


# ОПИСАНИЕ ФУНКЦИЙ И ВОЗМОЖНОСТЕЙ ПРЕДОСТАВЛЯЕМЫХ СКРИПТАМИ ДЛЯ АНАЛИЗА СИГНАЛОВ В ПРОГРАММЕ USB ОСЦИЛЛОГРАФ.

## 1 Использование скриптов анализатора.

Использование скриптов анализатора делает возможным осуществление автоматического анализа сигналов в файлах осциллограмм по созданному пользователем алгоритму. Результатами этого анализа является маркирование и комментирование характерных точек осциллограммы, получение текстовых отчетов и графиков. Данная функция программы базируется на использовании скриптовых языков **JScript** и **VBScript** с интеграцией в них функций, констант и объектов, реализованных в программе осциллографа. Текстовые и графические результаты анализа отображаются в окне отчета. Скриптовые файлы анализатора представляют собой обычные текстовые файлы с расширением **"\*.ajs"**, содержащие JScript, или **"\*.abs"**, содержащие VBScript. Для расширения функциональности интерфейсной части скриптов в программе предусмотрено использование **HTML** файлов. Файлы с расширением **"\*.asc"** - кодированный файл. Этот тип файлов создается путем кодирования исходных файлов (**Jscript**, **VBScript** или **HTML**) с целью защиты их содержимого. В случае использования **HTML** файлов, алгоритм анализа данных интегрируется в тело **HTML** в виде **JScript** и **VBScript** скриптовых блоков. При этом основное тело **HTML** описывает элементы пользовательского интерфейса и взаимодействие их с интегрированным скриптом. В случае использования **HTML** все обращения к элементам скриптового ядра программы осуществляются через объект **"Host"**. Для обеспечения нормальной функциональности скриптов анализатора на базе **HTML** файлов необходимо учитывать, что выполнение скрипта в данном случае происходит в одном потоке с основным кодом программы. Поэтому при запуске длительных процедур, анализирующих массив данных, в основном цикле процедуры необходимо вызывать функцию **CanContinue()**. Функция препятствует зависанию программы при обработке потока сообщений.

Для загрузки скриптовых файлов анализатора используется опция меню **"Анализ/Загрузить скрипт"** или кнопка  панели инструментов. Для запуска - опция меню **"Анализ/Выполнить скрипт"** или кнопка  панели инструментов. Выполнение скрипта прерывается при помощи опции меню **"Анализ/Прервать выполнение"** или кнопка  панели инструментов. Переключение между режимами отображения осциллограммы или отчета осуществляется опциями меню **"Отображать/Перейти к просмотру осциллограмм"** или **"Отображать/Перейти к просмотру отчета"**, а также кнопками  или  соответственно. Удаление полученного отчета и закрытие окна отчета осуществляется функцией меню **"Анализ/Удалить отчет"** или кнопкой  панели инструментов. Если запущенный скрипт создает один или несколько графиков, переключение между ними осуществляется при помощи закладок, расположенных в нижней - левой части окна отчета. Данный документ не включает в себя описание используемых скриптовых языков. Ниже приведено описание интегрируемых в скрипт элементов.

## 2 Использование панели анализатора.

Панель анализатора позволяет пользователю создавать свой инструментарий для проведения специфических измерений параметров записанного программой сигнала. Файл панели анализатора представляет собой **HTML** файл с интегрированным в него алгоритмом анализа данных. По аналогии с функциональностью файлов анализатора на базе **HTML** (см. выше). Программа поддерживает три типа файлов панели анализатора. Файлы с расширением **“.apn”** и **“.html”** идентичны, файлы **“.apc”** – кодированный файл. Функциональность файлов панели анализатора отличается от скриптов анализатора. Как и скрипты анализатора, скрипт панели может осуществлять доступ к массиву данных, для анализа записанного сигнала. Однако панель не создает текстовых или графических отчетов. Для реализации части функций панели в скриптовое ядро программы добавлена отдельная группа функций. Благодаря чему панель может участвовать в отрисовке экрана, получать и изменять параметры отображения сигнала, привязывать вызов отдельных скриптовых функции к ряду событий предусмотренных программой. Информация по функциям и событиям представлена ниже. Загрузка файла панели осуществляется опцией меню **"Анализ/Загрузить панель"** или кнопкой  панели инструментов.

### 3 Использование потоков данных.

Потоки предназначены для упрощения и ускорения процессов обработки данных, построенных на базе скриптов. Потоки классифицируются по направлению на входные - предоставляющие входные для обработки данные и выходные - накапливающие результаты обработки данных. На текущий момент программа поддерживает два подтипа входных и два подтипа выходных потоков. С момента создания потока функцией `CreateStream` (см. ниже описание функций) он жестко привязывается к одному из каналов в массиве данных осциллограммы. В зависимости от контекста, в котором происходит вызов этой функции, определяется подтип потока. Все потоки созданные внутри функции трансформации данных экранного буфера (функции зарегистрированной как обработчик события "`Transform`") получают или возвращают данные непосредственно из или в экранный буфер. Прочие потоки созданные вне контекста функции трансформации получают данные из основного потока данных или копируют результат во вторичный буфер экрана. Данные вторичного буфера замещают данные экранного буфера непосредственно перед отрисовкой экрана. Функционально два подтипа выходных потока отличаются тем, что данные первого, созданного в теле функции трансформации, участвуют в процессе обработки данных (вычисление разности каналов, расчете средних и пиковых значений сигнала). Поэтому размер блока данных этого подтипа увеличен в соответствующее количество раз текущему масштабу. Данные выходного потока второго подтипа не участвуют в указанных процессах обработки, а размер потока равен количеству точек экрана осциллограммы. Обновление данных второго подтипа выходного потока можно осуществлять в теле функций обработчиков событий "`Transform`" или "`BeforeDraw`", вызываемых всякий раз перед отрисовкой осциллограмм. Получение данных из входного потока происходит при непосредственном использовании объекта типа входной поток в выражениях обработки данных. Копирование очередной порции данных в выходной поток осуществляется при обращении к объекту типа выходной поток как к функции, основным параметром этого вызова является новое значение или объект источник данных. Каждое обращение к объекту входного или выходного потока перемещает текущее положение потока на величину считанных или установленных данных. Ниже приведены примеры на JavaScript иллюстрирующие процессы создания считывания и записи данных из или в потоки.

```
InStr = CreateStream(1,0);//создаем входной поток для канала 1  
OutStr = CreateStream(1,1);//создаем выходной поток для канала 1
```

```
while( cBuffer > 0 ){// повторять пока cBuffer > 0  
OutStr(1- exp(InStr));// уст. след. элемент потока OutStr как  $1-e^{\ln Str}$   
CBuffer--; } // уменьшить cBuffer на 1
```

Пример выше иллюстрирует последовательное преобразование блока данных канала 1, размером `cBuffer`, из входного потока `InStr` в выходной поток `OutStr` того же канала, по формуле  $y=1-e^x$ . Это типичный пример функции трансформации данных для обработчика события "`Transform`".

Следующие примеры иллюстрируют передачу блока данных между потоками.

*OutStr( InStr )* - заполнить выходной поток *OutStr*, данными входного потока *InStr*.

*OutStr( InStr, 200 )* - скопировать 200 элементов входного потока *InStr* в выходной поток *OutStr*.

*OutStr(InStr, 200, 100)* - скопировать 200 элементов входного потока *InStr* с усреднением в 100 элементов выходного потока *OutStr*.

*OutStr(InStr, 200, 300)* - скопировать 200 элементов входного потока *InStr* с интерполяцией в 300 элементов выходного потока *OutStr*.

В качестве источника данных в описанных выше функциях допускается использование массива данных. Пример ниже иллюстрирует эту возможность.

```
var aData = Array(-1, 2.5, 3.51, -12.15, 0.16 );//создаем массив с данными
```

```
OutStr(aData, aData.length, 30); /* копирует элементы массива aData в 30 элементов выходного потока OutStr с интерполяцией.*/
```

Для расширения функциональности, потоки снабжены базовыми функциями общими для входных и выходных потоков, и дополнительными функциями отличными для входных и выходных потоков. Описание этих функций представлено ниже.

## 4 Описание функций, констант и объектов, экспортируемых программой в скриптовое ядро анализатора.

Базовым объектом, представляющим основные свойства и функции обработки анализируемого файла осциллограмм, является объект **“Host”**. Функции этого объекта позволяют анализировать сигнал, выводить результаты этого анализа, а также создают другие объекты скриптового ядра программы, расширяющие возможности анализа, представления результатов и интерфейса пользователя.

### 4.1 Константы объекта **“Host”**.

**DataType** – Данная константа описывает тип данных в анализируемом файле. В текущей версии программы определены два типа данных:

**"DIG"** – данные записанные цифровым анализатором;

**"ANA"** – данные записанные аналоговым осциллографом.

**Channels** – Константа определяет количество каналов в анализируемом файле.

**Frequency** – Частота оцифровки данных в анализируемом файле.

**NumberOfSamples** – Определяет размер массива данных анализируемого файла.

**SelBegin, SelEnd** – Определяет индекс начального и конечного образцов выделенного участка осциллограммы в массиве данных анализируемого файла.

### 4.2 Основные функции объекта **“Host”**.

**GetChannelName( iChannel )** Функция возвращает имя канала для номера канала заданного числовым значением **iChannel** (\*1).

**HasChannellInversion( iChannel )** Функция возвращает значение **true** если опция инверсии канала **iChannel** включена (\*1).

**GetFullScale( iChannel )** Функция возвращает максимальное значение амплитуды для канала **iChannel** (\*1).

**GetChannelParam( iChannel, strParamName )** Функция возвращает значение параметра определенного именем **strParamName** для канала **iChannel** (\*1). Ниже представлены возможные значения для **strParamName** и соответствующий параметр, возвращаемый функцией:

**Visible** – возвращает **true** если сигнал отображается, или **false** иначе;

**Inversion** – аналогично функции **HasChannellInversion**;

**Average** – возвращает **true** если опция среднего значения активна, или **false** если выбрана опция отображения пиковых значений;

**Vdiv** – значение усиления мВ на деление;

**Offs** – вертикальное смещение для канала;

**Color** – выбранный для канала цвет (\*2).

**SetChannelParam( iChannel, strParamName, vParam )** Функция устанавливает значение **vParam** параметра определенного именем **strParamName** для канала **iChannel** (\*1). Возможные значения параметра **strParamName** указаны в описании ф. **GetChannelParam**. Для параметра усиления, определены следующие значения: 50, 100, 200, 500, 1000, 2000, 5000, 10000 мВ на деление.

**ValueAt( iChannel, iPos )** Функция возвращает значение напряжения для канала **iChannel** (\*1) с позицией в массиве данных осциллограммы **iPos**. Для аналогового сигнала значения возвращаются в вольтах. Для цифрового значения принимают величину 0 или 1.

**AveValueAt( iChannel, iPos, nPoints )** Функция возвращает среднее арифметическое значение напряжения для канала **iChannel** с позицией в массиве данных осциллограммы **iPos**, рассчитанное по **nPoints** количеству точек в диапазоне от **iPos - nPoints/2** до **iPos + nPoints/2**. Для аналогового сигнала значения возвращаются в вольтах. Для цифрового значения принимают величину 0 или 1 (\*1).

**ReportOut( strText )** Функция выводит текстовое сообщение **strText** в окно отчета. Возможно использование данной функции в файлах панели анализатора для вывода отладочных сообщений. В этом случае сообщения выводятся в окне редактора скриптовых файлов.

**SynchBy( iChannel, iType, Level, iFromPos, cSamples )** Функция осуществляет поиск **iType** - фронта сигнала с переходом через уровень - **Level** для канала - **iChannel** на участке массива данных с позиции - **iFromPos** на протяжении - **cSamples**.

Где: **iChannel** – номер канала (\*1), значение 0 – определяет поиск по любому каналу ( только для цифровых сигналов );

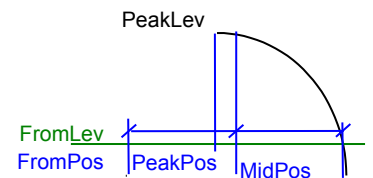
**iType** - искомый тип фронта, 0 - любой, 1 - передний, -1 - задний;

**Level** – искомый уровень в вольтах для аналогового сигнала или 0.

Если значение **cSamples** = 0 поиск осуществляется до конца массива данных. Если указанный фронт сигнала, удовлетворяющий входным параметрам, найден, функция возвращает его положение в массиве данных осциллограммы. Иначе возвращается -1.

**SynchByEx**( *iChannel*, *iType*, *Level*, *iFromPos*, *cSamples*, *dT*, *dV* ) Функция аналогична функции **SynchBy**, за исключением двух параметров *dT*, *dV* определяющих параметры селективности сигнала по времени и приращению напряжения соответственно. Параметр *dT* определяет интервал времени в сэмплах, где фронт должен оставаться неизменным. Значение *dV* определяет пороговое значение приращения напряжения на интервале времени заданным параметром *dT*. Параметры *dT* и *dV*, задаются как положительные числа не равные нулю.

**FindPeak**( *iChannel*, *iFromPos*, *cSamples*, *MinDelta*, *AveDepth*, *Mid* ) Функция осуществляет поиск положения пикового значения сигнала на канале *iChannel* (\*1), начиная с позиции в массиве данных *iFromPos* на интервале *cSamples*. Значение *MinDelta* определяет минимальное значения пика, а ее знак - условие поиска. Положительное значение определяет поиск максимума, отрицательное – минимума. Параметр *AveDepth* определяет количество точек арифметического усреднения данных при поиске пикового значения (1...512). При значении *Mid* = 0 функция возвращает положение **PeakPos**, пикового значения **PeakLev**, если *Mid* = 1 функция возвращает положение **MidPos** расположенное по центру от точек пересечения кривой сигнала с уровнем *FromLev*, одна из них *iFromPos* см. рисунок. Если искомый пик не найден функция возвращает "-1".



**CreateConfigure**( *strName* ) Функция создает и возвращает объект конфигурация с именем *strName*. Данное имя используется для сохранения параметров конфигурации в реестре.

**CreateGraphicView**( *strName* ) и **CreateGraphicViewEx**( *strName*, *vType* ) Функции создают и возвращает объект типа окно графика с именем *strName*. Параметр *vType* определяет тип графика.

**CreateGraphic**( *strName*, *strYname*, *vColour* ) Функция создает и возвращает объект типа график с именем *strName*, именем единицы измерения по координате 'Y' – *strYname*, и цветом *vColour* (\*2).

**ShowProgress**( *strMessage*, *nTotalSteps* ) Функция отображает сообщение *strMessage* и полосу прогресса в строке состояния основного окна программы. Значение *nTotalSteps* – определяет значение для 100% состояния полосы прогресса.

**HideProgress**() Удаляет сообщение и полосу прогресса из строки состояния основного окна.

**SetProgress**( *iPos* ) Устанавливает текущее положение полосы прогресса *iPos*, по отношению к *nTotalSteps* = 100%.

**GetCurLanguage( )** Возвращает идентификатор выбранного в программе языка интерфейса. Значения 0x409 – соответствует английскому США, а 0x419 – русскому.

**SetStatusText( strText )** Функция отображает текстовое сообщение **strText** в панели статуса программы. Пустое значение параметра **strText**, устанавливает значение по умолчанию типа “Готов...”.

**SetMarker( iPos, strComment )** Функция устанавливает маркер в позицию **iPos** в массиве данных с текстовым комментарием – **strComment**.

**SetVMarker( iChannel, MinValue, MaxValue, strComment )** Функция устанавливает маркер уровня/диапазона для канала **iChannel** (\*1). Диапазон определяется значениями напряжения **MinValue** и **MaxValue**. Если значения **MinValue** = **MaxValue** - функция определяет маркер уровня. При необходимости маркер может быть снабжен комментарием – **strComment**.

**CreateStream( iChannel, InOut)** Функция создает и возвращает объект типа поток данных канала **iChannel** (\*1). Параметр **InOut** - определяет направление потока, значение 0 – определяет входной поток, 1 – выходной.

#### 4.3 Функции объекта “Host” используемые файлами анализатора на базе HTML.

**CanContinue( )** Функция используется для контроля возможности продолжения цикла основного тела скрипта. Функция возвращает значение **true**, если дальнейшее выполнение цикла возможно или **false**, если его необходимо прервать. Использование функции обязательно при построении длительных циклов обработки данных.

**ClearResults( )** Функция удаляет текстовые и графические результаты анализа.

**Quit( )** Функция закрывает диалоговое окно анализатора на базе HTML скрипта.

#### 4.4 Функции объекта “Host” используемые файлами панели анализатора.

**SetEvent** ( **strName**, **Function** ) Функция устанавливает функцию **Function** -обработчик события **strName**. В программе определены следующие события:

"FileLoaded" - вызывается при загрузке нового файла;

"Information" - вызывается при нажатии кнопки «?» панели анализатора, если функция не определена, то при нажатии данной кнопки загружается файл «Помощь.htm», расположенный в той же папке что и файл скрипта;

"Markermoved" - вызывается при перемещении одного из измерительных маркеров, в качестве параметра функции передается индекс маркера изменившего свое положение;

"BeforeDraw" - вызывается перед отрисовкой осциллограммы, в качестве параметра функции передается объект типа **Draw**.

"AfterDraw" - вызывается после отрисовки осциллограммы, в качестве параметра функции передается объект типа **Draw**.

"Transform" - вызывается для осуществления трансформации данных экранного буфера, в качестве параметра функции передается размер этого буфера;

"ScrollScreen" - вызывается всякий раз, когда область отображения изменила свое положение;

"ChanParamChanged" – вызывается всякий раз, когда пользователь изменил параметры отображения канала, номер которого передается функции обработчику в качестве параметра;

**GetScreenPos**( ) Функция возвращает индекс первого отображаемого на экране сэмпла в массиве данных осциллограммы.

**SetScreenPos**( **iPos** ) Функция перемещает положение экрана для отображения сигнала начиная с позиции **iPos** массива данных.

**GetScreenWidth**( ) Функция возвращает ширину области отображения осциллограммы в точках.

**GetCurZoom**( ) Функция возвращает текущее значение масштаба. Данное значение определяет количество данных необходимое для отображения одной точки. Умножив это значение на ширину экрана в точках, можно получить полное количество данных необходимое для отображения текущего экрана.

**SetCurZoom**( **iZoom** ) Функция устанавливает текущее значение масштаба. Определены следующие значения масштаба: 1, 2, 5, 10, 20, 50, 100, 200, 500.

**GetScaleDots**( ) Функция возвращает количество точек приходящихся на сторону одной ячейки измерительной сетки экрана.

**GetChannelColor**( **iChannel** ) Функция возвращает значение цвета (\*2) используемого для отрисовки сигнала на канале **iChannel** (\*1).

**GetChannelVDot( iChannel )** Функция возвращает текущее значение вольт на точку для канала **iChannel** (\*1). Данная величина используется для пересчета напряжения в вертикальную координату для канала **iChannel**.

**GetChannelVOffs( iChannel )** Функция возвращает вертикальное смещение нулевой линии для заданного канала **iChannel** (\*1). Смещение отсчитывается от верхней части экрана в точках. Положительные значения означают, что нулевая линия канала расположена ниже крайней верхней линии экрана, отрицательные – выше.

**RedrawScreen( )** Функция вызывает полную перерисовку экрана осциллограммы. Обычно данная функция вызывается всякий раз, когда необходимо изменить информацию отображаемую скриптом при помощи функций обработчиков событий "**BeforeDraw**" или (и) "**AfterDraw**".

**UpdatePannel( )** Функция вызывает перерисовку окна панели анализатора.

**GetTMarkerPos( Index )** Функция возвращает положение измерительного маркера **Index**. Это положение является индексом в массиве данных осциллограммы. Значение параметра **Index** равное 0 и 1 соответствует основным маркерам А и В. Значения 2 ... 17 - адресуют 16 дополнительных маркеров 0....9, С...Н соответственно.

**SetTMarkerPos( Index, iPos)** Функция устанавливает измерительный маркер **Index** в позицию **iPos** в массиве данных осциллограммы. Значение параметра **Index** аналогично ф. **GetTMarkerPos**. Отрицательное значение параметра **iPos** для маркеров **Index=2...7** скрывает маркер.

#### **4.5 Объект конфигурация.**

Данный объект служит для задания конфигурационных параметров процесса анализа сигнала. Объект создается функцией **CreateConfigure** объекта *Host*. Ниже представлены функции объекта конфигурация.

**AddItem( strName, DefValue )** Функция добавляет переменную конфигурации **strName** со значением по умолчанию – **DefValue**. Если параметр **DefValue** задать в виде строки состоящей из отдельных элементов разделенных символом перевода каретки "**\n**", то данная переменная будут представлены в диалоговом окне конфигурации виде элемента выбора. В этом случае пользователю предлагается выбрать один из элементов списка. Функция **GetValue** в этом случае вернет порядковый индекс выбранного элемента начиная с нуля.

**Configure( )** Функция отображает диалоговое окно конфигурации наполненное элементами заданными предшествующим вызовом ф. **AddItem**. Если пользователь подтвердил корректность конфигурации нажатием на кнопку "ОК", функция возвращает **true**, или **false** в противном случае. Изначально значения всех переменных считываются из реестра.

**GetValue( strName )** Функция возвращает значение переменной **strName**, установленное пользователем в диалоговом окне конфигурации.

#### 4.6 Описание объекта **окно графика**.

Данный объект служит для отображения одного или большего числа графиков. Объект создается функцией **CreateGraphicView** или **CreateGraphicViewEx** объекта *Host*. Ниже представлены функции и параметры объекта окно графика.

**MainAxisName** Переменная определяет имя основной (общей) оси графика по координате X.

**AddGraphic( objGraphic )** Функция добавляет объект **objGraphic**, содержащий информацию о графике, для отображения в данном окне графика.

**AddGraphicEx( objGraphic, DrawType, vColour )** Функция аналогична ф. **AddGraphic**, параметр **DrawType** определяет тип прорисовки а **vColour** (\*2) цвет графика заданного параметром **objGraphic**. Текущая версия программы поддерживает следующие типы: 0 – линии, 1 – точки, 2 – линии и точки.

**SetDescription( strText )** Функция служит для задания текстового описания **strText**, выводимого в нижней части экрана окна графика.

**SetScale( vStep, vColour )** Функция служит для задания шага измерительной сетки **vStep** в единицах размерности оси X. Цвет линий сетки определяет параметр **vColour** (\*2).

**SetLimits( vMin, vMax )** Функция устанавливает значения минимума – **vMin** и максимума – **vMax** для координаты 'X'. Если эти значения не заданы, минимум и максимум определяется из суммарного массива точек для всех графиков, добавленных в окно графика.

**SetBkColour( vColour )** Функция устанавливает цвет заливки фона **vColour** (\*2), для окна графика.

**Show( )** Функция активизирует окно графика.

**Update( )** Функция инициирует перерисовку всех графиков в окне графика.

#### 4.7 Объект **график**.

Данный объект накапливает точки графика, определяющие зависимость исследуемой величины, значения которой отображаются по оси 'Y', от изменения параметра, значения которого расположены на оси 'X'. Объект создается функцией **CreateGraphic** объекта *Host*.

**AddPoint ( X, Y )** Функция добавляет новую точку графика, со значениями **X** и **Y**. Данными значениями могут быть любые вещественные числа.

**SetLimits( vMin, vMax )** Функция явно определяет диапазон изменения значений величины по оси 'Y' от минимума - **vMin** до максимума - **vMax**. Если данные значения не определены явно, они определяются программой из массива точек графика.

**SetScale( vStep, vColour )** Функция задает шаг измерительной сетки **vStep** в единицах размерности исследуемой величины значения которой отображаются по оси 'Y' данного графика. Цвет линий сетки задается параметром **vColour (\*2)**.

**Reset( )** Функция удаляет все точки графика.

#### 4.8 Объект **Draw**.

Данный объект служит для отображения графической и текстовой информации в окне осциллограммы. Объект передается функциям обработчикам событий "**BeforeDraw**" и "**AfterDraw**" в качестве параметра.

Рисование осуществляется в координатах экрана осциллограммы, крайние значения которого задаются константой типа прямоугольник **rScreen** объекта **Draw**. Область внутри экрана осциллограммы, ограниченная измерительной сеткой, определена константой типа прямоугольник **rScale** объекта **Draw**. Нулевые координаты расположены в верхнем левом углу экрана. Координата X - увеличивается слева на право, Y – сверху в низ.

**SetPen( vColour )** Функция устанавливает цвет **vColour (\*2)**, которым будет осуществляться отрисовка линий функциями **DrawLine** и **DrawRect**.

**SetBrush( vColour )** Функция устанавливает цвет **vColour (\*2)** кисти, используемой для заливки областей ф. **FillRect**.

**SetTextColour( vColour )** Функция устанавливает цвет **vColour (\*2)** которым будет осуществляться вывод текстовых сообщений ф. **DrawText**.

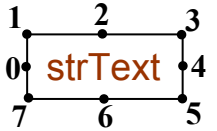
**SetFont( strName, iSize, iWeight, bItalic )** Функция устанавливает параметры шрифта используемого для вывода текстовых сообщений ф. **DrawText**. Параметр **strName** – определяет имя шрифта, **iSize** – размер шрифта, **iWeight** – толщина линий при выводе шрифта, **bItalic** – опция наклона шрифта (0 - выключена, 1 - включена).

**DrawLine( X1, Y1, X2, Y2 )** Функция соединяет линией точку заданную координатами **X1, Y1** с точкой заданную координатами **X2, Y2**.

**DrawRect( X1, Y1, X2, Y2 )** Функция рисует прямоугольник, верхний левый угол которого задан координатами **X1, Y1**. Координаты **X2, Y2** определяют его правый нижний угол соответственно.

**FillRect( X1, Y1, X2, Y2 )** Функция закрашивает прямоугольную область, верхний левый угол которой задан координатами **X1, Y1**. Координаты **X2, Y2** определяют ее правый нижний угол соответственно.

**DrawText( X, Y, P, strText )** Функция выводит текст **strText** в координаты заданные параметрами **X, Y**. Ориентация текста осуществляется по параметру **P**, который определяет положение заданной точки с координатами **X, Y** относительно ограничивающего текст прямоугольника. Рисунок ниже, наглядно отображает ориентацию данной точки в зависимости от значения **P**.



#### 4.9 Объект прямоугольник.

Объект типа **прямоугольник** используется для задания прямоугольной области. Координаты данной области определяются значениями **left** – задающей координату **X** левой стороны прямоугольника, **top** – задающей координату **Y** верхней стороны прямоугольника, **right** – задающей координату **X** правой стороны прямоугольника и **bottom** – задающей координату **Y** нижней стороны прямоугольника соответственно. Для объекта **прямоугольник** определены две функции, позволяющие определить размеры описываемой области.

**Width( )** Функция возвращает ширину прямоугольной области.

**Height( )** Функция возвращает высоту прямоугольной области.

#### 4.10 Объект поток.

Потоки с момента создания их ф. **CreateStream** объекта **Host**, привязываются к входному или выходному каналу данных осциллограммы. Ниже представлены общие для входных и выходных потоков функции.

**Reset( )** Функция сбрасывает поток в исходное состояние. Указатель на первый элемент потока устанавливается в нуль.

**GetPos( )** Функция возвращает текущую позицию потока.

**SetPos( iPos )** Функция устанавливает текущую позицию потока в позицию **iPos**.

**Offset( iOffs )** Функция перемещает положение потока от текущей позиции на значение **iOffs** от текущей позиции.

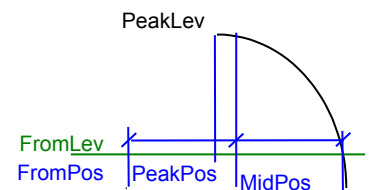
Список функций входных потоков представлен ниже.

**GetAverage( cSamples )** Функция рассчитывает и возвращает среднее значение сигнала для данных потока начиная с текущей позиции на отрезке **cSamples**. Текущая позиция потока перемещается на величину **cSamples**.

**FindFront ( Front, Level, dT, dV )** Функция осуществляет поиск фронта сигнала **Front** с пересечением уровня **Level** начиная с текущей позиции потока. Параметры **dT** и **dV** являются опциональными и необходимы для дополнительной селективности. Параметр **Front** определяет тип искомого фронта и может принимать следующие значения: 1 – передний, -1 задний, 0 – любой. Значение параметра **Level** задается в вольтах. Параметр **dT** определяет количество данных участвующих в вычислении среднего арифметического значения. Увеличение значения **dT**, позволяет

отфильтровать высокочастотную составляющую сигнала. Параметр  $dV$  определяет пороговое значения приращения напряжения. Высокочастотный сигнал имеет большее значение приращения напряжения, поэтому увеличение  $dV$  при условии уменьшения  $dT$ , позволяет селектировать высокочастотную составляющую сигнала. Параметры  $dT$  и  $dV$ , задаются как положительные числа не равные нулю. Функция возвращает тип найденного фронта, или 0 – если сигнал с заданными параметрами не найден. Если фронт найден, текущее положение потока устанавливается в искомую позицию, в противном случае текущее положение потока остается неизменным.

**FindPeak** ( **MinDelta**, **AveDepth**, **Mid**, **Samples** ) Функция осуществляет поиск пикового значения сигнала превышающего порог в **MinDelta**, начиная с текущей позиции потока. Параметры **AveDepth**, **Mid**, **Samples** - являются опциональными. Если **MinDelta** имеет положительное значение, функция осуществляет поиск максимума, в противном случае - минимума. Параметр **AveDepth** – определяет количество данных участвующих в вычислении среднего арифметического значения используемого при поиске пика сигнала. Увеличение данного параметра позволяет уменьшить влияние высокочастотной составляющей сигнала на результат. Если данное значение равно 1, усреднение не используется. Если пик найден и значение параметра **Mid**=0, функция устанавливает текущее положение потока в положение **PeakPos** найденного пикового значения **PeakLev**. Если значение **Mid**=1 функция корректирует положение найденного максимума **MidPos** на середину отрезка сечения дуги сигнала уровнем сигнала **FromPos** в текущей позиции потока **FromPos**. Смотри рисунок ниже. В случае успешного завершения функция возвращает найденное пиковое значение **PeakLev**. Иначе возвращаемое значение - **null**, а текущее положение потока остается неизменным.



Функции выходных потоков представлены ниже.

**Fill** (**Level**, **Samples**, **Type** ) Функция заполняет выходной поток значением **Level**, начиная с текущей позиции потока, на интервале **Samples**. Тип заполнения определяется параметром **Type**. Если **Type** = 0, **Samples** - элементов выходного потока заполняются значением **Level**. При **Type** = 1 функция линейно интерполирует значения на интервале от текущего положения потока до значения **Level** в конце интервала **Samples**. Параметры **Samples** и **Type** опциональные. По умолчанию **Samples** = 0 – до конца массива данных, **Type** = 0.

## 4.11 Комментарии.

(\*1) - Номер канала при вызове функций задается значениями от 1 до N, где N – максимальное количество каналов в анализируемой осциллограмме.

(\*2) – Параметр цвет, задается в виде шестнадцатиричных значений типа 0xRRGGBB, где поля RR – определяет красную, GG – зеленую, BB – голубую составляющие цвета. Значения этих полей могут изменяться в диапазоне от 00 до FF (255 в десятичной форме представления). Таким образом значение 0x000000 – соответствует черному цвету, 0xFFFFFFFF – белому, 0xFF0000 – ярко-красному, 0x00FF00 – ярко-зеленому, а 0x00FF00 – ярко-синему соответственно.

**JScript** файлы анализатора допускают использование директивы - `“//include “ИмяФайла””`. Если данная директива обнаружена, содержимое файла `<ИмяФайла>` вставляется в текст скрипта вместо текущей директивы.

Программа определяет следующие классы для файлов на базе HTML:  
*InnerEdge* – рамка с углублением;  
*OuterEdge* – выпуклая рамка;  
*Display* – элемент данного класса имеет рамку с углублением и цветом фона аналогичным цвету элементов измерительной панели программы.

## 5 ДОПОЛНЕНИЯ

### 5.1 Пример использования скрипта анализатора на JScript

```
// Алгоритм находит инверсные импульсы и определяет их
// длительность
If (DataType == "DIG") // Данные должны быть цифровыми
{// Создать конфигурацию, чтобы пользователь мог задать номер канала
var MyCong = CreateConfigure("Impulse width");

MyCong. AddItem("Канал", 1); // По умолчанию используется 1-й канал
MyCong. Configure(); // Вывести диалог конфигурации
DataChannel = GetValue("Канал"); // Получить номер анализируемого канала
// Проверить введенный пользователем номер канала
If (DataChannel > 0 && DataChannel <= Channels) {
cFound = 0;
ReportOut( "==== Запуск =====" );

for( Position = 0; Position < NumberOfSamples;)
{ // Получить положение очередного заднего фронта сигнала
StartPos = SynchBy(DataChannel, -1, 0, Position, 0);

If (StartPos >= 0) { // Задний фронт найден, найти след. передний
Position = SynchBy(DataChannel, 1, 1, StartPos, 0);

If (Position >= 0) {// Передний фронт найден, опред. длит. импульса в
мсек.
ImpulseWidth = 1000 * (Position – StartPos)/ Frequency;
cFound++;
// Вывести в отчет номер импульса и его длительность.
ReportOut( "Импульс N:" + cFound.toString(10) + " Длительность:" +
ImpulseWidth.toString(10) + " мс \n");
/* Установить маркер в начало импульса и отформатировать комментарий к
данному маркеру, содержащий величину длительности импульса */
SetMarker( StartPos, "Длительность:" + ImpulseWidth.toString(10) + " мс");
}
else break; // Передний фронт не найден, прервать выполнение
}
else break; // Задний фронт не найден, прервать выполнение
}
ReportOut( "==== Завершен =====" );
}
else ReportOut( "Недопустимый номер канала" );
}
else
ReportOut( "Данные должны быть записаны в режиме циф. анализатора");
```

## 5.2 Пример использования скрипта анализатора на HTML

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=windows-
1251">
<TITLE>Find front</TITLE><!-- Имя диалогового окна -->
</HEAD>
<SCRIPT LANGUAGE=javascript>// основное тело скрипта
  var strProgressText = "Process data:";
// Тело основной процедура осуществляющей поиск
function Find()
{
  // Получить значение номера канала из элемента IdChannel
  var iChannel = IdChannel.value;
  // Получить значение уровня сигнала из элемента idLevel
  var level = 1.0*idLevel.value;
  var iFront = 0;// Принять по умолчанию любой тип фронта
  var iPos = 0; // Начальное значение положения в массиве данных
  // Получить значение частоты дискретизации
  var iFrequency = Host.Frequency;

  if (FrontType[0].checked)// Проверить выбран ли нараст. фронт
    iFront = 1;// Установить тип фронта 1 (нарастающий)
  else if (FrontType[1].checked)// Проверить выбран ли убыв. фронт
    iFront = -1;// Установить тип фронта -1 (убывающий)

  // Очистить предыдущие результаты в случае повторного запуска
  функции
  Host.ClearResults();
  // Инициировать отображение статуса процесса обработки данных
  Host.ShowProgress(strProgressText, Host.NumberOfSamples);

  // Продолжать поиск фронта сигнала:iFront на канале:iChannel
  // с заданным уронем:level с текущей позиции:iPos
  while((iPos = Host.SynchBy(iChannel, iFront, level, iPos, 0)) >= 0
    && Host.CanContinue())//Проверить возможно ли продолжение цикла?
  {// если iPos >= 0, iPos - содержит положение в массиве данных,
    // удовлетворяющее заданным условиям

    //установить в позицию:iPos маркер с комментарием сод. время
    Host.SetMarker(iPos, ":"+iPos/iFrequency);
    // Вывести сообщение в окно отчета
    Host.ReportOut("Next marker at: "+iPos/iFrequency+" sec.\n")
    // Установить текущее положение полосы выполнения процесса
    Host.SetProgress(iPos);
  }
  // Убрать полосу выполнения процесса
  Host.HideProgress();
}
</SCRIPT>
```

```

<!-- Задать размеры диалог. окна в точках: ширина=250, высота=200 -->
<BODY width="250px" height="200px">
<!-- Далее в теле HTML определяются элементы управления -->
<TABLE width="100%" height="100%" cellPadding=0 cellSpacing=0>
<TBODY>
<TR vAlign=center>
<!-- Элемент выбора канала -->
  <TD id=idChanTxt width="1px">Channel:</TD><TD width="1px">
    <SELECT id=IdChannel></SELECT>
  </TD>
</TR>
<TR>
<!-- Элемент ввода уровня -->
  <TD id=idLevelTxt>Level [V]:</TD>
  <TD>
    <INPUT type=text id=idLevel size=8 maxlength=8 value="0.0">
  </TD>
</TR>
<!-- Элемент выбора типа фронта -->
<TR>
  <TD id=idFrontTxt>&nbsp;Signal front:</TD>
  <TD>
    <INPUT type=radio id=idRise name=FrontType>rised<BR>
    <INPUT type=radio id=idFall name=FrontType>fallen<BR>
    <INPUT type=radio id=idAny name=FrontType checked>any
  </TD>
</TR>
<TR>
<!-- Кнопки выхода и поиска -->
  <TD Align=right>
    <!-- Определить в качестве обработчика нажатия клавиши ф.
      Host.Quit() - закрыть диалоговое окно -->
    <INPUT id=idExit type=submit value=" Exit " onclick="Host.Quit() ">
  </TD>
  <TD Align=left>
    <!-- Определить в качестве обработчика нажатия клавиши ф.
      Find() - определенную выше и осуществляющую поиск -->
    <INPUT id=idRun type=submit value=" Find " onclick="Find() ">
  </TD>
</TR>
</TBODY>
</TABLE>

```

```
<SCRIPT LANGUAGE=javascript>
// Данный фрагмент скрипта запускается после загрузки тела HTML

// Установить опции элемента выбора канала:IdChannel
for(iChan=1; iChan<=Host.Channels; iChan++)
{
  // Создать элемент типа:OPTION для номер канала
  var oOption = document.createElement("OPTION");
  oOption.text=">" + iChan; // Установить текстовое значение опции
  oOption.value=iChan; // Установить числовое значение опции
  IdChannel.add(oOption); // Добавить опцию для элемента:IdChannel
}
// Проверить выбран ли в программе русский язык интерфейса
if(Host.GetCurLanguage() == 0x419)
{
  // Установить текст для русскоязычной версии диалога
  idChanTxt.innerHTML = "Канал:";
  idLevelTxt.innerHTML = "Уровень [B]:";
  idFrontTxt.innerHTML = "&nbsp;фронт сигнала:";
  idRise.replaceAdjacentText("afterEnd", "передний");
  idFall.replaceAdjacentText("afterEnd", "задний");
  idAny.replaceAdjacentText("afterEnd", "любой");
  idExit.value = " Выход ";
  idRun.value = " Найти ";
  strProgressText = "Обработка данных:";
} // В данном случае по умолчанию принята англоязычная версия
</SCRIPT>
</BODY>
</HTML>
```