

Программирование в АРМ LanMon 3.6

Ревизия 15 от 30.08.2007 г.

«МНПП Сатурн»
www.mnppsatur.ru

Содержание

Содержание.....	2
О документе.....	4
Введение.....	4
Описание скриптового языка APM LanMon.....	4
Синтаксис PascalScript.....	6
Синтаксис C++Script.....	7
Синтаксис JScript.....	9
Синтаксис BasicScript.....	10
Структура программы на скрипте.....	12
Типы данных.....	13
Приведение типов.....	14
Классы.....	15
Объект «Звук».....	15
Объект «Канал».....	15
Объект «Список каналов».....	16
Объект «Графический объект карты».....	16
Объект «Набор картинок».....	18
Объект «Картинка».....	18
Объект «Текст».....	19
Объект «Фигура».....	19
Объект «График».....	19
Объект «Прогресс бар» (Полоса заливки).....	19
Объект «Список».....	20
Объект «Область».....	20
Объект «ActiveX».....	20
Объект «Карта».....	21
Объект «Список карт».....	22
Объект «Группа каналов».....	22
Объект «Список групп каналов».....	23
Переменные.....	23
Функции общего назначения.....	24
Преобразование типов.....	24
Форматирование.....	24
Дата/время.....	25
Строковые функции.....	25
Математические функции.....	26
Другие.....	27
Функции специфичные для APM LanMon.....	27
Функции специального преобразования типов.....	27
Функции, специфичные для APM LanMon.....	28
Функции для работы с файлами.....	31
Функции для работы с отчетами RTF.....	32
Функции для работы с генератором отчетов.....	32
Функции фонового выполнения отчета в генераторе отчетов.....	34

Функции для работы со встроенным клиентом IP телефонии H323.....	35
Коды ошибок H323LastError().....	37
Объект «Абонент IP телефонии».....	37
Объект «Список абонентов IP телефонии».....	37
Обработчик событий IP телефонии.....	38
Функции по работе с трендами (графиками).....	39
Объект для управления настройками окна просмотра графиков.....	42
Функции для работы с драйверами устройств.....	42
Функции модуля GSM.....	43
Дискретные алармы.....	46
Перечисления и множества.....	47
Массивы.....	47
Многофайловые проекты.....	47
Обучение работе со скриптами.....	48
Редактор программ в APM LanMon.....	48
Клавиши редактирования в редакторе программ.....	50
Приложения.....	51
Расшифровка значений типа канала (свойство DTYPE).....	51
Расшифровка значений состояния канала (свойство STATE).....	55
Маска форматирования для функций Format и sprintf.....	55
Маска форматирования для функции FormatDateTime.....	57

О документе

Данный документ является описанием встроенных программных ресурсов APM 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 APM LanMon 2, начиная с версии 0.81, используется встроенный язык бейсик. Он позволяет производить вычисления, манипулировать встроенными объектами APM LanMon и их свойствами. Но он имеет ряд ограничений, которые не позволяют настраивать APM LanMon максимально гибко. Перечислим эти ограничения:

- Интерпретируемая структура языка: каждый раз при выполнении программы производится полный синтаксический анализ. Это требует значительных временных затрат и снижает общую производительность APM LanMon;
- Отсутствуют пользовательские процедуры и функции;
- Отсутствуют массивы;
- Обработчики событий не могут получать параметры;
- Каждый обработчик события находится в своем файле. В крупных проектах такой подход неоптимален;

В APM LanMon 3 встроен новый скриптовый движок. Он значительно превосходит старый бейсик по скорости работы и функциональным возможностям. Скорость работы возросла от 5 до 15 раз, в зависимости от особенностей применения. Скриптовый движок (далее скрипт) поддерживает четыре языка программирования: C++, объектный паскаль, бейсик и ява-скрипт. Более подробно особенности скрипта в APM LanMon описаны в следующей главе.

В версии APM LanMon 3 новый скриптовый движок существует параллельно со старым языком бейсик. Все старые проекты поддерживаются, т.е. совместимость снизу вверх полная. Тем не менее, старый бейсик надо считать морально устаревшим и в следующей версии APM LanMon (в версии 4) он не будет поддерживаться. Настоятельно рекомендуется все новые проекты делать с использованием скрипта и не пользоваться старым бейсиком.

Описание скриптового языка APM LanMon

Скриптовый язык в APM LanMon имеет мультязычную 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);

APM 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 APM LanMon содержит все процедуры, функции и обработчики событий проекта. Проект APM LanMon также может содержать массу выражений на скрипте (например, в динамических объектах карты).

Скрипт является компилируемым. Основная программа APM LanMon и все выражения на скрипте компилируются в псевдокод на этапе начала выполнения проекта, а затем псевдокод исполняется. Этим объясняется несколько возросшая задержка при старте APM LanMon. В противовес этому, мы имеем скорость работы скрипта, возросшую от 5 до 15 раз по сравнению со старым бейсиком.

Редактор программ имеет функции пошагового исполнения программы с просмотром значений переменных после каждого шага выполнения.

В APM LanMon 3 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 APM 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:

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 '
Block '
ProcedureHeading -> PROCEDURE Ident [FormalParameters]
FunctionDecl -> FunctionHeading ';' Block '
Block '
FunctionHeading -> FUNCTION Ident [FormalParameters] ':' Ident
FormalParameters -> '(' FormalParam/','... ')'
FormalParam -> [VAR | CONST] VarList

Синтаксис C++Script:

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:

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:

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 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 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 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()
    ' Добавьте 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, вы можете использовать следующие predefined типы в ваших скриптах:

Целочисленные:

Byte

Word

Integer

Longint

Cardinal

TColor

Логические:

Boolean

С плавающей точкой:

Real

Single

Double

Extended

Currency

TDate

TTime

TDateTime

Символьный:

Char

Строковый:

String

Вариантный тип:

Variant

Pointer

Массив:

Array

Соответствие некоторых типов C++Script стандартным типам:

int, long = Integer

void = Integer

bool = Boolean

float = double = Extended

JScript не имеет описаний типов - все типы являются Variant.

BasicScript может использовать описание типов (напр. `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);

int AsInteger(Variant v);

String AsString(Variant v);

Variant AsVariant(Variant v);

Эти очень удобны при использовании в выражениях на скрипте.

Классы

Вы не можете объявить класс в программе на скрипте, но вы можете использовать классы, которые APM LanMon экспортирует в скрипт. Например, TChannel – это класс канала LanMon. APM LanMon экспортирует класс TChannel в скрипт. Скрипт уже «знает» описание этого класса, все его свойства и методы.

Далее следует описание объектов APM LanMon и классов их описывающих. Описание методов и свойств приводится на языке Pascal.

Объект «Звук»

Предоставляет доступ к звуковой подсистеме APM 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. При чем если файл задан без пути – он будет искаться в поддиректории для звуков в папке проекта .\Wav.

Объект «Канал»

Канал представлен классом TChannel. Доступ к каналу возможен через список каналов. Динамически создать канал (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. Только для чтения.
property NAME: String	Название канала LanMon уровня A4. Только для чтения.
property DTYPE: Integer	Тип канала (тип значения свойства VALUE). Расшифровку смотрите в приложении «Расшифровка значений типа канала (свойство DTYPE)» или в файле dtype.txt Только для чтения.
property TIME: Extended	Метка времени: время последнего изменения свойств STATE, VALUE или Values. Время представлено типом TDateTime (double). Для чтения и записи.
property STATE: Integer	Качество (состояние) канала LanMon. Расшифровку смотрите в приложении «Расшифровка значений состояния канала (свойство STATE)» или в файле state.txt Свойства VALUE и Values имеют смысл только когда STATE равен нулю (т.е. состояние канала ОК). Для чтения и записи.
property VALUE: Variant	Значение канала. Тип значения зависит от типа канала и от значения STATE. 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 Change: bool	Если значение true – значит, произошло изменение свойств STATE, VALUE или Values. Источником такого изменения может быть сервер, драйвер оборудования или изменение соответствующего свойства из программы. Чтение свойства CHANGE сбрасывает его в false. Таким образом, к этому свойству можно обращаться только из одного места. Типовое использование этого свойства – в выражениях событийного тренда. Только для чтения.
property ID: Integer	Номер учетной записи сервера LanMon, от которой поступило данное значение канала. Используется в больших системах с сервером и несколькими опросчиками. Только для чтения.
property SYSM: Integer	Подсистема, к которой относится данный канал. Все подсистемы перечислены в файле ID SYSM.txt. Только для чтения.
property PARAM: Integer	Единица измерения значения, которое несет данный канал. Расшифровку смотри в файле ID PARAM.txt Только для чтения.

Объект «Список каналов»

Список каналов представлен классом TRTV. Есть глобальная переменная RTV: Class TRTV, которая служит для доступа к списку каналов.

Свойство/Метод	Значение
index property Items(p0: Integer): TChannel	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 «Графический объект карты»

Базовый класс для всех графических объектов карты это *TMonControl*. Каждому объекту карты можно назначить определенный канал. Поэтому часть свойств у объекта карты совпадает с классом *TChannel*. Доступ к объектам карты возможен через список объектов карты (класс *TMap*) или через создание переменной 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, ассоциированного с данным объектом. 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 NAME: String	Название канала LanMon уровня A4. Только для чтения.
property DTYPE: Integer	Тип канала (тип значения свойства VALUE). Расшифровку смотрите в приложении «Расшифровка значений типа канала (свойство DTYPE)» или в файле dtype.txt Только для чтения.
property TIME: Extended	Метка времени: время последнего изменения свойств STATE, VALUE или Values. Время представлено типом TDateTime (double). Для чтения и записи.
property STATE: Integer	Качество (состояние) канала LanMon. Расшифровку смотрите в приложении «Расшифровка значений состояния канала (свойство STATE)» или в файле state.txt Свойства VALUE и Values имеют смысл только когда STATE равен нулю (т.е. состояние канала ОК). Для чтения и записи.
property VALUE: Variant	Значение канала. Тип значения зависит от типа канала и от значения STATE. 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 SYSM: Integer	Подсистема, к которой относится данный канал. Все подсистемы перечислены в файле ID_SYSM.txt. Только для чтения.
property Text: String	Заголовок (текстовая подпись) объекта
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
property Map: Class TMap	Ссылка на карту, на которой данный объект располагается.
property x: Integer property y: Integer	Координаты центра объекта на карте. Для чтения и записи. Изменяя можно двигать объект.
property EnableBorder: boolean	Разрешено выделение объекта рамочкой в Run Time ? Для чтения и записи.
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	Если true то дискретные алармы, назначенные данному объекту не срабатывают. Замаскированный объект обводится красной рамкой. Это свойство хранится в файле карты. Для чтения и записи.
property MapIndex: Integer	Индекс карты в списке Maps.Items[MapIndex]. Только для чтения.
property Canvas: Class TCanvas	Используется для пользовательского рисования (Custom Draw).
procedure Invalidate	Принудительная перерисовка, объекта карты. Применяется после использования свойства Canvas для рисования.

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 «График»

Объект график – это движущийся во времени график. Объект представлен классом TMonGraph. Базовый класс для этого объекта TMonControl. Класс имеет следующие свойства и методы:

Свойство/Метод	Значение
property MinValue: Extended	Минимальное значение, отображаемое на графике. Сохраняется в файле карты. Только для чтения.
property MaxValue: Extended	Максимальное значение, отображаемое на графике. Сохраняется в файле карты. Только для чтения.
property ValueColor: Integer	Цвет значений на графике. Сохраняется в файле карты. Для чтения и записи.
property BackColor: Integer	Цвет фона графика. Сохраняется в файле карты. Для чтения и записи.

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 для этого объекта *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 «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. <i>Name</i> – Имя функции или свойства. <i>Type</i> – Строка, обозначающая тип вызываемого объекта. Может быть одним из следующих: "FUNC" – <i>Name</i> это имя функции

	<p>"PROPERTYGET" - <i>Name</i> это имя свойства, возвращающего значение</p> <p>"PROPERTYPUT" - <i>Name</i> это имя свойства, принимающего значение</p> <p><i>Params</i> – массив входных параметров для вызова функции или свойства</p> <p>Возвращаемое значение функции <i>Call(...)</i> это то, что возвращает вызванная функция объекта или свойство <i>PROPERTYGET</i>.</p>
<p>procedure OnActiveXEvent(Sender: TMonActiveX; MessageType: Integer; var Params: Variant)</p>	<p>Обработчик событий от объекта ActiveX. Параметры:</p> <p><i>Sender</i> - ссылка на объект карты TMonActiveX, от которого пришло событие;</p> <p><i>MessageType</i> - идентификатор события (можно посмотреть в списке функций для данного ActiveX. В скрипте определены константы идентификаторов типовых событий:</p> <p>DISPID_CLICK DISPID_DBLCLICK DISPID_KEYDOWN DISPID_KEYPRESS DISPID_KEYUP DISPID_MOUSEDOWN DISPID_MOUSEMOVE DISPID_MOUSEUP DISPID_READYSTATECHANGE</p> <p><i>Params</i> - массив параметров события. Используется и для выдачи параметров из обработчика.</p>

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 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 A1: Integer property A2: Integer property A3: Integer property A4: Integer	Адрес канала LanMon, ассоциированного с данной картой. Только для чтения.
property Ohrana: boolean	Состояние охраны карты. После старта АРМ LanMon всегда False. Для чтения и записи.
property Name: String	Название карты, отображаемое в ее заголовке. Для чтения и записи.
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 <code>.\BMP\</code> 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 <code>.\WAV\</code> 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 (<code>Maps.Items[MapIndex]</code>). Для чтения и записи.
property Tag: Integer	Может использоваться программистом произвольным образом. Для чтения и записи.
property PPR: Integer	Может использоваться программистом произвольным образом. Для чтения и записи. Доступно только из программы.
index property Items[p0: Integer]: TMonControl	Список всех графических объектов карты (базовый класс <code>TMonControl</code>). Только для чтения.
property Count: Integer	Количество объектов в свойстве <code>Items[]</code> . Максимальный индекс <code>Items[Count-1]</code> . Только для чтения.
procedure Show	Сделать карту видимой и выдвинуть ее на первый план.
procedure Hide	Сделать карту невидимой.
property Visible: Boolean	Карта видна сейчас ? Примечание: карта может быть видна, но перекрыта другой картой. В таком случае процедура <code>Show</code> выдвинет ее на первый план.

Объект «Список карт»

Все карты в APM 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 <code>TMap</code> . Только для чтения. Карта доступна по индексу.
property Count: Integer	Количество объектов в свойстве <code>Items[]</code> . Максимальный индекс <code>Items[Count-1]</code> . Только для чтения.

Объект «Группа каналов»

Группа каналов содержит список каналов. Группы каналов предназначены для создания охранных зон и т.п. Они создаются в редакторе групп. В программе на скрипте доступен ряд свойств группы. 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:	Есть хоть один неисправный канал в группе ? Канал считается

boolean	неисправным, когда его качество STATE>2. Только для чтения.
property Ohrana: boolean	Состояние охраны группы каналов. Для чтения и записи.
property State: Integer	Состояние группы: 0-все спокойно 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: Class TGroups.

Свойство/Метод	Значение
index property Items(p0: Integer): TGroup	Ссылка на объекты группа каналов TGroup. Группа каналов доступна по индексу. Только для чтения.
property Count: Integer	Количество объектов в свойстве Items[]. Максимальный индекс Items[Count-1]. Только для чтения.

Переменные

В скрипте всегда определено несколько ссылок на глобальные объекты APM LanMon:

- Application: Class T Application – Объект для управления программой APM LanMon.
- GraphSetup: Class TGraphSetup – Объект для управления настройками окна показа графиков.
- Groups: Class TGroups – список групп каналов. Группа представлена классом TGroup;
- 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 APM LanMon, а также в демонстрационных проектах.

Функции общего назначения

В скрипте можно использовать богатый набор стандартных функций.

Преобразование типов

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:	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

Variant=0; a8: Variant=0; a9: Variant=0; a10: Variant=0;): String	«Маска форматирования для функций Format и sprintf». <i>Пример:</i> print(sprintf("%.2u часов %.2u минут", 10, 6)); Выведет на печать строку: «10 часов 06 минут»
----------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Дата/время

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

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: <i>print(ExtractSubString("Hello, world !", 2, " "));</i> 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

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 символа в 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

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

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 числа с плавающей запятой в строке

function ValidDate(cDate: String): Boolean

Проверка правильности даты в строке

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

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 LMIF(Expr: Boolean; TrueValue, FalseValue: Variant): Variant	Если выражение <i>Expr</i> истинно, то функция <i>LMIF</i> возвращает значение выражения <i>TrueValue</i> . Иначе – возвращает значение выражения <i>FalseValue</i> . 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 APM LanMon.
procedure LMProtokol(Str: String; File: String=)	<p>Запись текстовой строки <i>Str</i> и метки времени в конец текстового файла <i>File</i>. Если <i>File</i> не задан или задана пустая строка – файлом считается <i>lanmon.log</i> в директории проекта.</p> <p>Пример записи сформатированной строки в файл <i>lanmon.log</i> на C++Script:</p> <pre>// “\x9” – это символ табуляции LMProtokol(sprintf("INFO\x9%s\x9%d", "hello !", 123));</pre>
function LMExec(AppName: String; ShowWindow: Word=1; Wait: Boolean=False; var ExitCode: Integer=Nil): Boolean	<p>Запуск внешней программы <i>AppName</i> или просмотр документа внешней программой. При запуске программы <i>AppName</i> может содержать аргументы командной строки.</p> <p><i>ShowWindow</i>:</p> <ul style="list-style-type: none"> 1 – показать в нормальном размере 2 – показать минимизированное окно 3 – показать распахнутое окно <p><i>Wait</i> – ждать завершения вызванного процесса. Если задать <i>True</i> – работа APM LanMon будет заморожена до завершения вызванного приложения. Можно ставить <i>True</i> ТОЛЬКО если вызываемая программа выполняется не более одной секунды и ТОЛЬКО если вам нужен <i>ExitCode</i>.</p> <p><i>ExitCode</i> – значение возвращаемое вызванной программой. Актуально только если <i>Wait=True</i>.</p> <p>Возвращает <i>True</i> если внешняя программа запущена успешно и <i>False</i> в противном случае.</p> <p>Примеры на C++:</p> <pre>// просмотр документа manager.pdf LMExec("c:\\program files\\LanMon</pre>

	<pre>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");</pre>
<pre>procedure LMShowMessage(Str: Variant; Color: TColor=cInfoBk; Left: Integer=-1; Top: Integer=-1)</pre>	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.
<pre>procedure Print(Str: Variant)</pre>	Печать строки Str в окне отладочной печати.
<pre>function LMSendControl(A1,A2,A3,A4,STATE,VAL: Integer): Integer</pre>	Послать команду управления драйверам оборудования и на сервер LanMon. Функция получает адрес канала, его состояние и значение. При ошибке возвращает нулевое значение. Например: <i>LMSendControl(1,2,3,4,0,1)</i>
<pre>function LMSendState(A1,A2,A3,A4,STATE: Integer; VAL: Variant): Integer</pre>	Послать состояние канала на сервер LanMon подобно тому, как это делает опросчик. Функция получает адрес канала, его состояние и значение. При ошибке возвращает нулевое значение. Например: <i>LMSendState(1,2,3,4,0,1)</i>
<pre>function LMSendStateLocal(A1,A2,A3,A4,STATE: Integer; VAL: Variant): Boolean</pre>	<p>Послать состояние канала самому себе. Подобно тому, как будто бы оно пришло от сервера или от драйвера оборудования. Функция получает адрес канала, его состояние и значение. При ошибке возвращает значение false. Ошибка может возникнуть в двух случаях:</p> <ul style="list-style-type: none"> - такого канала нет в дереве - произошла ошибка преобразования типов параметра VAL (например, канал целочисленный, а в функцию передана строка) <p>Пример: <i>LMSendStateLocal(1,2,3,4,0,1);</i></p>
<pre>function LMSendChannel(c: TChannel; Action: Integer=3): Boolean</pre>	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).
<pre>function LMServerStatus(type: Integer): Variant</pre>	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

	<p>строки.</p> <p>Возможные возвращаемые значения:</p> <p>0-"ОТКЛЮЧЕН!"</p> <p>1-"Подключение..."</p> <p>2-"Регистрация..."</p> <p>3-"Запись маски..."</p> <p>4-"Получение A1..."</p> <p>5-"Получение A2..."</p> <p>6-"Получение A3..."</p> <p>7-"Получение A4..."</p> <p>8-"Получение каналов..."</p> <p>9-"ОК"</p> <p>10-"ОТКЛЮЧЕН! (РЕЗЕРВНЫЙ)"</p> <p>11-"Подключение... (РЕЗЕРВНЫЙ)"</p> <p>12-"Регистрация... (РЕЗЕРВНЫЙ)"</p> <p>13-"Запись маски... (РЕЗЕРВНЫЙ)"</p> <p>14-"Получение A1... (РЕЗЕРВНЫЙ)"</p> <p>15-"Получение A2... (РЕЗЕРВНЫЙ)"</p> <p>16-"Получение A3... (РЕЗЕРВНЫЙ)"</p> <p>17-"Получение A4... (РЕЗЕРВНЫЙ)"</p> <p>18-"Получение каналов... (РЕЗЕРВНЫЙ)"</p> <p>19-"ОК (РЕЗЕРВНЫЙ)"</p>
procedure LMViewAnalogAlarm	Показать на экране окно аналоговых алармов.
procedure LMViewLog	Открыть окно журнала.
procedure LMViewArchive	Открыть окно ввода параметров для просмотра архива.
procedure LMViewArchiveOne(A1,A2,A3,A4: Word)	Показ окна с выборкой событий по указанному каналу LanMon. Выборка производится по локальному архиву событий, накопленному в поддиректории .\LOG\ текущего проекта. Глубина выборки по времени в днях задается в настройках проекта.
procedure LMViewAlarm	Открыть окно просмотра алармов.
procedure LMExit	В режиме отладки проекта: завершение отладки. В режиме выполнения проекта: завершение работы APM LanMon.
procedure LMOperatorChange	Вызов модального окна смены оператора. Пример использования смотрите в файле .\samples\operator.bas
function LMFormatString(A1,A2,A3,A4: Integer; MASK: String): String	<p>Сформатировать параметры канала в текстовую строку. Возвращает полученную строку. Первые 4 аргумента – адрес канала. Пятый аргумент маска форматирования:</p> <p>%DATE – дата изменения канала</p> <p>%TIME – время изменения канала</p> <p>%STATE – состояние канала</p> <p>%VALUEn(%mask) – Значение канала. n – номер значения. %mask – маска форматирования значения канала.</p> <p>%A1 - текстовая расшифровка адреса A1</p> <p>%A2 - текстовая расшифровка адреса A2</p>

	<p>%A3 - текстовая расшифровка адреса A3 %A4 - текстовая расшифровка адреса A4 %NL – вставка символов новой строки</p> <p>LMFormatString(1,2,3,4,,"%DATE %TIME %STATE %VALUEn(%mask) %A1 %A2 %A3 %A4 %NL")</p>
function AAState(Alarm: Variant; ResultType: Integer): Variant	<p>Получить состояние аналогового аларма.</p> <p>Первый аргумент - название аларма или его номер с нуля.</p> <p>Если второй аргумент 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</p> <p>Если второй аргумент равен единице, функция возвращает текстовую расшифровку состояния аларма.</p>
procedure Sleep(Milliseconds: int)	Пауза в выполнении программы на указанное количество миллисекунд.

Функции для работы с файлами

Описание	Комментарии
function addfile(File, Str: String): Integer	<p>Добавить строку в текстовый файл File. Первый аргумент – имя файла. Второй аргумент – строка. После записи строки в файл автоматически добавляются символы перевода строки.</p> <p>Возврат: 0 – ошибка открытия или записи файла 1- все хорошо</p>
function deletefile(File: String): Integer	Удаление указанного файла File. Удаление производится безвозвратно. Возвращает ноль при ошибке.
function LMGetCSV(File: String; Row,Col: Integer): String	<p>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.</p> <p>Пример использования смотрите в файле</p>

	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.

Функции для работы с отчетами RTF

function rptRun(TemplateFile, ReportFile: String): Integer	2	Формирует отчет из шаблона. Смотри пример: report.bas
procedure rptShow(Caption, ReportFile: String)	2	Показывает отчет из файла RTF на экране в специальном окне. Это окно может использоваться для просмотра произвольных текстовых и RTF файлов (без вызова функции rptRun).
function rptPrint(ReportFile: String): Integer	1	Печатает отчет из файла на принтере по умолчанию.

Механизм отчетов RTF и функции по работе с ним являются устаревшими. Рекомендуется пользоваться более современным механизмом статистики – генератором отчетов.

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 искажаться в поддиректории проекта «.\FR\». Функция frLoad() восстанавливает использование SystemADOConnection в качестве соединения по

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

	<p>подготовлен функциями frPrepare или frRun. формат_экспорта: 0 – TXT 1 – HTML 2 – XLS 3 – XML 4 – RTF 5 – BMP 6 – JPEG 7 – TIFF 8 – PDF</p> <p>Если второй аргумент – “имя файла” – не задан, то выдается диалоговое окно параметров экспорта. Если файл задан – производится экспорт без запросов пользователя. Функция возвращает ноль в случае ошибки.</p>
--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Функции фонового выполнения отчета в генераторе отчетов

function frAsyncPrepare(ClearLastReport: Boolean; OnReport: String='OnReportNotify'): Boolean	<p>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().</p> <p>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 окончании выполнения отчета (по умолчанию 'OnReportNotify'). Функция возвращает False если отчет уже выполняется.</p>
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)	<p>После завершения выполнения фонового отчета, запущенного функцией frAsyncPrepare, вызывается событие OnReportNotify. RetVal – код результата выполнения отчета: 0-при выполнении отчета произошла ошибка</p>

	<p>1-отчет успешно выполнен 2-выполнение отчета прервано вызовом frBreak</p> <p>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.</p> <p>В функции frAsyncPrepare может быть задано другое имя для обработчика события по выполнении отчета.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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 отправки 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 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 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	Выбор вкладки в окне разговора. View – номер вкладки с нуля: 0 – основная вкладка со списком абонентов

: Integer=0)	1 - дополнительная вкладка с объектом ActiveX
--------------	-----------------------------------------------

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
Ошибка звонка !	16

Объект «Абонент IP телефонии»

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

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

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.
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.
index property Items(index: Integer): TIPAbonent	Ссылка на абонента IP телефонии класс TIPAbonent по индексу от 0 до (Count-1). Только для чтения.

В следующем фрагмента вода 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
    ));
}
```

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 единократно)
hm323End			Разговор завершен
hm323Forwarded	Алиас того, кому переадресовали	Адрес того, кому переадресовали	Звонок был успешно переадресован.

	ВЫЗОВ	ВЫЗОВ	
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

Функции по работе с трендами (графиками)

Описание	Комментарии
procedure TrendClear	Удалить все графики в окне вывода графиков и сбросить назначенное увеличение (Zoom). Удаляются тренды, добавленные вызовами функции <i>TrendAssign</i> 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 или индекс тренда в списке. Если Trend равен -1 или «» 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:	Создание трендовой переменной из данных, накопленных в SQL сервере. Выполняется

<p>Variant; ADOConnection: TADOConnection): Integer</p>	<p>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 переменной. Далее можно использовать функции <i>TrendAssign</i> и <i>TrendShowWindow</i> для показа тренда. Параметры:</p> <p><i>Trend</i> – имя трендовой переменной. Должно содержать только символы, разрешенные в именах файлов.</p> <p><i>Number</i> – номер периода накопления (1 – текущий период).</p> <p><i>Sql</i> – текст SQL запроса к базе данных. Параметры запроса предваряются двоеточием.</p> <p><i>Params</i> – массив параметров запроса. Должен быть массивом Variant даже если надо передать одно значение ! Если параметры не используются можно передать ноль - 0.</p> <p><i>ADOConnection</i> – ссылка на подключение к базе данных через ADO. Можно указать одно из системных подключений SystemADOConnection, SystemADOConnection1, SystemADOConnection2. Системные подключения настраиваются в настройках проекта.</p> <p>Функция возвращает число строк в полученном результате выполнения запроса >= 0. Если возвращаемое значение меньше нуля – это код ошибки:</p> <ul style="list-style-type: none"> -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:	Создание трендовой переменной из данных,

<p>String; Number: Integer; A1,A2,A3,A4: Word; Start, End: TDateTime): Integer</p>	<p>накопленных локально при работе APM в поддиректории \LOG\ проекта. Выборка данных производится по каналу с адресом A1.A2.A3.A4. Функция создает файл тренда в поддиректории проекта \TREND\ Имя файла совпадает с именем трендовой переменной. Далее можно использовать функцию <i>TrendAssign</i> и <i>TrendShowWindow</i> для показа тренда.</p> <p>Параметры:</p> <p><i>Trend</i> – имя трендовой переменной. Должно содержать только символы, разрешенные в именах файлов.</p> <p><i>Number</i> – номер периода накопления (1 – текущий период).</p> <p><i>A1,A2,A3,A4</i> – адрес канала, для которого надо запросить данные.</p> <p><i>Start,End</i> – Интервал времени для запроса данных.</p> <p>Функция возвращает число найденных записей в полученном результате выполнения запроса ≥ 0. Если возвращаемое значение меньше нуля – это код ошибки:</p> <ul style="list-style-type: none"> -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";
<p>procedure TrendLine(Trend: String; Value: Extended; Number: Integer)</p>	<p>Создание тренда – линии из двух точек. Значения точек по оси времени берутся минимальное и максимальное из графиков, уже назначенных функцией <i>TrendAssign</i>.</p> <p><i>Trend</i> – имя трендовой переменной. Должно содержать только символы, разрешенные в именах файлов.</p> <p><i>Number</i> – номер периода накопления (1-99 единица означает текущий период накопления).</p> <p><i>Value</i> – значение обеих точек по вертикальной оси.</p> <p>После создания тренда его надо отобразить в один из графиков функцией <i>TrendAssign</i>.</p> <p>Данная процедура используется для рисования</p>

	горизонтальных границ в окне графиков.
procedure TrendGetMinMax(var MinValue: Extended; var MaxValue: Extended)	Получение минимального и максимального значений по всем графикам, отображенным функцией <i>TrendAssign</i> .
void OnTrendWindowButtonClick(int Button, int Info)	<p>Обработчик события по нажатию клавиш «Предыдущий период накопления» и «Следующий период накопления» в окне просмотра трендов.</p> <p>Button – номер нажатой кнопки. 1 - «Предыдущий период накопления». 2 - «Следующий период накопления».</p> <p>Info – Для кнопок №1 и №2 - номер требуемого периода накопления.</p> <p>Данный обработчик предназначен для просмотра динамически генерируемых трендов. Например, для трендов, созданных функцией <i>TrendCreateFromSQL</i>. При нажатии кнопки «Предыдущий период накопления» нужный тренд может быть динамически сгенерирован. Обработчик вызывается ТОЛЬКО если тренд назначен функцией <i>TrendAssign</i>.</p> <p>Для добавления обработчика данного события в код программы в редакторе программ выберите пункт меню «События / Нажатие клавиши в окне просмотра графиков».</p>

Объект для управления настройками окна просмотра графиков

В программе доступна ссылка на глобальный объект *GraphSetup* типа *TGraphSetup*. Этот объект содержит следующие свойства:

Описание	Комментарии
bool LeftAxisAutomatic	Масштаб по оси значений (вертикальной) устанавливать автоматически или вручную.
double LeftAxisMinimum	Минимальное значение на вертикальной оси, которое будет показано. Это свойство работает ТОЛЬКО если LeftAxisAutomatic=true
double LeftAxisMaximum	Максимальное значение на вертикальной оси, которое будет показано. Это свойство работает ТОЛЬКО если LeftAxisAutomatic=true
double LeftAxisIncrement	Инкремент разметки по вертикальной оси.

Функции для работы с драйверами устройств

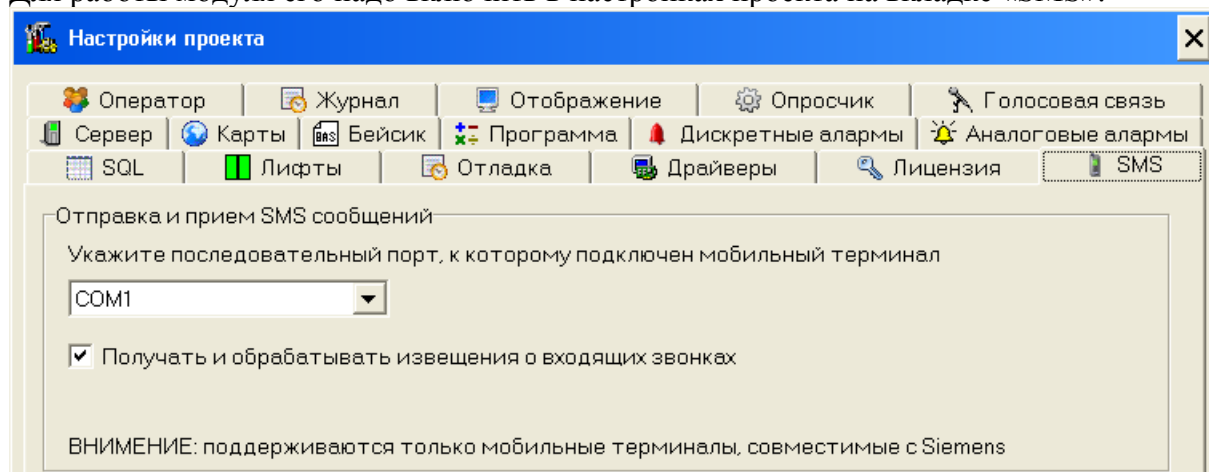
Описание	Комментарии
procedure DriverShow(Show: Boolean=True)	Показать или скрыть окно драйверов оборудования. В режиме выполнения проекта функции настройки,

	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: <i>String s = DriverChannelInfo(1,1,2,5); if(Length(s) > 0) LMShowMessage(s); else LMShowMessage("Информация об этом канале недоступна.");</i>

Функции модуля GSM

Модуль GSM предназначен для приема и отправки SMS сообщений и регистрации входящих голосовых вызовов. Данный модуль всегда включен в комплект поставки APM LanMon, но для его работы требуется отдельная лицензия. Модуль работает только с Siemens совместимым мобильным телефоном или мобильным терминалом, подключенным к компьютеру по последовательному интерфейсу. Все функции модуля GSM доступны из программы на скрипте.

Для работы модуля его надо включить в настройках проекта на вкладке «SMS»:



Надо указать последовательный порт компьютера, к которому подключен мобильный терминал. Для регистрации входящих голосовых вызовов установите галочку «Получать и обрабатывать извещения о входящих звонках».

Работа модуля SMS протоколируется в отдельном файле sms.log Этот файл находится в директории проекта.

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

	<p>пришло SMS сообщение. Номер необходимо указывать полностью, включая код страны и код сети. Например: +7-903-123-45-67</p> <p>или</p> <p>79031234567</p> <p>Все символы кроме цифр игнорируются.</p>
property String Text	<p>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.</p> <p>При наличии русских букв сообщение отправляется в кодировке юникоде. Это в два раза уменьшает максимально допустимую длину сообщения. В этом случае, максимальная длина сообщения составляет 80 символов.</p>
property bool Flash	<p>True – посылать автоматически всплывающее сообщение.</p> <p>False – посылать обычное SMS сообщение.</p> <p>Данное свойство действует только на посылку сообщения.</p> <p>Данное свойство действует, только если сообщение (свойство Text) не содержит русских букв.</p>
property bool Transliterate	<p>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.</p>

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	<p>True – Если номер телефона в поле Telephone представлен в интернациональном формате (номер начинается с кода страны, например с +7)</p> <p>False – Если номер телефона в поле Telephone представлен в местном формате.</p>
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	<p>Отправка SMS сообщения. Объект SMS сообщения (класс TSMS) должен быть предварительно создан. Функция возвращает значение <i>true</i> если мобильный терминал подключен к указанному в настройках порту, отвечает на команды и сообщение поставлено в очередь на отправку. Возвращает <i>false</i> в противном случае.</p>

	<p>Фрагмент кода для отправки SMS сообщения на C++ скрипт:</p> <pre> <i>TSMS</i> <i>p</i> = new <i>TSMS</i>; <i>p.Telephone</i> = "+7-903-123-45-67"; <i>p.Text</i> = "Hello, world"; <i>p.Flash</i> = false; <i>p.Transliterate</i> = false; if(<i>SMSSend</i>(<i>p</i>)) print("Сообщение поставлено в очередь на отправку."); else print("Ошибка отправки сообщения !"); delete <i>p</i>; </pre>
function SMSCount: Integer	<p>Модуль автоматически принимает входящие SMS сообщения и сохраняет их во внутреннем буфере. Функция <i>SMSCount</i> 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 <i>SMSReceive</i>.</p>
function SMSReceive(p: TSMS): Boolean	<p>Прием SMS сообщения. Объект SMS сообщения (класс <i>TSMS</i>) должен быть предварительно создан. Пример кода для приема SMS сообщения на C++ скрипт:</p> <pre> <i>TSMS</i> <i>p</i> = new <i>TSMS</i>; if(<i>SMSReceive</i>(<i>p</i>)) { print("Принято новое SMS сообщение !"); print("С номера: " + <i>p.Telephone</i>); print("Текст: " + <i>p.Text</i>); } delete <i>p</i>; </pre>
function RingCount: Integer	<p>Модуль автоматически регистрирует факт поступления входящего звонка и сохраняет все параметры звонка во внутреннем буфере. Функция <i>RingCount</i> 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 <i>RingReceive</i>.</p>
function RingReceive(p: TRing): Boolean	<p>Получение информации о входящем звонке. Объект звонка (класс <i>TRing</i>) должен быть предварительно создан. Пример кода для приема звонка на C++ скрипт:</p> <pre> <i>TRing</i> <i>p</i> = new <i>TRing</i>; if(<i>RingReceive</i>(<i>p</i>)) </pre>

```

{
    print("Принят входящий звонок !");
    print("С номера: " + p.Telephone);
}
delete p;

```

ВНИМАНИЕ: Для работы модуля приема SMS сообщений нужна отдельная лицензия. Эта лицензия может быть получена из USB ключа или от сервера LanMon (в этом случае сервер должен быть версии 3.32 или старше).

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 APM LanMon на разных языках;

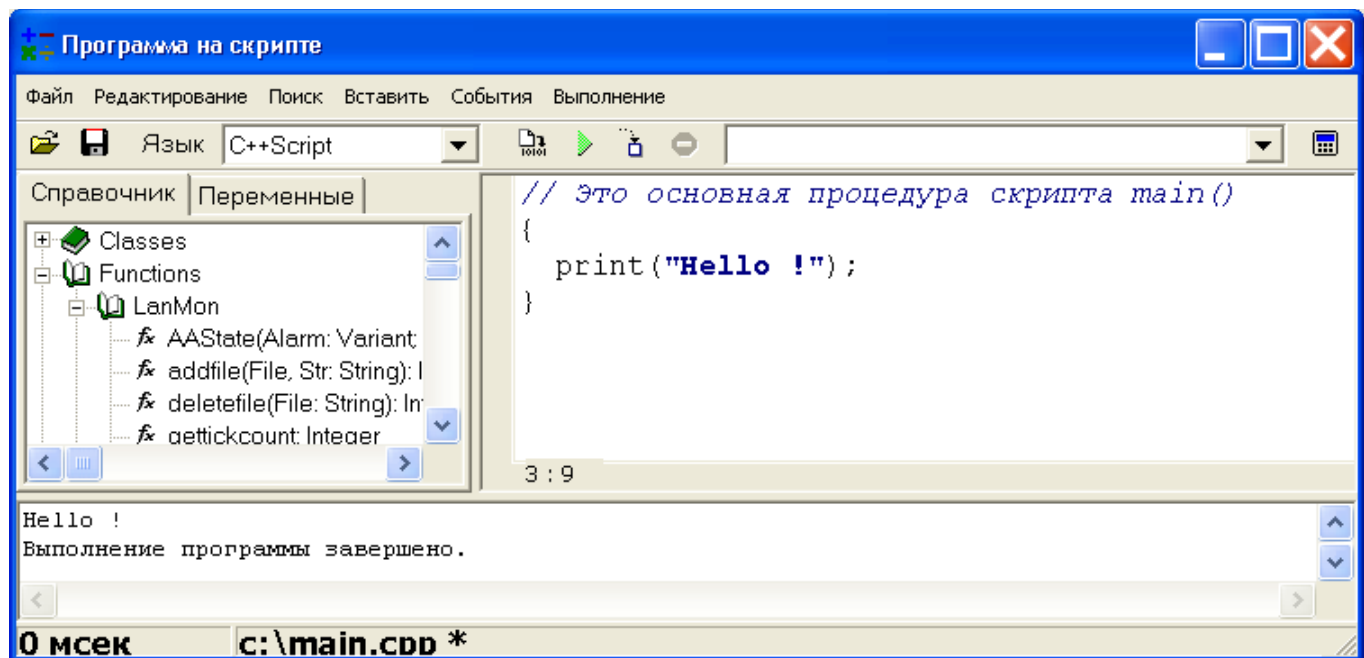
Откройте в редакторе программ одну из демонстрационных программ и выполните ее нажав клавишу «F9».

Редактор программ в APM LanMon

Для написания и отладки программ в APM 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 программы 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 отражаются 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 APM 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	0,1
2	byte	BYTE	0..255
3	int8	int8	-128...+127
4	int16	int16	-32k..+32k
5	int32	int32	- 2,147,483,648 2,147,483,648
6	float	float	$1.18 \cdot 10^{-38} < X < 3.40 \cdot 10^{38}$ 7-digit precision
7	char[16]	char[16]	Строка длиной до 16 символов включительно (может не завершаться нулем)
8	word	WORD	0...65535
9	double	double	$2.23 \cdot 10^{-308} < X < 1.79 \cdot 10^{308}$ 15-digit precision
10	byte	Температура	-128...+127 °C
11	byte	Состояние КД	0-Норма 1-Сработка
12	byte[6]	Состояние ОД	VALUE[0] 0-Норма 1-Сработка 2-Норма левый 3-Сработка левый 4-Норма правый 5-Сработка 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)

			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-Сработка 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 - Выкл 1 - Вкл 2 - Затоплен 3 - Обесточен 4 - Вода вкл. 5 - Вода выкл.
17	byte	Вентилятор	0 - Выкл. 1 - Вкл. 2 - Обесточен
18	byte	Кнопка (канал управления)	0 – Выключен 1 – Включен 2 – Выключен, нет питания 3 – Включен, нет питания
20	byte	ЭРУ	0-Норма 1- Затопление уровня 1 2- Затопление уровня 2 3- Затопление уровня 3 4- Затопление уровня 4
21	byte	Состояние локальной охранной зоны	0-охрана снята 1-на охране 2-снят с охраны, сработка 3-на охране, сработка 4-снят с охраны, тревога 5-на охране, тревога
22	byte	Диагностика	Качество работы в % (0-100)

24	byte[5]	чего-либо Лифт «Сатурн»	<p>VALUE[0]</p> <p>0- "Нет данных" *</p> <p>1- "Есть вызов"</p> <p>2- "Нажата кнопка Стоп"</p> <p>3- "Устройство защиты дифта: " (дописывается VAL[3])</p> <p>4- "Авария по сигналам"</p> <p>5- "Кабина в движении"</p> <p>6- "Дверь кабины открыта"</p> <p>7- "Все в порядке"</p> <p>8- "Выключен" *</p> <p>9- "Нет ответа по СОС-95" *</p> <p>10- "Снято питание лифта"</p> <p>11- "Долго нет движения лифта"</p> <p>12- "Вызов из МП"</p> <p>13- "Блок БДК"</p> <p>14- "Нет данных+Пассажир" *</p> <p>15- "Есть вызов+Пассажир"</p> <p>16- "Нажата кнопка Стоп+Пассажир"</p> <p>17- "Остановлен БЗЛ+Пассажир"</p> <p>18- "Авария по сигналам+Пассажир"</p> <p>19- "Кабина в движении+Пассажир"</p> <p>20- "Дверь кабины открыта+Пассажир"</p> <p>21- "Все в порядке+Пассажир"</p> <p>22- "Выключен" *</p> <p>23- "Нет ответа по СОС-95+Пассажир" *</p> <p>24- "Снято питание лифта"</p> <p>25- "Долго нет движения лифта+Пассажир"</p> <p>26- "Вызов из МП+Пассажир"</p> <p>27- "Блок БДК"</p> <p>VALUE [1] – Тип лифта</p> <p>VALUE [2] – Маска ошибок</p> <p>VALUE [3] – Состояние лифта:</p> <p>"Все в порядке",//0</p> <p>"Нет движения на бол. скорости",//1</p> <p>"Нет движения на мал. скорости",//2</p> <p>"Устройство безопасности",//3</p> <p>"Ошибка фаз ABC",//4</p> <p>"Движение без двигателя",//5</p> <p>"Команда из диспетчерской",//6</p> <p>"Перегревание электродвигателя",//7</p> <p>"" ,//8</p> <p>"" ,//9</p> <p>"" ,//10</p> <p>"" ,//11</p> <p>"" ,//12</p>
----	---------	----------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

			""//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 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 (свойство 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.

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 формируют 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.

и 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. По умолчанию точность – 2 цифры после запятой.

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 Pointer. The argument must be a pointer value. The value is converted to an 8 character string that represents the pointers value in hexadecimal.

e Scientific. The argument must be a floating-point value. The value is converted to a string of the form "-d.ddd...E+ddd". The resulting string starts with a minus sign if the number is negative. One digit always precedes the decimal point. The total number of digits in the resulting string (including the one before the decimal point) is given by the precision specifier in the format string—a default precision of 15 is assumed if no precision specifier is present. The "E" exponent character in the resulting string is always followed by a plus or minus sign and at least three digits.

g General. The argument must be a floating-point value. The value is converted to the shortest possible decimal string using fixed or scientific format. The number of significant digits in the resulting string is given by the precision specifier in the format string—a default precision of 15 is assumed if no precision specifier is present. Trailing zeros are removed from the resulting string, and a decimal point appears only if necessary. The resulting string uses fixed point format if the number of digits to the left of the decimal point in the value is less than or equal to the specified precision, and if the value is greater than or equal to 0.00001. Otherwise the resulting string uses scientific format.

n Number. The argument must be a floating-point value. The value is converted to a string of the form "-d,ddd,ddd.ddd...". The "n" format corresponds to the "f" format, except that the resulting string contains thousand separators.

m Money. The argument must be a floating-point value. The value is converted to a string that represents a currency amount. The conversion is controlled by the CurrencyString, CurrencyFormat, NegCurrFormat, ThousandSeparator, DecimalSeparator, and CurrencyDecimals global variables, all of which are initialized from the Currency Format in the International section of the Windows Control Panel. If

the format string contains a precision specifier, it overrides the value given by the CurrencyDecimals global variable.

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 задает 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.
dddddd	Дата в кратком формате по умолчанию.
ddddddd	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).
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 форматирования в краткой форме по умолчанию.
tt	Время с использованием форматирования 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 “с”.