug=1
```

```
'Отправить письмо и заданные файлы
SendMail(p,att)
‘Удалить письмо
delete p
‘Удалить список файлов
delete att
‘Пример вызова функции получения строки результата отправки почты
Print(SendMailResult())
```

Пример отправки почтового сообщения на языке JScript

Пример отправки сообщения через почтовый сервер с SSL шифрованием и приложением файлов:

```
var att= new TStringList;
var p= new TEMAIL;
//Список посылаемых файлов
att.Add("D:\\Tmp\\04_Измеритель.png");
att.Add("D:\\Tmp\\Export.xls");
//Письмо
p.FromName="APM-35";
p.FromAddress="user2016@yandex.ru";
p.Subject="Автоматическое сообщение с объекта 35";
//кому
p.EMailAddresses="user@mail.ru";
p.Text="Авария насоса 4 !";
//Настройка сервера перед отправкой
p.Host="smtp.yandex.ru";
p.Port=465;
p.Username="user2016";
p.Password="mypassword";
//Использовать SSL
p.UseSSL=True;
//Отладка
p.Debug=1;
//Отправить письмо и заданные файлы
SendMail(p,att);
//Удалить письмо
delete p;
//Удалить список файлов
delete att;
//Пример вызова функции получения строки результата отправки почты
Print(SendMailResult());
```

Пояснения к работе функции отсылки e-mail сообщений

1. Если требуется посылать файлы, то сначала нужно создать список имен посылаемых файлов как экземпляр класса TStringList, и заполнить его именами посылаемых файлов с полными путями.

Если задаётся несуществующий файл, то это не приведёт к ошибке – просто данный файл не буде послан – нечего посылать.

2. Создать экземпляр класса TEMAIL, и заполнить его всеми необходимыми данными. Любая опечатка или неточность в существенных полях работы с сервером приведёт к невозможности подключиться к почтовому серверу или отправить письмо. Следует обратить внимание на настройку шифрования UseSSL. Большинство современных почтовых SMTP серверов используют шифрование и порт сервера в этом случае отличается от стандартного порта 25 без шифрования. Следует предварительно поискать в сети Internet информацию о почтовом сервере, который используется для посылки сообщения: имя сервера, порт и использование SSL шифрования. Хорошая подготовка поможет сэкономить много времени.
3. Вызвать функцию *SendMail,* указав ей в качестве параметров письмо и список вложений. Если вложений нет, то допускается задать константу nil. Вызов этой функции является неблокирующим, т.е. эта функция не посылает письмо, а запускает новый программный поток, который и будет выполнять подключение к почтовому серверу и отправлять собственно письмо.
4. Удалить созданные динамически экземпляры классов TStringList и TEMAIL. Все необходимые действия программиста по отправке почты завершены – если имеется возможность, то письмо или письма будут отправлены.

Причины неудачного отправления почты могут быть следующими:

- Неправильно настроен почтовый сервер: *p.Host, p.Port, p.Username, p.Password*
- Неправильно настроено шифрование: *p.UseSSL*
- Неправильно задан email адрес получателя: *p.EMailAddresses*
- Нет подключения к сети Internet (выполните ping почтового сервера)
- Выбранный почтовый сервер не функционирует (выполните ping почтового сервера)

При выяснении ошибок можно использовать скриптовую функцию *SendMailResult*:

Print(SendMailResult());

Вызов возвращает результат отправки письма в виде строки. Пока возвращается пустая стока – выполняется работа с почтовым сервером. Сразу после завершения строка будет содержать ответ сервера о принятии письма, например, в случае успеха:

Ok: queued on smtp1m.mail.yandex.net as 1482221033-CFJ2bIVpRK-3qcWVWEV

Или в случае ошибки (неправильный логин или пароль доступа к серверу):

ERROR: Error: authentication failed:Invalid format.

Или в случае ошибки (неправильное имя сервера или порт):

ERROR: Connect timed out.

Удобным средством обнаружения ошибок является задание записи протокола в отладочный файл:

p.Debug=1;

Через некоторое время после начала отсылки сообщения при помощи фцнкции *SendMail* можно просмотреть текстовый файл, расположенный по пути:

C:\ProgramData\LanMon4\email.log

или

C:\Users\All Users\LanMon4\email.log

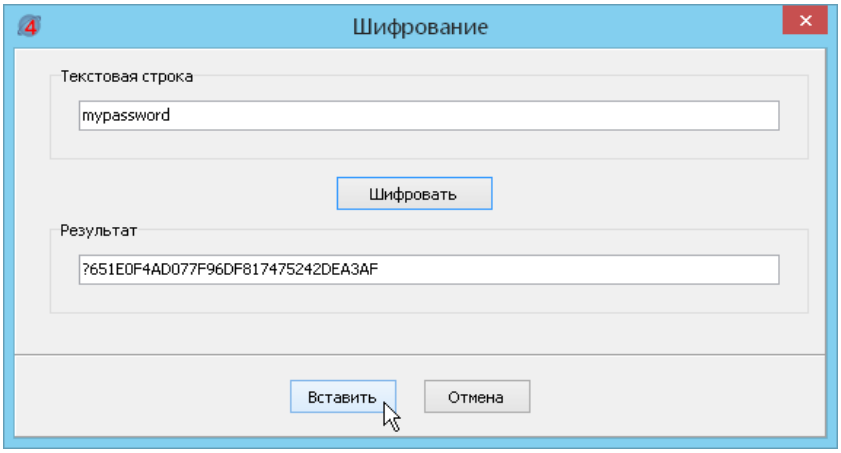
, который содержит отладочную информацию, например:

20.12.2016 11:36:33 Отправка почты по адресу: lanmon@mnppsaturn.ru
Почтовый сервер: smtp.yandex.ru:465
Использование SSL: Да
Сообщение: Авария насоса 4 !

20.12.2016 11:36:38 Результат: Ok: queued on smtp2p.mail.yandex.net as 1482222988-ZcKmyFIqN0-aRUC2MhY
Передача письма была выполнена за 5 секунд.

Шифрование полей объекта TEMAIL в программе Lanmon

Некоторые поля объекта TEMAIL, описывающего письмо, содержат приватную информацию, например, логин и пароль доступа к почтовому серверу. Если Вы хотите скрыть эти поля в тексте скриптовой программы, то следует их зашифровать. В тексте скриптовой программы будет храниться зашифрованная копия строк.
Для шифрования текста следует вызвать специальное окно, выбрав пункт меню «Вставить/Шифрованный текст…» окна редактора программы.
В открывшемся окне (показано на рисунке ниже) следует ввести нужный текст в поле «Текстовая строка» и нажать кнопку «Шифровать». В поле «Результат» появится зашифрованная копия введённого текста. Нажмите кнопку «Вставить» для вставления зашифрованного текста в текущую позицию редактирования текста скриптовой программы.



Таким образом можно зашифровать следующие поля объекта TEMAIL:

Описание	Комментарии
property String FromName	Имя отправителя письма, которое увидит получатель **Может быть зашифровано** Например: *FromName=”?84D53F1F11218228FFF2B539505FA794”*
property String FromAddress	Адрес электронной почты отправителя (От кого письмо) **Может быть зашифровано**
property String EMailAddresses	Адрес электронной почты получателя или получателей (Кому письмо) **Может быть зашифровано**
property String Username	Логин доступа к почтовому серверу **Может быть зашифровано**
property String Password	Пароль доступа к почтовому серверу **Может быть зашифровано**

Эти пять полей автоматически дешифруются при отправлении письма.
Например:
Следующий фрагмент скриптовой программы:

```
p.FromName="APM-29";
p.FromAddress="user2016@yandex.ru";
p.EMailAddresses="user@mail.ru";
p.Username="user2016";
p.Password="mypassword";
```

может выглядеть так:

```
p.FromName=" ?A222574C1EA50AC3889C186DF5BB217A";
p.FromAddress="?1BC37B70FB2CD6A56643086105D738CECCBFF5D66EFC5F80BB5F25EDA6
2E09F5 ";
p.EMailAddresses=" ?1A203F6F702071F36C69CA62D8988C31";
p.Username=" ?F203DD789629EDC211475BEA45CD059A";
p.Password=" ?4FC9C27B82D9CE6F56A2ECDF59D3D675";
```

При шифровании исходная строка дополняется случайными символами, поэтому при каждом нажатии кнопки «Шифровать» Вы можете получать различную зашифрованную строку, но при автоматической дешифрации случайные символы отбрасываются, остаётся только исходная строка.
Признаком шифрования является вопросительный знак в первом символе строки. Для шифрования используется алгоритм AES128.

Функции для управления хранителем экрана

Для управления хранителем экрана (заставкой) Windows APM LanMon предоставляет следующие функции:

Описание	Комментарии
function ScreenSaverIsActive: Boolean	Функция возвращает значение ***true*** если выполнение хранителя экрана разрешено и ***false*** если нет.
procedure ScreenSaverSetActive(Active: Boolean)	Функция разрешает (true) или запрещает (false) выполнение хранителя экрана.
function ScreenSaverGetTimeOut: Integer	Получение таймаута выполнения хранителя экрана в секундах.
procedure ScreenSaverSetTimeOut(TimeOut: Integer)	Установка таймаута выполнения хранителя экрана в секундах.
function ScreenSaverIsRunning: Boolean	Данная функция возвращает значение ***true*** если хранитель экрана выполняется и ***false*** если нет.
procedure ScreenSaverClose	Завершить выполняющийся хранитель экрана. Пример на C++ скрипте: *if(ScreenSaverIsRunning())* *ScreenSaverClose();*

Для запуска хранителя экрана используйте следующий код:

```
ScreenSaverSetActive(true);
ScreenSaverSetTimeOut(1);
```

Через одну секунду после выполнения этого кода хранитель экрана запустится. Для успешного запуска хранителя экрана он предварительно должен быть выбран и настроен в свойствах экрана.

Прочие функции

Описание	Комментарии
function InputString(Caption: String; var Value: String; Password: Boolean=False): Boolean	Функция осуществляет запрос ввода строки от оператора. Она получает следующие параметры: *Caption* – заголовок окна ввода строки *Value* – Значение строковой переменной *Value* выводится в поле ввода строки. Она же получает введенное значение. *Password* – если задано *true*, то вводимый текст заменяется на символ '*' Функция возвращает значение *true* если нажата клавиша «ОК» или *Enter* и *false* в противном случае.
function CheckPassword(Caption,Password,ErrorMess: String): Boolean	Функция осуществляет запрос у оператора пароля и его проверку. Она получает следующие параметры: *Caption* – заголовок окна ввода строки. Если задана пустая строка «», то в заголовке выводится «Введите пароль». *Password* – правильный пароль. Если введен указанный пароль функция возвращает значение *true*. *ErrorMess* – сообщение о неверно введенном пароле. Если задана пустая строка «», то при неверно введенном пароле выдается сообщение «Неверный пароль, доступ запрещен !».
function QuestionYesNo(Question: String; Caption: String=''; Position: TPosition=poOwnerFormCenter): Boolean	Функция осуществляет запрос у оператора подтверждения. Выводится окно с кнопками «ДА» «НЕТ». Функция получает следующие параметры: *Question* – текст запроса. *Caption* – заголовок окна. Если задана пустая строка «», то в заголовке выводится «Подтверждение». *Position* – задает положение окна на экране. Можно задать одно из следующих значений: *poDesigned* – в месте заданном разработчиками *poDefault* – размеры и позицию определяет Windows *poDefaultPosOnly* - позицию определяет Windows *poDefaultSizeOnly* - размеры определяет Windows *poScreenCenter* - по центру экрана *poDesktopCenter* - по центру рабочего стола *poMainFormCenter* - по центру главной формы *poOwnerFormCenter* – по центру текущей формы Если нажата кнопка «ДА» или клавиша «Enter» функция возвращает значение True. Если нажата

Описание	Комментарии
	кнопка «НЕТ» или клавиша «Esc» функция возвращает значение False.
function SendMessage(hWnd: Integer; Msg,wParam: Cardinal; lParam: Integer): Integer	Послать сообщение с кодом **Msg** окну с идентификатором **hWnd** или нескольким окнам. Функция возвращает управление после того, как сообщение будет обработано. Возвращаемое значение зависит от обработанного сообщения. Аналог одноименной функции Windows API.
function FindWindow(ClassName,WindowName: String): Integer	Получить идентификатор окна по имени класса и/или имени заголовка окна. Поиск производиться только среди окон верхнего уровня (дочерние окна не рассматриваются). Аналог одноименной функции Windows API. Чтобы рассматривать все классы окна надо указать **null**. Пример поиска окна по его заголовку на C++: *int handle;* *handle = FindWindow(null, "заголовок искомого окна");*

Дискретные алармы

Дискретные алармы предназначены для настройки оповещения на изменение дискретных величин: срабатывание охранных извещателей, переключение силового оборудования и т.п. Дискретный аларм можно создать в редакторе алармов. Для работы дискретного аларма необходимо привязать его к отображаемому объекту карты. При активизации алармов производится вызов обработчиков из программы на скрипте. Вызов обработчиков производится, только если обработчик определен в программе. Для добавления обработчика в программу используйте меню редактора программ: «События \ Сработка дискретного аларма тип 1» и «События \ Сработка дискретного аларма тип 2».

Обработчик на активизацию дискретного аларма тип 1 имеет следующие параметры:
int Alarm - номер дискретного аларма с 1.
WORD A1, A2, A3, A4 - адрес канала, который вызвал активизацию аларма.
bool & bAllow – разрешение активизации аларма. По умолчанию значение ***true***. Под активизацией аларма понимается появление окна алармов и включение проигрывания звукового оповещения.

Пример обработчика на активизацию дискретного аларма тип 1:

```
void OnDigitalAlarm(int Alarm, WORD A1, WORD A2, WORD A3, WORD A4, bool & bAllow)
{
        // Запретим активизацию аларма №27
        If( Alarm==27 ) bAllow=false;
        else bAllow=true;
}
```

Обработчик на активизацию дискретного аларма тип 2 имеет следующие параметры:
int Alarm - номер дискретного аларма с 1.
TMap map - ссылка на карту, где находится отображаемый объект, вызвавший активизацию аларма.
TMonControl control - ссылка на объект карты, вызвавший активизацию аларма.
bool & bAllow - разрешение активизации аларма (и появления окна алармов и звукового оповещения). По умолчанию значение ***true***.
bool & bVisual - разрешение появления окна алармов. По умолчанию значение ***true***.
bool & bSound - разрешение проигрывания звукового оповещения. По умолчанию значение ***true***.

Пример обработчика на активизацию дискретного аларма тип 2:

```
void OnDigitalAlarm2(int Alarm,
                     TMap map,
                     TMonControl control,
                     bool & bAllow,
                     bool & bVisual,
                     bool & bSound)
{
        // Запретим звуковое оповещение по активизацию аларма №27
        If( Alarm==27 ) bSound =false;
        else bSound =true;
}
```

Перечисления и множества

Скрипты поддерживает перечисления. Вы можете написать в скрипте:

```
Form1.BorderStyle := bsDialog;
Множества не поддерживаются. Тем не менее, вы можете оперировать элементами множества:
Font.Style := fsBold;  { Font.Style := [fsBold] в Delphi }
Font.Style := fsBold + fsItalic;        { Font.Style := [fsBold, fsItalic] }
Font.Style := 0;        { Font.Style := [] }
```

Массивы

Скрипт поддерживает все типы массивов: статические (одномерные, многомерные), динамические, вариантные. Вот пример скрипта, использующего три массива целых чисел, объявленных разным способом:

```
var
ar1: array[0..2] of Integer;
ar2: array of Integer;
ar3: Variant;
SetLength(ar2, 3);
ar3 := VarArrayCreate([0, 2], varInteger);
ar1[0] := 1;
ar2[0] := 1;
ar3[0] := 1;
```

Более подробный пример работы с массивами находится в файле .\program\samples\array.cpp

Многофайловые проекты

Вы можете разбивать большой скрипт на модули, подобно тому, как это делается в Object Pascal. Для использования модуля служит директива "uses" в паскале или директива “#include” в C++. Вот пример ее применения:

Файл unit1.pas:
```
uses 'unit2.pas';
begin
Unit2Proc('Hello!');
end.
```

Файл unit2.pas:
```
procedure Unit2Proc(s: String);
begin
        ShowMessage(s);
end;
begin
        ShowMessage('initialization of unit2...');
end.
```

Как видно, отличие от Object Pascal заключается в том, что мы указываем в uses имя файла с расширением в одинарных кавычках. Подключаемый файл должен иметь такую же структуру, как и основной. Код, заключенный в основной блок begin..end, будет выполнен при подключении модуля - это аналог секции initialization в Pascal.
В этом примере нельзя использовать unit1 из unit2 - это вызовет бесконечный цикл при попытке откомпилировать такой скрипт. Подобные ссылки невозможны, т.к. в скрипте нет аналогов паскалевским конструкциям interface/implementation.

Пользуясь директивой #language, можно писать многоязычные скрипты. Так, один модуль может быть написан на PascalScript, другой - на C++Script:

Файл unit1.pas:
```
uses 'unit2.pas';
begin
Unit2Proc('Hello from PascalScript!');
end.
```

Файл unit2.pas:
```
#language C++Script
void Unit2Proc(string s)
{
        ShowMessage(s);
}
{
```

```
    ShowMessage("unit2 initialization, C++Script");
}
```

Директива #language должна быть первой строкой в файле. Если директива присутствует, она перекрывает установленное значение типа скрипта.

Обучение работе со скриптами

Обучающие примеры программ на скрипте расположены в следующих поддиректориях:

- .\PROGRAM\samples\standart – базовые примеры программирования на разных языках;
- .\PROGRAM\samples\LanMon – примеры использования функций и классов АРМ LanMon на разных языках;

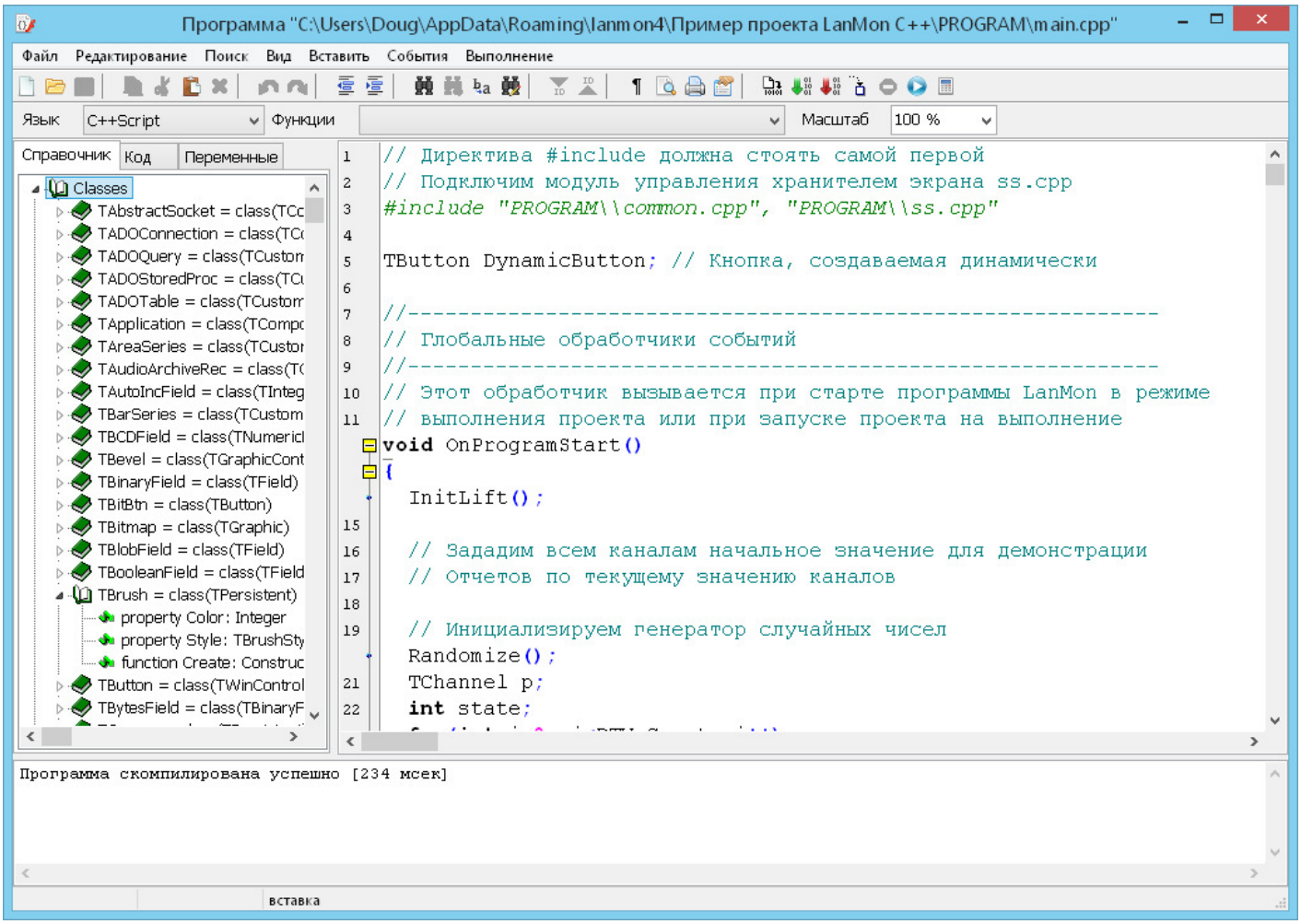
Откройте в редакторе программ одну из демонстрационных программ и выполните ее, нажав клавишу «F9».

Редактор программ в АРМ LanMon

Для написания и отладки программ в АРМ LanMon используется встроенный редактор программ. Редактор программ имеет следующие особенности:

- Редактор текста программы с подсветкой синтаксиса. Причем подсветка зависит от выбранного языка программирования.
- Список последних загруженных файлов доступен из меню «Файл».
- Быстрая навигация по тексту программы с помощью закладок: Ctrl+Shift+<цифра> - Установка закладки с номером 0..9 на текущей строке. Ctrl+<цифра> - Переход на установленную закладку.
- Быстрая навигация по тексту программы с помощью выбора процедуры или функции в выпадающем списке в левой части панели кнопок.
- Автоматическая вставка обработчиков глобальных событий в текст программы доступна из меню «События».
- Выявление ошибок в программе на этапе компиляции (кнопка панели «Компил.»).
- Пошаговое выполнение программы с просмотром списка текущих переменных и их значений. Возможность изменения значений переменных по ходу отладки программы.
- Возможность вычисления выражения на выбранном языке программирования (кнопка панели «Вычислить»). В выражении могут использоваться вызовы стандартных и определенных пользователем функций.

Редактор программ доступен только в режиме редактирования проекта. Редактор программ вызывается из меню «Программа \ Редактор программ…» или нажатием кнопки  на панели в главном окне. Вид окна редактора программ представлен на следующем рисунке.



В верхней части окна редактора программ расположено меню с доступом ко всем функциям редактирования и выполнения программы.
Ниже расположена панель с кнопками и выпадающими списками для управления наиболее часто используемыми функциями редактора.
Основную часть окна занимает редактор текста с подсветкой синтаксиса. Причем, подсветка синтаксиса зависит от выбранного языка программирования. Язык программирования выбирается в выпадающем списке на панели задач. В окне редактора текста есть контекстное меню, которое вызывается правой кнопкой мыши. В этом меню доступны дополнительные возможности, упрощающие редактирование кода программы.
В левой части окна расположены две вкладки: «Справочник» и «Переменные».
«Справочник» - это дерево всех классов, функций и глобальных переменных, доступных в программе. Любой элемент справочника можно перетащить мышью в редактор текста программы. В справочнике доступно контекстное меню (вызывается правой кнопкой мыши) с возможностью обновления дерева объектов.
На вкладке «Переменные» расположен список переменных программа и их значения. Список переменных постоянно обновляется в режиме пошаговой отладки программы. Значения переменных можно изменять по ходу отладки программы. В списке переменных доступно контекстное меню (вызывается правой кнопкой мыши) с дополнительными функциями.

В нижней части окна расположено окно отладочной печати. В этом окне отражается результаты компиляции и выполнения программы. Встроенная функция printf() производит печать текстовой строки в это окно, что удобно при отладке программы.
Внизу окна расположена строка статуса. Слева пишется время компиляции и выполнения программы в миллисекундах. Справа пишется полное имя файла, в котором хранится редактируемая программа. Символ «*» рядом с именем файла означает, что программа изменена в редакторе и требует сохранения. Предполагается, что программа в АРМ LanMon должна храниться в поддиректории.\PROGRAM\ текущего проекта.

Клавиши редактирования в редакторе программ.

Клавиша	Значение
Стрелки курсора	Перемещение курсора
PgUp, PgDn	Переход на предыдущую/последующую страницу
Ctrl+PgUp	Переход в начало текста
Ctrl+PgDn	Переход в конец текста
Home	Переход в начало строки
End	Переход в конец строки
Enter	Переход на следующую строку
Delete	Удаление символа в позиции курсора, удаление выделенного текста
Backspace	Удаление символа слева от курсора
Ctrl+Y	Удаление текущей строки
Ctrl+Z	Отмена последнего изменения (до 32 событий)
Shift+Стрелки курсора	Выделение блока текста
Ctrl+A	Выделить весь текст
Ctrl+U	Сдвиг выделенного блока на 2 символа влево
Ctrl+I	Сдвиг выделенного блока на 2 символа вправо
Ctrl+C, Ctrl+Insert	Копирование выделенного блока в буфер обмена
Ctrl+V, Shift+Insert	Вставка текста из буфера обмена
Ctrl+X, Shift+Delete	Перенос выделенного блока в буфер обмена
Ctrl+Shift+<цифра>	Установка закладки с номером 0..9 на текущей строке
Ctrl+<цифра>	Переход на установленную закладку
Ctrl+F	Поиск строки в тексте программы

Приложения

Расшифровка значений типа канала (свойство DTYPE)

Поле DTYPE	Размер поля данных VALUE	Тип канала	Пояснения
1	bit	BIT	0 или 1
2	byte	BYTE	0…255
3	int8	int8	-128…+127
4	int16	int16	-32768…+32767
5	int32	int32	-2147483648…2147483647
6	float	float	-3.4*10^38…3.4*10^38 (точность 7 знаков)
7	char[16]	char[16]	Строка длиной до 16 символов включительно (может не завершаться нулем)
8	word	WORD	0...65535
9	double	double	2.23*10^-308…1.79*10^308 (точность 15 знаков)
10	byte	Температура	-128…+127 °C
11	byte	Состояние КД	0-Норма 1-Срабатывание
12	byte[6]	Состояние датчика движения	VALUE[0] 0-Норма 1- Срабатывание 2-Норма левый 3- Срабатывание левый 4-Норма правый 5- Срабатывание правый --- VALUE [1]-признак наличия дополнительной информации: 0-“нет информации” --- 1-“есть только 1я амплитуда”: VALUE [2]-1я амплитуда VALUE [3]-порог --- 2-“есть 2 амплитуды и 2 частоты”: VALUE [2]-1я амплитуда (левая) VALUE [3]-1я частота (левая) VALUE [4]-2я амплитуда VALUE [5]-2я частота VALUE [6]-1ый порог (левый)

Поле DTYPE	Размер поля данных VALUE	Тип канала	Пояснения
			VALUE [7]-2ой порог ---------- 3-“есть только 1я амплитуда”: VALUE [2,3]-1я амплитуда (int16) VALUE [4,5]-порог (int16)
13	byte	Состояние ДД	0-Норма 1- Срабатывание 2-Отсутствие
14	byte	Фазный сигнал	0-Нет фазы 1-Есть фаза
15	byte[3]	Состояние ГД	0-Норма 1-Газ 2-Обрыв ЧЭ 3-Замыкание ЧЭ 4-Тест 5-Нет питания ---------- VAL[1]-признак наличия дополнительной информации: 0-“нет информации” ---------- 1-“есть только концентрация (% НКПР)”: VAL[2-5] – концентрация Float (4 байта)
16	byte	Насос	0 - Выключен 1 - Включен 2 - Затоплен 3 - Обесточен 4 – Есть вода включен 5 – Есть вода выключен
17	byte	Вентилятор	0 - Выключен 1 - Включен 2 - Обесточен
18	byte	Кнопка (канал управления)	0 – Выключен 1 – Включен 2 – Выключен, есть питание 3 – Включен, нет питания
20	byte	Датчик затопления	0-Норма 1- Затопление уровня 1 2- Затопление уровня 2 3- Затопление уровня 3 4- Затопление уровня 4

Поле DTYPE	Размер поля данных VALUE	Тип канала	Пояснения
21	byte	Состояние локальной охранной зоны	0-охрана снята 1-на охране 2-снят с охраны, срабатывание 3-на охране, срабатывание 4-снят с охраны, тревога 5-на охране, тревога
22	byte	Диагностика чего-либо	Качество работы в % (0-100)
24	byte[5]	Лифт «Сатурн»	VALUE[0] 0- "Нет данных" * 1- "Есть вызов" 2- "Нажата кнопка Стоп" 3- "Устройство защиты дифта: " (дописывается VAL[3]) 4- "Авария по сигналам" 5- "Кабина в движении" 6- "Дверь кабины открыта" 7- "Все в порядке" 8- "Выключен" * 9- "Нет ответа по СОС-95" * 10- "Снято питание лифта" 11- "Долго нет движения лифта" 12- "Вызов из МП" 13- "Блок БДК" 14- "Нет данных+Пассажир" * 15- "Есть вызов+Пассажир" 16- "Нажата кнопка Стоп+Пассажир" 17- "Остановлен БЗЛ+Пассажир" 18- "Авария по сигналам+Пассажир" 19- "Кабина в движении+Пассажир" 20- "Дверь кабины открыта+Пассажир" 21- "Все в порядке+Пассажир" 22- "Выключен" * 23- "Нет ответа по СОС-95+Пассажир" * 24- "Снято питание лифта" 25- "Долго нет движения лифта+Пассажир" 26- "Вызов из МП+Пассажир" 27- "Блок БДК" VALUE [1] – Тип лифта VALUE [2] – Маска ошибок VALUE [3] – Состояние лифта:

Поле DTYPE	Размер поля данных VALUE	Тип канала	Пояснения
			"Все в порядке",//0 "Нет движения на бол. скорости",//1 "Нет движения на мал. скорости",//2 "Устройство безопасности",//3 "Ошибка фаз АВС",//4 "Движение без двигателя",//5 "Команда из диспетчерской",//6 "Перегревание электродвигателя",//7 "",//8 "",//9 "",//10 "",//11 "",//12 "",//13 "",//14 ""//15 VALUE [4] – этаж Если VALUE [4]=0 – означает, что информация об этаже недоступна
25	byte	БГС	0 - Все в норме, нет вызова 1 - Есть вызов
26	byte	УИР-Р	0- Рычаг норма 1- Рычаг сдернут 2- Рычаг норма, вызов 3- Рычаг сдернут, вызов 4- Рычаг норма, разговор 5- Рычаг сдернут, разговор 6- Рычаг норма, ВПРАВО 7- Рычаг сдернут, ВПРАВО 8- Рычаг норма, вызов, ВПРАВО 9- Рычаг сдернут, вызов, ВПРАВО 10- Рычаг норма, разговор, ВПРАВО 11- Рычаг сдернут, разговор, ВПРАВО 12- Рычаг норма, ВЛЕВО 13- Рычаг сдернут, ВЛЕВО 14- Рычаг норма, вызов, ВЛЕВО 15- Рычаг сдернут, вызов, ВЛЕВО 16- Рычаг норма, разговор, ВЛЕВО 17- Рычаг сдернут, разговор, ВЛЕВО

Поле DTYPE	Размер поля данных VALUE	Тип канала	Пояснения
			18- Рычаг норма, ОБЕ 19- Рычаг сдернут, ОБЕ 20- Рычаг норма, вызов, ОБЕ 21- Рычаг сдернут, вызов, ОБЕ 22- Рычаг норма, разговор, ОБЕ 23- Рычаг сдернут, разговор, ОБЕ
254	byte	Карта	0 - карта снята с охраны оператором 1 - карта поставлена на охрану оператором 2 - карта снята с охраны с пульта 3 - карта поставлена на охрану с пульта 4 - карта снята с охраны автопилотом 5 - карта поставлена на охрану автопилотом
255	byte[12]	Оператор	VAL[0] - событие: 1 - запуск программы 2 - завершение программы 3 - начало смены оператора 4 - конец смены оператора 5 - подключение к главному серверу 6 - подключение к резервному серверу 7 - сервер отключился 8 - изменение конфигурации программы 9 - перезагрузка 10 - реакция на тревогу (подтверждение) 11 - нет реакции на дежурный режим 12 - датчик замаскирован 13 - датчик размаскирован 14 - карта поставлена на охрану 15 - карта снята с охраны 16 - на карте ... неисправно ... датчиков 17 - датчик выключен (как с пульта) 18 - датчик включен (как с пульта) VAL[1] - инициатор действия: 0 - сам оператор 1 - автопилот в смену данного оператора 2 – пульт ОПП VAL[2..9] - LM адрес карты или датчика VAL[10..11] - дополнительное слово

Расшифровка значений состояния канала тип 1 (свойства Quality, STATE)

Значение	Расшифровка
0	ОК
1	Выключен (т.е. сознательно не опрашивается)
2	Состояние не определено (нет никаких данных о состоянии канала). Например: драйвер устройства не загружен.
3	Неисправно устройство (датчик)
4	Неисправен контроллер
5	Значение недостоверно (показания датчика вышли за допустимые пределы)
6	Датчик не подключен. Нарушение линии связи с датчиком.

Расшифровка кода системы (свойство SYSM)

SYSM	Описание
0	*Неопределенная система*
1	Теплоснабжение
2	Электроснабжение
3	Водоснабжение
4	Газоснабжение
5	Лифтовая диспетчеризация
6	Охранная сигнализация
7	Пожарная сигнализация
8	Сигнализация загазованности
9	Диспетчеризация объекта (откачка воды, вентиляция и т.п.)
10	Служебная подсистема (например: протоколирование действий оператора или состояние работы контроллера системы)
11	Сигнализация затоплений
12	Система голосовой связи

Расшифровка единицы измерения значения канала (свойство PARAM)

PARAM	Описание
0	*Неопределенный тип*
1	Дискретный параметр (перечисление состояний)
2	Градус Цельсия
3	Тонны
4	кГс/см2
5	Гкал
6	м3/час
7	кВатт/час
8	0-100%
9	м3
10	Час
11	тонн/час
12	Гкал/час
13	Дискретный параметр с фильтрацией
14	МВт
15	ГДж
16	МПа
17	ГДж/час
18	МВт*ч

Расшифровка значений состояния канала тип 2 (свойство TChannel2:Quality)

Значение Quality	Наименование	Пояснение
0	ОК	Все работает. Поле значение канала (Value) достоверно. При всех других значениях Quality значение канала (Value) НЕ достоверно.
1	Выключен	Сознательно не опрашивается контроллером. Например: датчик временно отключен по причине неисправности.
2	Состояние не определено	Нет данных о состоянии канала. Например: драйвер устройства не загружен или опросчик - владелец данного канала не подключен к серверу или произошел запуск программы, но информация о состоянии датчиков еще не поступила.
3	Неисправен датчик	Неисправен датчик или устройство — источник первичной информации. (аппаратный уровень 1)
4	Неисправен контроллер	Неисправен контроллер, производящий первичную обработку сигнала от датчика или адресный расширитель. (аппаратный уровень 2)
5	Значение недостоверно	Показания датчика вышли за допустимые пределы измерения. (аппаратный уровень 1)
6	Датчик не подключен	Нарушение линии связи с датчиком. Например обрыв. (аппаратный уровень 1)
7	Нет связи	Неисправность канала связи между сервером (или регистратором) и контроллером. Кроме случая нарушения линии

Значение Quality	Наименование	Пояснение
		связи контроллера с датчиком (Quality=6). (аппаратный уровень 2)
8	Неисправен регистратор	Неисправен регистратор, производящий сбор информации от контроллеров. (аппаратный уровень 3)

Расшифровка назначения канала тип 2 (свойство TChannel2:Signification)

Значение	Описание
0	Не определено
1	Интегратор учета в импульсах. Например, интегратор от контроллера БРК-К или БТС.
2	Интегратор учета в физических единицах. Например, интегратор газа счетчика Омега: потребленный объем газа в м3.
3	Канал управления чем-либо. Например, управление клапаном счетчика Омега или канал управления БИУ.
4	Канал контроля чего-либо. Например, сухой контакт БИУ-Р или фаза БИУ.
5	Охранный извещатель или шлейф
6	Пожарный извещатель или шлейф
7	Состояние охраны
8	Сигнализатор загазованности
9	Датчик температуры
10	Извещатель пожарный ручной
11	Датчик затопления
12	Лифт
13	Датчик давления
14	Мощность. Например: тепловая мощьность в Гкал/ч
15	Энтальпия.
16	Мгновенный (текущий) расход. Например: объемный расход воды в м3/ч

Маска форматирования для функций Format и sprintf

Маска форматирования содержит два типа объектов: буквы и команды форматирования. Буквы копируются в получаемую строку. Команды форматирования выбирают аргументы из списка и форматируют их. Общий вид команды форматирования:

"%" [index ":"] ["-"] [width] ["." prec] type

Команда форматирования начинается с символа %. После % идет поля:

Необязательный индекс параметра, [index ":"]
Необязательный указатель выравнивания по левому краю, ["-"]
Необязательный указатель ширины, [width]
Необязательный указатель точности, ["." prec]
Символ, задающий тип форматирования аргумента;

Описание возможных значений для типа форматирования:

d ***Десятичное целое.*** Аргумент должен быть целого типа. Если формат содержит указатель точности – это означает, что получаемая строка должна содержать как минимум столько цифр. Если реальное количество цифр меньше – получаемая строка дополняется слева нулями.
u ***Десятичное целое без знака.*** То же что и «d», но не содержит знака в получаемой строке.
f ***Фиксированное.*** Аргумент должен быть числом с плавающей точкой. Значение преобразуется в строку в формате "-ddd.ddd...". Количество знаков после запятой определяется указанным спецификатором точности после точки. По умолчанию точность – 2 цифры после запятой. Пример: значение «1.12340» сформатированное по строке формата «%.3f» будет выглядеть как «1.123».
s ***Строка.*** Аргумент должен быть символом или строкой. Строка или символ вставляются на место указателя формата. Если указатель точности присутствует, то он определяет максимальный размер получаемой строки. Если строка в аргументе больше – она усекается.
x ***Шестнадцатеричный.*** Аргумент должен быть типа целого типа. Его значение преобразуется в строку шестнадцатеричных цифр. Если формат содержит указатель точности – это означает, что получаемая строка должна содержать как минимум столько цифр. Если реальное количество цифр меньше – получаемая строка дополняется слева нулями. Регистр получаемых символов зависит от регистра «x» (или «X»).
p ***Указатель.*** Аргумент должен быть указателем. Значение указателя преобразуется в строку из 8 символов - значение указателя в шестнадцатеричном виде.
e ***Научный.*** Аргумент должен быть числом с плавающей точкой. Его значение преобразуется в строку формата "-d.ddd...E+ddd". Результирующая строка начинается с символа минус если значение числа отрицательно. Одна цифра всегда предшествует десятичной точке. Полное число цифр в результирующей строке (включая одну перед десятичной точкой) задается спецификатором точности в строке формата. По умолчанию подразумевается точность 15, если спецификатор точности отсутствует. После символа экспоненты "E" в получаемой строке всегда следует символ плюс или минус и минимум три цифры.
g ***Общий.*** Аргумент должен быть числом с плавающей точкой. Его значение преобразуется в максимально короткую строку, с использованием фиксированного или научного формата. Число значащих цифр в получаемой строке задается спецификатором точности в строке формата. По умолчанию подразумевается точность 15, если спецификатор точности отсутствует. Нули в конце убираются из получаемой строки. Десятичная точка добавляется, только если это необходимо. При формировании результирующей строки используется фиксированный формат (**f**), если число цифр слева от десятичной точки меньше или равно указанной точности, и если значение больше или равно 0.00001. В противном случае, используется научный формат (**e**).
n ***Числовой.*** Аргумент должен быть числом с плавающей точкой. Его значение преобразуется в строку формата "-d,ddd,ddd.ddd...". Формат **n** соответствует формату **f**, за исключением того, что получаемая строка содержит разделители тысяч.
m ***Денежный.*** Аргумент должен быть числом с плавающей точкой. Его значение преобразуется в строку, которая представляет сумму денег. Преобразование управляется следующими глобальными

переменными: CurrencyString, CurrencyFormat, NegCurrFormat, ThousandSeparator, DecimalSeparator и CurrencyDecimals. Все эти переменные инициализируются из формата валюты, заданного в секции региональных настроек панели управления Windows. Если строка формата содержит спецификатор точности - он переопределяет значение, задаваемое глобальной переменной CurrencyDecimals.

Индекс аргумента, ширина и точность могут быть указана в строке формата (например: "%10d"), или косвенно указаны с использованием символа звездочки (например: "%*.*f"). При использовании звездочки, следующий аргумент в списке (должен быть целым числом) подставляется вместо звездочки. Например:

sprintf("%*.*f",8,2,123.456)

То же самое что:

sprintf("%8.2f ",123.456)

Указатель ширины залает минимальную ширину поля для конвертации. Если получаемая строка короче указанной ширины, она дополняется пробелами. По умолчанию сформатированная строка выравнивается по правому краю поля. Но если указан индикатор выравнивания по левому краю (символ “-“ предваряет указатель ширины), результат выравнивается по левому краю и дополняется пробелами после значения.

Индекс параметра задает текущий индекс аргумента для форматирования. Индекс первого аргумента – 0. Используя этот механизм, можно форматировать один аргумент много раз. Например: "sprintf ('%d %d %0:d %1:d', 10, 20)" возвращает строку «10 20 10 20».

Маска форматирования для функции FormatDateTime

Функция *FormatDateTime* поддерживает следующие команды форматирования:

Команда	Действие
c	Отображает дату и время, используя формат по умолчанию. Если время равно нулю (ноль часов ноль минут ноль секунд) то оно не отображается вообще.
d	День в числовом виде без дополнения нулем (1-31).
dd	День в числовом виде с дополнением нулем (01-31).
ddd	День недели как сокращение Sun-Sat.
dddd	День недели полностью Sunday-Saturday.
ddddd	Дата в кратком формате по умолчанию.
dddddd	Дата в длинном формате по умолчанию.
m	Месяц в числовом виде без дополнения нулем (1-12). Если m следует за h или за hh то минута отображается вместо месяца.
mm	Месяц в числовом виде с дополнением нулем (01-12). Если m следует за h или за hh то минута отображается вместо месяца.
mmm	Месяц как сокращение Jan-Dec.
mmmm	Месяц полностью January-December.
yy	Год в двух цифрах (00-99).

Команда	Действие
yyyy	Год в четырех цифрах (0000-9999).
h	Час без дополнения нулем (0-23).
hh	Час с дополнением нулем до двух цифр (00-23).
n	Минута без дополнения нулем (0-59).
nn	Минута с дополнением нулем до двух цифр (00-59).
s	Секунда без дополнения нулем (0-59).
ss	Секунда с дополнением нулем до двух цифр (00-59).
z	Миллисекунда без дополнения нулем (0-999).
zzz	Миллисекунда с дополнением нулем до трех цифр (000-999).
t	Время с использованием форматирование в краткой форме по умолчанию.
tt	Время с использованием форматирование в полной форме по умолчанию.
am/pm	Использовать 12 часовое время по предыдущей команде h или hh, и писать 'am' для времени до полудня и 'pm' для времени после полудня. Команда am/pm может быть указаны в верхнем или нижнем регистре: результат будет соответствовать.
a/p	Использовать 12 часовое время по предыдущей команде h или hh, и писать 'a' для времени до полудня и 'p' для времени после полудня. Команда a/p может быть указаны в верхнем или нижнем регистре: результат будет соответствовать.
ampm	Использовать 12 часовое время по предыдущей команде h или hh, и писать спецификатор по умолчанию в полученной строке.
/	Отображать разделитель даты по умолчанию.
:	Отображать разделитель времени по умолчанию.
'xx'/"xx"	Символы заключенные в одинарные или двойные кавычки отображаются как есть без форматирования.

Команды могут указываться в нижнем или верхнем регистре – результат будет один и тот же. Если строка формата пуста – форматирование будет произведено по команде “c”.