Содержание

Краткое пособие по созданию сценариев Artix Calliseum

Краткое описание

Система Artix Calliseum использует сценарии (скрипты), написанные на языке Pascal. В них можно использовать все стандартные конструкции языка, большинство стандартных функции базовых модулей, а также дополнительные функции по работе с телефонными платами, графическими и голосовыми файлами, по работе с внутренней базой системы и по соединению и выполнению SQL запросов к внешним базам данных.

Сценарии представляют собой текстовые файлы с расширением “PAS”, а также опционально могут иметь текстовый файл с расширением “VAR”, где описываются интерфейсные параметры, которые можно настроить в параметрах диалога в Конфигурации Администратора. Все файлы сценариев должны находиться в подкаталоге Scripts установленной системы. Для написания нового сценария достаточно создать или скопировать новый PAS файл. Вновь созданные файлы сценариев подключаются к Администратору автоматически при каждом открытии окна Конфигурации, а к Менеджеру – после перезагрузки. Компиляция уже имеющихся файлов сценариев происходит в Менеджере звонков в реальном времени, что позволяет одновременно править сценарии и их интерфейсные параметры и сразу же их тестировать.

Сценарии могут работать для входящих и исходящих звонков, при запуске и остановке линии, при запуске и выгрузке менеджера звонков, а также регулярно с некоторым периодом. В сценарии всегда предопределена переменная iRunType, которая указывает, каким образом его запустили, поэтому в одном сценарии можно реализовать разные алгоритмы, в зависимости от того как сценарий был запущен. Разумеется, некоторые функции и переменные актуальны только при определенных типах запусках, например, нельзя проиграть файл в сценарии, если он запущен не для обработки звонка.

Ввиду того, что система организована на потоках, следует знать примерную организацию работы менеджера звонков. Так, при его запуске всегда сначала создается один поток (thread), в контексте которого (контекст системы) работают сценарии, запускаемые однократно при запуске и выгрузке менеджера звонков, а также регулярно запускаемые сценарии. В любом случае сценарии, запускаемые при старте менеджера выполняются раньше, а запускаемые при его выгрузке - позже любого регулярно выполняющегося сценария. Аналогично ситуация обстоит со сценариями при запуске и выгрузке линии и сценариями обработки звонков. Отличие состоит только в том, что они работают не в контексте системы, а в контексте потока одной линии (контекст линии). И точно также сценарии, запускаемые при запуске линии выполняются раньше, а запускаемые при ее выгрузке - позже любого сценария, обрабатывающего звонки.

В Администраторе каждый сценарий рассматривается системой как отдельный диалог, который имеет свой набор параметров и, как и любой стандартный диалог, может виртуально копироваться в системе для использования разных значений интерфейсных параметров.

Написание сценариев

Простейший, ничего не делающий сценарий:

```
Begin
End.
```

Весь сценарий выполняется между конечной парой begin/end как при инициализации паскалевских модулей. Внутри сценария возможно написание любых функций и переменных. В сценарии нельзя использовать стандартные для паскалевского модуля секции UNIT, USES, INTERFACE, IMPLEMENTATION, INITIALIZATION, FINALIZATION. То есть общая структура файла сценария будет такая

```
Var ilocal:integer; {описание локальных переменных модуля}
Function funcA:Boolean;{описание функции 1}
Var ilocal:integer; {описание локальных переменных функции}
Begin
```

```
//.....
Result := True;
End;
Procedure funcB;{описание функции 2}
Begin
//.....
End;
Begin
funcB;//выполнение функции funcB
End.
```

Во всех сценариях можно использовать все средства и возможности современного языка паскаль от Delphi. Также в них предопределены все классы, константы переменные и функции стандартных модулей Delphi - System, Types, Windows, SysUtils, Variants, StrUtils и Classes.

Помимо стандартных, в каждом сценарии предопределены (их не надо объявлять) некоторые дополнительные типы, функции и переменные для реальной работы системы.

Переменные

Переменные, объявленные в теле сценария, вне всех функций являются локальными и валидными в контексте одного выполнения сценария. Объявление глобальных переменных, актуальных в пределах нескольких выполнений сценария, пользователю недоступно, вместо этого в системе предопределены несколько глобальных структур и переменных, которые пользователь может использовать по своему усмотрению, например, для хранения своих глобальных данных.

Предопределенные локальные переменные

Эти предопределенные переменные, как и все локальные, работают в контексте одного выполнения сценария и, поэтому, инициализируются при каждом запуске сценария заново.

```
Var iRunType: TRunType;
```

В этой переменной хранится тип запуска данного сценария, то есть, был ли он запущен во время звонка или, например, при запуске Менеджера.

```
Var iLine: integer;
```

Это переменная, хранящая номер линии текущего звонка и во многих функциях, связанных с внутренней базой и звонками она должна идти как первый параметр. Если сценарий был запущен не на линии или звонке, этот параметр равен «-1».

```
Var bLineResult: Boolean;
```

Это переменная, в которую необходимо записать True, если надо, чтобы считать сценарий успешно выполненным или False, если неудачно. При обработке исходящих звонков значение надо указывать обязательно, поскольку указывает системе, следует ли продолжать звонить этому абоненту. При обработке входящих звонков это значение используется для визуального отображения в свойствах линии в Менеджере звонков

Преопределенные глобальные переменные контекста линии

Все глобальные переменные с областью видимости линии, в том числе и параметры звонка, хранятся в переменной-структуре:

```
cuLine: TCUHandles;
```

Эта структура полностью описывает общие параметры текущей линии и звонка. Эту переменную нельзя использовать (ее адрес будет равен nil) если сценарий работает не на линии или при обработке звонка. Элемент структуры

cuLine.UserVars:TUserVars

предоставляет пользователю возможность работы с глобальными переменными контекста линии. То есть, например, в переменной cuLine.UserVars.Ints[0] можно хранить число обработанных сценарием звонков на каждой линии.

Инициализируются и финализируются эти переменные, как правило, в сценариях, работающих при запуске и выгрузке линий. На каждой линии создается свой экземпляр этой структуры.

Преопределенные глобальные переменные контекста системы

В системе также предопределены глобальные переменные контекста системы, доступные при любом выполнении сценария в любом месте, представленные в единственном экземпляре со временем жизни, равным времени работы менеджера звонков. Хранятся они все в структуре TUserVars:

UserVars: TUserVars

То есть, например, в переменной UserVars.Ints[0] можно хранить число обработанных сценарием звонков всей системы.

Переменные в этой структуре свободны для использования произвольным образом. Обычно они инициализируются и финализируются в сценариях, работающих при запуске и выгрузке менеджера звонков.

Одновременный доступ к глобальным переменным

Ввиду того, что сценарии выполняются в разных потоках и, присутствуют глобальные переменные, может возникнуть проблема одновременного безопасного доступа к ним из разных потоков. Эта проблема решается использованием критических секций, мьютексов и семафоров. Все, необходимые для этого функции и типы Windows зарегистрированы в системе, например структура TRTLCriticalSection и, поэтому, могут свободно использоваться в сценариях.

Когда можно не использовать критические секции?

Глобальные переменные контекста линии доступны и валидны только в потоке линии и поэтому всегда безопасны для использования произвольным образом. Также безопасны глобальные переменные контекста системы простых типов: целые числа, перечисления и множества, ввиду того, что запись и чтение из них одноатомны и производятся в одном такте процессора. Если глобальная переменная контекста системы любого типа используется только в сценариях, запускаемых при запуске и выгрузке менеджера или регулярно, то есть не используемая в контексте линии, то она также безопасна, так как эти запуски проходят только в одном потоке.

Когда обязательно использование критических секций?

Однозначно небезопасны и требуют использование критических секций глобальные нетривиальные переменные контекста системы, используемые в сценариях обработки линий и звонков, например строки, записи, объекты классов. Вообще, использование таких переменных не рекомендуется, вместо них желательно использовать локальные или глобальные переменные контекста линии, без критических секций. Дело в том, что частый одновременный доступ к совместному коду через критические секции может замедлять выполнение сценариев, а значит и всей системы.

Интерфейсные переменные

Интерфейсные переменные это локальные переменные сценария, то есть работающие в контексте одного выполнения сценария и предназначены для удобной настройки начальных значений в Администраторе, не изменяя сам сценарий. В проекции работы системы со стандартными диалогами это означает, что интерфейсные переменные сценария – это, фактически, параметры диалога сценария в Администраторе. Настраивая параметры диалога сценария в Администраторе, вы просто задаете начальное значение, которое получит соответствующая переменная при запуске сценария.

Для создания и хранения этих параметров предназначен специальный файл, имя которого совпадает с именем файла сценария, но имеет расширение VAR. Это текстовый файл, в каждой строке которого, указан один параметр. Характеристики параметра разделяются символом «;», их всего 4:

Название параметра

Это имя параметра, а также имя переменной, имеющей любой удобоваримый для паскаля вид.

Тип данных параметра

это тип данных хранящихся в параметре (переменной), возможные значения:

integer, string, double – стандартные типы pascal

digitalfields. language – фактически тип integer

currency – тип TCurrencyUnits, перечисление всех валют

pwdcheckmode – тип TVMailPwdCheckType, перечисление режимов проверки пароля

vmailusermode – тип TVMailUserMode, перечисление режимов обработки пользователя диалога

Описание параметра

Произвольная текстовая информация, описывающая параметр. Она используется для визуального отображения параметров сценария в Администраторе. Эта информация также доступна в теле сценария через другую переменную, у которой в названии добавлен суффикс «_desc». Например, для интерфейсного параметра ExtensionLength в сценарии будет доступна как глобальная переменная ExtensionLength, так и ExtensionLength_desc.

Значение по умолчанию

Значение параметра по умолчанию, в которое параметр устанавливается при первом использовании в Администраторе. В дальнейшем это значение также используется при нажатии кнопки «По умолчанию» в Конфигурации для восстановления начального значения параметра.

Пример параметра

Строка в VAR файле:

ExtensionLength;integer;Кол-во цифр в доб. номере;3

В этом случае в Администраторе в параметрах сценария появляется параметр «Кол-во цифр в доб. номере», с начальным значением «3». При работе сценария в нем будет доступна локальная переменная ExtensionLength, инициализируемая значением, указанным в Администраторе и переменная типа string ExtensionLength_desc со значением «Кол-во цифр в доб. номере».

Предопределенные параметры

Для любого сценария заранее предопределены (без определения их в VAR файле) следующие интерфейсные параметры, недоступные в теле сценария, указывающие менеджеру звонков, когда следует запускать диалог помимо звонка.

Выполнить один раз при запуске менеджера (тип Boolean) – указывает, должен ли сценарий выполняться при запуске Менеджера

Выполнить один раз при остановке менеджера (тип Boolean) – указывает, должен ли сценарий выполняться при выгрузке Менеджера

Выполнить один раз на линии при ее запуске (тип Boolean) – указывает, должен ли сценарий выполняться при запуске каждой линии

Выполнить один раз на линии при ее остановке (тип Boolean) – указывает, должен ли сценарий выполняться при остановке каждой линии

Выполнять периодически (тип Boolean) – указывает, должен ли сценарий выполняться регулярно. Система выполняет сценарий в отдельном потоке регулярно с определенным интервалом.

Период выполнения (сек) (тип integer) – периодичность в секундах между выполнениями сценария, если включен регулярный запуск

Функции по работе с платами

Данные функции предназначены для обработки звонков и должны использоваться только при работе сценария во время звонка. В других случаях функции не делают ничего.

cmPlayFile – проиграть звуковой файл

Синтаксис:

```
procedure cmPlayFile(iLine: integer; const sName, sDesc: string);
```

функция, проигрывающая файл. Выдает исключение EUserException, если абонент бросил трубку и EFaxException если система услышала сигнал факса.

Параметры:

iLine – Номер линии

sName – проигрываемый звуковой файл, используется как короткое имя, так и длинное. Если путь не указан, файл должен находиться в стандартном подкаталоге системы Voice/Rus

sDesc – описание файла

Пример

```
cmPlayFile(iLine, 'hello.wav', 'Приветствие');
```

cmRecFile – записать звуковой файл

Синтаксис:

```
procedure cmRecFile(iLine: integer; const sName, sDesc: string);
```

функция, записывающая файл. Выдает исключение EUserException, если абонент бросил трубку и EFaxException если система услышала сигнал факса.

Параметры:

iLine – Номер линии

sName – записываемый звуковой файл, используется как короткое имя, так и длинное. Если путь не указан, файл будет находиться в стандартном подкаталоге системы Voice/Rus

sDesc – описание файла

Пример

```
cmRecFile(iLine, 'c:\1.wav', 'Записываемое сообщение');
```

cmGetDigits – принять тоновые цифры

Синтаксис:

```
function cmGetDigits(iLine, Count: integer; var Digits: string; UserTimeOut: integer = -1): Char;
```

функция принимающая тоновые цифры от пользователя. Выдает исключение EUserException, если абонент бросил трубку и EFaxException если система услышала сигнал факса. Возвращает последний символ набранный пользователем.

Параметры:

iLine – Номер линии

Count – Максимальное число цифр, которое может ввести пользователь (0 – без ограничений). В любом случае сигнал # от пользователя завершает прием цифр.

Digits – строка, где находятся принятые цифры

UserTimeOut – таймаут пользователя на ввод цифры (-1 – стандартный, настраиваемый в конфигурации)

Пример

```
Var Digits:string;
cmGetDigits(iLine, 3, Digits);
```

cmGetPasswordDigits – принять тоновые цифры, не показывая их в логе

Синтаксис:

```
function cmGetPasswordDigits(iLine: integer; Count: integer; var Digits: string; UserTimeOut: integer = -1): Char;
```

Функция полностью аналогична cmGetDigits. но не показывает цифры в лог файле.

cmPlayDigits – проиграть число

Синтаксис:

```
procedure cmPlayDigits(iLine: integer; Number: Int64; iPortion: integer = 0);
```

функция, проигрывающая число. Выдает исключение EUserException, если абонент бросил трубку и EFaxException если система услышала сигнал факса.

Параметры:

iLine – Номер линии

Number – проигрываемое число

iPortion – порция, на которое делится число (0 – число читается как есть).

Пример

```
cmPlayDigits(iLine, strtointdef(Digits, 0), 0);
```

cmSendFax – отправить факс

Синтаксис:

```
function cmSendFax(iLine: integer; const sFaxFiles: string): Boolean;
```

функция, передающая факс. Возвращает True, если факс успешно передался и False, если нет. Исключений нет.

Параметры:

iLine – Номер линии

sFaxFiles – список графических файлов, разделенных «;». Используются как короткие имена, так и длинные. Если путь не указан, файл должен находиться в стандартном подкаталоге системы OutBox

Пример

```
cmSendFax(iLine, 'c:\test.tif');
```

cmRecFax – принять факс

Синтаксис:

```
function cmRecFax(iLine: integer; const sFaxFile: string): Boolean;
```

функция, принимающая факс. Возвращает True, если факс успешно пирнят и False, если нет. Исключений нет.

Параметры:

iLine – Номер линии

sFaxFile – один факсимильный файл. Используется как короткое имя, так и длинное. Если путь не указан, файл должен находиться в стандартном подкаталоге системы InBox

Пример

```
cmRecFax(iLine, 'c:\test.tif');
```

cmClrDigBuf – очистить буфер принятых цифр

Синтаксис:

```
procedure cmClrDigBuf(iLine: integer);
```

Функция, очищающая буфер введенных цифр. Часто нужна перед проигрыванием или записыванием файлов для очистки буфера цифр. Иначе если в буфере остаются цифры – следующий файл не проигрывается, а пропускается.

Параметры:

iLine – Номер линии

Пример

```
cmClrDigBuf(iLine);
```

cmPlayMoney – проиграть деньги

Синтаксис:

procedure cmPlayMoney(iLine: integer; Currency: TCurrencyUnits; Value: Double);

функция, проигрывающая деньги. Выдает исключение EUserException, если абонент бросил трубку и EFaxException если система услышала сигнал факса.

Параметры:

iLine – Номер линии

Currency – тип валюты

Value – проигрываемое число

Пример

cmPlayMoney(iLine, CU_RUR, 245.34);

cmPlayDate – проиграть дату, первая функция

Синтаксис:

procedure cmPlayDate(iLine: integer; Year, Month, Day: integer; Mode: TDateCase = DATE_CASE_GENITIVE; use_yyyy: Boolean = True); overload;

функция, проигрывающая дату. Выдает исключение EUserException, если абонент бросил трубку и EFaxException если система услышала сигнал факса.

Параметры:

iLine – Номер линии

Year, Month, Day – год месяц и число. Если значение равно -1, оно не проигрывается

Mode – падеж проигрываемой даты. Может быть родительный или именительный падеж.

use_yyyy – использовать или нет 4 цифры в годе (если False - проигрываются две цифры)

Пример

cmPlayDate(iLine, 2012, 3, 12, DATE_CASE_NOMINATIVE, True);

cmPlayDate – проиграть дату, вторая функция

Синтаксис:

procedure cmPlayDate(iLine: integer; DateTime: TDateTime; Mode: TDateCase = DATE_CASE_GENITIVE; useyear: Boolean = False; use_yyyy: Boolean = True); overload;

Функция полностью аналогична предыдущей, но использует для проигрывания стандартный тип TDateTime.

Параметры дополнительные:

Useyear – параметр указывает проигрывать, или нет год

Пример

cmPlayDate(iLine, Now, DATE_CASE_GENITIVE, True, True);

cmPlayTime – проиграть время, первая функция

Синтаксис:

procedure cmPlayTime(iLine: integer; hh: integer; mm: integer = -1; ss: integer = -1); overload;

функция, проигрывающая время. Выдает исключение EUserException, если абонент бросил трубку и EFaxException если система услышала сигнал факса.

Параметры:

iLine – Номер линии

hh, mm, ss – час минуты и секунды. Если значение равно -1, оно не проигрывается

Пример

cmPlayTime(iLine, 12, 23, 45);

cmPlayTime – проиграть время, вторая функция

Синтаксис:

procedure cmPlayTime(iLine: integer; DateTime: TDateTime; Use_hh, Use_mm, Use_ss: Boolean); overload;

Функция полностью аналогична предыдущей, но использует для проигрывания стандартный тип TDateTime.

Параметры дополнительные:

Use_hh, Use_mm, Use_ss – параметр указывает проигрывать, или нет час, минуты, секунды.

Пример

cmPlayTime(iLine, Now, True, True, True);

cmPlayTTSTextFile – проиграть текстовый файл методом текст-в-речь

Синтаксис:

procedure cmPlayTTSTextFile(iLine: integer; const sTextFile, sTextFileDesc: string);

функция, проигрывающая текстовый файл методом Текст-в-речь. Параметр Текст-в-речь в Конфигурации Администратор должен быть включен и настроен. Выдает исключение EUserException, если абонент бросил трубку и EFaxException если система услышала сигнал факса.

Параметры:

iLine – Номер линии

sTextFile– проигрываемый текстовый файл, используется как короткое имя, так и длинное. Если путь не указан, файл должен находиться в стандартном подкаталоге системы Texts

sTextFileDesc – описание файла

Пример

cmPlayTTSTextFile(iLine, 'text.txt', 'TTS text file');

cmPlayTTSText – проиграть текст методом текст-в-речь

Синтаксис:

procedure cmPlayTTSText(iLine: integer; const sText: string);

Функция полностью аналогична предыдущей, но использует для проигрывания обычный текст.

Параметры дополнительные:

sText – проигрываемый текст

Пример

```
cmPlayTTSText(iLine, 'Hello');
```

Функции по работе с внутренней базой

Данная группа функций предназначена для работы с внутренней базой системы. Эти функции доступны при запуске сценария в любом месте. Все функции возвращают True, при успешном завершении и False при неудачном.

cuGetUserByExtension – получить пользователя по внутреннему номеру

Синтаксис:

function cuGetUserByExtension(iLine: integer; var cuUser: TCUUser): Boolean;

функция, возвращающая пользователя Calliseum по его полю внутренний номер (Extension) и результат (True – пользователь найден, False – нет).

Параметры:

iLine – Номер линии

cuUser – данные пользователя системы

Пример

```
Var cuUser: TCUUser;
 cuUser.UserExtension := ‘12345’;
 cuGetUserByExtension(iLine, cuUser);
```

cuGetUserByLogin – получить пользователя по логину

Синтаксис:

function cuGetUserByLogin(iLine: integer; var cuUser: TCUUser): Boolean;

Функция полностью аналогична предыдущей, но использует для входа значение логина.

Пример

```
Var cuUser: TCUUser;
 cuUser.UserLogin := ‘12345’;
 cuGetUserByLogin (iLine, cuUser);
```

cuGetUserByLogin – получить пользователя по счёту

Синтаксис:

function cuGetUserByAccount(iLine: integer; var cuUser: TCUUser): Boolean;

Функция полностью аналогична предыдущей, но использует для входа значение счета.

Пример

```
Var cuUser: TCUUser;
 cuUser.Account := ‘12345’;
 cuGetUserByAccount (iLine, cuUser);
```

cuAddIncomingMessageByExtension – добавить сообщение по внутреннему номеру

Синтаксис:

```
function cuAddIncomingMessageByExtension(iLine: integer; const sUserExtension, sFiles: string): Boolean;
```

функция, добавляющая сообщение пользователю Calliseum по его полю внутренний номер (Extension) и возвращающая результат (True – сообщение добавлено, False – нет).

Параметры:

iLine – Номер линии

sUserExtension – внутренний номер пользователя

sFiles – список файлов, разделенных «;». Используются как короткие имена, так и длинные. Если путь не указан, файл должен находиться в стандартном подкаталоге системы InBox

Пример

```
cuAddIncomingMessageByExtension(iLine, ‘12345’, 'c:\test.tif');
```

cuAddIncomingMessageByLogin – добавить сообщение по логину

Синтаксис:

```
function cuAddIncomingMessageByLogin(iLine: integer; const sUserLogin, sFiles: string): Boolean;
```

Функция полностью аналогична предыдущей, но использует для входа значение логина пользователя.

Параметры дополнительные:

sUserLogin – логин пользователя

Пример

```
cuAddIncomingMessageByLogin(iLine, ‘12345’, 'c:\test.tif');
```

cuAddMail – добавить исходящее Email сообщение

Синтаксис:

```
function cuAddMail(iLine: integer; const sUserLogin, sAddress, sSubject, sBodyText, sAttachs: string): Boolean;
```

функция, добавляющая исходящее Email сообщение и возвращающая результат (True – сообщение добавлено, False – нет).

Параметры:

iLine – Номер линии

sUserLogin – от кого сообщение (не обязательно)

sAddress – адрес «Кому»

sSubject – тема сообщения

sBodyText – тело сообщения

sAttachs – список вложенных файлов, разделенных «;». Используются как короткие имена, так и длинные. Если путь не указан, файл должен находиться в стандартном подкаталоге системы Voice/Rus, Images или Texts в зависимости от типа файлов

Пример

```
cuAddMail(iLine, '209', 'test@artix.ru', 'test subject', 'test body', 'greet.wav');
```

cuAddSMS – добавить исходящее SMS сообщение

Синтаксис:

```
function cuAddSMS (iLine: integer; const sUserLogin, sPhone, sText: string): Boolean;
```

функция, добавляющая исходящее SMS сообщение и возвращающая результат (True – сообщение добавлено, False – нет).

Параметры:

iLine – Номер линии

sUserLogin – от кого сообщение (не обязательно)

sPhone – кому передать сообщение

sText – текст сообщения

Пример

```
cuAddSMS(iLine, '209', '+79037116005', 'Hello');
```

cuAddOutboundMessage – добавить исходящее сообщение на телефон

Синтаксис:

```
function cuAddOutboundMessage(iLine: integer; iMessageType: TMessageType; const sUserLogin, sPhone, sFiles: string): Boolean;
```

функция, добавляющая исходящее сообщение для оповещения на телефон и возвращающая результат (True – сообщение добавлено, False – нет).

Параметры:

iLine – Номер линии

iMessageType – тип сообщения

sUserLogin – от кого сообщение (не обязательно)

sPhone – кому передать сообщение

sFiles – список файлов, разделенных «;». Используются как короткие имена, так и длинные. Если путь не указан, файл должен находиться в стандартном подкаталоге системы Voice/Rus, Images или Texts в зависимости от типа файлов

Пример

```
cuAddOutboundMessage(iLine, MT_VOICE, '209', '209', 'greet.wav');
```

Функции по работе с графическими файлами

Здесь описаны функции по работе с любыми графическими файлами (формат Adobe PDF также считается графическим). Тип файлов определяется только по расширению имени файла. Во всех функциях используются только длинные имена файлов. Эти функции доступны при запуске сценария в любом месте. Все функции возвращают True, при успешном завершении и False при неудачном.

Предопределенные константы

DEFAULT_WHITE_TOLERANCE = 0; – баланс белого по умолчанию

DEFAULT_BLACK_TOLERANCE = 5; – баланс черного по умолчанию

Image2Image – преобразовать список графических файлов

Синтаксис:

function Image2Image(sInputFiles: string; const sOutputFiles: string; mergetype: TMergeImageType = MultiPage; compression: TTiffCompressType = COMPRESSION_DEFAULT; XResolution: integer = 0; YResolution: integer = 0; method: TDitherType = DefaultDither; ToleranceWhite: integer = DEFAULT_WHITE_TOLERANCE; ToleranceBlack: integer = DEFAULT_BLACK_TOLERANCE; pageunits: TPageUnits = uPixel; pageheight: double = 0; piDescFilesCount: PInteger = nil): Boolean;

функция, конвертирующая графический файл

Параметры:

sInputFiles – список исходных файлов, разделенных «;», которые необходимо конвертировать.

sOutputFiles – имя преобразованного файла. Если предполагается появление нескольких файлов, например, при конвертировании многостраничного TIF в одностраничные JPEG, имя преобразованного файла должно представлять строку-формат для функции sprintf с обязательным полем %d, куда будут подставляться порядковые номера файлов.

Пример названий для создания множественных файлов:

«c:\abc%03d.jpg», преобразованные файлы будут иметь вид abc000.jpg, abc001.jpg,

«c:\abc%d», преобразованные файлы будут иметь вид abc0.jpg, abc1.jpg,

Mergetype – общий тип преобразованного файла

Compression – компрессия преобразованного файла (только для TIFF формата)

XResolution, YResolution – горизонтальное и вертикальное разрешение преобразованного файла (только для форматов, имеющих разрешение)

Method – способ преобразования графического файла (только для цветных и серых рисунков)

ToleranceWhite, ToleranceBlack – балансы белого и черного, используемые при преобразовании цветных и серых изображений в черно-белые (возможные значение 0-10)

Pageunits – единицы измерения при переразбитии файла на страницы (только для mergetype = AdjustedMultiPage)

Pageheight – длина каждой страницы переразбитого файла (только для mergetype = AdjustedMultiPage)

piDescFilesCount – указатель на integer, в котором будет храниться число получившихся файлов

Примеры

Image2Image('c:\test.tif','c:\test2.tif', MultiPage, MMR, 300, 300); – поменять разрешение и компрессию TIFF файла

Image2Image('c:\test.tif','c:\test%02d.jpg'); – преобразовать TIFF файл в несколько JPEG файлов

ImageSplitPage – разбить одну страницу графического файла на две

Синтаксис:

```
function ImageSplitPage(const sInputFile, sOutputFiles: string; iPage: integer; dPortion: Double): Boolean;
```

функция, разбивающая страницу в исходном файле на две

Параметры:

sInputFile – исходный файл, в котором необходимо разбить страницу

sOutputFiles – имя преобразованного файла (см Image2Image)

iPage – номер страницы, которую надо разбить (нумерация начинается с «1»)

dPortion – доля первой половинки разбитой страницы (если 0.5 – страница делится пополам)

Пример

```
ImageSplitPage('c:\test.tif', 'c:\test2.tif', 1, 0.5);
```

ImageMergePages – склеить несколько страниц графического файла в одну

Синтаксис:

```
function ImageMergePages(const sInputFile, sOutputFile: string; iStartPage, iEndPage: integer): Boolean;
```

функция, склеивающая несколько страниц идущих подряд в исходном файле в одну

Параметры:

sInputFile – исходный файл, в котором надо склеить страницы

sOutputFile – имя преобразованного файла (см Image2Image)

iStartPage – номер первой склеиваемой страницы (нумерация начинается с «1»)

iEndPage – номер последней склеиваемой страницы

Пример

```
ImageMergePages('c:\test.tif', 'c:\test2.tif', 1, 2);
```

ImageDeletePages – удалить некоторые страницы графического файла

Синтаксис:

```
function ImageDeletePages(const sInputFile, sOutputFile, sPages: string): Boolean;
```

функция, удаляющая несколько страниц в исходном файле

Параметры:

sInputFile – исходный файл, в котором надо удалить страницы

sOutputFile – имя преобразованного файла (см Image2Image)

sPages – список удаляемых страниц, разделенных «;» (нумерация начинается с «1»)

Пример

```
ImageDeletePages ('c:\test.tif', 'c:\test2.tif', '1');
```

ImageRotatePage – повернуть страницу графического файла на 180º

Синтаксис:

function ImageRotatePage(const sInputFile, sOutputFile: string; iPage: integer): Boolean;

функция, поворачивающая одну страницу на 180º в исходном файле

Параметры:

sInputFile – исходный файл, в котором надо повернуть страницу

sOutputFile – имя преобразованного файла (см Image2Image)

iPage – номер удаляемой страницы (нумерация начинается с «1»)

Пример

ImageRotatePage('c:\test.tif', 'c:\test2.tif', 1);

ImageExtractPages – извлечь некоторые страницы из списка графических файлов

Синтаксис:

function ImageExtractPages(sInputFiles: string; const sOutputFiles, sPages: string; piDescFilesCount: PInteger = nil): Boolean;

функция, извлекающая некоторые страницы из списка исходных файлов

Параметры:

sInputFile – список исходных файлов, разделенных «;», из которых надо извлечь страницы

sOutputFile – имя преобразованного файла (см Image2Image)

sPages – список извлекаемых страниц, разделенных «;» (нумерация начинается с «1»)

piDescFilesCount – число полученных файлов (см Image2Image)

Пример

ImageExtractPages('c:\test.tif', 'c:\test2.tif', '1');

Text2Image – преобразовать текст в графический файл

Синтаксис:

function Text2Image(const sText, sOutputFile: string; XResolution, YResolution: integer; compression: TTiffCompressType = COMPRESSION_DEFAULT; method: TDitherType = DefaultDither; ToleranceWhite: integer = DEFAULT_WHITE_TOLERANCE; ToleranceBlack: integer = DEFAULT_BLACK_TOLERANCE): Boolean;

функция, генерирующая графический файл из текста

Параметры:

sText – текст, из которого надо генерировать графический файл

sOutputFile, XResolution, YResolution, compression, method, ToleranceWhite, ToleranceBlack –параметры полученного файла (см Image2Image)

Пример

Text2Image('Тест', 'c:\test2.tif', 200, 200);

TextOnImagePage – поместить текст в имеющийся графический файл

Синтаксис:

function TextOnImagePage(const sText, sInputFile, sOutputFile: string; iPage: integer; textlocation: TTextOnImageLocation): Boolean;

функция, помещающая текст в графический файл

Параметры:

sText – текст, который надо поместить

sInputFile – исходный файл, в который надо поместить текст

sOutputFile – имя преобразованного файла (см Image2Image)

iPage – номер страницы, на которую помещается текст (нумерация начинается с «1»)

textlocation – положение текста на странице

Пример

TextOnImagePage('Тест', 'c:\test.tif', 'c:\test2.tif', 1, topcenter);

PrintImages – распечатать список графических файлов

Синтаксис:

function PrintImages(const sImageFiles, sPrinter: string; const sPages: string): Boolean;

функция, распечатывающая графические файлы на принтер

Параметры:

sImageFiles – список печатаемых файлов, разделенных «;»

sPrinter – имя принтера

sPages – список печатаемых страниц, разделенных «;» (нумерация начинается с «1»). Пустая строка – все страницы.

Пример

PrintImages('c:\test.tif', 'Adobe PDF', '');

ExtractFirstPage – извлечь первую страницу из списка графических файлов

Синтаксис:

function ExtractFirstPage(const sInputFiles: string): string;

функция, извлекающая первую страницу из списка исходных файлов и возвращающая имя файла с первой страницей

Параметры:

sInputFiles – список исходных файлов, разделенных «;»

Пример

Var sfirstpage:string;

…

sfirstpage := ExtractFirstPage ('c:\test.tif');

GetImagesPageCount – получить число страниц списка графических файлов

Синтаксис:

function GetImagesPageCount(sInputFiles: string): integer;

функция, возвращающая общее число страниц списка исходных файлов

Параметры:

sInputFiles – список исходных файлов, разделенных «;»

Пример

GetImagesPageCount ('c:\test.tif');

GetImageResolution – получить разрешение графического файла

Синтаксис:

function GetImageResolution(const sInputFile: string; out iXResolution, iYResolution: integer): Boolean;

функция, показывающая разрешение исходного файла

Параметры:

sInputFile – исходный файл

iXResolution, iYResolution – горизонтальное и вертикальное разрешение файла

Пример

```
var
iX, iY: integer;
…
GetImageResolution('c:\test.tif', iX, iY);
```

GetImageCompression – получить компрессию TIFF файла

Синтаксис:

function GetImageCompression(const sInputFile: string; out compression: TTiffCompressType): Boolean;

функция, показывающая компрессию исходного TIFF файла

Параметры:

sInputFile – исходный TIFF файл

compression – компрессия исходного TIFF файла

Пример

```
var
iComp:TTiffCompressType;
…
GetImageCompression ('c:\test.tif', iComp);
```

Image2Text – распознать графический файл (OCR)

Синтаксис:

function Image2Text(const sInputFile, sOutputFile, sLangPrefix, sPages: string; encoding: TOCRTextEncoding = ENCODING_UTF16): Boolean;

функция, генерирующая текст из графического файла

Параметры:

sInputFile – исходный графический файл

sOutputFile – имя сгенерированного текстового файла

sLangPrefix – язык, используемый для распознавания. Языки – это файлы, хранящиеся в подкаталоге системы TessData, их имена имеют вид «префикс.traineddata», например «rus.traineddata». Для указания нужного языка необходимо использовать префикс этого файла. В системе имеются два языка – русский и английский, но поддерживаемых языков значительно больше, их можно скачать с сайта Тессеракта - http://code.google.com/p/tesseract-ocr/downloads/list и скопировать в каталог TessData – менеджер звонков их автоматически подхватит после перезапуска.

sPages – список распознаваемых страниц, разделенных «;» (нумерация начинается с «1»). Пустая строка – все страницы.

encoding – кодировка распознанного текстового файла

Пример

Image2Text('c:\test.tif', 'c:\test.txt', 'rus', '', ENCODING_UTF16);

Функции по работе со звуковыми файлами

Здесь описаны функции по работе со звуковыми файлами, используемыми системой. Тип файлов определяется только по расширению имени файла, в системе используются расширения WAV (обычный звуковой файл Windows) и MP3 (звуковой файл с кодировкой MPEG). Во всех функциях используются только длинные имена файлов. Эти функции доступны при запуске сценария в любом месте. Все функции возвращают True, при успешном завершении и False при неудачном.

Предопределенные константы

Форматы звуковых файлов

CM_AUDIO_Unknown = $10000000; – неизвестный формат

CM_AUDIO_Unsupported = $20000000; – неподдерживаемый формат

CM_AUDIO_BestFormat = $00000000; – наилучший формат для текущего набора плат

CM_AUDIO_8bit_6kHz_aLaw = $00000001; – формат ALaw, 8 Bit, 6 kHz, Mono

CM_AUDIO_8bit_8kHz_aLaw = $00000002; – формат ALaw, 8 Bit, 8 kHz, Mono

CM_AUDIO_8bit_11kHz_aLaw = $00000004; – формат ALaw, 8 Bit, 11 kHz, Mono

CM_AUDIO_8bit_6kHz_uLaw = $00000008; – формат uLaw, 8 Bit, 6 kHz, Mono

CM_AUDIO_8bit_8kHz_uLaw = $00000010; – формат uLaw, 8 Bit, 8 kHz, Mono

CM_AUDIO_8bit_11kHz_uLaw = $00000020; – формат uLaw, 8 Bit, 11 kHz, Mono

CM_AUDIO_8bit_6kHz_PCM = $00000040; – формат PCM, 8 Bit, 6 kHz, Mono

CM_AUDIO_8bit_8kHz_PCM = $00000080; – формат PCM, 8 Bit, 8 kHz, Mono

CM_AUDIO_8bit_11kHz_PCM = $00000100; – формат PCM, 8 Bit, 11 kHz, Mono

CM_AUDIO_16bit_6kHz_PCM = $00000200; – формат PCM, 16 Bit, 6 kHz, Mono

CM_AUDIO_16bit_8kHz_PCM = $00000400; – формат PCM, 16 Bit, 8 kHz, Mono

CM_AUDIO_16bit_11kHz_PCM = $00000800; – формат PCM, 16 Bit, 11 kHz, Mono

Voice2Voice – преобразовать звуковой файл

Синтаксис:

function Voice2Voice(const sSource, sDestination: string; iDestinationFormat: integer): Boolean;

функция, преобразующая звуковой файл из одного формата в другой

Параметры:

sSource – исходный звуковой файл. Можно использовать несколько файлов, разделенных «;», в этом случае преобразованный файл будет сначала склеен в один

sDestination – имя преобразованного звукового файла

iDestinationFormat – формат преобразованного файла

Пример

```
Voice2Voice('c:\test.wav ', 'c:\test2.wav', CM_AUDIO_16bit_11kHz_PCM);
```

VoiceGetFileFormat – получить формат звукового файла

Синтаксис:

function VoiceGetFileFormat (const sSource: string): integer;

функция, возвращающая формат исходного WAV файла

Параметры:

sInputFile – исходный WAV файл

Пример

```
Var iFormat:integer;
…
iFormat :=VoiceGetWavFormat('c:\test.wav');
```

VoiceGetDuration – получить продолжительность звукового файла

Синтаксис:

function VoiceGetDuration(const sSource: string): double;

функция, возвращающая продолжительность исходного звукового файла

Параметры:

sSource – исходный звуковой файл

Пример

```
AddLog(iLine, 'I', floattostrf(VoiceGetDuration('c:\test.wav'),ffFixed,10,2)+' sec');
```

Общие функции

Здесь описаны некоторые вспомогательные функции.

AddLog – добавить строку в лог-файл

Синтаксис:

procedure AddLog(iLine: integer; const Mode, sLog: string);

функция, добавляющая строку в лог файл callman.log.

Параметры:

iLine – Номер линии

Mode – тип событие (‘E’ – ошибка, ‘W’ – предупреждение, ‘I’ – информация)

sLog – строка

Пример

AddLog(iLine, 'I', ‘Привет’);

Работа с внешней базой данных

Для работы с внешними базами данных используются два предопределенных класса: TDBConnection - для создания соединения и выполнения запросов, только изменяющих данные и TDBQuery - для выполнения любых запросов.

TDBConnection – класс, определяющий соединение с внешней базой данных

Класс, определяющий и устанавливающий соединение с сервером внешней базы данных.

Методы и свойства:

TDBConnection.Create – конструктор класса TDBConnection

Синтаксис:

constructor Create(Provider: TDBProvider; const sDatabaseName, sUserName, sUserPassword, sServer, sParameters: string; iPort: integer = 0);

Параметры:

Provider – тип сервера

sDatabaseName – имя или файл базы данных

sUserName – логин пользователя базы данных. Если он не нужен – оставить пустым.

sUserPassword – пароль пользователя базы данных. Если он не нужен – оставить пустым.

sServer – адрес или имя внешнего сервера базы данных. Если он не нужен или локальный – оставить пустым.

sParameters – дополнительные параметры, специфические для конкретного сервера базы данных, разделенных «;». Если они не нужны – оставить пустым. Полный список параметров можно получить функцией GetParameters.

iPort – порт внешнего сервера базы данных. Если он не нужен или по умолчанию – оставить нулевым.

Пример

```
Var db:TDBConnection;
...
db := TDBConnection.Create(DB_MSSQL, 'calliseum', 'sa', '', '', 0);
```

TDBConnection.GetParameters – получить список параметров базы данных

Функция возвращает полный список всех свойств данного соединения вместе с их значениями по умолчанию. Параметры и их значения отделяются знаком «=», разные параметры разделяются знаком «;».

Синтаксис:

```
function GetParameters: string;
```

Параметры:

Нет

Пример

```
var db:TDBConnection;
…
db := TDBConnection.Create(DB_MSSQL, 'calliseum', 'sa', '', '', 0);
AddLog(iLine, 'I', 'Specific database server options: ' + db.GetParameters);
```

TDBConnection.Open – открыть соединение к базе данных

Процедура открывает и устанавливает соединение с внешней базой данных. Для любых действий с данными, база должна быть открыта.

Синтаксис:

```
procedure Open;
```

Параметры:

Нет

Пример

```
Var db:TDBConnection;
…
db.Open;
```

TDBConnection.Close – закрыть соединение к базе данных

Процедура закрывает соединение с внешней базой данных. При удалении объекта вызывается автоматически.

Синтаксис:

```
procedure Close;
```

Параметры:

Нет

Пример

```
Var db:TDBConnection;
…
db.Close;
```

TDBConnection.ExecSQL – выполнить SQL, изменяющий данные

Функция выполняет SQL запрос, изменяющий данные, например, INSERT или UPDATE

Синтаксис:

```
function ExecSQL(const Text: string; const Params: array of variant): variant;
```

Функция возвращает либо вариантный Null, либо значение out параметра Result, если он присутствует.

Параметры:

Text – текст SQL запроса, параметры указываются через двоеточие.

Params – набор вариантных значений параметров

Пример

```
Var db:TDBConnection;
...
db.ExecSQL('INSERT INTO Config (User_login, Extension) VALUES (:i1, :i2)',['Ivanov', 234]);
```

TDBConnection.ExecSQLEx – выполнить SQL, изменяющий данные

Функция выполняет SQL запрос, изменяющий данные, например, INSERT или UPDATE. Функция аналогична ExecSQL, но параметры передаются по другому - парами.

Синтаксис:

```
function ExecSQLEx (const Text: string; const Params: array of variant): variant;
```

Функция возвращает либо вариантный Null, либо значение out параметра Result, если он присутствует.

Параметры:

Text – текст SQL запроса, параметры указываются через двоеточие.

Params – набор вариантных имен и значений параметров, разделенных попарно, сначала имя параметра, потом его значение. Параметры могут инициализироваться в произвольном порядке.

Пример

```
Var db:TDBConnection;
...
db.ExecSQLEx('INSERT INTO Config (User_login, Extension) VALUES (:i1, :i2)', ['i1', 'Ivanov', 'i2', 234]);
```

TDBConnection.ExecProc – выполнить встроенную процедуру

Функция выполняет встроенную процедуру

Синтаксис:

```
function ExecProc(const Name: string; const Params: array of variant): variant;
```

Функция возвращает вариантный результат выполнения встроенной процедуры

Параметры:

Name – имя встроенной процедуры

Params – набор вариантных значений параметров

Пример

```
Var db:TDBConnection;
…
db.ExecProc('Make_sum', [12, 234]);
```

TDBConnection.ExecProcEx – выполнить встроенную процедуру

Функция выполняет встроенную процедуру. Функция аналогична ExecProc, но параметры передаются по другому - парами.

Синтаксис:

```
function ExecProcEx (const Name: string; const Params: array of variant): variant;
```

Функция возвращает вариантный результат выполнения встроенной процедуры

Параметры:

Name – имя встроенной процедуры

Params – набор вариантных значений параметров, разделенных попарно, сначала имя параметра, потом его значение. Параметры могут инициализироваться в произвольном порядке.

Пример

```
Var db:TDBConnection;
…
db.ExecProcEx('Make_sum', ['i1', 12, 'i2', 234]);
```

TDBConnection.StartTransaction – начать транзакцию

Процедура начинает транзакцию

Синтаксис:

```
procedure StartTransaction;
```

Параметры:

Нет

Пример

```
Var db:TDBConnection;
…
db.StartTransaction;
```

TDBConnection.Commit – завершить транзакцию

Процедура завершает транзакцию, все изменения записываются в базу

Синтаксис:

```
procedure Commit;
```

Параметры:

Нет

Пример

```
Var db:TDBConnection;
…
db.Commit;
```

TDBConnection.Rollback – отменить транзакцию

Процедура отменяет транзакцию и все сделанные изменения

Синтаксис:

```
procedure Rollback;
```

Параметры:

Нет

Пример

```
Var db:TDBConnection;
…
db.Rollback;
```

TDBConnection.Connected – есть ли соединение?

Свойство, показывающее, установлено ли соединение с внешней базой данных

Синтаксис:

```
property Connected: Boolean;
```

Пример

```
Var db:TDBConnection;
…
if db.Connected then AddLog(iLine, 'I', 'Database is connected');
```

TDBQuery – класс для выполнения любых SQL запросов

Класс, предназначенный для выполнения любых SQL запросов.

TDBQuery.Create – конструктор класса TDBQuery

Синтаксис:

```
constructor Create(DBConnection: TDBConnection; const sSQLText: string);
```

Параметры:

DBConnection – объект установленного соединения

Пример

```
var
db: TDBConnection;
```

```
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA', 'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
```

TDBQuery.Open – выполнить SQL, возвращающий данные (выборку)

Процедура выполняет SQL запрос, возвращающий данные (выборку), например, SELECT

Синтаксис:

```
procedure Open(const Text: string; const Parameters: array of variant);
```

Параметры:

Text – текст SQL запроса, параметры указываются через двоеточие.

Parameters – набор вариантных значений параметров

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA', 'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
```

TDBQuery.Close – закрывает SQL запрос

Процедура закрывает текущий запрос и позволяет выполнить следующий запрос

Синтаксис:

```
procedure Close;
```

Параметры:

Нет

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA', 'masterkey', '', 0);
```

```
db.Open;
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
dq.Close;
```

TDBQuery.Next – переход к следующей записи

Процедура переводит курсор базы на следующую запись выборки

Синтаксис:

```
procedure Next;
```

Параметры:

Нет

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA',
'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
dq.Next;
```

TDBQuery.Prior – переход к предыдущей записи

Процедура переводит курсор базы на предыдущую запись выборки

Синтаксис:

```
procedure Prior;
```

Параметры:

Нет

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA',
'masterkey', '', 0);
db.Open;
```

```
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
dq.Prior;
```

TDBQuery.First – переход на первую запись

Процедура переводит курсор базы на первую запись выборки

Синтаксис:

```
procedure First;
```

Параметры:

Нет

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA',
'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
dq.First;
```

TDBQuery.Last – переход на последнюю запись

Процедура переводит курсор базы на последнюю запись выборки

Синтаксис:

```
procedure Last;
```

Параметры:

Нет

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA',
'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
```

```
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
dq.Last;
```

TDBQuery.GetValues – получить данные записи

Процедура передает одномерный вариантный массив со всеми данными текущей записи

Синтаксис:

```
procedure GetValues(out Values: variant);
```

Параметры:

Values – возвращаемый набор всех значений текущей записи в виде одномерного массива

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
cb: integer;
s: string;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA', 'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
dq.GetValues(vt);
s := '';
for cb := 0 to VarArrayHighBound(vt, 1) do
if not VarIsNull(vt[cb]) then s := s + ' ' + vt[cb];
AddLog(iLine, 'I', 'All data from record: '+s);
```

TDBQuery.GetValue – получить значение одного поля по его имени

Функция выдает значение одного поля в записи по его имени.

Синтаксис:

```
function GetValue(const FieldName: string):Variant;
```

Функция возвращает вариантное значение поля. Если поле не найдено, функция возвращает вариантный Null.

Параметры:

FieldName – имя поля

Пример

```
var
```

```
db:TDBConnection;
dq: TDBQuery;
vt: Variant;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA', 'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
if not VarIsNull(dq.GetValue('Extension')) then AddLog(iLine, 'I', 'Value found: ' + dq.GetValue('Extension'));
```

TDBQuery.Lookup – получить данные по условию

Функция находит данные в выборке по условию. Перехода курсора не происходит.

Синтаксис:

```
function Lookup(const KeyFields: string; const KeyValues: variant; const ResultFields: string): variant;
```

Функция возвращает вариантное значение, если результирующее поле одно, или одномерный вариантный массив, если поисковых полей несколько. Запись считается найденной, если поля из KeyFields имеют значения KeyValues. Если запись не найдена, функция возвращает вариантный Null.

Параметры:

KeyFields – набор полей (поисковых полей), по которым идет проверка условия, разделенных «;»

KeyValues – вариантное значение, если поисковое поле одно, или одномерный вариантный массив, если поисковых полей несколько

ResultFields – набор результирующих полей, данные которых должны возвратиться, разделенных «;»

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
vt: Variant;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA', 'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
vt:=dq.LookUp('User_login; Extension', VarArrayOf(['Ivan', '212']), 'User_Login;Registered_At');
if not VarIsNull(vt) then AddLog(iLine, 'I', 'Lookup results: ' + vt[0] + ' ' + datetimetostr(vt[1]));
```

TDBQuery.Locate – переход на запись по условию

Функция переводит курсор базы на последнюю запись выборки в соответствии с условиями

Синтаксис:

function Locate(const KeyFields: string; const KeyValues: variant; Options: TLocateOptions): boolean;

Функция возвращает результат True, если запись, удовлетворяющая условиям, найдена. Запись считается найденной, если поля из KeyFields имеют значения KeyValues.

Параметры:

KeyFields – набор полей (поисковых полей), по которым идет проверка условия, разделенных «;»

KeyValues – вариантной значение, если поисковое поле одно, или одномерный вариантный массив если поисковых полей несколько

Options – дополнительные параметры поиска

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA', 'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
if dq.Locate('User_login; Extension', VarArrayOf(['Vasiliy', '211']), [loCaseInsensitive, loPartialKey]) then
AddLog(iLine, 'I', 'Record is located');
```

TDBQuery.Eof – конец ли это выборки?

Свойство, показывающее достигнут ли конец выборки

Синтаксис:

```
property Eof: Boolean;
```

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA', 'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
```

```
while not dq.EOF do
begin
…
 dq.Next;
end;
```

TDBQuery.Bof – начало ли это выборки?

Свойство, показывающее достигнуто ли начало выборки

Синтаксис:

```
property Bof: Boolean;
```

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA',
'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Open('SELECT * FROM CONFIG WHERE Extension = :Extension AND Number1=:Number1', ['209', 1001]);
dq.Last;
while not dq.BOF do
begin
…
 dq.Prior;
end;
```

TDBQuery.Execute – выполнить SQL, изменяющий данные

Процедура выполняет SQL запрос, изменяющий данные, например, INSERT или UPDATE

Синтаксис:

```
procedure Execute(const Text: string; const Parameters: array of variant);
```

Параметры:

Text – текст SQL запроса, параметры указываются через двоеточие.

Params – набор вариантных значений параметров

Пример

```
var
db:TDBConnection;
```

```
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA', 'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Execute('INSERT INTO Config (User_login, Extension) VALUES (:i1, :i2)',['Ivanov', 234]);
```

TDBQuery.RowsAffected – число измененных строк

Свойство, показывающее количество измененных или добавленных строк

Синтаксис:

```
property RowsAffected:Integer;
```

Пример

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\a.fdb', 'SYSDBA', 'masterkey', '', 0);
db.Open;
dq := TDBQuery.Create(db);
dq.Execute('INSERT INTO Config (User_login, Extension) VALUES (:i1, :i2)',['Ivanov', 234]);
if dq.RowsAffected > 0 then AddLog(iLine, 'I', 'Data has changed');
```

Специфические параметры разных серверов баз данных

Специфические параметры серверов баз данных передаются в виде строки с параметрами и их значениями, разделенными «;», причем все значения пишутся без кавычек. Полный список всех параметров можно получить с помощью функции GetParameters. Некоторые важные параметры популярных серверов баз данных описаны ниже.

Пример использования параметров

```
var
db:TDBConnection;
dq: TDBQuery;
…
db := TDBConnection.Create(DB_INTERBASE, 'c:\program files\artix calliseum\db\calliseum.fdb', 'SYSDBA', 'masterkey', '', 'UseUnicode=True; SQLDialect=3', 0);
```

Interbase, Firebird

SQLDialect – используемый в базе диалект SQL (возможные значения «1» или «3»)

Role – роль пользователя (возможные значения – имя роли)

Protocol – сетевой протокол для доступа к серверу (возможные значения «TCP», «NetBEUI» и «SPX»)

UseUnicode – все принимаемые строковые значения будут в кодировке юникод (возможные значения «True» и «False»)

Charset – кодировка базы данных (возможные значения - любые поддерживаемые сервером базы данных типа «UTF8» и т.д.)

MS SQL Server

Authentication – способ аутентификации MS SQL (возможные значения «auWindows», если Windows аутентификация, при этом параметры логин и пароль при создании соединения не используются, и «auServer», если аутентификация MS SQL сервера, она используется по умолчанию)

InitialFileName – имя файла базы данных

Oracle

ConnectMode – тип соединения с базой данных (возможные значения «cmNormal» – соединение с ролью обычного пользователя, «cmSysOper» – соединение с ролью SYSOPER, «cmSysDBA» – соединение с ролью SYSDBA, cmSysASM – соединение с ролью SYSASM)

Direct – соединение осуществляется по TCP/IP без клиента Oracle (возможные значения «True» и «False»)

ThreadSafety – клиент Oracle будет работать в потоко-безопасном режиме (возможные значения «True» и «False»)

DateFormat – формат даты (возможные значения типа «MM/DD/YYYY» и т.д.)

DateLanguage – язык строковых данных сервера (возможные значения типа «Russian», «French», «German» и т.д.)

UseUnicode – все принимаемые строковые значения будут в кодировке юникод (возможные значения «True» и «False»)

Charset – кодировка базы данных (возможные значения - любые поддерживаемые сервером базы данных типа «UTF8» и т.д.)

MySQL

Protocol – сетевой протокол для доступа к серверу (возможные значения «mpDefault» - автоматический выбор протокола, «mpTCP» - соединение по TCP/IP, «mpSocket», - соединение через сокеты, «mpPipe» - соединение через Named Pipes, «mpMemory» - соединение через SharedMem и «mpSSL» - соединение с помощью SSL)

Direct – соединение осуществляется по TCP/IP без клиента MySQL (возможные значения «True» и «False»)

SSLCACert – для соединения с помощью SSL, полный путь к корневому файлу сертификату, выданному центром сертификации

SSLCert – для соединения с помощью SSL, полный путь к личному файлу сертификату

SSLChipherList – для соединения с помощью SSL, список дозволенных шифров

SSLKey – для соединения с помощью SSL, полный путь к файлу - ключу

Compress– передаваемые данные будут автоматически сжиматься (возможные значения «True» и «False»)

UseUnicode – все принимаемые строковые значения будут в кодировке юникод (возможные значения «True» и «False»)

Charset – кодировка базы данных (возможные значения - любые поддерживаемые сервером базы данных типа «UTF8» и т.д.)

PostgreSQL

SSLCACert – для соединения с помощью SSL, полный путь к корневому файлу сертификату, выданному центром сертификации

SSLCert – для соединения с помощью SSL, полный путь к личному файлу сертификату

SSLChipherList – для соединения с помощью SSL, список дозволенных шифров

SSLKey – для соединения с помощью SSL, полный путь к файлу - ключу

UseUnicode – все принимаемые строковые значения будут в кодировке юникод (возможные значения «True» и «False»)

Charset – кодировка базы данных (возможные значения - любые поддерживаемые сервером базы данных типа «UTF8» и т.д.)

Предопределенные структуры и перечисления

TRunType

Тип, перечисление, определяющее время и тип запуска сценария во время его выполнения

TRunType = (RT_CALL, RT_GLOBALSTART, RT_GLOBALSTOP, RT_LINESTART, RT_LINESTOP, RT_REGULAR)

RT_CALL – сценарий выполняется во время звонка

RT_GLOBALSTART – сценарий выполняется при запуске Менеджера

RT_GLOBALSTOP – сценарий выполняется во выгрузке Менеджера

RT_LINESTART – сценарий выполняется для каждой линии при ее запуске

RT_LINESTOP – сценарий выполняется для каждой линии при ее остановке

RT_REGULAR – сценарий выполняется регулярно, с определенным периодом

TCurrencyUnits

Тип, перечисление, определяющее валюту

TCurrencyUnits = (CU_RUR, CU_USD, CU_YE, CU_EUR, CU_KKZ, CU_AZE);

CU_RUR – рубли

CU_USD – доллары

CU_YE- условные единицы

CU_EUR – евро

CU_KKZ – тенге

CU_AZE – манаты

TDateCase

Тип, перечисление, определяющее падеж проговариваемой даты

TDateCase = (DATE_CASE_NOMINATIVE, DATE_CASE_GENITIVE);

DATE_CASE_NOMINATIVE – родительный падеж

DATE_CASE_GENITIVE) – именительный падеж

TPriorities

Тип, перечисление, определяющее приоритет пользователя

TPriorities = (PRIORITY_LOW, PRIORITY_NORMAL, PRIORITY_HIGH);

PRIORITY_LOW – низкий приоритет

PRIORITY_NORMAL – нормальный приоритет

PRIORITY_HIGH – высокий приоритет

TAdminActions

Тип, перечисление, определяющее последнее действие администратора

TAdminActions = (ADMIN_NOACTIONS, ADMIN_USER_ADD, ADMIN_USER_MODIFY, ADMIN_USER_MARKDELETE, ADMIN_USER_MAILBOXCLEAR, ADMIN_STATISTICSCLEAR, ADMIN_EXTENDEDSECURITYOFF, ADMIN_USER_MARKADD);

ADMIN_NOACTIONS – не было действия

ADMIN_USER_ADD – пользователь был добавлен

ADMIN_USER_MODIFY – пользователю поменяли параметры

ADMIN_USER_MARKDELETE – пользователя пометили для удаления

ADMIN_USER_MAILBOXCLEAR – пользователю очистили входящий ящик

ADMIN_STATISTICSCLEAR – статистика была очищена

ADMIN_EXTENDEDSECURITYOFF – было включена расширенная безопасность

ADMIN_USER_MARKADD – пользователя пометили для добавления

TVMailPwdCheckType

Тип, перечисление, определяющее режим работы системы по режиму проверки пароля пользователя по телефону

TVMailPwdCheckType = (PWD_CHECK_STRICT, PWD_USER_DEFINED, PWD_NOT_CHECK_STRICT);

PWD_CHECK_STRICT – всегда проверять пароль

PWD_USER_DEFINED –проверять пароль, если указано в свойствах пользователя

PWD_NOT_CHECK_STRICT – никогда не проверять пароль

TVMailUserMode

Тип, перечисление, определяющее режим работы пользователя системы в диалоге

TVMailUserMode = (USER_PROMPT_LIST_MESSAGES, USER_PROMPT_SEND_MESSAGES)

USER_PROMPT_LIST_MESSAGES – абоненту будет предложено прослушать свои сообщения

USER_PROMPT_SEND_MESSAGES – абоненту будет предложено принять и переслать сообщение

TMessageType

Тип, перечисление, определяющее тип исходящего звонка. На любой тип исходящего звонка можно повесить сценарий.

TMessageType = (MT_ALL, MT_VOICE, MT_FAX, MT_CONF, MT_CONFFIRST, MT_PAGING, MT_ROUTING, MT_PHONENOTIFY, MT_FAXNOTIFY, MT_SENDTOMBOX, MT_FAXCHECK, MT_ALARM, MT_USER1, MT_USER2, MT_USER3, MT_USER4, MT_USER5, MT_USER6, MT_USER7, MT_USER8, MT_USER9, MT_USER10);

MT_ALL – для внутреннего пользования

MT_VOICE – голосовое сообщение

MT_FAX – факсимильное сообщение

MT_CONF – участник конференции

MT_CONFFIRST – первый участник конференции

MT_PAGING – пейджинг

MT_ROUTING – не используется

MT_PHONENOTIFY – оповещение по телефону о наличии новых сообщений

MT_FAXNOTIFY – оповещение по факсу, переправка принятых факсов на факс пользователя

MT_SENDTOMBOX – отправка сообщения в ящик пользователю, система при использовании этого типа никуда не звонит

MT_FAXCHECK – проверка наличия факс аппарата на линии

MT_ALARM – будильник

MT_USER1, MT_USER2, MT_USER3, MT_USER4, MT_USER5, MT_USER6, MT_USER7, MT_USER8, MT_USER9, MT_USER10 – пользовательские типы, могут использоваться произвольным способом.

TDitherType

Тип, перечисление, определяющее способ преобразования графического файла

TDitherType = (DefaultDither, FloydSteinbergDither, ModifiedFloydSteinbergDither)

DefaultDither – стандартное, простое размытие

FloydSteinbergDither – размытие по методу Флойда-Штейнберга

ModifiedFloydSteinbergDither – размытие по модифицированному методу Флойда-Штейнберга

TOCRTextEncoding

Тип, перечисление, определяющее кодировку распознанного текста

TOCRTextEncoding = (ENCODING_UTF8, ENCODING_UTF16)

ENCODING_UTF16 – кодировка Unicode-16 (Windows)

ENCODING_UTF8 – кодировка UTF-8

TMergeImageType

Тип, перечисление, определяющее режим конвертирования графических файлов

TMergeImageType = (SinglePage, MultiPage, MultiFile, AdjustedMultiPage)

SinglePage – результатом конвертирования будет одностраничный файл

MultiPage – результатом конвертирования будет многостраничный файл, если это возможно, если же нет – результат будет как если бы значение было MultiFile

MultiFile – результатом конвертирования будут несколько одностраничных файлов

AdjustedMultiPage – результатом конвертирования будет многостраничный файл, переразбитый на страницы определенной длины

TTextOnImageLocation

Тип, перечисление, определяющее положение вставляемого в графический файл текста

TTextOnImageLocation = (topleft, topcenter, topright, centerleft, centercenter, centerright, bottomleft, bottomcenter, bottomright)

topleft – текст будет помещен в левом верхнем углу

topcenter – текст будет помещен посередине вверху

topright – текст будет помещен в правом верхнем углу

centerleft – текст будет помещен слева посередине

centercenter – текст будет помещен в центре

centerright – текст будет помещен справа посередине

bottomleft – текст будет помещен в левом нижнем углу

bottomcenter – текст будет помещен посередине внизу

bottomright – текст будет помещен в правом нижнем углу

TPageUnits

Тип, перечисление, определяющее единицы измерения

TPageUnits = (uCentimeter, uInch, uPixel)

uCentimeter – сантиметры

uInch – дюймы

uPixel – пикселы

TTiffCompressType

Тип, перечисление, определяющее тип компрессии TIFF файла

TTiffCompressType = (MH, MR, MMR, COMPRESSION_DEFAULT)

MH – компрессия MH, TIFF G3

MR – компрессия MR, TIFF G3, two dimensional

MMR – компрессия MMR, TIFF G4

COMPRESSION_DEFAULT – компрессия по умолчанию (MMR)

TDBProvider

Тип, перечисление, определяющее тип сервера внешней базы данных

TDBProvider = (DB_MSSQL, DB_INTERBASE, DB_ORACLE, DB_ACCESS, DB_DBF, DB_DB2, DB_MYSQL, DB_POSTGRE, DB_SYBASE, DB_ADVANTAGE)

DB_MSSQL – MS SQL Server

DB_INTERBASE – Interbase или Firebird

DB_ORACLE – Oracle

DB_ACCESS – MS Access

DB_DBF – dBase DBF

DB_DB2 – DB2

DB_MYSQL – My SQL

DB_POSTGRE – PostGRE SQL

DB_SYBASE – Sybase ASE

DB_ADVANTAGE – Advantage

TLocateOption

Тип, перечисление, определяющее свойства поиска данных в выборке

TLocateOption = (loCaseInsensitive, loPartialKey)

loCaseInsensitive – поиск осуществляется без учета регистра

loPartialKey – поиск осуществляется не только значения как целого, но и как подстроки

TLocateOptions

Тип, множество

TLocateOptions = set of TLocateOption

TUserVars

Тип, структура, определяющая пользовательские переменные, доступные для произвольного использования

TUserVars= record

Ints: array [0 .. 9] of integer; – Набор пользовательских переменных с типом integer

Strings: array [0 .. 9] of string; – Набор пользовательских переменных с типом string

Pointers: array [0 .. 9] of pointer; – Набор пользовательских переменных с типом pointer

Floats: array [0 .. 9] of double; – Набор пользовательских переменных с типом double

end;

TCUHandles

Тип, структура, определяющая параметры линии и звонок

TCUHandles = record

ani: string; – AOH/ANI – только для входящих линий

dnis: string; – DID/DNIS – только для входящих линий

iResult: integer; – результат функций работы с платой

sFaxRemoteId: string; – удаленный Fax ID (доступен после приема/отправки факса)

iFaxBaudRate: integer; – скорость приема/передачи (доступна после приема/отправки факса)

iFaxReceivedFilesCount: integer; – Количество принятых факсимильных файлов (доступно после приема факса)

iFaxSentPagesCount: integer; – число отправленных/полученных страниц (доступно после приема/отправки факса)

UserVars:TUserVars; – Пользовательские переменные, глобальные в пределах каждой линии

end;

TCUUser

Тип, определяющий пользователя системы

TCUUser = record

UserID: Int64; – ID пользователя в базе

UserLogin: string; – логин

UserPassword: string; – пароль

UserPassword2: string; – пароль ограниченного доступа

UserName: string; – Имя, фамилия

UserExtension: string; – внутренний номер

RegisteredAt: TDateTime; – Дата регистрации

LastModifiedAt: TDateTime; – Дата последнего изменения

VoiceGreeting: string; – Файл голосового приветствия

FaxGreeting: string; – Файл факсимильного приветствия

FaxLocalId: string; – Индивидуальный Fax Id

EmailAddress: string; – Email пользователя

EmailAddress2: string; – Второй Email пользователя

PasswordOn: Boolean; – Включена ли проверка пароля

Password2On: Boolean; – Включен ли пароль ограниченного доступа

ForwardOn: Boolean; – Включен ли форвард лист

NotifyByEmailOn: Boolean; – Включено ли оповещение на Email

VoiceGreetingOn: Boolean; – Включено ли голосовое приветствие

LastLoggedAt: TDateTime; – Последний вход в систему

TotalDuration: integer; – общее время работы в секундах

UserInfo: string; – поле пользователя – информация

Fax: string; – Факс

Number1: Double; – Число 1

Number2: Double; – Число 2

Number3: Double; – Число 3

Date1: TDate; – Дата 1

Date2: TDate; – Дата 2

Date3: TDate; – Дата 3

UserOn: Boolean; – Включен ли пользователь

Phone: string; – Телефон

NotifyByPhoneOn: Boolean; – Включено ли оповещение на телефон

Account: string; – Счет

Address: string; – Поле для свободного использования - адрес

CopyDir: string; – Директория для копирования входящих сообщений

CopyDirOn: Boolean; – Включено ли копирование входящих сообщений

SentCopyDir: string; – Директория для копирования исходящих сообщений

SentCopyDirOn: Boolean; – Включено ли копирование исходящих сообщений

NotifyByFaxOn: Boolean; – Включено ли оповещение на факс

NotifyBySmsOn: Boolean; – Включено ли оповещение по СМС

SMSPhone: string; – телефон для отправки СМС

NotifyByPrinterOn: Boolean; – Включено ли оповещение на принтер

Printer: string; – Принтер

FirstPageOnly: Boolean; – Только ли печатать первую страницу

Supervised: Boolean; – Визируется ли пользователь

Priority: TPriorities; – приоритет пользователя

IsOperator: Boolean; – Является ли оператором

AdminName: string; – Администратор, который делал последнее изменение

AdminCode: TAdminActions; – Код последнего действия Администратора

AutoDelete: integer; – Режим автоудаления сообщений

AdminAction: string; – Описание последнего действия Администратора

EERSON: Boolean; – поле для EERS отчетов

RITSID: string; – поле для EERS отчетов

GEIDID: string; – поле для EERS отчетов

SOEID: string; – поле для EERS отчетов

AppFuncCode: string; – поле для EERS отчетов

AppFuncCodeDesc: string; – поле для EERS отчетов

MaxConfereesCount: integer; – максимальное число участников в конференции

GroupId: Int64; - Id группы

SendStatusReports: Boolean; - Включено ли оповещение о постановке в очередь исходящих сообщений

SendConfirmations: Boolean; -Включено ли оповещение о результатах попыток отправки исходящих сообщений

end;

TCUHandles.iResult

В этом поле указываются результаты последней функции работы с платой

Для TCUHandles.iResult возможны следующие значения:

TM_NORMTERM = $00000; – нормальное завершение

TM_MAXDTMF = $00001; – максимум числа цифр принят

TM_MAXSIL = $00002; – максимальная тишина достигнута

TM_MAXNOSIL = $00004; – максимальная НЕ-тишина достигнута

TM_LCOFF = $00008; – сигнал LCOFF получен (для аналоговых линий)

TM_IDDTIME = $00010; – максимальная пауза между вводом цифр достигнута

TM_MAXTIME = $00020; – максимальное время выполнения исчерпано

TM_DIGIT = $00040; – пришла тоновая цифра

TM_PATTERN = $00080; – пришел сигнал отбоя звонка

TM_USRSTOP = $00100; – удаленно прервали функцию

TM_EOD = $00200; – нормальное завершение

TM_TONE = $02000; – пришел сигнал факса

TM_ERROR = $80000; – ошибка оборудования