Майкл Бернштейн 10 июля 2009 года Ашкелон.

Виртуальный осциллограф сервера

Динамическая библиотека для Win32:
(для Microsoft Windows 95, 98, NT4, XP, 2000, 2003, Vista, Windows 7).

Оригинальное название файла "Osc_DLL.dll", размер файла 710144 байт,

сумма MD5 (RFC тысячу триста двадцать одна стандартная) является: 4123c94af3bf4895d5c8be0ad28269b6

Экспортируемые функции:

```
AtOpenLib,
ScopeCreate,
ScopeDestroy,
ScopeShow,
ScopeHide,
ScopeCleanBuffers,
Быстрое обновление,
ShowNext,
ExternalNext,

ScopeGetFormLeft,
ScopeGetFormTop,
ScopeGetFormWidth,
ScopeGetFormHeight,
ScopeGetScreenWidth,
ScopeGetScreenHeight,
ScopeGetCellPixelSize,
ScopeGetCellSampleSize,
ScopeGetPanelState,
ScopeGetAmpScale,
ScopeGetAmpOffset,
ScopeGetTriggerLevel,
ScopeGetTriggerSourse,
ScopeGetActiveTriggerEdge,
ScopeGetTrigHoldOffSmplsNum,

ScopeSetPanelState,
ScopeSetFormPos,
ScopeSetFormSize,
ScopeSetCellPixelSize,
ScopeSetCellSampleSize,
ScopeSetCaption,
ScopeSetAmpScale,
ScopeSetAmpOffset,
ScopeSetTriggerLevel,
ScopeSetTriggerSourse,
ScopeSetActiveTriggerEdge,
ScopeSetTrigHoldOffSmplsNum,


DllGetClassObject,
DllCanUnloadNow,
DllRegisterServer,
DllUnregisterServer;
```

Функциональное описание.

AtOpenLib.

Декларация Паскаль-стиль:

```
AtOpenLib: функция (PRM: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
INT (__cdecl * AtOpenLib) (INT PRM);
```

Надо вызывать эту функцию только один раз - только после загрузки DLL в главную программу.
Для динамического нагружения DLL:
вызвать функцию API Windows, '' LoadLibrary, определить указатель на '' AtOpenLib по телефону
Функция API "GetProcAddress" Окна и затем вызвать "AtOpenLib.
В случае статически связанной DLL просто позвонить "AtOpenLib" один раз в начале программы.
Функция "AtOpenLib 'должен быть вызван первым, перед вызовом любой другой функции, экспортируемой по
Эта DLL. Если 羡 tOpenLib называется второй раз это вызовет ошибку в программе.
Обратите внимание, что "AtOpenLib 'не является основной функцией DLL и ОС никогда не называют его автоматически.
Входной аргумент 'Prm "этой функции является формальным и зарезервированы для использования в будущем.
"Prm" может быть любое значение.
Если функция работает должным образом - то он возвращает значение максимального размера осциллографа

случаи, которые можно создать с помощью текущей библиотеки динамической компоновки.
Если функция делает любую ошибку - то он возвращает код ошибки -1 (минус один).

Перед закрытием программы с использованием библиотеки DLL его необходимую осциллографа, чтобы остановить называя
любая функция DLL, уничтожить все созданные экземпляры осциллографа "ScopeDestroy"
Функция DLL, а затем выгрузить DLL осциллографа, используя функцию API '' FreeLibrary
из MS Windows - если он был загружен динамически.

ScopeCreate.

Декларация Паскаль-стиль:

```
  ScopeCreate: функция (PRM: Integer; P_IniName: Указатель; P_IniSuffix: Указатель): Integer; Cdecl;
```

Заявление С-стиль:

```
  INT (__cdecl * ScopeCreate) (INT Prm, символ * P_IniName, символ * P_IniSuffix);
```

Эта функция создает новый экземпляр осциллографа. Новый экземпляр инициализируется полностью
и не требует какого-либо дополнительного сброса. Окно (форма) нового экземпляра невидим.
Позвоните, чтобы функция "ScopeShow" DLL, чтобы принести новое окно осциллографа на экране компьютера.

Аргумент "Prm" является формальным и зарезервированы для будущего. Может быть любое значение.
Аргументы 'P_IniName' и 'P_IniSuffix "являются указателями на строки с нулем в конце ANSI персонажа.

"P_IniName" является указателем на строку, которая является именем файла 'INI' описывает свойства нового
Экземпляр осциллографа.

Если строка путь содержит символ двоеточия (два пятна - ':') - он будет признан
абсолютное полный путь. Если строка пути не содержит двоеточие - тогда новый экземпляр
осциллографа будет искать файл его '' INI в папке собственное приложение в каталоге ().

Длина этой строки должна быть не больше, чем 100 символов.
Если строка отметил имеет нулевую длину - чем осциллограф будет выглядеть
для '' INI файл с именем по умолчанию: "Scope_Desk.ini. Если файл с требуемым именем делает
не существует, - то новый осциллограф будет применяться к себе умолчанию и усваиваемые свойства.

"P_IniSuffix" является указателем на заканчивающаяся нулем ANSI, содержащий последние символы
имен разделов. Например:
Если он указывает на нулевой длины, чем новый осциллограф будет искать настройки
в случае отказа [Осциллограф] и [] Разное разделы файла "INI".
Если он указывает на "ABC" строка - то новый осциллограф будет искать настройки
в [Oscilloscope_ABC] и [] Miscellaneous_ABC разделы файла "INI".
Если раздел с необходимыми имя не существует, - то новый осциллограф будет использовать по умолчанию
и усваиваемые свойства. Длина строки не суффиксом должен быть больше, чем 100 символов.

Разделы объяснение:

[Осциллограф] - раздел содержит свойства осциллографа.
OscVrulFontName = Arial - Имя шрифта для вертикальных линеек цифр.
OscVrulFontSize = 16 - размер шрифта для вертикальных линеек цифр.
OscVrulFontStyleBold = 0 - Жирный этап для вертикальных линеек цифр.
OscVrulFontStyleItalic = 0 - Курсив этап для вертикальных линеек цифр.
OscBackColor = 00000000 - цветной экран осциллографа.
OscGridColor = 00404040 - экран цвет сетки осциллографа.
OscBeam0Color = 0000FF00 - Цвет луча первого осциллографа.
OscBeam1Color = 00FF00FF - Цвет луча второго осциллографа.
OscBeam2Color = 0000FFFF - Цвет луча третьего осциллографа.
OscVrulGenFontColor = 000000FF - Цвет цифр вертикальную линейку, когда это справедливо для всех лучей.
OscHorScaleColor = 000000FF - Цвет текста в поле, которое находится на верхнем левом углу графического поля осциллографа:
он отображает горизонтальное разрешение сигнала образца в одной ячейке основной осциллографа
графическое поле маркировки сетку. Другими словами, он отображает горизонтальную скорость развертки.
OscZeroLineColor = 00808080 - Цвет нулевой горизонтальной линии сетки экрана цвета осциллографа.
ShowSecondBeam = 1 - Второй луч виден (если имеется) или невидимым (если равен нулю).
ShowThirdBeam = 1 - третий луч виден (если имеется) или невидимым (если равен нулю).
OscOscCaption = Осциллограф - Строка основного заголовка окна осциллографа (форма).
ShowTestButton = 1 - Кнопка теста измерения производительности / видимой (если есть) или невидимым (если равен нулю).
BfrSmplSize = 500000 - Размер (в образцах сигнала) из FIFO для каждого созданного пучка (канал).
BuffersRecordType = 1 - Тип созданного элемента FIFO: если 1 - тип двойной точности с плавающей, 0 - одноместное плавающей.
CreateSecondBeam = 1 - Если он не равен нулю - создает второй луч и выделяет память для буфера FIFO его,
в противном случае - нет.
CreateThirdBeam = 1 - Если он не равен нулю - создает третий луч и выделяет память для буфера FIFO его,
в противном случае - нет.
SpreadViewTiming = 1 - Если он не равен нулю - вставляет незначительную задержку в случайное синхронизации сигнала
Визуализация для предотвращения биений (пульсирующая) эффект между развертки монитора компьютера,
развертки лучей осциллографа и занимает времени выхода образцов через
Функция "ShowNext.
GridRefreshDivider = 41 - Как часто сетка имеет в обновлении. Это значение определяет количество горизонтальных луча
полный экран развертки проходит после чего сетка восстановление начинается.
EventsProcDivider = 1000 - это свойство создано для ленивых людей, призывающих к функции "ShowNext 'библиотеки
непосредственно из потока части главного окна программы.
Это значение определяет количество звонков в функции "ShowNext ', после которого начинается
алгоритм "ProcessMessages.
"ProcessMessages" позволяет приложению обрабатывать сообщения, что в настоящее время
в очереди сообщений. ProcessMessages циклов цикл обработки сообщений Windows, до
он пуст, а затем возвращает управление функции "ShowNext».
Примечание: Пренебрегая обработки сообщений влияет только на вызывающее приложение
ProcessMessages, не другие приложения. В длительных операций, называя
ProcessMessages периодически позволяет приложению реагировать другие сообщения.
Примечание: ProcessMessages не позволяет приложению перейти в режиме ожидания,

в то время как функция Windows API, «HandleMessage 'делает.
Если это значение равно нулю - алгоритм "ProcessMessages" не работает никогда.
GridCellPixelsSize = 40 - Определяет геометрические размеры ячеек Сетка в пикселях экрана. От 10 до 150.
Клетки всегда квадрат.
TriggerEnable = 0 - режим запуска развертки - если он равен нулю, то триггер отключен,
если он больше нуля - триггер работает на повышение направление наклона сигнала,
если она меньше нуля - триггер работает на падение направление наклона сигнала.
FirstTriggerLevel = 0,00 - уровень реагирования триггера для первого луча - если триггер включен и
выбран первый луч в качестве источника сигнала для триггера.
SecondTriggerLevel = 0,00 - То же самое, как "FirstTriggerLevel ', но для второго луча.
ThirdTriggerLevel = 0,00 - То же самое, как "FirstTriggerLevel ', но для третьего луча.
ExtrnTriggerLevel = 0,00 - То же самое, как "FirstTriggerLevel ', но для внешнего входного сигнала триггера.
TriggerSource = 0 - Она определяет источник сигнала для запуска. Если это 0 (ноль), то вызывают источник
Первый луч 1 - второй луч, 2 - третий луч и 3 - входной сигнал внешнего запуска.
TimeSamplesPerCell = 20,0 - это горизонтальная (время) масштаб. Значение показывает количество выборок сигнала в
за один шаг горизонтальной экранной сетки - на одну ячейку этой сетки.
Это читает на старте осциллографа и тенденцию к ближайшей стандартной величины
горизонтальных координат масштаб.
Beam0VertScale = 20,0 - это вертикальная (амплитуда) масштаб первого луча. Это значение показывает амплитуду
размерность (коэффициент масштабирования) из одной вертикальной стадии экранной сетки - одна ячейка
этой сетки.
Beam1VertScale = 20,0 - То же самое, как "Beam0VertScale ', но для второго луча.
Beam2VertScale = 20,0 - То же самое, как "Beam0VertScale ', но для третьего луча.
Beam0VertOffset = 0.0 - это вертикальная (амплитуда) смещение нулевой базы для первого луча с середины
Высота экрана осциллографа.
Beam1VertOffset = 0,0 - то же, как "Beam0VertOffset ', но для второго луча.
Beam2VertOffset = 0,0 - то же, как "Beam0VertOffset ', но для второго луча.
ScopeFormTop = 108 - Экран компьютера вертикаль координаты верхней части формы прицела (окно) в пикселях.
Точка происхождения в левом верхнем углу экрана компьютера.
ScopeFormLeft = 108 - Компьютерная экрана горизонтальная координата левой стороне формы прицела (окно)
в пикселях экрана.
ScopeFormHeight = 580 - форма (окно) Высота прицела в пикселях экрана компьютера.
ScopeFormWidth = 765 - форма (окно) ширина прицела в пикселях экрана компьютера.
ScopeStayOnTop = 0 - Если он не равен нулю, то форма осциллографа (окно) всегда остается перед другими окнами
В окне этот режим осциллографа не представить "ScopeFormTop» и «полей» ScopeFormLeft.
ScopeShowHints = 0 - Если он не равен нулю, то сфера показывает подсказки о цели его визуальных компонентов управления.
ShowTriggerControls = 1 - если он равен нулю - скрывает все элементы управления триггера, в противном случае нет.
ShowOscJaggies = 0 - Если она равна нулю, луч нанесены путем подключения сигнала сдержанный точек с
одного сегмента линии (первый интерполяции заказ). Если он не равен нулю, смежный
точки света связаны вертикальными и горизонтальными отрезками, при этом показывая
луч в "шагов" для лучшего просмотра образцов.
ShowOscHexVal = 0 - Если это не ноль - то 'значение' текст поле осциллографа интерфейс имеет
шестнадцатеричное представление.
ShowHexCheckBox = 1 - Если это не равно нулю - ". Hex", то флажок графического интерфейса осциллографа видна.
Если он равен нулю, то 'значение' текстовое поле имеет десятичной с представлением дробное значение.
Если он не равен нулю - "стоимость" текстовое поле имеет шестнадцатеричное целое представление.
ScopeShowAutoSweep = 0 - Если он не равен нулю - форма объема показывает "Auto развертки" проверить контроль окно запуска.
ScopeAutoSweep = 0 - Если он не равен нулю, а поле "ScopeShowAutoSweep 'не равно нулю и" Auto развертки "флажок
включена, то, триггер работает в режиме автоматического качания.
ScopeAutoSweepDivider = 4 - коэффициент прореживания в режиме автоматического качания. Следует отметить, что осциллограф не показывает
все образцы сигнала в режиме триггера авто развертки. Однако все отсчеты сигнала
будут сохранены в буферах Сфера 痴
ScopeInputDumpSize = 32000 - Определяет этап ввода в буфер осциллографа демпинга.
Если это 0 (ноль или отрицательное значение), - то осциллограф не имеет никакого вход
захоронения буфер, в противном случае это поле определяет длину буфера и сброса
Функции "ShowNext '' & 'ExternalNext работать через свалку буфер.
Наличие этого буфера не снизить производительность осциллографа, но решает
Время - критические ситуации. Если это значение больше нуля и меньше, чем 32000, - то
демпинг длина буфера будет указано, как 32000.

[Разное] - раздел, содержащий разные свойства, например осциллографа.
ReadDelayPerSample = 0 - задержка в мс. для одного образца сигнала чтения из файла и показывая в осциллограф.

Если функция работает должным образом - то он возвращает ручку нового осциллографа.
Действительно ручка должна быть любым целым числом со значением, которое больше нуля и не более максимального
количество экземпляров осциллографа возвращается функцией '' AtOpenLib.
Если возвращенный дескриптор не соответствует выше условия - то это ошибка, и новый
Осциллограф не создается.
Вернулся действует ручка используется для вызова к экспортируемой функции операционной каждого DLL,
с экземпляров, созданных осциллографа.

---------- ----------

ScopeDestroy.

Декларация Паскаль-стиль:

```
  ScopeDestroy: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeDestroy) (INT ScopeHandle);
```

Полное уничтожение ocilloscope инстанции и освободить память, используемую для данного экземпляра.
Дает возможность создавать количество новых осциллографов в одном больше, чем раньше
Осциллограф уничтожения.
Аргумент '' ScopeHandle является ручка уничтожения осциллограф. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Если функция работает должным образом - то он возвращает целое значение, который
равна входной аргумент '' ScopeHandle, в противном случае код ошибки возвращается и
Экземпляр осциллографа не разрушается.

Перед удалением из ocilloscope Например, необходимого, чтобы остановить называя
в этом случае по любой функции DLL.

ScopeShow.

Декларация Паскаль-стиль:

```
  ScopeShow: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeShow) (INT ScopeHandle);
```

Показывает - приносит окно осциллографа (форма) в экране.

Аргумент '' ScopeHandle является ручка, показывая осциллограф. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Если функция работает должным образом - то он возвращает целое значение, которое
равно входного аргумента '' ScopeHandle, в противном случае код ошибки возвращается и
Экземпляр осциллографа не показали.

Внимание! Частое призвание к этой функции вызовет замедления скорости производительности осциллографа.

ScopeHide.

Декларация Паскаль-стиль:

```
  ScopeHide: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeHide) (INT ScopeHandle);
```

Скрывает - удаляет окно осциллографа (форма) от экрана. Не разрушает
- Делает его невидимым только.

Аргумент '' ScopeHandle является ручка укрытия осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Если функция работает должным образом - то он возвращает целое значение, которое
равно входного аргумента '' ScopeHandle, в противном случае код ошибки возвращается и
Экземпляр осциллографа не скрыты.

Внимание! Частое призвание к этой функции вызовет замедления скорости производительности осциллографа.

ScopeCleanBuffers.

Декларация Паскаль-стиль:

```
  ScopeCleanBuffers: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeCleanBuffers) (INT ScopeHandle);
```

Очищает все данные сигнала из буферов осциллографа и очищает экран осциллографа.

Аргумент '' ScopeHandle является ручка чистки осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Если функция работает должным образом - это то возвращает целое значение, которое
равно входного аргумента '' ScopeHandle, в противном случае код ошибки возвращается и
Экземпляр осциллографа не очищается.

Эта функция работает достаточно медленно - использование способности триггера осциллографа
для запуска развертки по горизонтали от левой части экрана осциллографа
но не функция "ScopeCleanBuffers.

Внимание! Частое призвание к этой функции вызовет замедления скорости производительности осциллографа.

ShowNext.

Декларация Паскаль-стиль:

```
  ShowNext: функция (ScopeHandle: Integer; PArrDbl: Указатель): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ShowNext) (INT ScopeHandle дважды * PArrDbl);
```

Выход (рисунок) следующего этапа лучей сигнала осциллографа.

Аргумент '' ScopeHandle является ручка осциллограф показ сигнала. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Аргумент '' PArrDbl это указатель на массив трех переменных типа двойного
(64-битные двойной точности с плавающей точкой в формате IEEE 754).
Во-первых член массива соответствует первого луча осциллографа, второго члена - в
Второй луч и третий - третий луч.
Новые этапы лучей будет обращено на экране осциллографа немедленно.
Эта функция не проверяет достоверность данных указал аргументом "PArrDbl '.
Если острый данные некорректны, - то библиотека будет остановить текущий поток и покажет
Ошибка выполнения окно исключением.

Если функция работает должным образом - то он возвращает целое значение, которое
равно входного аргумента '' ScopeHandle, в противном случае код ошибки возвращается и
Экземпляр осциллографа не показать следующий этап сигналов.

ExternalNext.

Декларация Паскаль-стиль:

```
  ExternalNext: функция (ScopeHandle: Integer; PDbl: Указатель): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ExternalNext) (INT ScopeHandle дважды * PDbl);
```

Отправка осциллограф новое значение амплитуды внешнего сигнала для триггера.

Аргумент '' ScopeHandle является ручка осциллограф показ сигнала. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Аргумент '' PDbl это указатель на переменную типа двойного (64-битной двойной точности
с плавающей точкой IEEE формат 754). Это новое значение амплитуды внешнего сигнала для триггера.
Эта функция не проверяет достоверность данных указал аргументом "PDbl '.
Если острый данные некорректны, - то библиотека будет остановить текущий поток и покажет
Ошибка выполнения окно исключением.

На самом деле эта функция просто пишет новый внешний значение сигнала в внутренней переменной
экземпляра осциллографа.
Осциллограф будет реагировать на это значение только в момент вызова функции, чтобы '' ShowNext.
Если вызов "ExternalNext" несколько раз один за другим без вызова '' ShowNext
Функция - то только последний аргумент призван функции "ExternalNext 'будет сохранен
в внутренней переменной экземпляра осциллографа.

Выбрать '' для внешнего источника синхронизации (TriggerSource = 3 в INI. Файле) для настройки
Экземпляр осциллографа реагировать на внешний сигнал запуска.

Если функция работает должным образом - то он возвращает целое значение, который
равна входной аргумент '' ScopeHandle, в противном случае код ошибки возвращается и
Экземпляр осциллографа не принимает очередной этап внешнего сигнала.

Быстрое обновление.

Декларация Паскаль-стиль:

```
  QuickUpDate: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * QuickUpDate) (INT ScopeHandle);
```

Быстро освежает экран осциллографа. Вызов этой функции, как правило,
не требуется. Рекомендуется для использования в ситуациях, когда интенсивный поток данных
вдаваясь в осциллографа прекращены или прерваны в течение длительного времени.

Аргумент '' ScopeHandle является ручка обновления осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Если функция работает должным образом - то он возвращает целое значение, который
равна входной аргумент '' ScopeHandle, в противном случае код ошибки возвращается и
Экземпляр осциллографа не обновляется.

Внимание! Частое призвание к этой функции вызовет замедления скорости производительности осциллографа.

ScopeGetFormLeft.

Декларация Паскаль-стиль:

```
  ScopeGetFormLeft: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeGetFormLeft) (INT ScopeHandle);
```

Возвращает горизонтальную ординату левой границы формы (окна) осциллографа на экране компьютера.
Вернулся позиция представлена в пикселях экрана компьютера.

Дело происхождения экране компьютера (0; 0) является левый верхний угол, его положительные направления
вниз и вправо.

Если она возвращает -2147483647 - это ошибка.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

ScopeGetFormTop.

Декларация Паскаль-стиль:

```
  ScopeGetFormTop: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeGetFormTop) (INT ScopeHandle);
```

Возвращает вертикальную ординату верхней границы формы (окна) осциллографа на экране компьютера.
Вернулся позиция представлена в пикселях экрана компьютера.

Дело происхождения экране компьютера (0; 0) является левый верхний угол, его положительные направления
вниз и вправо.

Если она возвращает -2147483647 - это ошибка.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

ScopeGetFormWidth.

Декларация Паскаль-стиль:

```
  ScopeGetFormWidth: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeGetFormWidth) (INT ScopeHandle);
```

Эта функция возвращает горизонтальную геометрические размеры форм (окна) осциллографа в пикселях экрана компьютера.

Если она возвращает ноль или отрицательное значение - это ошибка.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

ScopeGetFormHeight.

Декларация Паскаль-стиль:

```
  ScopeGetFormHeight: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeGetFormHeight) (INT ScopeHandle);
```

Эта функция возвращает вертикальную геометрическую форму размер (окна) осциллографа в пикселях экрана компьютера.

Если она возвращает ноль или отрицательное значение - это ошибка.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

ScopeGetScreenWidth.

Декларация Паскаль-стиль:

```
  ScopeGetScreenWidth: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeGetScreenWidth) (INT ScopeHandle);
```

Эта функция возвращает горизонтальную геометрические размеры экрана сигнала осциллографа в пикселях экрана компьютера.

Если она возвращает ноль или отрицательное значение - это ошибка.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

ScopeGetScreenHeight.

Декларация Паскаль-стиль:

```
  ScopeGetScreenHeight: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeGetScreenHeight) (INT ScopeHandle);
```

Эта функция возвращает вертикальную геометрическую размер экрана сигнала осциллографа в пикселях экрана компьютера.

Если она возвращает ноль или отрицательное значение - это ошибка.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

ScopeGetCellPixelSize.

Декларация Паскаль-стиль:

```
  ScopeGetCellPixelSize: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeGetCellPixelSize) (INT ScopeHandle);
```

Эта функция возвращает геометрический размер ячейки сетки осциллографа (шаг сетки, интервал) в пикселях экрана.

Форма ячеек сетки квадратной всегда.

Если она возвращает ноль или отрицательное значение - это ошибка.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

ScopeGetCellSampleSize.

Декларация Паскаль-стиль:

```
  ScopeGetCellSampleSize: функция (ScopeHandle: Integer): Double; Cdecl;
```

Заявление C-стиль:

```
  дважды (__cdecl * ScopeGetCellSampleSize) (INT ScopeHandle);
```

Эта функция возвращает масштаб по горизонтали осциллографа в ячейки сетки в образцах время сигнала.

Если она возвращает ноль или отрицательное значение - это ошибка.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

ScopeGetPanelState.

Декларация Паскаль-стиль:

```
  ScopeGetPanelState: функция (ScopeHandle: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeGetPanelState) (INT ScopeHandle);
```

Эта функция возвращает 1 (один), если панель управления упомянутой осциллографа скрыл состояние
или 2 (два) - если это видно.

Если она возвращает другое значение - это ошибка.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

ScopeSetPanelState.

Декларация Паскаль-стиль:

```
  ScopeSetPanelState: функция (ScopeHandle: Integer; VisState: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeSetPanelState) (INT ScopeHandle, INT VisState);
```

Эта функция устанавливает панель управления упомянутой осциллографа в видимый или невидимый государства.
Если аргумент '' VisState не равна нулю - то функция будет показывать панель, в противном случае - скрыть.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Если функция работает должным образом - то он возвращает целое значение, которое
равно входного аргумента '' ScopeHandle, в противном случае код ошибки возвращается и
состояние панели управления осциллографа не изменилось.

Внимание! Частое призвание к этой функции вызовет замедления скорости производительности осциллографа.

ScopeSetFormPos.

Декларация Паскаль-стиль:

```
  ScopeSetFormPos: функция (ScopeHandle: Integer; FormLeft: Integer; FormTop: Integer): Integer; Cdecl;
```

Заявление C-стиль:

```
  INT (__cdecl * ScopeSetFormPos) (INT ScopeHandle, INT FormLeft, INT FormTop);
```

Эта функция устанавливает новый геометрическое положение формы (окна) осциллографа на экране компьютера.
Новая должность определяется в пикселях экрана по "FormLeft» и аргументов "FormTop '.
Аргумент '' FormLeft определяет горизонтальную ординату левой границы формы.
Аргумент '' FormTop определяет вертикальную ординату верхней границы формы.
Дело происхождения экране компьютера (0; 0) является левый верхний угол, его положительные направления
вниз и вправо.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Если функция работает должным образом - то он возвращает целое значение, которое
равно входного аргумента '' ScopeHandle, в противном случае код ошибки возвращается и
Положение осциллографа на экране не меняется.

Внимание! Частое призвание к этой функции вызовет замедления скорости производительности осциллографа.

ScopeSetFormSize.

Декларация Паскаль-стиль:

```
  ScopeSetFormSize: функция (ScopeHandle: Integer; FormWidth: Integer; FormHeight: Integer): Integer; Cdecl;
```

Заявление С-стиль:

```
  INT (__cdecl * ScopeSetFormSize) (INT ScopeHandle, INT FormWidth, INT FormHeight);
```

Эта функция устанавливает новый геометрический размер формы (окна) осциллографа в пикселях экрана.
Аргумент '' FormWidth определяет новую ширину формы, 'FormHeight "- новую высоту.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Если функция работает должным образом - то он возвращает целое значение, которое
равно входного аргумента '' ScopeHandle, в противном случае код ошибки возвращается и
Размер формы осциллографа не изменилось.

Внимание! Частое призвание к этой функции вызовет замедления скорости производительности осциллографа.

ScopeSetCellPixelSize.

Декларация Паскаль-стиль:

```
  ScopeSetCellPixelSize: функция (ScopeHandle: Integer; CellPixelSize: Integer): Integer; Cdecl;
```

Заявление С-стиль:

```
  INT (__cdecl * ScopeSetCellPixelSize) (INT ScopeHandle, INT CellPixelSize);
```

Эта функция устанавливает новый геометрический размер ячейки сетки осциллографа (шаг сетки, интервал) в пикселях экрана.
Новый шаг сетки определяется "CellPixelSize" функции и может быть от 10 до 150.

Форма ячеек сетки квадратной всегда.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

Если функция работает должным образом - то он возвращает целое значение, которое
равно входного аргумента '' ScopeHandle, в противном случае код ошибки возвращается и
Размер осциллографа из ячейки сетки не изменяется.

Внимание! Частое призвание к этой функции вызовет замедления скорости производительности осциллографа.

ScopeSetCellSampleSize.

Декларация Паскаль-стиль:

```
  ScopeSetCellSampleSize: функция (ScopeHandle: Integer; CellSampleSize: Двухместный): Integer; Cdecl;
```

Заявление С-стиль:

```
  INT (__cdecl * ScopeSetCellSampleSize) (INT ScopeHandle дважды CellSampleSize);
```

Эта функция устанавливает горизонтальное масштабирование осциллографа в ячейки сетки в образцах время сигнала.
Шкала Нового ячейки определяется аргументом "CellSampleSize 'функции.
Осциллограф принесет масштаб ячейки до ближайшего стандартного значения шкалы.
Например: если "CellSampleSize" 53, - то шкала новая ячейка будет указано, как 50.
Здесь перечислены стандартные значения (нормированных) весов:
0,1; 0,2; 0,5; 1; 2; 5; 10; 20; 50; 100; 200; 5 часов; 1000; 2000 года; 5000; 10000; 20000.

Аргумент '' ScopeHandle является ручка экземпляра упомянутого осциллографа. Эта ручка возвращается
по специальностям "ScopeCreate ', когда был создан этот экземпляр.

If the function works properly - then it returns signed integer value which is
equal to 'ScopeHandle' input argument, otherwise code of error returned and
oscilloscope's horizontal scale of grid cell is not changed.

Внимание! Frequent calling to this function will cause of slowing down of oscilloscope's performance speed.

ScopeSetCaption.

Pascal-style declaration:

```
  ScopeSetCaption : function (ScopeHandle : Integer; P_ZeroTermStr : Pointer) : Integer; cdecl;
```

C-style declaration:

```
  int (__cdecl * ScopeSetCaption) (int ScopeHandle, char * P_ZeroTermStr);
```

This function writes new text string into the caption of mentioned oscilloscope's window (form).

The caption is a text allocated on the top of screen form.

Argument 'P_ZeroTermStr' is a pointer to null-terminated ANSI character's string of new caption.
Its length can be not more than 250 characters.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If the function works properly - then it returns signed integer value which is
equal to 'ScopeHandle' input argument, otherwise code of error returned and
oscilloscope's caption string is not changed.

Внимание! Frequent calling to this function will cause of slowing down of oscilloscope's performance speed.

ScopeSetAmpScale.

Pascal-style declaration:

```
  ScopeSetAmpScale : function (ScopeHandle : Integer; BeamNum : Integer; AmpScale : Double) : Integer; cdecl;
```

C-style declaration:

```
  int (__cdecl * ScopeSetAmpScale) (int ScopeHandle, int BeamNum, double AmpScale);
```

This function sets vertical (amplitude) scale of mentioned oscilloscope's beam.
Functional argument 'AmpScale' defines new amplitude value of one grid cell in signal's amplitude units.

The oscilloscope will bring cell's vertical scale to nearest standard scale value.
For instance: if 'AmpScale' is 52.78467 - then new cell's vertical scale will be stated as 50.0 .
Here are listed some of standard (normalized) scale's values:
... 0.00001; 0.00002; 0.00005; 0.0001; 0.0002; 0.0005; 0.001; 0.002; 0.005; 0.01; 0.02; 0.05;
    0.1; 0.2; 0.5; 1; 2; 5; 10; 20; 50; 100; 200; 500; 1000; 2000; 5000; 10000; 20000; 50000... etceteras.

Argument 'BeamNum' defines beam for rescale: If it is 0 - it is first beam, 1 - second and 2 - third.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If the function works properly - then it returns signed integer value which is
equal to 'ScopeHandle' input argument, otherwise code of error returned and
oscilloscope's beam amplitude scale will be not changed.

Внимание! Frequent calling to this function will cause of slowing down of oscilloscope's performance speed.

ScopeSetAmpOffset.

Pascal-style declaration:

```
  ScopeSetAmpOffset : function (ScopeHandle : Integer; BeamNum : Integer; AmpOffset : Double) : Integer; cdecl;
```

C-style declaration:

```
  int (__cdecl * ScopeSetAmpOffset) (int ScopeHandle, int BeamNum, double AmpOffset);
```

This function sets vertical (amplitude) offset (bias) of mentioned oscilloscope's beam.

Functional argument 'AmpOffset' defines vertical (amplitude) offset of zero base for mentioned beam
from middle of oscilloscope's screen height in signal's amplitude units - NOT in pixels of screen.

Argument 'BeamNum' defines beam for rescale: If it is 0 - it is first beam, 1 - second and 2 - third.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If the function works properly - then it returns signed integer value which is
equal to 'ScopeHandle' input argument, otherwise code of error returned and
oscilloscope's beam amplitude offset will be not changed.

Внимание! Frequent calling to this function will cause of slowing down of oscilloscope's performance speed.

ScopeGetAmpScale.

Pascal-style declaration:

```
  ScopeGetAmpScale : function (ScopeHandle : Integer; BeamNum : Integer) : Double; cdecl;
```

C-style declaration:

```
  double (__cdecl * ScopeGetAmpScale) (int ScopeHandle, int BeamNum);
```

This function returns oscilloscope's vertical (amplitude) scale of mentioned beam.
Returned value is value of one grid cell in signal's amplitude units.

Argument 'BeamNum' defines beam number: If it is 0 - it is first beam, 1 - second and 2 - third.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If it returns zero or negative value - it is an error.

ScopeGetAmpOffset.

Pascal-style declaration:

```
  ScopeGetAmpOffset : function (ScopeHandle : Integer; BeamNum : Integer) : Double; cdecl;
```

C-style declaration:

```
  double (__cdecl * ScopeGetAmpOffset) (int ScopeHandle, int BeamNum);
```

This function returns oscilloscope's vertical (amplitude) offset (bias) of mentioned beam.
Returned value represents vertical (amplitude) offset of zero base for mentioned beam
from middle of oscilloscope's screen height in signal's amplitude units - NOT in pixels of screen.

Argument 'BeamNum' defines beam number: If it is 0 - it is first beam, 1 - second and 2 - third.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

ScopeGetTriggerLevel.

Pascal-style declaration:

```
  ScopeGetTriggerLevel : function (ScopeHandle : Integer; BeamNum : Integer) : Double; cdecl;
```

C-style declaration:

```
  double (__cdecl * ScopeGetTriggerLevel) (int ScopeHandle, int BeamNum);
```

This function returns oscilloscope's amplitude trigger's actuation level of mentioned beam.
Returned value represents amplitude trigger's level for mentioned beam in signal's amplitude units.

Argument 'BeamNum' defines beam number: If it is 0 - it is first beam, 1 - second and 2 - third.
If it is 3 - it is trigger's input for 'External' synchronization.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

ScopeSetTriggerLevel.

Pascal-style declaration:

```
  ScopeSetTriggerLevel : function (ScopeHandle : Integer; BeamNum : Integer; TrigLevel : Double) : Integer; cdecl;
```

C-style declaration:

```
  int (__cdecl * ScopeSetTriggerLevel) (int ScopeHandle, int BeamNum, double TrigLevel);
```

This function sets oscilloscope's trigger's amplitude level of mentioned beam.

Functional argument 'TrigLevel' defines trigger's amplitude actuation level of mentioned beam in signal's amplitude units.

Argument 'BeamNum' defines beam for rescale: If it is 0 - it is first beam, 1 - second and 2 - third.
If it is 3 - it is trigger's input for 'External' synchronization.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If the function works properly - then it returns signed integer value which is
equal to 'ScopeHandle' input argument, otherwise code of error returned and
oscilloscope's trigger's amplitude level will be not changed.

ScopeGetTriggerSourse.

Pascal-style declaration:

```
  ScopeGetTriggerSourse : function (ScopeHandle : Integer) : Integer; cdecl;
```

C-style declaration:

```
  int (__cdecl * ScopeGetTriggerSourse) (int ScopeHandle);
```

This function returns currently active oscilloscope's trigger sourse.

Returned value represents channel number: If it is 0 - it is first beam, 1 - second and 2 - third.
If it is 3 - it is trigger's input for 'External' synchronization.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If returned value is -1 - it is function error. Probably oscilloscope instance with mentioned 'ScopeHandle'
doesn't exist.

ScopeSetTriggerSourse.

Pascal-style declaration:

```
  ScopeSetTriggerSourse : function (ScopeHandle : Integer; BeamNum : Integer) : Integer; cdecl;
```

C-style declaration:

```
  int (__cdecl * ScopeSetTriggerSourse) (int ScopeHandle, int BeamNum);
```

This function sets active oscilloscope's trigger sourse.

Argument 'BeamNum' defines beam for trigger input: If it is 0 - it is first beam, 1 - second and 2 - third.
If it is 3 - it is trigger's input for 'External' synchronization.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If the function works properly - then it returns signed integer value which is
equal to 'ScopeHandle' input argument, otherwise code of error returned and
oscilloscope's trigger sourse will be not changed.

ScopeGetActiveTriggerEdge.

Pascal-style declaration:

```
  ScopeGetActiveTriggerEdge : function (ScopeHandle : Integer) : Integer; cdecl;
```

C-style declaration:

```
  int (__cdecl * ScopeGetActiveTriggerEdge) (int ScopeHandle);
```

This function returns currently active direction of oscilloscope's trigger edge.

Returned value represents sweep trigger mode - if it is zero then trigger is disabled,
if it is 1 - trigger works on rising direction of signal slope,
if it is -1 - trigger works on falling direction of signal slope.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If returned value is -2 - it is function error. Probably oscilloscope instance with mentioned 'ScopeHandle'
doesn't exist.

ScopeSetActiveTriggerEdge.

Pascal-style declaration:

```
  ScopeSetActiveTriggerEdge : function (ScopeHandle : Integer; TriggerEdge : Integer) : Integer; cdecl;
```

C-style declaration:

```
  int (__cdecl * ScopeSetActiveTriggerEdge) (int ScopeHandle, int TriggerEdge);
```

This function sets active direction of oscilloscope's trigger edge or disables the trigger.

Argument 'TriggerEdge' defines new active direction of oscilloscope's trigger edge:
if it is zero then trigger is disabled,
if it is more than zero - trigger works on rising direction of signal slope,
if it is less than zero - trigger works on falling direction of signal slope.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If the function works properly - then it returns signed integer value which is
equal to 'ScopeHandle' input argument, otherwise code of error returned and
oscilloscope's trigger mode will be not changed.

ScopeGetTrigHoldOffSmplsNum.

Pascal-style declaration:

```
  ScopeGetTrigHoldOffSmplsNum : function (ScopeHandle : Integer) : Integer; cdecl;
```

C-style declaration:

```
  int (__cdecl * ScopeGetTrigHoldOffSmplsNum) (int ScopeHandle);
```

This function returns 'Hold OFF' parameter of oscilloscope's trigger.

Returned value represents amout of signal's samples between trigger's latching event
and start of new horizontal sweep of oscilloscope's beams (from left side of oscilloscope's screen).

Its positive value defines 'Holded off' - delayed start of sweep.
Its negative value defines beforehand trigger latch start - forwarded start of sweep.
Forwarded mode of oscilloscope's trigger can be not implemented.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If it returns -2147483647 - it is an error - probably oscilloscope instance with mentioned
'ScopeHandle'doesn't exist.

ScopeSetTrigHoldOffSmplsNum.

Pascal-style declaration:

```
  ScopeSetTrigHoldOffSmplsNum : function (ScopeHandle : Integer; HoldOffVal : Integer) : Integer; cdecl;
```

C-style declaration:

```
  int (__cdecl * ScopeSetTrigHoldOffSmplsNum) (int ScopeHandle, int HoldOffVal);
```

This function sets 'Hold OFF' parameter of oscilloscope's trigger.

Argument 'HoldOffVal' defines amout of signal's samples between trigger's latching event
and start of new horizontal sweep of oscilloscope's beams (from left side of oscilloscope's screen).

Its positive value defines 'Holded off' - delayed start of sweep.
Its negative value defines beforehand trigger latch start - forwarded start of sweep.
Forwarded mode of oscilloscope's trigger can be not implemented.

'ScopeHandle' argument is handle of mentioned oscilloscope instance. This handle is returned
by 'ScopeCreate' function when this instance was created.

If the function works properly - then it returns signed integer value which is
equal to 'ScopeHandle' input argument, otherwise code of error returned and
oscilloscope's trigger 'Hold OFF' parameter will be not changed.

DllGetClassObject, DllCanUnloadNow, DllRegisterServer and DllUnregisterServer functions
support interface to regsvr32.exe MS Windows utility and do not have to be used.
Can be used only if required.

The functional purpose of the visual interface elements of oscilloscope.

The window (form) of oscilloscope contains a group of standard elements.
These and only these elements are the oscilloscope痴 interface elements.

The most graphical field having grid is the main oscilloscope痴 screen.
All channel beams are downloaded here.

Vertical column of numbers is the additional graphical field, displaying the textual
information concerning the vertical (amplitude) signal scale and offset.

Textual field, found on the upper left corner of oscilloscope's screen
displays horizontal sample signal resolution in one cell of the main oscilloscope's
graphical field's marking grid. In other words, it displays the horizontal sweep speed.

Panel located under oscilloscope's graphical field contains all oscilloscope's
managing elements. Our continuing description refers to elements contained on this panel.

Buttons with black vertical arrows - vertical zoom management.
Vertical zoom management is available with the aid of vertical ruler field as well:
move mouse pointer on vertical ruler field, press and hold right mouse button depressed
and move mouse pointer vertically.

Buttons with black horizontal arrows - horizontal zoom management.

Button with '0' (zero) - attributing the vertical scale zero point to the middle
height of the main graphic field.

Buttons with red diagonal arrows - changes the geometrical size of the main graphical
field grid cells (vertically as well as horizontally).

Buttons with red vertical arrows - rolls (shift) the vertical scale and signal up & down.
Vertical rolling management is available with the aid of vertical ruler field as well:
move mouse pointer at vertical ruler field, press and hold left mouse button depressed
and move mouse pointer vertically. Can use mouse痴 scroll wheel.

Buttons with red horizontal arrows - rolls (shift) the signal horizontally.
Can be done by depressing 都hift or 田ontrol buttons and rotating mouse痴 scroll wheel.

Button with picture of hand points to scale - search for the beam. Shifts the vertical
scale & the beam, placing the last beam sample in the middle of graphical field height.
Works for all beams, changing from one to the other with every click of the button.

Button with picture of magnifying glass - returns to the previous vertical & horizontal
zoom scales which are changed by selecting & enlarging of the main graphical field section
by pressing on the right mouse-button.

Textual field 'VALUE:' - textual field designed to obtain the instantaneous value of the
channel's amplitude while moving the mouse cursor in the main graphical field.
Deduces the signal amplitude value in the moment of time corresponding to the horizontal
mouse's cursor position in the oscilloscope graphical field.
It is possible to switch it to other beams by double clicking mouse痴 left button.
Check 'Hex.' check box for hexadecimal value representation.

'Jaggies' check box - changes the graphical style of the beams' representation.
If this element is not selected, the beam is plotted by connecting the signal痴 discreet
points with a single line segment (first order interpolation). If this element is enacted,
the adjacent beam points are connected by vertical & horizontal segments, thus displaying
the beam in "steps" for better viewing of samples.

'Trigger' button puts the oscilloscope in the trigger mode. In this mode the next
horizontal sweep cycle occurs only if the front of the first beam crosses the
amplitude level determined in the textual field on the right of this button.
Current mode of trigger shown by picture on this button: if it is diagonal cross - the trigger
is disabled, if it is an image of rising edge with arrow directed upwards - the trigger
works on positive (rising) slope of signal, if it is an image of falling edge with arrow
directed downwardss - the trigger works on negative (falling) slope of signal.

'for Beam N' or 'for External' button defines the signal source for trigger.

'Manual' check box - It's selection determines the manual trigger set mode - i.e.
only with a single horizontal sweep cycle & only after clicking the 'Wait for next...' button.
In the manual mode 'Wait for next...' button will be disabled until event of trigger
не бывает.

'Auto sweep' check box (can be invisible) - is trigger's mode control also.
Selection of this check box brings oscilloscope into auto sweep trigger mode.
Note that oscilloscope shows not all samples of signal in this trigger mode.
However, all signal samples will be saved in scope痴 buffers

'ScopeAutoSweepDivider' field of ini. file defines decimation factor for auto sweep mode.
The auto sweep mode works in the next style: If trigger did not work out on it's predefined
event during predefined amount of signal samples - then oscilloscope runs one horizontal
sweep screen pass and then comes back to waiting stage until trigger's event or until
next expiration of amount of unshown signal samples.
Amount of unshown signal samples defined as product of value of 'ScopeAutoSweepDivider'

field of ini. file and amount of samples per one screen pass for current horizontal scale of oscilloscope. Auto sweep trigger mode only works if 'Auto sweep' check box is visible, enabled and checked.

Note that ALL signal's samples are recorded in oscilloscope's buffers without data loss (omission) in ALL trigger's modes. This means that REAL signal's data sequence will be restored on the screen after repainting of signal's beams by any of oscilloscope's controls.

'Beams' button commutes screen control (buttons with arrows and so on) between beams.
If this button has caption 'Beams' - then control connected to all visible beams.
Pressing on this button will change caption to 'Beam 0', next pressing - to 'Beam 1', next - to 'Beam 2' and next - again to 'Beams'. Screen control will work for beam having same number as caption: first beam is controlled if caption is 'Beam 0', second - if caption is 'Beam 1', third - if caption is 'Beam 2'.
Graphical field, displaying the textual info concerning the vertical (amplitude) signal scale will change color of digits in accordance with color of currently controlled beam. Offset and zoom of this vertical scale will bring conformity with currently controlled beam. If this button has stage 'Beams', but offsets and/or zoom of beams are different, then vertical scale will show parameters for first beam.
It is very easy to bring all beams to same offset and zoom: when button has stage 'Beams' press 'Zero' (0) button.
If this button has stage 'Beams', and offsets and zoom of beams are equal - then vertical scale will apply to digits color described in the 'INI' file in the section [Oscilloscope] (taking into account suffix) in the field 'OscVrulGenFontColor'.
Switching by 'Beams' button has no effect on:
textual field 'VALUE:',
'Trigger' button and it's textual field,
textual field of horizontal scale,
buttons of horizontal zoom management,
buttons of horizontal rolling management,
'STEPS' check box,
and button with picture of magnifying glass.
Double click of left mouse button when mouse pointer is located on vertical scale ruler will gives same effect like pressing on 'Beams' button.

'Disconnect' button located in lower right corner of oscilloscope's form.
This button disconnects oscilloscope from data stream going through 'ShowNext' DLL exported function. 'ShowNext' function does not affect calling external process. 'ShowNext' function just gives a work-out like dummy (empty) function - does nothing.
After disconnection oscilloscope obtains any additional abilities which are not allowed in the online mode: select gap of signal using left button of mouse, copy selected gap of signal onto clipboard (as text) or save it to text file, select file and read it onto oscilloscope's screen, select area of oscilloscope's screen for zoom using right button of mouse, to measure of oscilloscope graphical performance (if field ShowTestButton=1 in 'INI' file.). Recurring pressure to this button will not allow additional abilities and will bring oscilloscope into online mode again.

'Menu' button is enabled when oscilloscope disconnected from data stream.
Opens main menu of oscilloscope.

'Read from file' button is enabled when oscilloscope disconnected from data stream and file for reading selected by means of 'Open file...' menu item.
Note that just selecting file for reading will not initiate reading of the file. In this situation full path and name of selected file will placed on the caption of oscilloscope's form.
Pressing on 'Read from file' button will clear oscilloscope's signal buffers and will begin reading selected file and loading read data into lines of beams.
Special slowing down of this output is possible through 'ReadDelayPerSample' filed in [Miscellaneous] (with suffix) section in the 'INI' file.
Stated parameter of this field in mS. for one sample of signal.
Recurring pressure on this button will clear oscilloscope's signal buffers and reread file again.

'Test/Performance' button is visible if 'ShowTestButton' filed in the [Oscilloscope] (with suffix) section in the 'INI' file is 1 (one).
This button is enabled when oscilloscope disconnected from data stream.
Runs measurement of graphical operating speed for 10 - 20 seconds. Shows results of test in opening dialog window.

Official web site of the project is: http://www.oscilloscope-lib.com

The web mirror: http://brnstin.googlepages.com

My E_Mail: brnstin@zahav.net.il ; brnstin@cheerful.com