WinПОС

Пакет Обработки Сигналов

Руководство программиста

Издание второе

(2.9)

БЛИЖ.409801.002-04 33

Оглавление

Об этой книге

Руководство программиста содержит подробное описание интерфейса программирования (API) WinПОС, примеры и рекомендации по оформлению программ. Отдельная часть руководства посвящена работе со встроенным редактором сценариев.

Книга рассчитана на пользователей WinПОС знакомых с азами программирования. Написание сценариев (Visual Basic Script или VBS) не требует от программиста высокой квалификации. Однако, для разработки подключаемых модулей (плагинов) может быть полезно знакомство с концепцией объектно-ориентированного программирования (ООП) и основами технологии OLE и ActiveX.

Структура книги

Часть 1 рассказывает о возможностях и области применения программного интерфейса (API) WinПОС. Описывается внутренняя структура приложения.

Часть 2 призвана помочь в выборе языка и среды программирования в зависимости от сложности решаемой задачи.

В *части 3* подробно описаны интерфейсы объектов WinПОС, свойства и методы.

Часть 4 содержит описания процедур упрощенного вызова алгоритмов.

В *пятой части* описан встроенный редактор-отладчик сценариев WinПОС.

В *Приложении* приводятся примеры и описание расположения на диске исходных текстов примеров сценариев, программ и подключаемых модулей (плагинов).

В конце книги помещен алфавитный *Указатель методов*.

Принятые обозначения

Для облегчения восприятия текста в руководстве используются следующие обозначения.

< >	В угловых скобках даются обозначения клавиш и их комбинаций, например, <Ctrl>
→	Символ → разделяет уровни меню. Таким образом, запись **Файл→Открыть…** говорит о том, что в меню **Файл** следует выбрать пункт **Открыть…**
Файл	Жирным шрифтом выделены названия пунктов меню или элементов диалоговых окон, которые можно выбрать, активизировать при помощи мышки.
Сигнал	Курсивом выделены названия глав руководства, окон WinПОС.
`signal`	Равномерным шрифтом выделен текст или числа, которые необходимо ввести с клавиатуры, прототипы функций, имена параметров и тексты примеров.
ⓘ	Важная информация, совет или рекомендация.

При описании интерфейсов также используются графические обозначения:

 - свойство,

 - метод,

 - возвращаемое значение.

Часть 1. Введение

WinПОС в вариантах поставки **профи** и **вибро** позволяет создавать свои собственные алгоритмы обработки сигнала, автоматизировать процесс обработки входного сигнала от выбора входного файла до документирования результатов обработки.

Области применения программного интерфейса (API) WinПОС:

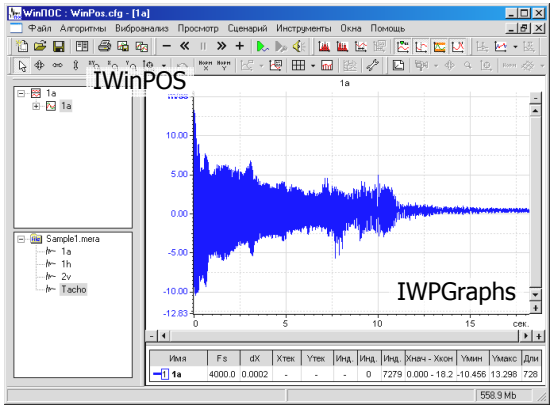
- циклическая обработка большого объема данных,
- вычисления по узкоспециализированным формулам,
- автоматический поиск характерных значений в результатах вычислений,
- программная генерация, к примеру, эталонных сигналов с заданными свойствами,
- формирование шаблонных отчетов, таблиц,
- чтение и запись данных в нестандартных форматах и т.п.

Структура приложения WinПОС

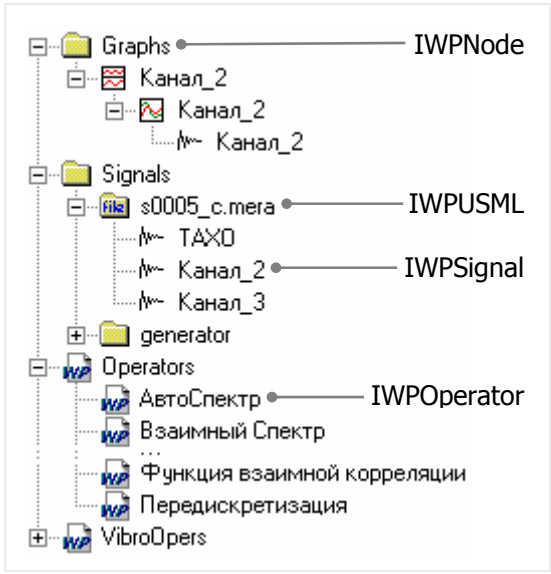
WinПОС является модульным приложением. В основе его программной модели лежит концепция объектно-ориентированного программирования (ООП). Условно объекты WinПОС можно разделить на группы, по их функциональной принадлежности:

- объекты, реализующие пользовательский интерфейс (меню, панели инструментов, окна настроек),
- объекты графической подсистемы (страницы, графики, линии),
- объекты, предоставляющие доступ к данным (сигналы, файлы данных),
- объекты, отвечающие за структурированное хранение данных (дерево объектов WinПОС)
- объекты, реализующие математические алгоритмы (операторы).

Наиболее важные объекты доступны извне, что позволяет задействовать возможности WinПОС не только через пользовательский интерфейс, но и программно. Тем самым пользователи могут с помощью некоторого инструмента программирования автоматизировать часто встречающиеся задачи, которые нельзя было предвидеть в ходе разработки приложения. Такие объекты принято называть объектами ActiveX, а приложение – OLE-сервером. Объекты WinПОС предоставляют интерфейсы, вкратце описанные ниже.



IWinPOS – основной интерфейс приложения, всё взаимодействие с приложением так или иначе осуществляется через него, т.к. через этот интерфейс доступны не только графические элементы WinПОС (графическая подсистема доступна через вызов метода `GraphAPI()`), но и реализован доступ к дереву объектов.



Дерево объектов WinПОС – это структурированное хранилище данных. Отдельные его поддеревья отображаются в окнах *дерева сигналов* (ветка “\Signals”), *дерева графиков* (ветка “\Graphs”), меню *Алгоритмы* (ветка “\Operators”) или панелях *Менеджера сигналов*.

Каждый объект (узел, «лист» дерева) доступен через интерфейс **IWPNode**. Метод `GetObject()` позволяет получить любой объект WinПОС из загруженных в его дерево. Посредством интерфейса **IWPNode** можно также осуществлять циклический перебор объектов.

К узлу дерева может быть подключен один из объектов WinПОС:

- страница, график, линия (доступны через интерфейс **IWPGraphs**),
- файл УСМЛ или МЕРА (доступен через интерфейс **IWPUSML**),
- сигнал (**IWPSignal**),
- оператор (**IWPOperator**).

Часть 2. Варианты применения

WinПОС предоставляет пользователю интерфейсы, с помощью которых можно создавать свои подключаемые модули или приложения, работающие с данными и алгоритмами WinПОС, практически в любой современной среде программирования. Для примеров выбраны *Microsoft Visual Basic Script* и *Borland Delphi*. VBScript входит в состав поставки Microsoft Windows, не требует отдельного компилятора, а несложная среда редактирования сценариев включена в состав WinПОС. Delphi заслуженно пользуется славой самой удобной среды быстрой разработки приложений (RAD) и идеально подходит для создания небольших приложений частного применения.

Рассмотрим плюсы и минусы программирования сценариев или приложений этими средствами.

VBScript

⊕ не нужен компилятор или отдельная среда разработки,

⊕ требуются лишь минимальные навыки программирования,

⊖ невозможно написать приложение с собственными диалогами для настройки или формами.

Delphi

⊕ можно создавать диалоги и формы для настроек любой степени сложности, использовать многочисленные компоненты Delphi, создавать специфические отчеты,

⊕ легко писать сложные собственные алгоритмы для обработки данных,

⊖ требуется установленный пакет Borland Delphi и соответствующие навыки программирования.

Таким образом, VBScript больше подходит для написания небольших сценариев автоматизации работы WinПОС или несложных алгоритмов и слабо подходит для обработки значительных объемов данных, а Delphi лучше применять для написания собственных быстродействующих алгоритмов обработки и создания приложений, требующих дополнительных настроек или формирования специализированных отчетов.

Ниже рассматриваются способы совместного выполнения сценариев, приложений, модулей и WinПОС.

VBScript

Самый простой способ написать свой сценарий на VBScript – скопировать пример из *Справочной системы* или непосредственно с диска, вставить его в *Редактор сценариев* (меню *Сценарий*) и переделать его, добавив необходимую функциональность. Элементы управления *Редактором сценариев* описаны в *Части 5. Встроенный редактор сценариев*. Готовый сценарий можно запустить на выполнение несколькими способами:

- из редактора сценариев - *Выполнить программу* (F5),
- из главного окна WinПОС, пункт меню **Выполнить сценарий** или кнопкой быстрого запуска сценариев в *Панели инструментов*,
- из командной строки (“winpos.exe myscript.wps”).

Вот так на VBScript будет выглядеть классический пример:

```
sub main
      DebugPrint “Hello, world!”
end sub
```

Строка «Hello, world!» будет напечатана в окне отладочной печати Редактора сценариев (этот пример имеет смысл запускать только первым способом, т.к. иначе отладочная печать не будет задействована).

Delphi

С помощью RAD Delphi можно создавать не только приложения, которые будут обращаться к объектам и методам WinПОС, но и писать подключаемые модули, в виде динамических библиотек (DLL). Такие модули могут встраиваться в среду WinПОС. Например, легко можно добавить на панель инструментов WinПОС кнопки, вызывающие функции пользовательской библиотеки.

Приложения

Приложение (EXE-файл) может получить доступ к объектам их свойствам и методам WinПОС путем создания прокси-класса следующим образом:

```
var WinPOS: TWinPOS;
...
WinPOS:= TWinPOS.Create(nil);
```

Далее можно обращаться к методам WinПОС:

```
// открываем УСМЛ с помощью стандартного диалога WinПОС
FileName:= WinPos.USMLDialog();
```

Важно! С точки зрения приложения, WinПОС является выделенным (т.е. «out-of-process» или «outproc») сервером. А это означает, что вызовы методов объектов WinПОС будут происходить с неизбежными задержками, обусловленными небыстрым межпроцессным взаимодействием. Таким образом, отдельное приложение плохо подходит для создания собственных алгоритмов, циклически обращающихся к методам GetY сигналов или подобным. Первый пример (генератор синуса) наглядно показывает эту особенность. В то же время при разовых обращениях к сигналам или алгоритмам задержки практически неразличимы. Для задач, требующих постоянного взаимодействия с объектами WinПОС более подходят подключаемые модули, описанные ниже.

Подключаемые модули (Plug-Ins)

С помощью RAD Delphi можно создавать не только приложения, которые будут обращаться к объектам и методам WinПОС, но и писать подключаемые модули, в виде динамических библиотек (DLL). Такие модули могут встраиваться в среду WinПОС. Например, легко можно добавить на панель инструментов WinПОС кнопки, вызывающие функции пользовательской библиотеки.

Для такого модуля WinПОС будет локальным сервером, и, таким образом, дополнительные временные издержки будут минимальными. Пользовательские алгоритмы по быстродействию практически не будут уступать встроенным.

Подключаемый модуль должен содержать COM-класс с дуальным интерфейсом, предоставляющим три метода: Connect(), Disconnect() и NotifyPlugin().

```
function Connect(const app: IDispatch): Integer;
function Disconnect: Integer;
function NotifyPlugin(what: Integer; var param: OleVariant):
Integer;
```

При запуске WinПОС вызывает метод Connect(), передавая указатель на себя, при завершении работы вызывается Disconnect(), а другие сообщения WinПОС передает, вызывая NotifyPlugin() с кодом и параметрами сообщения.

WinПОС загружает плагины по списку, сохраняемому в реестре. Добавление плагина в список и удаление из списка удобно совмещать с процедурами регистрации. Для этого требуется перекрыть DllRegisterServer и DllUnregisterServer.

Создание плагина шаг за шагом

1. Создайте новую библиотеку (DLL): **File→New→Other…→ ActiveX → ActiveX Library**.
2. Сохраните библиотеку: **File→Save**. В диалоге сохранения укажите имя библиотеки, например, «MyPlugin» и нажмите кнопку **Save**.
3. Создайте новый COM-объект: **File→New→Other…→ActivyX→COM Object**. Появится диалог *COM Object Wizard* (Рис. 2.1).

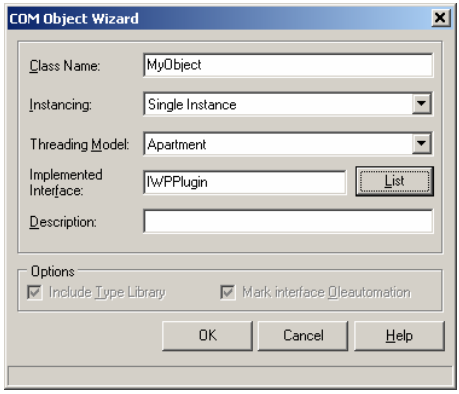


Рис 2.1 Диалог создания COM-объекта

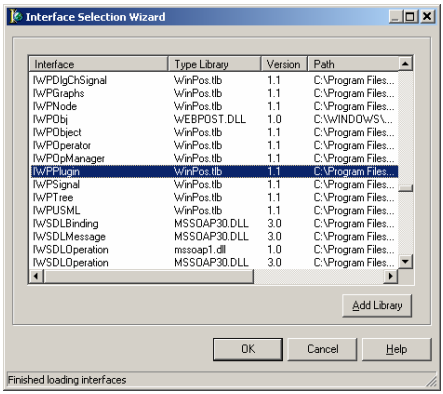


Рис 2.2 Диалог выбора интерфейса

- Укажите название класса в поле *Class Name*.
- Нажмите кнопку **List**. Откроется диалог *Interface Selection Wizard* (Рис. 2.2).
- Выберите интерфейс *IWPPlugin* (см. рис.) и нажмите **OK**. Откроется окно *Type Library*. Обратите внимание, что в левой части окна показаны новая библиотека и новый класс. Окно можно закрыть, для повторного вызова используйте **View→Type Library**.

4. Добавьте в секцию *uses* исходного файла библиотеки (здесь это MyPlugin.dpr) необходимые модули. Обычно это: SysUtils, Classes, Consts, Windows, ComServ, Registry.
5. Перекройте функцию DllRegisterServer, как показано ниже.

Здесь второй параметр метода reg.writestring() – наименование нового COM-класса. Оно состоит из имен новой библиотеки и класса, разделенных точкой. Эти имена можно увидеть в левой части окна *Type Library*.

```
function DllRegisterServer: hresult; stdcall;
var     reg: tregistry;
        buffer: array[0..255] of char;
begin
  reg := tregistry.Create;
  try
    reg.rootkey := hkey_local_machine;
    getmodulefilename(hinstance, buffer, 255);
    reg.openkey('\Software\MERA\Winpos\COMPlugins', True);
    reg.writestring(string(buffer), 'MyPlugin.MyObject');
  finally
    reg.Free;
   end;
  Result := comserv.dllregisterserver;
end;
```

6. Перекройте функцию DllUnregisterServer, как показано ниже.

```
function DllUnregisterServer: hresult; stdcall;
var
  reg: tregistry;
var
  buffer: array[0..255] of char;
begin
  reg := tregistry.Create;
  try
    reg.rootkey := hkey_local_machine;
    getmodulefilename(hinstance, buffer, 255);
    if (reg.openkey('\Software\MERA\Winpos\COMPlugins',False)) then
      reg.DeleteValue(string(buffer));
  finally
    reg.Free;
  end;
  Result := comserv.dllunregisterserver;
end;
```

7. Перекройте методы Connect(), Disconnect() и NotifyPlugin() (см. Unit1.pas).

В приведенном ниже примере в методе Connect() создается новая панель инструментов и кнопка на ней. В методе NotifyPlugin() по нажатию кнопки создается форма, которой передается управление.

Создайте форму и нарисуйте изображение для кнопки панели инструментов – это bitmap 16x16 (можно использовать встроенный редактор ресурсов: **Tools→Image Editor**).

```
var ID_Run1 : Integer=0;      // Идентификатор (код) команды
var bar_ID : Integer;         // Дескриптор панели инструментов

function TMyObject.Connect(const app: IDispatch): Integer;
var hbmp:THandle;
begin
   WP:=app as IWinPOS;
   ID_Run1 := WP.RegisterCommand();  //Получение кода команды
   bar_ID:=WP.CreateToolbar();       //Создание панели инструментов
                                     //Загрузка изображения кнопки
   hbmp:=LoadBitmap(HInstance,'TOOLBAR');
                                     //Создание кнопки
   WP.CreatetoolbarButton( bar_ID, ID_Run1, hbmp,
                           'Новое действие'#10' Новое действие');
   Result:= 0;
end;

function TMyObject.Disconnect: Integer;
begin
   Result:= 0;
end;

function TMyObject.NotifyPlugin(what: Integer;
  var param: OleVariant): Integer;
var cmdln : AnsiString;
begin
  try
    if HiWord(what)=ID_Run1          //Проверка кода команды
    then
      begin                          //Создание формы
        Application.CreateForm(TForm1, Form1);
        Form1.Show;
      end
  except
  end;
  Result:= 0;
end;
```

8. Скомпилируйте плагин. Зарегистрируйте его с помощью regsvr32. Закройте и снова запустите WinПОС. Должна появиться новая панель с копкой вызова подключаемого модуля.

Регистрация и отмена регистрации подключаемых модулей производится стандартным инструментом Windows:

```
regsvr32 myplugin.dll
regsvr32 /u myplugin.dll
```

В директории DelphiCommon (см. параграф *Установка и размещение файлов примеров*) находятся файлы Winpos_ole_TLB.pas и POSBase.pas.

Winpos_ole_TLB.pas создается автоматически и включает описания OLE-интерфейсов WinПОС (файл подключается автоматически при наследовании интерфейса IWPPlugin).

POSBase.pas содержит функции типа `RunXXXX()`, упрощающие доступ к алгоритмам WinПОС, и ряд констант (подключение этого файла может быть полезным). См. также главу 3 *Процедуры упрощенного вызова алгоритмов*.

Другие средства

Как упоминалось выше, помимо *VBScript* и *Delphi*, в качестве среды для разработки приложений или подключаемых модулей могут быть использованы и другие средства. Для работы с ними потребуется выполнить последовательность действий, направленную на генерацию соответствующего программного модуля по библиотеке типов (Type Library – TLB) WinПОС.

Так в *Borland C++ Builder*, как и в *Delphi*, последовательность будет такой:

Project→Import Type Library…→[выбрать winpos_ole]**→Create Unit**,

а в *Microsoft Visual C++* такой:

View→ClassWizard…→AddClass…→From a type library→[winpos.exe].

Часть 3. Интерфейсы WinПОС

Большинством объектов WinПОС можно манипулировать при помощи интерфейсов, перечисленных ниже.

IWinPOS	- основной интерфейс приложения
IWPGraphs	- интерфейс графической подсистемы
IWPSignal	- интерфейс сигнала
IWPUSML	- интерфейс пакетных файлов (УСМЛ и МЕРА)
IWPOperator	- интерфейс вызова математических алгоритмов
IWPNode	- элемент дерева объектов WinПОС

Следующие главы содержат описания методов этих интерфейсов в нотации ODL. Такая нотация, а ODL расшифровывается как Object Description Language (язык описания объектов), предпочтительна при описании OLE-интерфейсов. Ниже приведена таблица соответствия типов в разных языках.

ODL	Delphi	VBScript	описание
BSTR	String	BSTR	символьная строка
long	Integer	Variant	целое (32 бита)
short	Smallint	Variant	короткое целое (16 бит)
IDispatch*	IDispatch	IDispatch	указатель на интерфейс, производный от IDispatch
VARIANT_BOOL	Boolean	Variant	Булева(логическая) переменная
void	[procedure]*	[sub]*	*функция, возвращающая void, т.е. процедура
VARIANT	OleVariant	Variant	переменная

IWinPOS

Основной интерфейс приложения. С его помощью осуществляется управление средой WinПОС, доступ к дереву объектов, чтение и запись данных, взаимодействие с подключаемыми модулями, вызываются методы документирования и процедуры отладки.

Свойства

SelectedGraph

```
BSTR SelectedGraph
```

Имя выбранного графика в дереве графиков WinПОС. Свойство доступно **только по чтению.**

SelectedSignal

```
BSTR SelectedSignal
```

Имя выбранного сигнала в дереве сигналов WinПОС. Свойство доступно **только по чтению.**

Методы

Открытие и сохранение файлов данных

LoadUSML

```
IDispatch* LoadUSML(BSTR path)
```

Загрузить файл УСМЛ или МЕРА и поместить в дерево сигналов WinПОС.

[out,retval]	Объект, поддерживающий интерфейс IWPUSML
path	Имя файла

SaveUSML

```
void SaveUSML(BSTR Name, BSTR FileName)
```

Сохранить папку дерева сигналов WinПОС как файл формата МЕРА или УСМЛ.

Name	Полное имя папки в дереве сигналов WinПОС
FileName	Имя файла

LoadSignal

```
IDispatch* LoadSignal(BSTR path, long type)
```

Загрузить двоичный или текстовый файл данных и поместить результаты в дерево сигналов WinПОС.

	Объект, поддерживающий интерфейс IWPSignal
path	Имя файла
type	Тип данных файла. Список значений типа приведен в таблице ниже.

Тип файла данных	Значение константы	Описание
FT_TextWiz	3	Текстовый(ASCII) файл, открывается с помощью мастера настройки
FT_UChar	4	Массив беззнаковых целых (1 байт)
FT_INT16	5	Массив целых со знаком (2 байта)
FT_WORD	6	Массив беззнаковых целых (2 байта)
FT_INT32	7	Массив целых со знаком (4 байта)
FT_Float	8	Массивы вещественных чисел (4 байта)
FT_Double	9	Массивы вещественных чисел (8 байт)
FT_XLS	10	Столбцы таблицы Microsoft Excel

SaveSignal

```
void SaveSignal(BSTR Name, BSTR FileName, long type)
```

Сохранить сигнал как двоичный или текстовый файл данных.

Name	Полное имя сигнала в дереве сигналов WinПОС
FileName	Имя файла
type	Тип данных файла. Список значений типа приведен в описании метода LoadSignal.

Доступ к объектам WinПОС

CreateSignal, CreateSignalXY

```
IDispatch* CreateSignal(long type)
IDispatch* CreateSignalXY(long xtype, long ytype)
```

Создать новый сигнал. CreateSignalXY() создает сигнал, у которого ось X может быть неравномерной, значения могут быть заданы методом SetX(). **ВНИМАНИЕ!** Для сигнала, созданного CreateSignal(), метод SetX() не имеет смысла!

	Объект, поддерживающий интерфейс IWPSignal
`type`	Тип данных сигнала. Список значений типа приведен в таблице ниже. `xtype`, `ytype` – типы значений по X и Y.

Тип данных сигнала	Значение константы	Описание
`VT_I1`	`16`	Целые, 1 байт
`VT_UI1`	`17`	Целые беззнаковые, 1 байт
`VT_I2`	`2`	Целые, 2 байта
`VT_UI2`	`18`	Целые беззнаковые, 2 байта
`VT_I4`	`3`	Целые беззнаковые, 4 байта
`VT_R4`	`4`	Вещественные, 4 байта
`VT_R8`	`5`	Вещественные, 8 байт

GetInterval

```
IDispatch* GetInterval(IDispatch* src, long start, long count)
```

Возвращается сигнал, являющийся интервалом исходного сигнала. Метод используется при обработке диапазона исходного сигнала.

	Объект, поддерживающий интерфейс IWPSignal
`src`	Исходный сигнал
`start`	Начало отрезка, 0 .. (src.size-1)
`count`	Кол-во значений, 0 .. (src.size-start)

GetOversampled

```
IDispatch* GetOversampled(IDispatch* src, double freq)
```

Возвращаемый виртуальный сигнал позволяет трактовать данные исходного сигнала так, как будто они получены с другой частотой дискретизации. Новые значения интерполируются линейно, алгоритм передискретизации не вызывается, фильтрация не используется. Вследствие этого не рекомендуется указывать новую частоту меньше исходной.

	Объект, поддерживающий интерфейс IWPSignal
`src`	Исходный сигнал

`freq`	Новая частота

GraphAPI

```
IDispatch* GraphAPI()
```

Получить интерфейс графической подсистемы.

[out, retval]	Объект, поддерживающий интерфейс IWPGraph

Link

```
IDispatch* Link(BSTR Path, BSTR Name, IDispatch*
Object)
```

Поместить объект в дерево **WinПОС**.

[out, retval]	Объект, поддерживающий интерфейс IWPNode
`Path`	Путь в дереве **WinПОС**
`Name`	Имя объекта
`Object`	Объект, который требуется поместить в дерево

Unlink

```
void Unlink(IDispatch* Object)
```

Удалить объект из дерева.

`Object`	Объект, который требуется удалить из дерева

GetObject

```
IDispatch* GetObject(BSTR path)
```

Найти объект в дереве **WinПОС** по имени.

[out, retval]	Указатель на интерфейс запрашиваемого объекта
`path`	Строка, путь в дереве **WinПОС**

GetNode

```
IDispatch* GetNode(IDispatch* Object)
```

Получить позицию размещения (т. н. точку монтирования, узел) объекта в дереве **WinПОС**.

[out, retval]	Объект, поддерживающий интерфейс IWPNode
`Object`	Объект.

Управление средой WinПОС

USMLDialog

```
BSTR USMLDialog()
```

Выбрать файл УСМЛ или МЕРА с помощью стандартного диалога WinПОС.

	Полное имя файла

Refresh

```
void Refresh()
```

Производится обновление всех окон WinПОС. Рекомендуется использовать после вызовов методов, изменяющих состояние WinПОС, таких, как Link().

DoEvents

```
void DoEvents()
```

Обработать все события, накопившиеся за время работы продолжительной задачи. Процедура используется в процессе длительных вычислений для того, чтобы избежать эффекта "зависания" программы. DoEvents() приостанавливает выполнение сценария и позволяет **WinПОС** обработать оконные сообщения.

AddTextInLog

```
void AddTextInLog(BSTR text,BSTR
exttext,VARIANT_BOOL show)
```

Добавить строку текста в журнал.

`text`	Текстовая строка для журнала
`exttext`	Строка дополнительных параметров
`show`	true, если надо отобразить окно журнала, false, если нет

Взаимодействие с подключаемыми модулями

Методы этого раздела предоставляют доступ к управляющим элементам среды WinПОС (главному окну, панелям инструментов, меню) и предназначены для использования при создании подключаемых модулей.

MainWnd

```
long MainWnd()
```

Метод возвращает хэндл (handle) главного окна WinПОС, который может потребоваться подключаемым модулям, если в них определены собственные окна и диалоги.

	хэндл (handle) главного окна WinПОС

RegisterCommand

```
long RegisterCommand()
```

Метод возвращает уникальное число, которое можно использовать как идентификатор команды.

	Число, уникальный код команды, события.

CreateToolbar

```
long CreateToolbar()
```

Создать новую панель инструментов.

	Указатель на панель инструментов

CreatetoolbarButton

```
long CreatetoolbarButton(long bar, long command,
long picture, BSTR hint)
```

Добавить кнопку-«инструмент» на панель инструментов.

	Ненулевое в случае успешного добавления кнопки, 0 – в противном случае
`bar`	Указатель на панель инструментов. Можно получить при помощи CreateToolbar()
`command`	Присвоенная команда. Можно получить при помощи RegisterCommand ()
`picture`	Хендл изображения для кнопки
`hint`	Текст подсказки

ShowToolbar

```
void ShowToolbar(long bar, long visible)
```

Отображать или скрыть панель инструментов.

bar	Указатель на панель инструментов. Можно получить при помощи CreateToolbar()
visible	1 – отображать, 0 - скрыть

CreatemenuItem

```
long CreatemenuItem(long Command, long reserved,
BSTR text, long style, long picture)
```

Создать новый пункт меню.

	Ненулевое в случае успешного добавления строки меню, 0 – в противном случае
Command	Присвоенная команда. Можно получить при помощи RegisterCommand ()
reserved	Каждый байт этого числа (если не равен FF) представляет собой позицию в подменю данного уровня, куда будет добавлен новый пункт меню. Пример: 0xFFFF0301 - четвертое подменю главного меню, после второй позиции (начало отсчета - 0)
text	Строка меню
style	Вид элемента меню. Для обычного элемента меню устанавливается в 0. Значения констант стилей приведены в таблице ниже.
picture	Хендл изображения для пункта меню. Используется только при установленном стиле MF_BITMAP.

Вид элемента меню	Значение
MF_ENABLED	0h
MF_GRAYED	1h
MF_DISABLED	2h
MF_UNCHECKED	0h
MF_CHECKED	8h
MF_USECHECKBITMAPS	200h
MF_STRING	0h

Вид элемента меню	Значение
MF_BITMAP	4h
MF_OWNERDRAW	100h
MF_POPUP	10h
MF_MENUBARBREAK	20h
MF_MENUBREAK	40h
MF_UNHILITE	0h
MF_HILITE	80h

Документирование, печать результатов

SaveImage

```
boolean SaveImage(BSTR fname, BSTR comment)
```

Сохранить отображаемую страницу графиков в файл или буфер обмена.

	true – если операция прошла успешно, false – в противном случае
`fname`	Имя файла. Если передана пустая строка – изображение будет помещено в буфер обмена
`comment`	Строка подписи. Например, «Рис.1 Исходные уровни». Если задана пустая строка – подпись не печатается.

PrintPreview, Print

```
void PrintPreview(BSTR comment)
void Print(BSTR comment)
```

Распечатать отображаемую страницу графиков. PrintPreview выдает окно предварительного просмотра. Print – печатает на принтер, используя текущие настройки принтера и страницы.

`comment`	Строка подписи. Например, «Рис.1 Исходные уровни». Если задана пустая строка – подпись не печатается.

VBScript. Работа с двоичными файлами данных

Эти методы расширяют ограниченные возможности VBScript в части работы с двоичными файлами. Нецелесообразно обращаться к этим методам из Delphi, т.к. в Delphi можно вызывать функции для работы с файлами напрямую.

FileOpen

```
BSTR FileOpen(long isOpen, BSTR ext, BSTR fname,
long flags, BSTR filter)
```

Открыть стандартный файловый диалог и выбрать имя файла.

	Полное имя файла
`isOpen`	true - диалог открытия файла, false - сохранения
`ext`	Расширение файла по умолчанию
`fname`	Начальное имя файла
`flags`	Флаги для настройки внешнего вида и поведения диалога. Некоторые наиболее полезные приведены в таблице ниже, остальные можно найти в файле POSBase.pas и соответствующей литературе (Описание структуры OPENFILENAME).

`filter`	Набор фильтров для диалога. Например, строка *"USML files|*.usm|All files|*.*||"* предложит выбрать файлы с расширением .usm или все файлы.

Флаг	Значение	Описание
`OFN_ALLOWMULTISELECT`	`200h`	Разрешает выбирать несколько файлов сразу.
`OFN_CREATEPROMPT`	`2000h`	Если пользователь указал файл, которого не существует, то диалог запросит разрешение создать новый файл с указанным именем.
`OFN_FILEMUSTEXIST`	`1000h`	В поле *Имя файла* разрешается вводить только имена существующих файлов. Если указан этот флаг и введено некорректное имя файла - выдается предупреждение. Используется совместно с OFN_PATHMUSTEXIST.
`OFN_NOCHANGEDIR`	`8h`	Возвращает текущей папке первоначальное значение, если пользователь переходит из одной папки в другую при поиске файлов.
`OFN_NONETWORKBUTTON`	`20000h`	Убирает кнопку **Сеть** диалога
`OFN_NOREADONLYRETURN`	`8000h`	Поле *Только для чтения* не выбрано и возвращаемый файл не находится защищенной от копирования папке.
`OFN_OVERWRITEPROMPT`	`2h`	Диалог *Сохранить как* выдаст предупреждающее сообщение, если файл уже существует. Пользователь должен подтвердить перезапись файла.
`OFN_PATHMUSTEXIST`	`800h`	Пользователь может указать только существующий путь и имена файлов. При вводе неправильного имени файла или пути появится предупреждающее окно.

OpenFile

```
long OpenFile(BSTR Path, long flags)
```

Открыть или создать новый файл.

	Файловый хэндл, используется в дальнейших вызовах (см. параметр hFile)
`Path`	Имя файла (с путем)
`flags`	Тип доступа. Значения приведены в таблице ниже.

Флаг	Значение	Описание
`READ_WRITE`	`100h`	Открыть файл по чтению и записи (GENERIC_READ | GENERIC_WRITE), 0 – только по чтению (GENERIC_READ).
`SHARE_READ`	`1000h`	Этот файл одновременно можно будет открыть только по чтению (FILE_SHARE_READ), 0 – по чтению и записи (FILE_SHARE_READ | FILE_SHARE_WRITE).

CloseFile

```
void CloseFile(long hFile)
```

Закрыть файл.

hFile	Файловый хэндл

SeekFile

```
long SeekFile(long hFile, long Pos, long flags)
```

Изменить позицию файлового указателя.

	Новая позиция файлового указателя
hFile	Файловый хэндл
Pos	Позиция, в которую желательно переместить файловый указатель
flags	Точка отсчета при перемещении. Значения приведены в таблице ниже.

Флаг	Значение	Описание
FILE_BEGIN	0	За ноль принимается начало файла
FILE_CURRENT	1	Ноль – текущая позиция указателя файла
FILE_END	2	Ноль – конец файла

ReadByte, ReadWord, ReadLong, ReadFloat, ReadDouble

```
VARIANT ReadByte(long hFile)
VARIANT ReadWord(long hFile)
VARIANT ReadLong(long hFile)
VARIANT ReadFloat(long hFile)
VARIANT ReadDouble(long hFile)
```

Прочитать из двоичного файла число в соотв. формате.

	Число, прочитанное из файла
hFile	Файловый хэндл

WriteByte, WriteWord, WriteLong, WriteFloat, WriteDouble

```
void WriteByte(long hFile, short Value)
void WriteWord(long hFile, long value)
void WriteLong(long hFile, long Value)
void WriteFloat(long hFile, float value)
void WriteDouble(long hFile, double Value)
```

Записать в двоичный файл число в соотв. формате.

hFile	Файловый хэндл
Value	Число, которое должно быть записано в файл

VBScript. Отладка

DebugPrint, DebugPrintLn

```
void DebugPrint(VARIANT arg)
void DebugPrintLn(VARIANT arg)
```

Отладочная печать в окно вывода *Редактора сценариев*. Применяется только при запуске из *Редактора сценариев*. DebugPrintLn() в отличие от DebugPrint() переводит строку.

Arg	Символьная строка. Пример: *DebugPrintLn "Max= "+FormatNumber(max,6,0,0,0)+";"*

IWPGraphs

Интерфейс графической подсистемы.

Для управления графиками сначала требуется получить данный интерфейс, вызвав метод GraphAPI интерфейса IWinPOS.

Последовательность действий при создании новой страницы для отображения в ней сигнала (например, результата работы скрипта или плагина):

```
// получаем доступ к графической подсистеме WinПОС
api := GraphAPI as IWPGraphs;

// создаем новую страницу для графиков
hPage := api.CreatePage;

// страница всегда создается с одним графиком, получаем его
hGraph := api.GetGraph(hPage, 0);

// график всегда имеет как минимум одну ось Y, получаем ее
hYAxis := api.GetYAxis(hGraph, 0);

// создаем  новую линию в графике
api.CreateLine(hGraph, hYAxis, signal.Instance);

// нормализуем график
api.NormalizeGraph(hGraph);
```

Методы

CreatePage

```
long CreatePage()
```

Создать новую страницу. Будут учтены настройки по умолчанию.

	Указатель на новую страницу

DestroyPage

```
void DestroyPage(long hPage)
```

Удалить страницу.

`hPage`	Указатель на страницу

CreateGraph

```
long CreateGraph(long hPage)
```

Создать новый график. Будут учтены настройки по умолчанию.

	Указатель на новый график
`hPage`	Указатель на страницу, в которой требуется создать график

DestroyGraph

```
void DestroyGraph(long hGraph)
```

Удалить график.

`hGraph`	Указатель на график

CreateYAxis

```
long CreateYAxis(long hGraph)
```

Добавить новую ось ординат.

	Указатель на новую ось
`hGraph`	Указатель на график, куда будет добавлена ось

DestroyYAxis

```
void DestroyYAxis(long hAxis)
```

Удалить ось.

hAxis	Указатель на ось

CreateLine

```
long CreateLine(long hGr, long hAx, long hSig)
```

Создать новую линию. Будут учтены настройки по умолчанию.

	Указатель на новую линию
hGr	Указатель на график, в который будет добавлена линия
hAx	Указатель на Y-ось
hSig	Указатель на сигнал

DestroyLine

```
void DestroyLine(long hLine)
```

Удалить линию.

hLine	Указатель на линию

GetPageCount

```
long GetPageCount()
```

Число страниц графиков.

	Число страниц графиков

GetGraphCount

```
long GetGraphCount(long hPage)
```

Число графиков на странице.

	Число графиков на странице
hPage	Указатель на страницу

GetYAxisCount

```
long GetYAxisCount(long hGr)
```

Число Y-осей графика.

	Число Y-осей
hGr	Указатель на график

GetLineCount

```
long GetLineCount(long hGr)
```

Число линий в графике.

	Число линий в графике
hGr	Указатель на график

GetPage

```
long GetPage(long nPage)
```

Получить страницу по порядковому номеру.

	Указатель на страницу
nPage	Порядковый номер страницы

GetGraph

```
long GetGraph(long hPage, long nGraph)
```

Получить график по порядковому номеру.

	Указатель на график
hPage	Указатель на страницу
nGraph	Порядковый номер графика

GetYAxis

```
long GetYAxis(long hGr, long nAxis)
```

Получить Y-ось по порядковому номеру.

	Указатель на ось
hGr	Указатель на график
nAxis	Порядковый номер оси

GetLine

```
long GetLine(long hGr, long nLine)
```

Получить линию по порядковому номеру.

	Указатель на линию
hGr	Указатель на график
nLine	Порядковый номер линии

GetSignal

```
IDispatch* GetSignal(long hLine)
```

Получить ссылку на сигнал, который изображает линия hLine.

⧉	Объект, поддерживающий интерфейс IWPSignal
`hLine`	Указатель на линию

GetXCursorPos, SetXCursorPos

```
void GetXCursorPos(long hGraph, double* px, BOOL bSecond)
void SetXCursorPos(long hGraph, double x, BOOL bSecond)
```

Получить или установить позицию курсора.

`hGraph`	Указатель на график
`x`	Позиция курсора
`px`	Адрес переменной для возврата позиции курсора
`bSecond`	Работать с позицией второй линии курсора (только для разностного курсора)

GetPageRect, SetPageRect

```
void GetPageRect(long hPage, long* left, long* top, long* right, long* bottom)
void SetPageRect(long hPage, long left, long top, long right, long bottom)
```

Получить, установить размеры страницы графиков.

`hPage`	Указатель на страницу
`left`	Координаты левого края страницы
`top`	Координаты верха страницы
`rigth`	Координаты правого края страницы
`bottom`	Координаты низа страницы

SetPageDim

```
void SetPageDim(long hPage, long mode, long width, long height)
```

Установить режим расположения графиков.

`hPage`	Указатель на страницу
`mode`	Тип расположения графиков. Возможные варианты

	приведены в таблице ниже.
`width`	Число графиков на странице по горизонтали
`height`	Число графиков на странице по вертикали

Флаг	Значение	Описание
`PAGE_DM_VERT`	0	Графики располагаются вертикально
`PAGE_DM_HORZ`	1	Графики располагаются горизонтально
`PAGE_DM_TABLE`	2	Графики располагаются в виде таблицы width*height

GetXMinMax, SetXMinMax

```
void GetXMinMax(long hGR, double* pmin, double* pmax)
void SetXMinMax(long hGr, double min, double max)
```

Получить, установить границы по оси абсцисс. Таким образом, можно установить или получить видимый диапазон сигнала.

`hGr`	Указатель на график
`min, pmin`	Минимальное значение или указатель на него
`max, pmax`	Максимальное значение или указатель на него

GetYAxisMinMax, SetYAxisMinMax

```
void GetYAxisMinMax(long hAxis, double* pmin, double* pmax)
void SetYAxisMinMax(long hAxis, double min, double max)
```

Получить, установить границы выбранной оси ординат.

`hAxis`	Указатель на Y-ось
`min, pmin`	Минимальное значение или указатель на него
`max, pmax`	Максимальное значение или указатель на него

NormalizeGraph

```
void NormalizeGraph(long hGr)
```

Выполнить нормализацию графика.

`hGr`	Указатель на график

Invalidate

```
void invalidate(long hGraph)
```

Обновить поле отрисовки графика.

`hGraph`	Указатель на график

ActiveGraphPage

```
long ActiveGraphPage()
```

Получить указатель на активную страницу графиков.

	Указатель на активную страницу графиков

ActiveGraph

```
long ActiveGraph(long hPage)
```

Получить указатель на активный график.

	Указатель на активный график
`hPage`	Указатель на страницу графиков

Folder2Graphs, Folder2GraphsRecursive

```
void Folder2Graphs(IDispatch* Node)
void Folder2GraphsRecursive(IDispatch* Node)
```

Разместить все сигналы указанной папки или пакетного файла на новой странице. Второй вариант метода обходит вложенные папки.

`Node`	Объект, поддерживающий интерфейс IWPNode

Locate

```
IDispatch* Locate(long hGrItem)
```

Найти элемент графика в дереве по указателю.

	Объект, поддерживающий интерфейс IWPNode
`hGrItem`	Указатель на страницу графиков, график или линию

SetPageOpt

```
void SetPageOpt(long hPage, long opt, long mask)
```

Установить параметры выбранной страницы.

`hPage`	Указатель на страницу
`opt`	Битовое поле: установить (1) или снять (0) бит состояния. См. таблицу ниже.

mask	Маска. Показывает, какие биты поля opt следует изменить. См. таблицу ниже.

Флаг	Значение	Описание
PGOPT_SHOWNAME	1	Отображение названия страницы
PGOPT_SINGLEX	2	Флаг одной оси X на страницу
PGOPT_SINGLEY	4	Флаг одной оси Y на страницу
PGOPT_SINCCURS	8	Синхронизация курсоров

SetGraphOpt

```
void SetGraphOpt(long hGraph, long opt, long mask)
```

Установить параметры выбранного графика.

hGraph	Указатель на график
opt	Битовое поле: установить (1) или снять (0) бит состояния. См. таблицу ниже.
mask	Маска. Показывает, какие биты поля opt следует изменить. См. таблицу ниже.

Флаг	Значение	Описание
GROPT_SHOWNAME	1h	Флаг отрисовки названия
GROPT_YINDENT	2h	10% отступ для линий сверху и снизу
GROPT_SUBGRID	4h	Флаг отрисовки пунктирных линий на сетке
GROPT_GRIDLABS	8h	Значения линий в узлах сетки
GROPT_LINENUMS	10h	Показывать номера линий
GROPT_AUTONORM	20h	Автоматически нормализоовать график при добавлении новых линий
GROPT_POLAR	40h	Полярные координаты
GROPT_AXCOLUMN	80h	Флаг размещения осей Y друг над другом
GROPT_AXROW	100h	Флаг размещения осей Y друг за другом

GetAxisOpt, SetAxisOpt

```
void GetAxisOpt(long hGraph, long hAxis, long* opt, double *minR, double *maxR, BSTR *szname, BSTR *szftempl, long *color)
void SetAxisOpt(long hGraph, long hAxis, long opt, long mask, double minR, double maxR, BSTR szname, BSTR szftempl, long color)
```

Получить / установить параметры выбранной оси.

hGraph	Указатель на график (нужен для оси абсцисс)
hAxis	Указатель на ось (для оси абсцисс: 0)
opt	Битовое поле: установить (1) или снять (0) бит

	состояния. См. таблицу ниже.
mask	Маска. Показывает, какие биты поля opt следует изменить. Также показывает, надо ли изменить имя или формат оцифровки оси. См. таблицу ниже.
minR,maxR	Отображаемый диапазон оси (с флагом AXOPT_RANGE – полный диапазон, т.е. пределы масштабирования)
szname	Имя оси (обычно берется размерность)
szftempl	Формат оцифровки (описан в *Руководстве Пользователя*, ч.5, *Создание графиков*, *Настройка графиков*)
color	Цвет в формате RGB (белый = FFFFFFh, черный = 0h)

Флаг	Значение	Описание
AXOPT_LOG	1h	Логарифмический масштаб
AXOPT_FZERO	2h	Добавлять нули в конце числа ("1.500" вместо "1.5")
AXOPT_TIME	4h	Добавить шкалу времени в формате «чч.мм.сс.мск»
AXOPT_COLOR*	8h	Установить вручную цвет оцифровки оси
AXOPT_RANGE*	10h	Установить полный диапазон оси
AXOPT_NAME*	20h	Установить имя или размерность оси
AXOPT_FORMAT*	40h	Установить формат оцифровки

* - используются только в поле mask

SetLineOpt

```
void SetLineOpt(long hLine, long opt, long mask, long width, long color)
```

Установить параметры выбранной линии.

hLine	Указатель на линию
opt	Битовое поле: установить (1) или снять (0) бит состояния. См. таблицу ниже.
mask	Маска. Показывает, какие биты поля opt следует изменить. Также показывает, надо ли изменить толщину или цвет линии. См. таблицу ниже.
width	Толщина линии
color	Цвет в формате RGB (белый = FFFFFFh, черный = 0h)

Флаг	Значение	Описание
LNOPT_LINE2BASE	1h	Добавлять вертикальные линии от значения до 0
LNOPT_ONLYPOINTS	2h	Флаг соединения точек линиями
LNOPT_VISIBLE	4h	Флаг отображения/скрытия линии

LNOPT_HIST	8h	В виде гистограммы
LNOPT_HISTTRANSP	40h	"Прозрачная" гистограмма
LNOPT_PARAM	80h	В виде Y(idx), с реальными значениями на шкале оси X
LNOPT_INTERP	300h	Порядок интерполяции (два бита!)
LNOPT_COLOR*	10h	Изменить цвет линии. См. поле color
LNOPT_WIDTH*	20h	Изменить толщину линии. См. поле width

* - используются только в поле mask

AddLabel

```
void AddLabel(long hLine, long mode, double x, double offsX, double offsY, BSTR text)
```

Добавить выноску.

hLine	Указатель на линию
mode	Тип выноски. См. таблицу ниже
x	Значение времени, к которому привязана выноска
offsX, offsY	Положение выноски в поле графика, выраженное в процентах относительно размеров графика
text	Текст выноски, если mode = LAB_TEXT

Флаг	Значение	Описание
LAB_SINGLE	0	На одну линию
LAB_MULTI	1	На все линии
LAB_TEXT	2	Текстовая выноска

AddComment

```
void AddComment(long hGr, BSTR text, double x, double y, double dx, double dy)
```

Добавить комментарий.

hGr	Указатель на график
text	Текст комментария
x, y	Положение левого верхнего угла комментария, выраженное в процентах относительно размеров графика
dx, dy	Размеры комментария, выраженные в процентах относительно размеров графика

SaveSession, LoadSession

```
BOOL SaveSession(BSTR path)
BOOL LoadSession(BSTR path)
```

Сохранить текущий сеанс работы и загрузить ранее сохраненный сеанс.

[retval]	Результат операции (TRUE – успешно)
`path`	Путь на диске к файлам сессии.

IWPSignal

Интерфейс сигнала.

Свойства

size

```
long size
```

Количество значений (измерений) сигнала.

DeltaX

```
double DeltaX
```

Шаг по оси X для сигнала с равномерной осью абсцисс. Для сигнала с неравномерной осью DeltaX=0.

StartX

```
double StartX
```

Начальное значение по оси X для сигнала с равномерной осью абсцисс. Для сигнала с неравномерной осью StartX содержит значение абсциссы первого элемента сигнала.

SName

```
BSTR SName
```

Имя сигнала.

NameY

```
BSTR NameY
```

Единицы измерения значений сигнала, строка.

NameX

```
BSTR NameX
```

Единицы измерения оси абсцисс, строка.

Comment

```
BSTR Comment
```

Комментарий, дополнительная, расширенная текстовая информация по данному сигналу.

Characteristic

```
long Characteristic
```

Характеристика сигнала, влияет на вид графика. Возможные значения приведены в таблице.

Characteristic	Значение	Описание
SC_NORMAL	0	Обычный сигнал
SC_SPECTR	1	Спектр
SC_LOGSPEC	2	Логарифмический спектр
SC_LOGX	4	Сигнал с логарифмической осью абсцисс
SC_AMP	8	Амплитуда
SC_FASE	16	Фаза
SC_PARAM	32	Параметрический сигнал

MinY, MaxY

```
double MinY
double MaxY
```

Минимальное и максимальное значения сигнала. Если MaxY< MinY, то это означает, что минимальное и максимальное значения сигнала еще не рассчитаны.

MinX, MaxX

```
double MinX
double MaxX
```

Минимальное и максимальное значения оси абсцисс сигнала. Если сигнал содержит изменения параметра во времени, то это начало и окончание регистрации параметра. Доступны только по чтению, изменить MinX и MaxX можно, изменив StartX и DeltaX или, при неравномерном шаге, с помощью метода SetX().

k0, k1

```
double k0
double k1
```

Коэффициенты калибровочной характеристики, заданной как линейная функция: $y = k_1 \cdot (x – k_0)$.

Методы

Instance

```
long Instance()
```

Возвращает указатель на объект, предоставляющий этот интерфейс. Используется для передачи объекта в качестве параметра.

↵	Указатель на объект

IndexOf

```
long IndexOf(double x)
```

Возвращается индекс (порядковый номер) значения, соответствующего заданному времени, а если точного значения для этого времени нет – индекс ближайшего значения.

↵	Номер элемента сигнала из диапазона 0..(size-1)
x	Заданное значение времени (оси абсцисс)

GetY, GetX

```
double GetY(long index)
double GetX(long index)
```

Возвращается значение элемента сигнала по оси ординат или абсцисс.

↵	Значение сигнала по оси ординат или абсцисс
index	Номер элемента сигнала из диапазона 0..(size-1)

GetYX

```
double GetYX(double x, int pow)
```

Возвращается значение сигнала, соответствующего заданному времени, а если точного значения для этого времени нет – интерполированное значение. Pow определяет метод интерполяции.

↵	Значение сигнала
x	Заданное значение времени (оси абсцисс)
pow	Вид интерполяции: 0 – отсутствует (берется последнее по времени значение), 1 – линейная интерполяция, 2 – квадратный полином, 3 – интерполяция кубическими локальными сплайнами.

SetY, SetX

```
void SetX(long index, double value)
void SetY(long index, double value)
```

Устанавливается значение элемента сигнала по оси ординат или абсцисс. SetX не имеет смысла для сигналов с равномерной осью X, для таких сигналов следует задавать свойства StartX и DeltaX.

index	Номер элемента сигнала из диапазона 0..(size-1)
value	Новое значение

IWPUSML

Интерфейс пакетных файлов (УСМЛ и МЕРА).

Свойства

FileName

```
BSTR FileName
```

Полное имя файла.

ParamCount

```
long ParamCount
```

Количество параметров пакетного (УСМЛ или МЕРА) файла.

Name, Test, Date

```
BSTR Name
BSTR Test
BSTR Date
```

Название изделия, испытания и дата испытания в формате “dd.mm.yy”

Методы

Instance

```
long Instance()
```

Возвращает указатель на объект, предоставляющий этот интерфейс. Используется для передачи объекта в качестве параметра.

[out]	Указатель на объект

Parameter

```
IDispatch* Parameter(long index)
```

Возвращает из пакетного файла сигнал по порядковому номеру.

	Объект, поддерживающий интерфейс IWPSignal
index	Порядковый номер сигнала из диапазона 0..(ParamCount-1)

FileSave

```
void FileSave()
```

Сохранить изменения в файле.

AddParameter

```
void AddParameter(IDispatch* signal)
```

Добавить сигнал в файл УСМЛ или МЕРА.

signal	Объект, поддерживающий интерфейс IWPSignal

DeleteParameter

```
void DeleteParameter(long index)
```

Удалить параметр с указанным порядковым номером.

index	Порядковый номер сигнала из диапазона 0..(ParamCount-1)

IWPOperator

Интерфейс вызова математических алгоритмов. В принятой терминологии *оператор* – это математический *алгоритм* в совокупности с *параметрами* его выполнения.

Свойства

Name

```
BSTR Name
```

Сокращенное название алгоритма.

Fullname

```
BSTR Fullname
```

Полное название алгоритма.

nSrc, nDst

```
long nSrc
long nDst
```

Число входных и выходных параметров. Например, для амплитудного спектра nSrc=1, nDst=1, а для функции взаимной корреляции nSrc=2, nDst=1.

Методы

Instance

```
long Instance()
```

Возвращает указатель на объект, предоставляющий этот интерфейс. Используется для передачи объекта в качестве параметра.

	Указатель на объект

Exec

```
long Exec(VARIANT src, VARIANT src2, VARIANT dst,
VARIANT dst2)
```

Выполнить алгоритм. Если фактическое число входных (выходных) сигналов алгоритма меньше двух, то неиспользуемые параметры игнорируются.

	Код ошибки. В случае успеха равен нулю.
`src`	Первый входной сигнал
`src2`	Второй входной сигнал
`dst`	Первый выходной сигнал
`dst2`	Второй выходной сигнал

Error

```
long Error()
```

Получить код последней ошибки.

	Код ошибки. В случае отсутствия ошибок равен нулю.

MsgError

```
BSTR MsgError()
```

Получить текстовое описание последней ошибки.

↵	Текстовое описание последней ошибки.

getProperySet

```
BSTR getProperySet()
```

Получить список опций алгоритма.

↵	Строка имен опций алгоритма, перечисленных через запятую.

setProperty

```
void setProperty(BSTR name, VARIANT value)
```

Установить значение выбранного свойства алгоритма.

`name`	Имя свойства
`value`	Новое значение свойства

getProperty

```
VARIANT getProperty(BSTR name)
```

Прочитать значение выбранного свойства алгоритма.

↵	Значение свойства
`name`	Имя свойства

loadProperties

```
void loadProperties(BSTR values)
```

Установить значение набора свойств алгоритма.

`values`	Новые значения набора свойств. Строка формата « имя_свойства1 = значение_свойства1 , имя_свойства2 = значение_свойства2 , … » . Пример: *" kindFunc = 3 , numPoints = 1024 , nBlocks = 1 "*. Значения пропущенных свойств не присваиваются (сохраняются значения по умолчанию).

getPropertyValues

```
BSTR getPropertyValues()
```

Прочитать значение всех свойств алгоритма.

Строка формата « имя_свойства1 = значение_свойства1 , имя_свойства2 = значение_свойства2 , … » .

SetupDlg

```
long SetupDlg()
```

Вызвать диалог настройки опций алгоритма и выбора исходных сигналов.

Результат выполнения диалога. В таблице ниже приведены возвращаемые значения.

Результат	Значение	Описание
IDOK	1	Алгоритм запущен на выполнение
IDCANCEL	2	Отмена операции
IDERROR	-1	Ошибка открытия диалога

IWPNode

Узел дерева объектов WinПОС, «точка монтирования» объекта.

Свойства

Name

```
BSTR Name
```

Имя узла, объекта.

ChildCount

```
long ChildCount
```

Количество дочерних элементов данного узла. Например, для узла пакетного файла это число сигналов.

Методы

Instance

```
long Instance()
```

Возвращает указатель на объект, предоставляющий этот интерфейс. Используется для передачи объекта в качестве параметра.

	Указатель на объект

Root

```
IDispatch* Root()
```

Указатель на корневой узел дерева объектов **WinПОС**.

	Объект, поддерживающий интерфейс IWPNode

AbsolutePath, RelativePath

```
BSTR AbsolutePath()
BSTR RelativePath(IDispatch* baseNode)
```

Полный или относительный путь узла.

	Строка, путь узла.
`baseNode`	Узел, от которого вычисляется относительный путь

Reference

```
IDispatch* Reference()
```

Объект, на который ссылается данный узел.

	Объект, на который ссылается данный узел.

IsDirectory

```
long IsDirectory()
```

Проверка, является ли данный узел папкой или пакетным файлом.

	1 - если папка, 0 - в противном случае.

GetReferenceType

```
long GetReferenceType()
```

Тип объекта, на который ссылается данный узел.

	Тип объекта. Варианты типов приведены в таблице ниже.

Тип	Значение	Описание
OT_FOLDER	0	Простая папка
OT_PFILE	1	Файл формата УСМЛ или МЕРА
OT_SIGNAL	2	Сигнал

Link

```
IDispatch* Link(IDispatch* Object, BSTR name, long flag)
```

Поместить объект в список дочерних узлов данного узла.

	Объект, поддерживающий интерфейс IWPNode
Object	Объект, который требуется поместить в дерево
name	Имя объекта
flag	Если узел с таким именем уже существует, то при flag=1 будет изменено имя нового узла («Имя» перейдет в «Имя#1»), при flag=0 старый объект будет замещен.

Unlink

```
void Unlink(BSTR objname)
```

Удалить дочерний узел с заданным именем данного узла.

objname	Строка, имя узла для удаления

IsChild

```
long IsChild(IDispatch* testNode)
```

Проверить, является ли узел дочерним для данного узла.

	1 – является, в противном случае – 0.
testNode	Объект, поддерживающий интерфейс IWPNode

GetNode

```
IDispatch* GetNode(BSTR path)
```

Получить дочерний узел по имени.

	Объект, поддерживающий интерфейс IWPNode
path	Путь к дочернему узлу

At

```
IDispatch* At(long index)
```

Получить дочерний узел по порядковому номеру.

	Объект, поддерживающий интерфейс IWPNode
`index`	Порядковый номер, 0 .. (ChildCount-1)

Часть 4. Вызов алгоритмов

Алгоритмы доступны через дерево объектов **WinПОС**. То есть к алгоритмам можно обращаться по имени, выбирая из дерева объектов. Последовательность вызовов такая: получить оператор, загрузить необходимые настройки, выполнить оператор. Например, так выглядит вызов автоспектра с текущими настройками:

```
var oper : IWPOperator;
…
oper:= WINPOS.GetObject('/Operators/АвтоСпектр') as IWPOperator;
oper.Exec (signal, signal, refvar(dst), refvar(dst2));
…
```

Загрузить настройки алгоритма можно либо поочередно, методом **setProperty**(), либо одновременно, методом **loadProperties**(), см. выше описание интерфейса IWPOperator. Значения пропущенных параметров не присваиваются (сохраняются значения по умолчанию). Так выглядит тот же вызов автоспектра с указанием настроек:

```
var oper : IWPOperator;
…
oper:= WINPOS.GetObject('/Operators/АвтоСпектр') as  IWPOperator;
oper.loadProperties(' kindFunc = 3 , numPoints = 1024 , typeWindow
= 1 ');
oper.Exec(signal, signal, refvar(dst), refvar(dst2));
…
```

Процедуры упрощенного вызова алгоритмов

Вызов наиболее употребимых алгоритмов автоматизирован. В файле **POSBase.pas** (для VBScript – в файле **WinPOS.wps**) реализованы процедуры, оформленные в стиле вызовов «Командного режима ПОС», упрощающие вызов алгоритмов. Пример, приведенный выше, можно переписать так:

```
RunFFT(signal, dst, dst2, Opt, Err);
```

Названия процедур приведены ниже вместе с описанием настроек алгоритмов. Обозначения: `Src, Src2, Dst, Dst2` – это переменные, указывающие на объекты с интерфейсом IWPSignal, `Err` – код ошибки (0, если ошибок нет), `Opt` – строка настроек, где параметры со значениями перечислены через запятую: « имя_свойства1 = значение1 , имя_свойства2 = значение2 , … » . Пример: *" kindFunc = 3 , numPoints = 1024 , nBlocks = 1 "*.

Алгоритмы на основе быстрого преобразования Фурье (БПФ)

Алгоритмы, выполняющие БПФ, имеют ряд общих настроек:

`type`	Тип функции. См. таблицу ниже.
`kindFunc`	В зависимости от значения поля type, может содержать значения из разных наборов констант. Подробнее см. таблицы ниже
`method`	Метод расчета: 0 – БПФ, 1 - ДПФ
`numPoints`	Число точек, по которым вычисляется БПФ: 32…1048576
`nBlocks`	Число порций усреднения: 1…(длина сигнала/numPoints)
`ofsNextBlock`	Смещение порций относительно друг друга: 1, numPoints/4, numPoints/2, numPoints*3/4, numPoints
`typeWindow`	Тип весовой функции. См. таблицу ниже.
`typeMagnitude`	Тип значений. См. таблицу ниже.
`isMO`	Центрирование: 1 – включено, 0 – выключено
`isFill0`	Дополнение нулями: 1 – дополнять, 0 – нет
`fMaxVal`	Максимальные значения: 1 – максимальные, 0 – усредненные
`fLog`	Логарифмирование: 1 – результат в дБ, 0 – нет
`log_kind`	0 – 20*logX, 1 – 10*logX
`log_fOpZn`	Использовать опорное значение: 1 – использовать, 0 - нет
`log_OpZn`	Опорное значение
`fPrSpec`	Выполнить преобразование спектра
`prs_kind`	Вид преобразования: 0 – 1, 1 – $1/\omega$, 2 – $1/\omega^2$, 3 – $2\sqrt{2}/\omega^2$, 4 – $1*\omega$, 5 – $1*\omega^2$
`prs_loFreq`	Нижняя частота
`prs_s2n`	Отношение сигнал/шум
`prs_fCorr`	Использовать функцию-корректор
`prs_typeCorr`	Тип функции: 0 – пользовательская (задана в prs_strCorr), 1 – функция A, 2 – функция B, 3 – функция C
`prs_strCorr`	Функция-корректор (строка вида "x1 y1 x2 y2 x3 y3...")
`f3D`	Флаг трехмерного представления результатов: 1 – результат – трехмерный спектр, 0 - нет
`fSwapXZ`	Время по оси X: 1– по оси X – время, по оси Z – частота; 0 – по оси X – частота, по оси Z – время (для 3D)

Значения `type`

Константа	Значение	Описание
AUTOSPECTR	0	автоспектр
CROSS	20	взаимный спектр
COHEREN	30	функции когерентности
TRANS	40	передаточных функций

COMPLEX	50	комплексный спектр

Значения `typeWindow`

Константа	Значение	Описание
SINGLEWIN	1	прямоугольная функция
TRIANGLEWIN	2	треугольная функция
HANNINGWIN	3	функция Хэннинга
BLACKMANWIN	4	функция Блэкмана
FLATTOP	5	Flat-Top

Значения `typeMagnitude`

Константа	Значение	Описание
MEAD	1	эффективные
PEAK	2	амплитудные значения

АвтоСпектр

Быстрый вызов:

```
procedure RunFFT(const Src : OleVariant; var Dst, Dst2, Opt,
Err : OleVariant)
```

Настройки:

`type`	0 (AUTOSPECTR)

`kindFunc` может принимать следующие значения:

Константа	Знач.	Описание
SPM	1	спектр плотности мощности
SM	2	спектр мощности
SPP	3	спектр плотности энергии
SMAG	4	спектр амплитудный
SRI	5	комплексный спектр в виде реальной и мнимой части
SMF	6	комплексный спектр в виде модуля и фазы

Октавный спектр

Быстрый вызов: нет

Настройки:

`type`	0 (AUTOSPECTR)
`fFlt`	Метод расчета спектра: 1 – полосовые фильтры, 0 – БПФ
`fQual`	1 – использовать фильтры высокой точности, 0 – простые

`kindFunc` может принимать следующие значения:

Константа	Знач.	Описание
Oktav1	10	октавный спектр
Oktav3	11	третьоктавный спектр
Oktav12	12	1/12-октавный спектр

Oktav24	13	1/24-октавный спектр

Комплексный спектр

Быстрый вызов:

```
procedure RunComplexFFT(const Real, Imag : OleVariant; var
Dst, Dst2, Opt, Err : OleVariant)
```

Настройки:

`type`	50 (COMPLEX), `kindFunc` игнорируется

Взаимный спектр

Быстрый вызов:

```
procedure RunCrossFFT(const Src, Src2 : OleVariant; var Dst,
Dst2, Opt, Err : OleVariant)
```

Настройки:

`type`	20 (CROSS)

`kindFunc` может принимать следующие значения:

Константа	Знач.	Описание
CrSPM	21	спектр плотности мощности
CrRI	22	взаимный спектр в виде действ. и мнимой части
CrMF	23	взаимный спектр в виде модуля и фазы

Функция когерентности

Быстрый вызов:

```
procedure RunCoher(const Src, Src2 : OleVariant; var Dst, Opt,
Err : OleVariant)
```

Настройки:

`type`	30 (COHEREN)

`kindFunc` может принимать следующие значения:

Константа	Значение	Описание
COHERF	31	функция когерентности
COP	32	когерентная выходная мощность
S_N	33	отношение сигнала к шуму
NOTCOP	34	некогерентная выходная мощность
NOTCHR	35	функция некогерентности

Передаточная функция

Быстрый вызов:

```
procedure RunFuncTransfer(const Src, Src2 : OleVariant; var
Dst, Dst2, Opt, Err : OleVariant)
```

Настройки:

`type`	40 (TRANS)

`kindFunc` может принимать следующие значения:

Константа	Значение	Описание
H1	41	передаточная функция H1
H2	42	передаточная функция H2

Преобразование спектра

Быстрый вызов: нет

Настройки:

`kind`	Вид преобразования: 0 – 1, 1 – $1/\omega$, 1 – $1/\omega^2$, 3 – $2\sqrt{2}/\omega^2$, 4 – $1*\omega$, 5 – $1*\omega^2$
`loFreq`	Нижняя частота
`signal2noise`	Отношение сигнал/шум
`useCorrector`	Использовать функцию-корректор
`strCorrector`	Функция-корректор (строка вида "x1 y1 x2 y2 x3 y3...")
`typeCorr`	Тип функции: 0 – пользовательская (в strCorrector), 1 – функция A, 2 – функция B, 3 – функция C

Алгоритмы фильтрации

Рекурсивная фильтрация

Быстрый вызов:

```
procedure RunIIRFiltering(const Src : OleVariant; var Dst,
Opt, Err : OleVariant)
```

Настройки:

iType	Тип аппроксимации. См. таблицу ниже.
iKind	Вид фильтра. См. таблицу ниже.
nOrder	Количество двухполюсников (порядок): 1…20
nRipple	Неравномерность (процент) в полосе пропускания: 1…5
fsr	Частота среза (для ФНЧ, ФВЧ)
fn	Частота среза нижняя (для полосового)
fv	Частота среза верхняя (для полосового)
fs	Частота опроса
HO	Коэффициент фильтра

Значения iKind

Константа	Значение	Описание
LowPass	1	ФНЧ
BandPass	2	полосовой фильтр
HighPass	3	ФВЧ

Значения iType

Константа	Значение	Описание
Butterworth	1	Фильтр Баттерворта
Chebyshev	2	Фильтр Чебышева
Elliptic	3	эллиптический фильтр

Нерекурсивная фильтрация

Быстрый вызов:

```
procedure RunFIRFiltering(const Src : OleVariant; var Dst,
Opt, Err : OleVariant)
```

Настройки:

iType	Тип аппроксимации (с использованием рядов Фурье). Игнорируется
iKind	Вид фильтра. См. таблицу ниже.
iTypeWin	Тип весового окна. См. таблицу ниже.
nOrder	Количество коэффициентов (порядок), нечетное число: 1…1001
fsr	Частота среза (для ФНЧ, ФВЧ)
fn	Частота среза нижняя (для полосового и режекторного)
fv	Частота среза верхняя (для полосового и режекторного)

`fs`	Частота опроса

Значения `iKind`

Константа	Значение	Описание
LowPass	1	ФНЧ
BandPass	2	полосовой фильтр
HighPass	3	ФВЧ
BandStop	4	режекторный

Значения `iTypeWin`

Константа	Значение	Описание
HANN	2	окно Хэнна
HAMMINGWIN	3	окно Хэмминга

Медианная фильтрация

Быстрый вызов: нет

Настройки:

`Type`	Тип фильтра: 0 – дискретный, 1 – аналоговый
`nPoints`	Количество точек
`Level`	Порог (только для аналогового фильтра)
`LevelLow`	Нижний уровень (для дискретного фильтра)
`LevelHi`	Верхний уровень (для дискретного фильтра)
`bAuto`	Автоматическое определение уровней (для дискр.)

Действия над сигналами

Интегрирование (Первообразная)

Быстрый вызов:

```
procedure RunIntegral(const Src : OleVariant; var Dst, method,
numpointsAverg, typeRezult, Err : OleVariant)
```

Настройки:

`method`	Метод интегрирования. См. таблицу ниже.
`typeRezult`	Центрирование, 1 – включено, 0 – выключено
`numpointsAverg`	Число точек осреднения (только для RC)
`flagDelPerProcess`	Подавление переходного процесса: 1– вкл., 0 – выкл. (только для вибро)
`npointsPerProcess`	Длина переходного процесса (только для вибро)
`fsr`	Частота среза фильтрации (только для вибро)

Константа	Значение	Описание
AILER_INT	1	метод Эйлера
HANNING_INT	2	метод Хэннинга
RC_INT	3	метод RC-цепочек
VIBRO_INT	4	виброинтегрирование

Дифференцирование

Быстрый вызов:

```
procedure RunDiff(const Src : OleVariant; var Dst, method, Err
: OleVariant)
```

Настройки: `method` может принимать следующие значения:

Константа	Значение	Описание
THREE_POINTS	3	трехточечный метод
FIVE_POINTS	5	пятиточечный метод

Нормирование

Быстрый вызов: нет

Настройки:

`HiFront`	Верхняя граница
`LoFront`	Нижняя граница
`EnaShift`	Сдвиг значений сигнала относительно 0: 1 – разрешить (меняются статистические характеристики: МО, дисперсия и т.п.), 0 - запретить

Центрирование

Быстрый вызов: нет *Настройки:* нет

Арифметические операции

Быстрый вызов: нет

Настройки:

`kind`	Вид операции. Может принимать значения, приведенные в таблице.
`const`	Константа (для операций с одним сигналом)

Константа	Значение	Описание
CONST_PLUS	0	прибавление константы const
CONST_MINUS	1	вычитание константы const
CONST_MULTI	2	умножение на константу const
CONST_DIV	3	деление на константу const
BUF_PLUS	4	сложение значений двух сигналов
BUF_MINUS	5	вычитание значений второго сигнала из значений первого
BUF_MULTI	6	перемножение значений двух сигналов
BUF_DIV	7	деление значений первого сигнала на значения второго

Логарифмирование

Быстрый вызов: нет

Настройки:

`kind`	20logX (0) или 10logX (1)
`useOpZn`	Использовать опорное значение (1), иначе (0) - максимум
`OpZn`	Опорное значение

Передискретизация

Быстрый вызов:

```
procedure RunResampling(const Src : OleVariant; var Dst :
OleVariant; Freq, Method, FltType : OleVariant; var Err : OleVariant)
```

Настройки:

`freq`	Новая частота дискретизации
`kind`	Виды интерполяции. См. таблицу ниже.
`type`	Тип фильтрации. См. таблицу ниже.
`srcdt`	Сохранять исходный тип данных

Значения `kind`

Константа	Значение	Описание
NOINT	0	Интерполяция отсутствует
LINEINT	1	Линейная интерполяция
PARABINT	2	Интерполяция полиномом второго порядка
SPLINE3INT	3	Кубические локальные сплайны

Значения `type`

Константа	Значение	Описание
NOFLT	0	Фильтрация отсутствует
IIRFLT	1	Рекурсивная фильтрация
FIRFLT	2	Нерекурсивная фильтрация

Преобразование Гильберта

Быстрый вызов: нет

Настройки:

`nPoints`	Число точек, по которым вычисляется БПФ: 32…1048576
`nBlocks`	Число порций усреднения: 1…(длина сигнала/nPoints)
`isMO`	Центрирование: 1 – включено, 0 – выключено

Огибающая

Быстрый вызов: нет

Настройки:

`kind`	Метод: 0 - пик-детектор, 1 – преобразование Гильберта
`coef`	Коэффициент (K) для метода пик-детектора

Если выбран метод преобразования Гильберта, к этому алгоритму также применяются настройки преобразования Гильберта (см. выше).

Исследование сигналов

Вероятностные характеристики

Элементы результирующего сигнала (Dst) содержат значения вероятностных характеристик исходного сигнала.

Константа	Смещение	Описание
IDX_MO	0	Математическое ожидание
IDX_D	1	Дисперсия
IDX_SIG	2	Среднеквадр.отклон.
IDX_A3	3	Асиметрия
IDX_A4	4	Эксцесс
IDX_MAG	5	Амплитуда

Таким образом, чтобы получить, например, дисперсию сигнала, следует вызвать метод Dst.GetY(1) после выполнения алгоритма.

Быстрый вызов: нет *Настройки:* нет

Плотность вероятности

Быстрый вызов:

```
procedure RunPRV(const Src : OleVariant; var Dst, npoints,
type, Err : OleVariant)
```

Настройки:

npoints	Число точек вычисляемой характеристики
type	Метод расчета и представления плотности распределения вероятности. См. таблицу ниже.

Константа	Знач.	Описание
PARZEN	1	ПРВ, метод ядерных оценок
HIST	2	ПРВ, расчет в виде гистограммы
PARZNORM	8	вероятность попадания, метод ядерных оценок
HISTNORM	4	вероятность попадания, расчет в виде гистограммы

Функция автокорреляции

Быстрый вызов:

```
procedure RunAutoCorel(const Src : OleVariant; var Dst,
npoints, eps, Err : OleVariant)
```

Настройки:

npoints	Количество точек для построения корреляционной функции
eps	Возвращается оценка статистической погрешности

Функция взаимной корреляции

Быстрый вызов:

```
procedure RunCrossCorel(const Src, Src2 : OleVariant; var Dst,
npoints, eps, Err : OleVariant)
```

Настройки:

npoints	Количество точек для построения корреляционной функции
eps	Возвращается оценка статистической погрешности

Параметрический график

Быстрый вызов: нет

Настройки:

type	0 – параметрический график, 1 – полярный, 2 – параметрический для сигналов с одинаковой дискретизацией (берутся значения с одинаковыми индексами)

Часть 5. Встроенный редактор сценариев

Для написания собственных быстродействующих алгоритмов обработки, для обработки значительных объемов данных, и создания приложений, опирающихся на WinПОС, но требующих дополнительных настроек или способных формировать специализированные отчеты, лучше всего подходит Borland Delphi. Также можно использовать Borland C++ Builder, Microsoft Visual C++, Visual Basic или FoxPro.

Однако для написания небольших сценариев автоматизации работы WinПОС или несложных алгоритмов больше подходит Visual Basic Script. VBScript входит в состав поставки Microsoft Windows, не требует отдельного компилятора, а удобная среда редактирования и отладки сценариев включена в состав WinПОС.

Редактор сценариев (Рис. 11.1) открывается через меню **Сценарий→Редактор сценариев…**

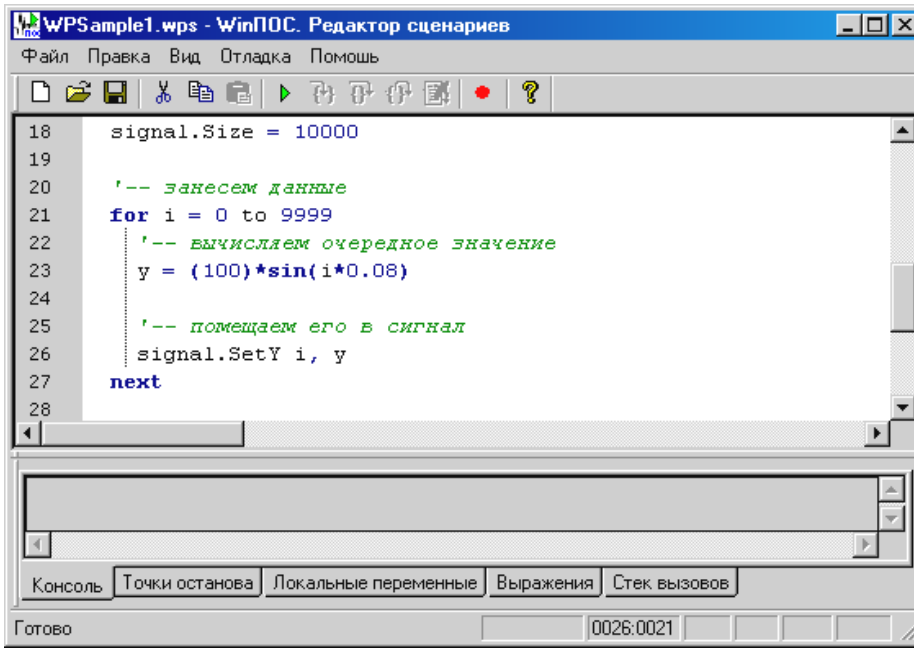


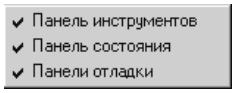
Рис. 5.1. Окно подготовки сценариев обработки

Редактор сценариев – это текстовый редактор с синтаксической подсветкой и стандартным набором инструментов, к которым можно обратиться с помощью

меню, панели инструментов и горячих клавиш. Редактор также предоставляет все необходимые инструменты для выполнения сценариев под отладкой: точки останова и выполнение по шагам, просмотр локальных переменных и стека вызовов, вычисление выражений.

Отличительные особенности встроенного редактора сценариев:

- кнопки управления отладкой на панели инструментов,
- синтаксическая подсветка,
- контроль парности скобок и визуализация отступов,
- линейка нумерации строк с полосой точек останова (расположена слева),
- панели отладки (под областью редактирования).



С помощью меню **Вид** можно отключать и подключать визуальные элементы редактора.

Синтаксическая подсветка позволяет сократить количество ошибок при наборе текста сценария и облегчает восприятие текста:

синим жирным шрифтом выделяются зарезервированные слова VBScript,

синим – символы,

курсивом – строковые константы,

зеленым курсивом – комментарии, а

идентификаторы – обычным черным шрифтом.

Парные скобки при наборе подсвечиваются зеленым цветом:

```
y = 100*sin(i*0.08)
```

Символ табуляции отмечается вертикальной полосой, что позволяет более наглядно изображать вложенность циклов и условий.

Режим редактирования

Команды режима редактирования сценария:

Панель инстр.	Меню	Сочетание клавиш	Описание
	Файл→Новый сценарий	Ctrl+Shift+N	Очистить окно для нового сценария
	Файл→Открыть сценарий…	Ctrl+Shift+O	Открыть файл для редактирования
	Файл→Сохранить сценарий	Ctrl+Shift+S	Сохранить редактируемый файл
	Файл→Сохранить сценарий как…		Сохранить сценарий под новым именем
	Файл→Закрыть редактор	Alt+F4	Закрыть окно редактора
	Правка→Отменить изменения	Ctrl+Z	Отменить последние действия
	Правка→Вырезать	Ctrl+X	Вырезать выделенный фрагмент в буфер
	Правка→Копировать	Ctrl+C	Копировать выделенный фрагмент в буфер
	Правка→Вставить	Shift+V	Вставить в позицию курсора текст из буфера
	Правка→Выделить все	Shift+A	Выделить весь текст
	Правка→Найти…		Поиск заданной строки
	Правка→Заменить…		Поиск и замена строки
	Правка→Перейти к строке…		Переход к строку с заданным номером
	Помощь→Содержание…	F1	Справка по объектам WinПОС.

Диалог *Перейти к строке* (Рис. 5.2, меню **Правка→Перейти к строке…**) помогает при навигации в пространном сценарии. Номер строки можно увидеть как на линейке слева, так и в панели состояния (позиция курсора, строка - первое число в паре “llll:cccc”).

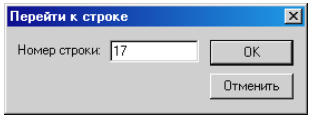


Рис. 5.2. Перейти к строке

Диалог *Поиск* (Рис. 5.3, меню **Правка→Найти…**) позволяет найти все вхождения заданной в поле *Искать текст* строки, с учетом регистра (**Учитывать регистр**) и положения в окружающем тексте (**Слова целиком** и **Регулярные выражения**). Кнопки **Далее** и **Назад**

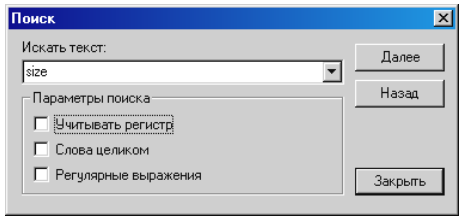


Рис. 5.3. Диалог поиска

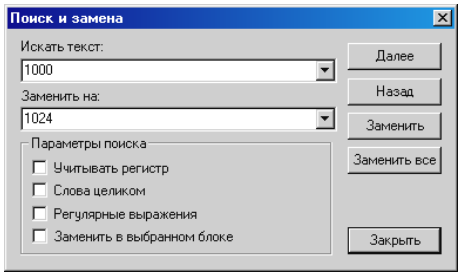


Рис. 5.4. Диалог поиска и замены

задают направления поиска относительно текущей позиции курсора. Диалог *Поиск и замена* (Рис. 5.4, меню **Правка→Заменить…**) повторяет диалог *Поиск*, но позволяет заменить найденную строку на указанную в поле *Заменить на*. Кнопка **Заменить** выполняет однократную замену, **Заменить все** – автоматически заменяет все найденные участки текста. Флажок **Заменить в выбранном блоке** позволяет ограничить область изменения текста.

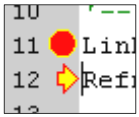
Режим отладки

Команды режима выполнения и отладки сценария:

Панель инстр.	Меню	Сочетание клавиш	Описание
	Отладка→Начать/Продолжить исполнение	Ctrl+F10	Переход в режим выполнения сценария
	Отладка→Вкл. /Выкл. точку останова	F9	Установка брекпоинта
	Отладка→Шаг внутрь	F11	Выполнение по шагам со заходом в процедуры
	Отладка→Шаг вперед	F10	Выполнение по шагам
	Отладка→Шаг наружу	Ctrl+F11	Выход из процедуры
	Отладка→Остановить отладку	Alt+F10	Прекращение выполнения сценария
	Отладка→Останов на старте		Режим остановки после запуска

ⓘ В режиме отладки становится недоступным редактирование текста сценария.

Переход в режим отладки происходит по кнопке . Если не были установлены точки останова (, брекпоинты), и не была включена опция остановки после запуска (**Отладка→Останов на старте**), сценарий будет выполнен полностью.



Для установки точки останова на выбранной строке нужно нажать кнопку или <F9>. Брекпоинт будет поставлен на текущей строке, в поле линейки появится соответствующий значок – красная точка.

Для продолжения выполнения сценария можно снова нажать кнопку . Можно также продолжить выполнение сценария по шагам, с помощью кнопок , и . Позиция строки, которая будет выполняться следующей, отмечается желтой стрелкой в поле линейки.

Отладочные панели

Отладочные панели дают полную картину состояния выполняемого сценария в каждый момент времени (позволяют следить за ходом выполнения сценария, изменением содержимого переменных и т.п.).

Консоль

На *Консоль* (рис. 5.5) направляется отладочная печать (функции DebugPrint() и DebugPrintLn()).

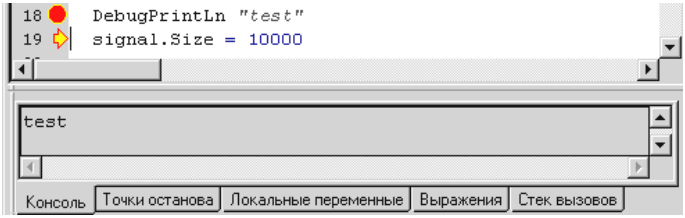


Рис. 5.5. Консоль

Точки останова

Закладка *точек останова* (рис. 5.6) показывает список брекпоинтов с номерами строк, состоянием и счетчиком количества проходов через строку. Состояние точки (*Активна* или *Блокирована* – изображается серой) можно изменить через контекстное меню панели, удалить точку можно не только по <F9>, но и через контекстное меню.

Строка	Состояние	Число проходов	
19	Активна	0	
24	Активна	0	

Консоль | Точки останова | Локальные переменные | Выражения | Стек вызовов

Рис. 5.6. Закладка точек останова

Локальные переменные

Значения и типы *локальных переменных* можно просмотреть на одноименной закладке (рис. 5.7).

Имя	Тип	Значение	Описание
signal	Object	{...}	
i	Integer	18	
y	Double	97.7864602435...	

Консоль | Точки останова | Локальные переменные | Выражения | Стек вызовов

Рис. 5.7. Закладка локальных переменных

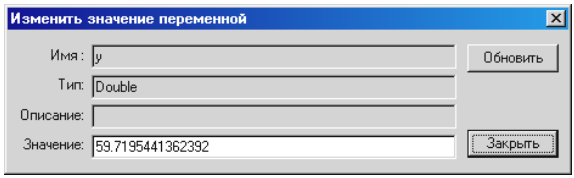


Рис. 5.8. Диалог просмотра и изменения переменной

Двойной щелчок мышки на строке переменной открывает диалог 5.8, в котором можно не только увидеть значение переменной, но и изменить его (поле *Значение,* кнопка **Обновить**).

Выражения

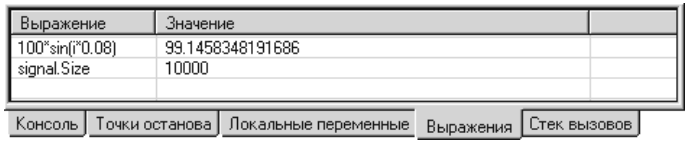


Рис. 5.9. Закладка вычисления выражений

Закладка *выражения* (рис. 5.9) дает возможность вычисления любых выражений, записанных в синтаксисе VBScript.

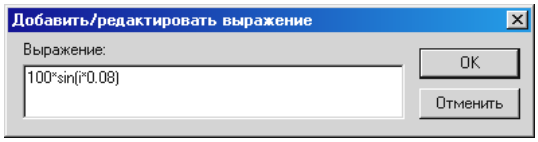


Рис. 5.10. Диалог добавления выражений

Новое выражение можно добавить, удалить и изменить с помощью контекстного меню, откуда вызывается диалог редактирования выражений (рис. 5.10).

Выделив строку сценария, ее также можно скопировать на закладку *Выражения* через контекстное меню редактора.

Стек вызовов

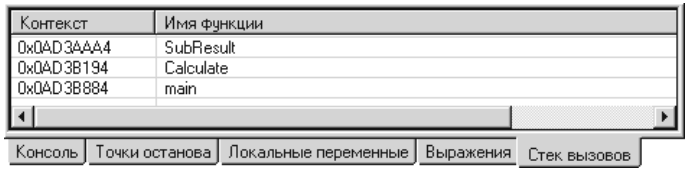


Рис. 5.11. Стек вызовов

Последняя закладка, *Стек вызовов* (рис. 5.11), помогает ориентироваться при отладке сценария, состоящего из большого количества процедур, и просто незаменима при написании кода с рекурсивными вызовами. Верхней в стеке является текущая процедура.

Приложение. Примеры

При установке **WinПОС** в своем рабочем каталоге создает подкаталог Samples, в котором можно найти примеры реализации сценариев на VBScript и примеры создания приложений и подключаемых модулей на Delphi.

`Samples \`	
`VBS`	- примеры сценариев на Visual Basic Script
`Delphi`	- примеры приложений на Delphi
`DelphiPlugIn`	- пример подключаемого модуля на Delphi
`DelphiCommon`	- служебные модули (Delphi)

В каталогах VBS и Delphi находится по 6 примеров программ.

Номер примера	Описание	Используемые возможности
1	Генератор сигнала. Создается сигнал и заполняется значениями, вычисляемыми с помощью функции.	✓ создание сигнала ✓ добавление сигнала в дерево WinПОС ✓ доступ к значениям сигнала
2	Отображение сигнала в графике. Создается и заполняется сигнал, готовый сигнал помещается в график.	✓ создание сигнала с неравномерной осью X ✓ доступ к графической подсистеме WinПОС ✓ отображение сигнала на графике
3	Загрузка произвольного УСМЛ файла, обработка сигнала (АвтоСпектр).	✓ загрузка пакетного файла ✓ вызов процедуры RunFFT()
4	Вызов диалога выбора файла, загрузка двоичного файла данных, обработка сигнала (вызов алгоритма WinПОС по имени, задание опций), печать.	✓ работа с диалогом открытия файла ✓ загрузка двоичного файла ✓ выполнение оператора через Exec() ✓ вызов метода PrintPreview()
5	Загрузка УСМЛ файла, вычисление АвтоСпектра. Доп. обработка результата: по спектру находим три значения частоты, в которых величины амплитуд максимальны, и	✓ дополнительная обработка результатов выполнения алгоритмов WinПОС ✓ создание страницы с двумя графиками (исходный сигнал и результат)

	печатаем эти значения	
6	Загрузка УСМЛ файла, обработка сигнала: передискретизация, фильтрация, автоспектр.	✓ последовательная выполнение цепочки алгоритмов ✓ отображение нескольких параметров в одном графике

Для удобства восприятия примеры на Delphi скомпонованы в отдельные, включаемые директивой «$I», файлы, каждый из которых содержит только тело процедуры. Для компиляции этих файлов в качестве отдельных модулей-«юнитов» необходимо дополнить их стандартным обрамлением: unit, interface, implementation,…

В каталоге DelphiPlugIn расположен пример подключаемого модуля (plug-in), добавляющего на панель инструментов WinПОС кнопку, по которой запускается один из примеров из каталога Delphi.

В каталоге DelphiCommon расположен автоматически созданный по библиотеке типов (TLB) файл с описанием интерфейсов **WinПОС** (WinPOS_ole_TLB.pas) и файл PosBase.pas, содержащий процедуры упрощенного доступа к алгоритмам **WinПОС** и описания констант.

Ниже приведен пример программы, которая создает тестовый сигнал, выполняет расчет спектра и отображает результат в графическом виде.

```
// Пример 1.
// Создание сигнала, вызов алгоритма, отрисовка в графиках.
program Sample1;

uses
  Forms,
  Winpos_ole_TLB in '../DelphiCommon/Winpos_ole_TLB.pas',
  PosBase in '../DelphiCommon/PosBase.pas';

var signal: IWPSignal;
    y : Double;
    dst1, dst2, Opt, Err : OleVariant;
    api : IWPGraphs;
    i, hPage, hGraph, hGraphFFT, hAxis, hAxisFFT : Integer;

begin
  Application.Initialize;

  with WINPOS do
  begin

  // 1) создаем сигнал со значениями типа Double
  signal:= CreateSignal(VT_R8) as IWPSignal;
```

```
  if Assigned(signal) then   // если сигнал создан
    begin

      // помещаем сигнал в дерево
      Link('/Signals/generator', 'sinus', signal as IDispatch);
      Refresh();

      signal.size:= 10000; // зададим длину сигнала

      for i:= 0 to 9999 do // занесем данные
      begin
        y:= (100)*sin(i*0.08); // вычисляем очередное значение
        signal.SetY(i, y); // помещаем его в сигнал
      end;

      // 2) применим к сигналу оператор "Автоспектр"
      RunFFT(signal, dst1, dst2, Opt, Err);

      // положим результат в дерево WinПОС
      Link('/Signals/Result', 'spectrum', dst1);

      // 3) отобразим исходный и результирующий сигналы
      // получаем доступ к графической подсистеме WinПОС
      api:= GraphAPI as IWPGraphs;

      // создаем новую страницу для графиков
      hPage:= api.CreatePage;

      // страница всегда создается с областью для рисования
      hGraph:= api.GetGraph(hPage,0);

      // создаем дополнительный график для спектра
      hGraphFFT:= api.CreateGraph(hPage);

      // получаем ось Y
      hAxis:= api.GetYAxis(hGraph,0);

      // создаем  новую линию в графике
      api.CreateLine(hGraph, hAxis, signal.Instance);

      // получаем ось Y во втором графике
      hAxisFFT:= api.GetYAxis(hGraphFFT,0);

      // создаем новую линию в графике спектра
      api.CreateLine(hGraphFFT, hAxisFFT, dst1.Instance);

      // нормализуем графики
      api.NormalizeGraph(hGraph);
      api.NormalizeGraph(hGraphFFT);

      Refresh;
    end;
  end; // with
end.
```

Следующий пример - программа для формирования нестандартного варианта экспресс-отчета. В цикле обрабатывается файл УСМЛ или МЕРА, параметры которого раскладываются по страницам 3x3 графика, устанавливается новый масштаб по оси Y, позволяющий оценить исходные уровни параметров заданного файла.

```
// Пример 2.
// Программа для формирования экспресс-отчета
program Express;

uses
  Forms,
  Winpos_ole_TLB in '../DelphiCommon/Winpos_ole_TLB.pas',
  PosBase in '../DelphiCommon/PosBase.pas';

var FileName  : string;
    signal    : IWPSignal;
    usml      : IWPUSML;
    api       : IWPGraphs;
    hPage, hGraph, hAxis, hLine : Integer;
    i, j, nGr, nPg : Integer;
    range, min, max : Double;

const nVer : Integer = 3;
const nHor : Integer = 3;

begin
  Application.Initialize;

  // WINPOS определяется и инициализируется в модуле POSBase.pas
  with WINPOS do
  begin
    // открываем УСМЛ с помощью стандартного диалога WinПОС
    FileName:= USMLDialog();

    if fileName<>'' then // если файл был выбран
    begin
      // получаем доступ к графической подсистеме Winpos
      api:= GraphAPI as IWPGraphs;

      usml:= LoadUsml(fileName) as IWPUSML; //загружаем УСМЛ

      // так мы сразу попадем в создание новой страницы (см. ниже)
      nGr:= nHor*nVer;
      nPg:= 0;
      for i:=0 to usml.ParamCount-1 do
      begin
        // теперь можно взять сигнал по порядковому номеру в файле
        signal:= usml.Parameter(i) as IWPSignal;
        if (nGr = nHor*nVer) then
        begin
```

```
          // создаем новую страницу для графиков
          hPage:= api.CreatePage;

          // устанавливаем вид 3x3
          api.SetPageDim(hPage, PAGE_DM_TABLE, nVer, nHor);

          for j:=2 to nHor*nVer do
            api.CreateGraph(hPage);

          Inc(nPg);
          nGr:= 0;
        end;

        hGraph:= api.GetGraph(hPage, nGr);
        // получаем ось Y
        hAxis:= api.GetYAxis(hGraph,0);

        // создаем  новую линию в графике
        api.CreateLine(hGraph, hAxis, signal.Instance);

        // нормализуем график
        api.NormalizeGraph(hGraph);

         range:= signal.MaxY - signal.MinY;
         max:= signal.MaxY + range*10;
         min:= signal.MinY - range*10;
         api.SetYAxisMinMax(hAxis, min, max);

         Inc(nGr);
      end;
    end;

    Refresh;
  end;
end.
```

Указатель методов