TPercent Type

Unit

CtcBase

Declaration

TPercent = 0..100;

Description

The TPercent type is used for cardinal percentage values. Used by the PercentDone property.

PrevFieldText Property

Applies to

TParseTool

Declaration

property PrevFieldText : string;

Description

The PrevFieldText property sets FieldText to the previous field, i.e. decrements FieldNumber, and returns the new value of FieldText.

The property is read-only.

CoreTools for Delphi

Welcome to CoreTools for Delphi.

This is a shareware product. This allows you to try the software before you buy it. After evaluating this toolset, if you decide to continue using it, you are required to register the product. You should the read the How to install section first. See Registering your product for information on registering your toolset.

Components

TFileTool

TSearchTool

TScanTool

TParseTool

TSysInfoTool

Libraries

Character routines	CRC-16 routines	DPMI routines
File handling routines	String conversion routines	String-handling routines
PZStr handling routines	Swapping routines	System routines

Product Information

How to use this manual	How to install	License information
Registering your toolset	Technical Support	Address, Phone, Email, etc.
Deploying applications	Technical notes	File inventory
Release notes	**Free upgrade entitlement**	Copyright acknowledgements

Registering by post

Send your remittance of $30 US (GBP 23 in Europe) in the form of CHEQUE or EUROCHEQUE, to the address above, along with the completed registration form (Printable Registration form).

When the cheque has been cleared, you will normally receive same-day postal acknowledgement and confirmation of your serial number.

Enter the registration details by editing the About property of one of the components, and your unique license file will be automatically created in the Windows directory.

That's all there is to it!

See also

About property
Printable Registration form

ErrorCode Property

Applies to:

All CoreTools for Delphi components

Declaration

property ErrorCode: TCtError;

Description

Stores a code that identifies the last error trapped by the component.

Registration form

CoreTools Registration

Company Name ____

Name ____

Address ____

____

City ____

State ______________ ZIP __________ Country___________

Phone ______________________________

Product **CoreTools for Delphi**

Number to order ____________

x $30.00 ____________

Total ____________

Send To:
Core Software Limited
3 Tearne Street, St Johns,
Worcester. WR2 6BL, UK

Deploying your application

When deploying your application, created using CoreTools for Delphi, you should also include the file CORE.LIC, which your application's deployment program should insert in the WINDOWS directory of the target PC.

You will find your copy of CORE.LIC in your own WINDOWS directory, where it will have been installed when you registered your toolset.

This procedure is necessary to ensure that CoreTools for Delphi operates in non-design mode.

See also

Design-mode enabled

How to install

To install these components, we suggest that you:

1. Create a new sub-directory of your main delphi directory (e.g. C:\DELPHI\CMPNTS where C is the drive where you installed Delphi).
2. Extract all the files in the **CORE1A.ZIP** file into it.
3. Add this directory to your **LIB** search list (**Options|Environment|Library path**).
4. Add the **CTREG.PAS** file to your library (**Options|Install components**), this installs all CoreTools for Delphi components and their associated property editors. You can edit this file later to configure which components you wish to install permanently.
5. Rebuild your Delphi library (remember to back up your old one as a precaution - better safe than sorry).

Installing Help

To merge the component help into the Delphi help you must add the **CORETOOL.KWF** file to the **DELPHI.HDX** file by using the **HELPINST.EXE** program that comes with Delphi.

1. Launch the **HELPINST.EXE** application.
2. Open the **DELPHI.HDX** file (usually in \DELPHI\BIN)
3. Add the **CORETOOL.KWF** file.
4. Choose **Save**.

You should now be able to get context-sensitive help with the CoreTools for Delphi components.

Until the components are registered, they will only work correctly whilst Delphi is also running, i.e. they are design mode enabled.
Get the toolset registered to be able to use the components in your applications.

See Also

Registering your product
Inventory of files

CtIsASCII Function

Unit
CtlChar

Declaration
function CtIsASCII (C: char): boolean;

Description
Tests *C* to check if it is an ASCII character. The check performed is: C < Chr(128).

Parameters
C	the character on which to perform the test.

Returns
True if the test is successful, otherwise returns False.

Exceptions
None

See also
CtIsAlNum
CtIsAlpha
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsLower
CtIsPrint
CtIsPunct
CtIsReal
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

FieldCount Property

Applies to

TParseTool

Declaration

property FieldCount: integer;

Description

The FieldCount property contains the number of lines in the Fields property. The Fields property is of a type that is a descendant of TStringList and the FieldCount property is the same as Fields.Count.

Design mode enabled

Until the product has been registered and you have been issued with your unique serial number, the CoreTools for Delphi components are design-mode enabled. This means that they will only run when Delphi is also running. If you try to run an application you have developed using the components and the CORE.LIC file is not present in your WINDOWS directory, then the controls will halt your application and display a message dialog explaining that your application will not run until the CoreTools license is present in the WINDOWS directory.

See also

Registering your product

About property

Deploying your own applications

About Property

Applies to

All CoreTools components

Declaration

property About: TAboutInfo

Description

The About property provides product and component information for the CoreTools for Delphi product.

The property editor associated with the About property also acts as the product registration mechanism. By selecting the **Register** button you enter the **Registration Info** page, where you can enter your name and the unique serial number that is sent to you when you register your CoreTools for Delphi toolset. It is important that you enter your name in the exact form specified in your registration acknowledgement.

When the name and serial number have been entered correctly, the property editor creates a file called CORE.LIC in your WINDOWS directory. The license file is the persistent record of your license information and must be present for applications using the components to run correctly. If the license file is lost for any reason, you have only to repeat the registration entry process to duplicate the file.

See also

Design-mode enabled

Deploying your own applications

FilePath Property

Applies to

TFileTool, TScanTool, TParseTool

Declaration

property FilePath : string;

Description

The FilePath property returns the path portion of FileName. The property is read-only.

TCtDateText Type

Unit

CtlDate

Declaration

TCtDateText = string[10];

Description

Variables of this type store dates in the short format.

See also

TCtTimeText

TDateTextPropertyEditor

TCtWinSettings Type

Unit

CtlSys

Declaration

TCtWinSet = (win80x87, winCPU286, winCPU386, winCPU486, winENHANCED, winPAGING, winPMODE, winSTANDARD, winWIN286, winWIN386);

TCtWinSettings = set of TCtWinSet;

Description

win80x87	System contains an Intel math coprocessor.
winCPU286	System CPU is an 80286.
winCPU386	System CPU is an 80386.
winCPU486	System CPU is an i486.
winENHANCED	Windows is running in 386-enhanced mode. The winPMODE flag is always set when winENHANCED is set.
winPAGING	Windows is running on a system with paged memory.
winPMODE	Windows is running in protected mode. In Windows 3.1, this flag is always set.
winSTANDARD	Windows is running in standard mode. The winPMODE flag is always set when winSTANDARD is set.
winWIN286	Same as winSTANDARD.
winWIN386	Same as winENHANCED.

See also

CoreTools system routines

Fields Property

Applies to

TParseTool

Declaration

property Fields : TSelectList;

Description

The Fields property maintains a list of fields, containing the individual fields that have been parsed from LineText.

In addition, FieldText can be aliased as Fields.Strings[FieldNumber].

Caveat

Do not use the associated Objects property, inherited from TStringList, as that is used to store positional data for the respective fields.

OnScan Event

Applies to:

TScanTool, TParseTool

Description

Whenever an Action Scan command has been initiated the component triggers the OnScan event for each line in the file. This creates a simple mechanism to allow the developer to code file modifying or parsing applications, without writing elaborate scanning and parsing code.

OnError Event

Applies to:
All CoreTools components
Description
The OnError event occurs whenever an error is trapped by any of the CoreTools for Delphi components. i.e. they attempt to convert all exceptions to events.   If an OnError event handler has not been defined, then an exception is generated and it is the responsibility of the user to have defined a try..exception construct to catch the exception.

OnFinish Event

Applies to:

TFileTool, TSearchTool, TScanTool, TParseTool

Description

Whenever an Action initiated command is complete, then the components call the user defined OnFinish event. This enables the developer to display messages or to take any other appropriate action.

OnStart Event

Applies to:
TFileTool, TSearchTool, TScanTool, TParseTool

Description
Whenever an Action command is set, the components verify that all property settings are in order and then triggers the OnStart event. This allows the user to display messages or whatever is appropriate for the application.

If the other property values are incorrect or inconsistent, then an OnError event is triggered.

Release Notes

Version 1a - August 10, 1995

Original release.

Action Property

Applies to

TFileTool, TParseTool, TScanTool, TSearchTool

Declaration

Each component has its own declaration.

TFileTool component

TParseTool component

TScanTool component

TSearchTool component

Description

Specifies an Action command to be performed by the component.

CtReg Unit

Description

This unit performs registration of the components in CoreTools for Delphi.   The file is supplied in Pascal format, so that the user can select which   components to install and to alter the component palette page on which they appear by default.

Several of the components cannot be removed when component registering. CtcBase provides the base class used by the other components and must be present, along with TAboutInfoPropertyEditor which is the editor for the About property on all CoreTools for Delphi components.

Routines

Register

LineText Property

Applies to

TScanTool, TParseTool

Declaration

property LineText : string;

Description

The LineText property returns the current line of text being scanned.
The property value can also be represented as Lines.Strings[LineNumber].

See also

LineCount
LineDups
LineNumber
LinePos
Lines
LineSort

FieldSeparators Property

Applies to

TParseTool

Declaration

property FieldSeparators : string;

Description

The FieldSeparators property is the field separator string. This can be altered at run time, and results in Fields being regenerated.

AutoDisplay Property

Applies to

All CoreTools components

Declaration

property AutoDisplay: boolean;

Description

When this property is set true, the component will perform all actions immediately, without waiting for an Action command. For example, the TFileTool component will update all relevant file properties as the FileName property is modified.

Registering via CompuServe

If you want to register via CompuServe, GO SWREG and enter the product ID #6246

The cost will be charged on your next CIS Direct Debit.

After CompuServe has sent us your User ID and your address, we will email an acknowledgement to you, as quickly as possible. (normally same day). The acknowledgement will advise you of your unique serial number.

Enter the registration details by editing the About property of one of the components, and your unique license file will be automatically created in the Windows directory.

That's all there is to it!

See also

About property

File Name Property

Applies to

TFileTool, TScanTool, TParseTool

Declaration

property FileName : TFileName;

Description

The FileName property specifies the file name that will be parsed.

The file name can include a path. For example, to open the file README.TXT in the directory C:\TEMP, set FileName to C:\TEMP\README.TXT.

The TFileName property editor allows navigational selection of a file name in the Object Inspector.

The FileName property can be set to the name of a file that doesn't exist in the current directory.

See also

faCopy
faCopyVerify
faCRC16
faCreate
faDelete
faKill
faMove
faMoveVerify

FileNameOnly Property

Applies to

TFileTool, TScanTool, TParseTool

Declaration

property FileNameOnly : TCtFileNameOnly;

Description

The Filenameonly property returns FileName, without the path. The property is read-only.

WinSettings Property

Applies to
TSysInfoTool

Declaration
property WinSettings: TCtWinSettings

Description
This set property returns the system processor and memory configuration.

PercentDone Property

Applies to
TScanTool, TParseTool

Declaration
property PercentDone : TPercent;

Description
The PercentDone property returns the percentage of the file that has been scanned so far.

The value returned is based on the number of lines scanned, and not the number of bytes in the file.

LineDups Property

Applies to

TScanTool, TParseTool

Declaration

property LineDups : TDuplicates;

Description

The LineDups property determines whether duplicate strings are allowed in Lines. If the list is not sorted (see LineSort), the value of LineDups has no effect. These are the possible values:

Value	Meaning
dupIgnore	Attempts to add a duplicate string to a sorted string list are ignored
dupAccept	Duplicate strings can be added to a sorted string list
dupError	Adding a duplicate string results in an EListError exception

FieldFilter Property

Applies to

TParseTool

Declaration

property FieldFilter : boolean;

Description

Determines whether to remove leading and trailing quotes from fields when the FieldMatch property is set to True.

Free upgrade entitlement

When you have registered your CoreTools for Delphi toolset, you will have the added bonus of one free upgrade to the next version of the product. Afterwards, upgrades will be sold as separate products and will require additional payment of a discounted upgrade price.

You will not have to register your free upgrade, as the toolset registration checking mechanism is already enabled to accept the upgrade. Simply, monitor the shareware library where you obtained this copy, and, when the next version is released, download it and start using it straight away.

License Agreement

IMPORTANT - READ CAREFULLY BEFORE USING THE SOFTWARE. By using the SOFTWARE you are agreeing to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, you are not authorized to use the SOFTWARE. The terms of this Agreement apply specifically to the SOFTWARE identified herein and supersede, to the extent of any inconsistencies any other agreement that might be packaged with the SOFTWARE.

LICENSE AGREEMENT

(CoreTools for Delphi)

SOFTWARE LICENSE

1. GRANT OF LICENSE.

This Agreement permits you to make and use copies of the software program identified above which includes the files identified as CTREG.PAS, CTREG.DCR, CTCBASE.DCU, CTCFBASE.DCU, CTCFILE.DCU, CTCPARSE.DCU, CTCSCAN.DCU, CTCSERCH.DCU, CTCSYSIN.DCU, CTEABOUT.DCU, CTEABOUT.DFM, CTEDATE.DCU, CTEDATE.DFM, CTEFILEN.DCU, CTESELCT.DCU, CTESELCT.DFM, CTETIME.DCU, CTETIME.DFM, CTLCHAR.DCU, CTLCRC16.DCU, CTLDATE.DCU, CTLDPMI.DCU, CTLERROR.DCU, CTLFILE.DCU, CTLSTRNG.DCU, CTLSWAP.DCU, CTLSYS.DCU, CTLZSTRN.DCU, CTSWITCH.INC, CORETOOL.HLP, CORETOOL.KWF (the "SOFTWARE") for your internal use only provided each copy is used only on a single computer or by a single person. The SOFTWARE is in "use" on a computer when it is loaded into temporary memory (i.e., RAM) or installed into permanent memory (e.g., hard disk, CD-ROM, or other storage device) of that computer.

2. COPYRIGHT.

The SOFTWARE (including any images, "applets", photographs, animations, video, audio, music, and text incorporated into the SOFTWARE) is owned by Core Software Ltd or its suppliers and is protected by copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g., a book or musical recording). You may not copy any printed materials accompanying the SOFTWARE.

3. OTHER RESTRICTIONS.

You may not rent or lease the SOFTWARE, but you may transfer the SOFTWARE and accompanying printed materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. You may not reverse engineer, decompile, or disassemble the SOFTWARE. If the SOFTWARE is an update or has been updated, any transfer must include the most recent update and all prior versions.

4. ADDITIONAL TERMS FOR DEVELOPMENT PRODUCTS

The products, including the one in this package, may include one or more libraries intended to help you develop your own application programs. You may write and compile your own application programs using the SOFTWARE contained in this package. If you are the licensed, registered user of this product, you may use, reproduce, give away, or sell any program you write using this product, in executable form only, without additional license or fees, subject to all the conditions in this statement.

You will remain solely responsible to anyone receiving your programs for support, service, upgrades, or technical or other assistance, and such recipients will have no right to contact Core Software Ltd for such services and or assistance.

Your programs may not merely be a set or subset of any of the libraries, code, or other files included in this package.

You cannot use the libraries or components to produce other components, other than for use in your own programs.

Registering your product

CoreTools for Delphi was created by:

Core Software Limited

Mail:	**3 Tearne Street, St Johns, WORCESTER, WR2 6BL, UK**
Email:	**100041.3143@compuserve.com**
CIS:	**100041,3143**
Phone:	**+44 1905 420784**

This is a shareware product. This allows you to try the software before you buy it. When, after evaluating the product, you decide to continue using it, you are required to register your toolset by sending us the registration fee of $30 (GBP 23 in Europe).

There are 2 ways to register your product.

Registering via CompuServe SWREG service:

Registering via postal mail and cheque:

How to use this manual

Manual organization

This manual is organized into logical topics. You can either use the table of contents to move from topic to topic, or the search button to locate information on a particular subject.

Included topics

Since most components share a large number of properties, events, and methods, only new topics are included for each component. To obtain help on common topics, use the Delphi manual. We strongly recommend you take advantage of Delphi's object browser (**View|Browser**). It provides complete information on every component.

Writing conventions

The manual is written using the following conventions:

The names of applications, products, or services are in bold italic (***Windows 95***).

Special references such as parameters, and points of interest, are in bold (**Application name**).

Keyboard commands, menu, buttons, and other items requiring direct response are in bold (**Close**).

File names and directories are in small capitals (PROGMAN.EXE).

Examples for property values are in purple, italic letters (*Fantastic new product)*

Examples for typed commands are in small, dark blue, Courier type (`TipDialog.Execute`).

Ellipses (...) are used in source code examples to represent non-pertinent code.

Important notes are in dark red (NOTE: This is dangerous).

Exceptions to the above conventions can be found in jumps and pop-ups.

TFileTool Component

Unit

CtcFile

Description

This component allows the developer to modify existing file attributes, create new files, and rename or delete existing files.
Files can also be copied and moved.
The component also provides information relating to the currently selected file, such as its size, CRC-16 value, and date/time stamp.

Properties

About	Provides product and component version information and facilities to register your product.
Action	Specifies an action to be performed by the component.
ErrorCode	Returns a code that identifies the last error trapped by the component.
FileAtts	Sets/returns the attributes for thecurrent file.
FileCRC	Returns the CRC-16 value for the currently selected file.
FileDate	Sets/returns the date for the currently selected file.
FileExists	Returns whether the specified file exists.
FileName	Sets the currently selected file.
FileNameExt	Returns the extension part of the currently selected file's full name.
FileNameOnly	Returns the name part only of the currently selected file's full name.
FilePath	Returns the path of the currently selected file.
FileSize	Returns the size in bytes of the currently selected file.
FileTime	Sets/returns the time of the currently selected file.
NewFileExt	Sets a new extension part to the currently selected file's full name.
NewFileName	Sets a new name for the currently selected file.
TransferName	Sets the destination name for a file copy or file move action.

Events

OnError	Fired when TFileTool encounters an error.
OnFinish	Fired when TFileTool has completed a command.
OnStart	Fired when TFileTool starts a command.

See also

Component logical model

TSearchTool Component

Unit

CtcSerch

Description

This component performs seraches for files, on both mask setting and file attribute settings. The results of the search are stored in report string lists, one for the directories and one for file, which allows simple processing of the search results.
The report lists can be sorted and the developer can specify whether the full path name is required in the reports.

Properties

About	Provides product and component version information and facilities to register your product.
Action	Specifies an action to be performed by the component.
ErrorCode	Returns a code that identifies the last error trapped by the component.
FileAtts	Sets/returns the file attributes to be searched for.
AutoDisplay	Allows AutoDisplay search without setting an Action command.
ListDirs	The report list containing directories located during the search.
ListDirItems	Returns the number of items in the ListDirs property.
ListFiles	The report list containing the files located during the search.
ListFilesItems	Returns the number of items in the ListFiles property.
ListsFullReport	Sets whether the full path name should be used in the reports.
ListsSortReport	Sets whether the reports should be sorted into ascending order.
SearchMask	Sets the file mask for the search, set to *.* by default.
SearchPath	Sets the start path for the search.
SearchSubDirs	Sets whether sub-directories are to be searched, in addition to the directory specified in SearchPath.

Events

OnStart	Fired when TSearchTool starts a command.
OnFinish	Fired when TSearchTool has completed a command.
OnError	Fired when TSearchTool encounters an error.

See also

Component logical model

TScanTool Component

Unit

CtcScan

Description

The TScanTool component allows you to quickly scan text files. It provides a mechanism to read a text file without writing lots of code and error checking.

TScanTool can read lines up to 255 characters long while scanning files. If a line exceeds this limit, TScanTool truncates the line and generates an Error event.

TScanTool is different from most other components in that it does not generate run time errors. Instead, it fires the OnError event and sets its own internal ErrorCode property. Therefore, TScanTool ignores errors by default. You can trap errors by trapping the OnError event.

Properties

About	Provides product and component version information and facilities to register your product.
Action	Specifies an action to be performed by the component.
AutoDisplay	Specifies immediate actiion or whether to wait for an Action command.
ErrorCode	Returns a code that identifies the last error trapped by the component.
FileName	Sets or returns the name of the file to be scanned by the component.
FileNameExt	Returns the DOS file name extension for FileName.
FileNameOnly	Returns the name of the file without the path.
FilePath	Returns the name of the file path.
LineCount	Returns the number of lines in the file being parsed.
LineDups	Specifies if duplicate lines are to be ignored, allowed or if an error is to be fired.
LineNumber	Returns the current line number in the file being parsed.
LinePos	Sets or returns the position of the current line within the file.
Lines	A list containing the file being parsed.
LineSort	Indicates whether the lines in the file are to be sorted before being parsed.
LineText	Returns the value of the current text line.
PercentDone	Returns a number between 0 and 100 that indicates how much of the current file has been scanned.

Methods

Events

OnStart	Fired when TScanTool opens a file for scanning, after Action is set to Start.
OnFinish	Fired when TScanTool has completed scanning a file.
OnError	Fired when TScanTool encounters an error.
OnScan	Fired for each line in the input file while the component scans a file.

See also

Component logical model

TParseTool Component

Unit

CtcParse

Description

The TParseTool component allows you to quickly scan and parse text files. It is the ideal tool for doing simple data manipulation changing its format, checking its validity, retrieving items, generating reports, and the like without having to write a lot of code to scan and parse files.

The TParseTool component is similar to the original AWK language in concept and purpose, but it is substantially different in syntax, since the TParseTool is embedded in event driven Object Pascal. The most apparent omissions are expression evaluation and a pattern matching mechanism.

TParseTool can read lines up to 255 characters long while scanning files. If a line exceeds this limit, TParseTool truncates the line and generates an Error event.

TParseTool is different from most other components in that it does not generate run time errors. Instead, it fires the OnError event and sets its own internal ErrorCode property. Therefore, TParseTool ignores errors by default. You can trap errors by trapping the OnError event.

Properties

About	Provides product and component version information and facilities to register your product.
Action	Specifies an action to be performed by the component.
AutoDisplay	Specifies immediate action or to wait for an Action command.
ErrorCode	Returns a code that identifies the last error trapped by the component.
FieldConvert	Determines whether to change the case of each line as it is read
FieldCount	The number of fields in LineText.
FieldFilter	Determines whether to remove leading and trailing quotes from fields when the FieldMatch property is set to True.
FieldMatch	Determines whether the TParseTool parser should regard quote-delimited strings as single fields.
FieldNumber	The index value into the Fields list.
Fields	Sets or returns the value of a field within the current line.
FieldSeparators	Field separator string.
FieldText	The text field, represented by Fields[FieldNumber].
FileName	Sets or returns the name of the file to be scanned by the component.
FileNameExt	Returns the DOS file name extension for FileName.
FileNameOnly	Returns the name of the file without the path.
FilePath	Returns the name of the file path.
LineCount	Returns the number of lines in the file being parsed.
LineDups	Specifies if duplicate lines are to be ignored, allowed or if an error is to be fired.
LineNumber	Returns the current line number in the file being parsed.
LinePos	Sets or returns the position of the current line within the file.
Lines	A list containing the file being parsed.
LineSort	Indicates whether the lines in the file are to be sorted before being parsed.
LineText	Returns the value of the current text line.
NextFieldText	Returns the next field to that in FieldText.
PercentDone	Returns a number between 0 and 100 that indicates how much of the current file has been scanned.
PrevFieldText	Returns the previous field to that in FieldText.

Methods

None

Events

OnError	Fired when TParseTool encounters an error.
OnFinish	Fired when TParseTool has completed scanning a file.
OnScan	Fired for each line in the input file while the component scans a file.

OnStart	Fired when TParseTool opens a file for scanning, after Action is set to Start.

See also

Component logical model

CtSwapByte Procedure

Unit
CtlSwap

Declaration
procedure CtSwapByte (var x, y: byte);

Description
Swaps the contents of two byte variables.

Parameters
x and *y* are the values to be swapped.

Exceptions
ECtError   Message = **CtlSwap: CtSwapByte**

See also
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines

TCtError Type

Unit

CtlError

Declaration

TCtError = (feNone, feOpenFile, feCloseFile, feInvalidDirectory, feReadError, feInvalidFile, feDateConversion, feTimeConversion, feSettingFileAttr, feCreateFile, feDeleteError, feRenameError, feSameSourceAndTarget, feErrorDuringCopy, feLineTooLong, feFieldTooLong, feCRCError);

Description

The enumeration contains the various errors returned by the CoreTools components.

If an error is detected, then the ErrorCode property is set to the appropriate error code. If the OnError event has a handler defined, then that procedure is called, otherwise an ECtError exception is generated, and it is the responsibility of the developer to handle the error condition.

Error Codes

feNone	No error condition exists
feOpenFile	Error occurred when opening a file
feCloseFile	Error occurred when closing a file
feInvalidDirectory	The directory does not exist
feReadError	Error occurred when reading a file
feInvalidFile	The file does not exists
feDateConversion	Attempted invalid date conversion.
feTimeConversion	Attempted invalid time conversion
feSettingFileAttr	Error occurred when setting file attributes
feCreateFile	Unable to create a file.
feDeleteError	Unable to delete a file.
feRenameError	Unable to rename a file.
feSameSourceAndTarget	Same file used for both source and destination during file copy.
feErrorDuringCopy	Error occurred while copying a file.
feLineTooLong	The line being parsed exceeds 255 characters.
feFieldTooLong	A field exceeds 255 characters.
feCRCError	CRC checking error

TCtFileExt Type

Unit
CtlFile

Declaration
TCtFileExt = string[4];

Description
The TCtFileExt type stores the 4 byte DOS file name extension, including the period, e.g '**.TXT**'.

TCtFileNameOnly Type

Unit

CtlFile

Declaration

TCtFileNameOnly = string[8];

Description

The TCtFileNameOnly type stores a file name without the extension.

TScanActions Type

Unit

CtcScan

Declaration

TScanActions = (saDormant, saScan, saNextLine, saSave, saQuit);

Description

The TScanActions enumeration contains the valid commands for the TScanTool and TParseTool components.

Commands

saDormant	No command is being executed.
saScan	Commands the component to scan each line of FileName, the text for which is stored in Lines, and will trigger an OnScan event for each line in the text file. The current text line is stored in LineText. scanQuit can be used to stop the command if necessary.
saNextLine	Causes the component to scan immediately the next line of text, i.e. LineNumber is incremented and LineText is altered appropriately.
saSave	The Lines property will be saved to FileName, OnStart and OnFinish events will be triggered. If any errors occur then the OnError event is triggered. scanQuit can be used to stop the command if necessary.
saQuit	Stops the current command, the OnFinish event will be triggered immediately.

TSearchActions Type

Unit

CtcSerch

Declaration

TSearchActions = (raDormant, raSearch, raQuit);

Description

The TSearchActions enumeration specifies valid actions for the TSearchTool component.

Commands

raDormant	No command is being executed.
raSearch	This command starts the search for the specified files and/or directories.
raQuit	Stops the search process.

TFileActions Type

Unit

CtcFile

Declaration

TFileActions = (faDormant, faCopy, faCopyVerify, faCRC, faCreate, faDelete, faKill, faMove, faMoveVerify);

Description

The valid Action commands for the TFileTool component.

Action commands

faDormant	No command being executed.
faCopy	Copies FileName to TransferName
faCopyVerify	Copies FileName to TransferName and checks the CRC-16 values of both files.
faCRC16	Calculates the CRC-16 value for FileName.
faCreate	Creates a new file called FileName
faDelete	Deletes FileName if the files properties allow deletion.
faKill	Deletes FileName regardless of the files properties.
faMove	Moves FileName to TransferName, regardless of the files properties.
faMoveVerify	Moves FileName to TransferName and checks the CRC-16 values of both files The move ignores the files properties.

TCtFullFileName Type

Unit

CtlFile

Declaration

TCtFullFileName = string[12];

Description

The TCtFullFileName type stores a DOS file name, including the period and extension.

TSelectList Type

Unit

CtcBase

Declaration

TSelectList   = TStringList;

Description

The TSelectList type is used for storing lists of items that have been selected by the CoreTools for Delphi components.

This TStringList descendant object maintains a list of strings. You can add, delete, insert, move, and exchange strings using the Add, Delete, Insert, Move, and Exchange methods. The Clear method clears all the strings in the list of strings. The Count property contains the number of strings in the list. Each string list object has a Strings property that lets you access a particular string by its position in the list of strings. To find the position of a string in the list, use the IndexOf method.

If you want to add several strings at once to a list of strings, use the AddStrings method. You can assign one strings object to another using the Assign method.

To determine if a particular string exists in the list of strings, call the Find method. You can store strings in a file and then load them all at one using the LoadFromFile method. To save the strings to a file, use the SaveToFile method.

TTextConvert Type

Unit

CtcBase

Declaration

TTextConvert = (cvtNone, cvtUpper, cvtLower);

Description

The TTextConvert enumeration is used to specify the text conversion type required.

TCtTimeText Type

Unit

CtlDate

Declaration

TCtTimeText = string[10];

Description

The TCtTimeText type stores time in the 24 hour hh:mm:ss format.

See also

TCtDateText
TTimeTextPropertyEditor

CtcBase Unit

Description
CtcBase is the unit containing the abstract super-class, TCoreTool, from which all CoreTools for Delphi components descend.

Types
TCoreTool
TCtError
TPercent
TSelectList
TTextConvert
TMaxBuffer

Variables
CoreBuffer

Exceptions
ECtError

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

TAboutInfoPropertyEditor property editor

Description

This property editor allows the developer to edit the About property in all CoreTools for Delphi components.

This property editor is associated with the About property and also acts as the product registration mechanism. By selecting the **Register** button you enter the **Registration Info** page, where you can enter your name and the unique serial number that is sent to you when you register your CoreTools for Delphi toolset. It is important that you enter your name in the exact form specified in your registration acknowledgement.

When the name and serial number have been entered correctly, the property editor creates a file called CORE.LIC in your WINDOWS directory. The license file is the persistent record of your license information and must be present for applications using the components to run correctly. If the license file is lost for any reason, you have only to repeat the registration entry process to duplicate the file.

Caveat:

If the property editor is not registered with the current Delphi library, then the CoreTools toolset cannot be registered.

CoreTools for Delphi components

The components in CoreTools for Delphi, version 1a, are:

TFileTool
TSearchTool
TScanTool
TParseTool
TSysInfoTool

FieldNumber Property

Applies to

TParseTool

Declaration

property FieldNumber : integer;

Description

The FieldNumber property returns the current field in the Fields property, used when the FieldText and NextFieldText properties are being used to parse the current line, LineText.

FieldMatch Property

Applies to

TParseTool

Declaration

property FieldMatch: boolean;

Description

Determines whether the parser should regard quote-delimited strings as single fields. This feature was added to facilitate parsing of quote and comma-delimited files such as those exported by most database and spreadsheet programs.

Double or single quotes are both recognized, but one does not match the other.

Unmatched quotes in a line are not ignored, i.e. parsing continues until end-of-line and no error is generated.

FieldConvert Property

Applies to

TParseTool

Declaration

property FieldConvert : TTextConvert;

Description

The FieldConvert property allows the user to specify what type of conversion should be performed on the fields stored in the Fields property. Possible choices are none, upper case, and lower case.

FieldText Property

Applies to

TParseTool

Declaration

property FieldText : string;

Description

The FieldText property returns the value of the current field. This property is exactly the same as the Fields.Strings[FieldNumber] property.

FileNameExt Property

Applies to

TFileTool, TScanTool, TParseTool

Declaration

property FileNameExt : TCtFileExt;

Description

The FileNameExt property returns the DOS file name extension for FileName. The property is read-only.

LineCount Property

Applies to
TScanTool, TParseTool

Declaration
property LineCount : integer;

Description
The LineCount property stores the number of lines in the file being parsed.

The file is read into the Lines property, taking into account the LineDups and LineSort property settings, and the LineCount property reflects the number of lines read.

The limitation on file sizes that can be parsed is that imposed by the integer type, i.e. 32,767 lines, along with the availability of free memory in the heap.

If the AutoDisplay property is true, then the file is read immediately the FileName property is set, otherwise it is not read until the Action command to scan is issued.

LineNumber Property

Applies to

TScanTool, TParseTool

Declaration

property LineNumber : integer;

Description

The LineNumber property stores the current line being parsed.

LineText can also be represented as Lines.Strings[LineNumber].

LinePos Property

Applies to
TScanTool, TParseTool

Declaration
property LinePos : longint;

Description
The LinePos property stores the physical position in the file of the start of the current line, LineText.
The physical position is the number of bytes from the start of the file.

See also
LineNumber property

Lines Property

Applies to

TScanTool, TParseTool

Declaration

property Lines : TSelectList;

Description

The Lines property stores the file being parsed.

See also

LineNumber

LineSort Property

Applies to

TScanTool, TParseTool

Declaration

property LineSort : boolean

Description

The LineSort property determines whether or not the lines in the file will be sorted in alphabetical order before being parsed.

See also

LineCount
LineDups
LineNumber
LinePos
Lines
LineText

CtIsCntrl Function

Unit

CtlChar

Declaration

function CtIsCntrl (C: char): boolean;

Description

Tests *C* to check if it is a control character. The check performed is: #00-#31 or #127.

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsDigit
CtIsGraph
CtIsLower
CtIsPrint
CtIsPunct
CtIsReal
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

NextFieldText Property

Applies to

TParseTool

Declaration

property NextFieldText : string;

Description

The NextFieldText property sets FieldText to the next field, i.e. increments FieldNumber, and returns the new value of FieldText.

The property is read-only.

CtcScan Unit

Description

This unit contains the implementation of the TScanTool component.

Types

TScanActions
TScanTool

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

CtcParse Unit

Description

This unit is the implementation of the TParseTool component.

Types

TParseTool

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

CoreSrch Unit

Description

This unit contains the implementation of the TSearchTool component.

Types

TSearchActions
TSearchTool

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

CtcFile Unit

Description

This unit contains the implementation of the TFileTool component.

Types

TFileActions
TFileTool

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

CteAbout Unit

Description

This unit contains the definition and implementation of TAboutInfoPropertyEditor, which is the property editor used to edit the About property on all CoreTools for Delphi components.

Types

TAboutInfoPropertyEditor

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

CteFilen Unit

Description

This unit contains the implementation of the TFileNamePropertyEditor class. This is the property editor associated with properties of the TFileName class.

Types

TFileNamePropertyEditor

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

CteSelct Unit

Description

This unit contains the implementation of the TSelectListPropertyEditor class. This is the property editor associated with properties of the TSelectList type.

Types

TSelectListPropertyEditor

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

CteTime Unit

Description

This unit contains the implementation of the TTimeTextPropertyEditor property editor. This is the property editor associated with properties of the TCtTimeText type.

Types

TTimeTextPropertyEditor

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

Units

The unit files constituting CoreTools for Delphi are:

CtReg	CtcBase	CtcFBase	CtcFile
CtcParse	CtcScan	CtcSerch	CtcSysIn
CteAbout	CteDate	CteFilen	CteSelct
CteTime	CtlChar	CtlCRC16	CtlDate
CtlError	CtlFile	CtlStrng	CtlSwap
CtlSys	CtlZStrn		

See also

Modifying the palette

CteDate Unit

Description

This unit contains the implementation of the TDateTextPropertyEditor property editor. This is the property editor associated with properties of the TCtDateText type.

Types

TDateTextPropertyEditor

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

Modifying the component palette

There are five components making up the CoreTools for Delphi toolset. It may be that you do not need all of the components installed in your component palette, in which case it is possible to tailor the component registration to suit your own needs.

The component and property editor registration is all handled in the CORETOOL.PAS unit file. If you examine the file you will see that the **Register** procedure performs all registrations. The technique to employ is to comment-out the components and associated property editors that you do not want installed at the current time.

TCoreTool Component

Unit

CtcBase

Description

TCoreTool is the abstract super-class from which all CoreTools for Delphi components descend.

See also

Component logical model

FileExists Property

Applies to

TFileTool

Declaration

property FileExists : boolean;

Description

The FileExists property indicates whether the file specified by FileName actually exists.

FileAtts Property

Applies to
TFileTool, TSearchTool

Declaration
property FileAtts : TFileType;

Description
The FileAtts property stores the set of file attributes for the currently specified file.

If the AutoDisplay property is true then the changes/search will take place immediately, otherwise it will not happen until the user sets the Action property.

Errors trapped
feInvalidFile
feDateConversion
feSettingFileAttr

Note
TFileType is declared in the Delphi unit FileCtrl.

FileDate Property

Applies to

TFileTool

Declaration

property FileDate : TCtDateText

Description

The FileDate property returns the date, in text format, of the current file..

Errors trapped

feOpenFile

feDateConversion

See also

FileTime

FileSize Property

Applies to

TFileTool

Declaration

property FileSize : longint;

Description

The FileSize property returns the size of the currently selected file. The property is read-only.

FileTime Property

Applies to

TFileTool

Declaration

property FileTime : TCtTimeText;

Description

The FileTime property returns the time stamp of the currently selected file. The time stamp can also be modified by setting this property to the required new time.

Errors trapped

feOpenFile
feTimeConversion

FileCRC Property

Applies to

TFileTool

Declaration

property FileCRC : TCtCRC16

Description

The FileCRC property stores the CRC value of the currently selected file.

NewFileExt Property

Applies to

TFileTool

Declaration

property NewFileExt : TCtFileExt

Description

The NewFileExt property accepts user input of a new filename extension. If the AutoDisplay property is set, then the file name is modified immediately, otherwise it will be modified when an actChangeExt command is set in the Action property.

If the file rename is successful then the NewFileExt property is cleared, otherwise the ErrorCode property is set to feRenameError and NewFileExt remains unaltered.

See also

NewFileName

NewFileName Property

Applies to

TFileTool

Declaration

property NewFileName : TCtFullFileName

Description

The NewFileName property accepts user input of a new filename. If the AutoDisplay property is set, then the file name is modified immediately, otherwise it will be modified when an actChangeExt command is set in the Action property.

If the file rename is successful then the Newfilename property is cleared, otherwise the ErrorCode property is set to feRenameError and NewFileName remains unaltered.

See also
NewFileExt

ListDirs Property

Applies to

TSearchTool

Declaration

property ListDirs : TSelectList;

Description

The ListDirs property stores a list of the directories scanned during the search operation.

See also

SearchSubDirs

ListFiles Property

Applies to

TSearchTool

Declaration

property ListFiles : TSelectList;

Description

The ListFiles property stores a list of the files matched during the search process.

See also

ListsFullReport

ListsSortReport

CtGetUser Function

Unit
CtlSys

Declaration
function CtGetUser : string;

Description
The CtGetUser function returns the registered user name from the USER library.

Target
Windows, DOS Protected Mode (WinAPI unit)

Returns
The function returns the registered user name retrieved from the USER library.

Exceptions
ECtError	Message = **CtGetUser**

See also
CtGetCompany
CtGetUserAndCompany
CoreTools system routines

ListsFullReport Property

Applies to

TSearchTool

Declaration

property ListsFullReport : boolean;

Description

The ListsFullReport property decides whether a full file description, including drive and path, is inserted in the ListDirs and ListFiles lists. If the property is true then a full report is prepared, otherwise only the shortened version is prepared.

If the AutoDisplay property is true then the search will take place immediately, otherwise it will not happen until the user sets the Action property to fsaSearch.

See also
ListsSortReport

ListsSortReport Property

Applies to
TSearchTool

Declaration
property ListsSortReport : boolean;

Description
The ListsSortReport property decides whether the ListDirs and ListFiles string lists should be sorted in ascending order.

This is the equivalent of ListDirs.Sorted and ListFiles.Sorted.

If the AutoDisplay property is true then the search will take place immediately, otherwise it will not happen until the user sets the Action property to fsaSearch.

See also
ListsFullReport

SearchMask Property

Applies to

TSearchTool

Declaration

property SearchMask : string

Description

The SearchMask property allows the user to specify the file, files or directory to be searched for.

The property is set to *.* by default.

If the AutoDisplay property is true then the search will take place immediately, otherwise it will not happen until the user sets the Action property to fsaSearch.

SearchPath Property

Applies to

TSearchTool

Declaration

property SearchPath : TFileName;

Description

The SearchPath property allows the user to specify the directory from which the search will take place.

If the AutoDisplay property is true then the search will take place immediately, otherwise it will not happen until the user sets the Action property to fsaSearch.

SearchSubDirs Property

Applies to

TSearchTool

Declaration

property SearchSubDirs : boolean;

Description

The SearchSubDirs property specifies whether the search should take place in sub-directories as well as that specified by SearchPath.

If the AutoDisplay property is true then the search will take place immediately, otherwise it will not happen until the user sets the Action property to fsaSearch.

ListDirItems Property

Applies to

TSearchTool

Declaration

property ListDirItems : integer;

Description

The ListDirItems property stores the number of items in the ListDirs property.

CtGetUserAndCompany Procedure

Unit
CtlSys

Declaration
procedure CtGetUserAndCompany (var User, Company: string);

Description
The CtGetUserAndCompany procedure returns the registered company name and user name from the USER library.

Target
Windows, DOS Protected Mode (WinAPI unit)

Parameters
User	The registered user name.
Company	The registered company.

Returns
The function returns the registered user and company names in the reference parameters.

Exceptions
ECtError	Message = **CtGetUserAndCompany**

See also
CtGetCompany
CtGetUser
CoreTools system routines

CtGetEnvVar Function

Unit

CtlSys

Declaration

function CtGetEnvVar (VarName: string): string;

Description

Returns the environment variable parameters for the specified variable. The environment searched is that belonging to the current (running) task.

The returned string starts with the first character after the equals sign (=) in the environment entry specified by *VarName*.

Target

Windows, DOS Protected Mode (WinAPI unit)

Parameters

VarName	The environment variable to search for.

Returns

The function returns a string containing the environment variable declaration, or an empty string if *VarName* is not found.

See also

CtGetEnv

CtGetEnvUsed

CoreTools system routines

CtGetEnvUsed Function

Unit
CtlSys

Declaration
function CtGetEnvUsed: cardinal;

Description
The CtGetEnvUsed function returns the number of bytes actually stored in the DOS environment for the current (running) task.

Comments
This function is useful in measuring how efficiently you are using the space reserved for the environment.

Target
Windows, DOS Protected Mode (WinAPI unit)

Returns
The number of bytes used in the environment string.

See also
CtGetEnv
CtGetEnvVar
CoreTools system routines

Copyright acknowledgements

Environment Property

Applies to
TSysInfoTool

Declaration
property Environment: TSelectList;

Description
This list contains the programs environment variables, split into separate list items.

See also
CtGetEnv
CtGetEnvUsed
CtGetEnvVar

CtlError Unit

Description

This unit contains the CoreTools for Delphi error handling subsystem.

Types

ECtError
TCtError

CtGetCompany Function

Unit
CtlSys

Declaration
function CtGetCompany : string;

Description
The CtGetCompany function returns the registered company name from the USER library.

Target
Windows, DOS Protected Mode (WinAPI unit)

Returns
The function returns a string containing the registered company in the USER library.

See also
CtGetUser
CtGetUserAndCompany
CoreTools system routines

TFileNamePropertyEditor property editor

Unit

CteFilen

Description

The TFileNamePropertyEditor is dedicated to the TFileName type, and allows graphical editing of all properties of that type.

If the facility is not required, then simply comment-out the registration of the property editor in the CORETOOL.PAS file and rebuild the component library.

Note

TFileName   is declared in Delphi's SysUtils unit.

TSelectListPropertyEditor property editor

Unit

CteSelct

Description

This property editor is devoted to the TSelectList type, and allows viewing of the list contents.

Note

If the editor is not required, then simply comment-out the editor registration in CORETOOL.PAS, rebuild the component library, and the default TStringList editor will be implemented instead.

TTimeTextPropertyEditor property editor

Unit

CteTime

Description

This property editor is devoted to the TCtTimeText type, and allows graphical setting of the time string.

If the editor is not required, then simply comment-out the editor registration in CTLREG.PAS, rebuild the component library, and the default string editor will be implemented instead.

TDateTextPropertyEditor

Description

This property editor allows the developer to set a TCtDateText property value, using a calendar type editor.

If the facility is not required, then simply comment-out the registration of the property editor in the CORETOOL.PAS file and rebuild the component library.

DOS Version Constants

Unit

CtlSys

Declaration

CtDOSMajor : byte;

CtDOSMinor : byte;

Description

These typed constants are initialized at boot up to the current DOS version parameters.

Example

For DOS 6.2:

```
        CtDOSMajor = 6
        CtDOSMinor = 20
```

See also

CtWinMajor
CtWinMinor
CoreTools system routines

CtlSys Unit

Description
This unit contains the CoreTools for Delphi System Library.

Constants
CtDOSMajor
CtDOSMinor
CtWinMajor
CtWinMinor

Types
TCtWinSet
TCtWinSettings

Routines
CtGetCompany
CtGetEnv
CtGetEnvUsed
CtGetEnvVar
CtGetSysDir
CtGetUser
CtGetUserAndCompany
CtGetWinDir
CtGetWinSettings

See also
CoreTools system routines

TCtCRC16 Type

Unit

CtlCRC16

Declaration

TCtCRC16 = word;

Description

The type is used to store the CRC-16 (16 bit Cyclic Redundancy Check) value of a file.

DPMI Constants

Unit

CtlDPMI

Declaration

DPMI API functions

dpmiGetDesc	:word = $0000;	{ Allocate descriptors }
dpmiFreeDesc	:word = $0001;	{ Free descriptor }
dpmiSegToDesc	:word = $0002;	{ Map segment to descriptor }
dpmiGetSelInc	:word = $0003;	{ Get selector increment value }
dpmiGetSegBase	:word = $0006;	{ Get segment base address }
dpmiSetSegBase	:word = $0007;	{ Set segment base address }
dpmiSetSegSize	:word = $0008;	{ Set segment size }
dpmiGetRealInt	:word = $0200;	{ Get real mode interrupt handler }
dpmiSetRealInt	:word = $0201;	{ Set real mode interrupt handler }
dpmiGetExcept	:word = $0202;	{ Get exception handler }
dpmiSetExcept	:word = $0203;	{ Set exception handler }
dpmiGetInt	:word = $0204;	{ Get interrupt handler }
dpmiSetInt	:word = $0205;	{ Set interrupt handler }
dpmiGetRMCB	:word = $0303;	{ Get real mode call-back }
dpmiGetVersion	:word = $0400;	{ Get version information }

Flag masks

flagCarry	:word = $0001;
flagParity	:word = $0004;
flagAuxiliary	:word = $0010;
flagZero	:word = $0040;
flagSign	:word = $0080;
flagTrap	:word = $0100;
flagInterrupt	:word = $0200;
flagDirection	:word = $0400;
flagOverflow	:word = $0800;

CtGetEnv Procedure

Unit

CtlSys

Declaration

procedure CtGetEnv (L: TStringList);

Description

The CtGetEnv procedure inserts the environment string of the current (running) task into *L*, separating and inserting each variable into the list.

The procedure clears the list before inserting the environment variables.

The list can be parsed later using the TParseTool component to split each variable declaration into its separate parts.

Comments

Unlike an application, a dynamic-link library (DLL) does not have a copy of the environment string. As a result, a library must call this function to retrieve the environment string into a native Delphi string list.

Parameters

L	The list in which to insert the environment variables.

Target

Windows, DOS Protected Mode (WinAPI unit)

See also

CtGetEnvUsed

CtGetEnvVar

CoreTools system routines

CtGetSysDir Function

Unit

CtlSys

Declaration

function CtGetSysDir: string;

Description

The CtGetSysDir function retrieves the path of the Windows system directory.

The path that this function retrieves does not end with a backslash unless the system directory is the root directory. For example, if the system directory is named **WINDOWS\SYSTEM** on drive **C**, the path of the system directory retrieved by this function is **C:\WINDOWS\SYSTEM**. If Windows is installed in the root directory of drive **C**, the path retrieved is **C:\**.

The CtForceRightBackslash function will add the backslash character to the returned string if such a character is required.

Comments

The system directory contains such files as Windows libraries, drivers, and font files. Applications should not create files in the system directory. If the user is running a shared version of Windows, the application will not have write access to the system directory.

Applications should create files only in the directory returned by the CtGetWinDir function.

Target

Windows

Returns

The function returns the system directory as a Pascal string.

See also

CtGetWinDir

CoreTools system routines

CoreTools string conversion routines

Description

Routines to convert values between different types.

A PZStr is only a PChar with a fixed memory allocation of 256 bytes. Therefore, the standard Delphi routines for conversion between String, PString and PChar can be freely used on the PZStr type. Routines have only been provided to supplement those already existing in Delphi.

Routines

CtInsituPasToStr	Converts a Pascal string to a null terminated string, using the original string memory.
CtInsituStrToPas	Converts a null terminated string to a Pascal string, using the original PChar memory.
CtPCharToPZStr	Converts a null terminated string to a PZStr.
PercentToText	Converts a cardinal value to a text string with % character appended.
CtStringToPZStr	Converts a Pascal string value to a PZStr value.
CtPZStrToString	Converts a PZStr value to a Pascal string.
CtPZStrToPString	Converts a PZStr value to a PString, optionally allows deallocation of the original value.

Caveat

PZStr strings should only be deallocated with the CtFreePZStr function.

See also

Pascal-string handling routines
PZStr handling routines
Swapping routines

CtInsituPasToStr Function

Unit

CtlStrng

Declaration

function CtInsituPasToStr (P: PString): PChar;

Description

This function converts a Pascal string to a null terminated string, using the original string memory.

Parameters

P	A pointer to the Pascal string to be converted.

Returns

The function returns a pointer to the same memory as the P argument, the zero terminated string having been converted to a Pascal string.

Exceptions

ECtError: **CtlStrng: CtInsituPasToStr**

See also

CtInsituStrToPas
CoreTools string conversion routines
CoreTools string handling routines

CtlStrng Unit

Description

This unit contains the CoreTools for Delphi Pascal string library routines.

Routines

CtBoolToResult
CtBoolToState
CtBoolToString
CtCase
CtCharCount
CtCharPos
CtCharPosNext
CtCharReplace
CtCJustify
CtCount
CtDetab
CtEntab
CtFill
CtFirstCapital
CtFirstCapitalPos
CtForceRightChar
CtForceRightBackslash
CtInsituPasToStr
CtInsituStrToPas
CtIsCharInString
CtIsNumeric
CtLeft
CtLJustify
CtLowerCase
CtLRotate
CtLShift
CtLStrip
CtLStripSet
CtOverlay
CtParse
CtParseClean
CtPercentToText
CtPosSet
CtRCharPos
CtRemove
CtReplace
CtReverse
CtRight
CtRJustify
CtRPos
CtRRotate
CtRShift
CtRStrip
CtRStripSet
CtSplit

CtSqueeze
CtStrip
CtStripAll
CtStripSetAll
CtTrim
CtUpperCase
CtWordCount
CtWordExtract
CtWordNext
CtWordPos
CtWordProperCase

See also

CtlSwap Unit
CtlZStrn Unit
CoreTools string handling routines

TMaxBuffer Type

Unit

CtcBase

Declaration

TMaxBuffer = array[1..65521] of byte;

Description

Largest buffer that can be allocated on the heap.

CoreTools Pascal string handling library

Description

A library of routines to supplement the Delphi Pascal string handling routines.

Routines

CtBoolToResult	Returns a string showing the status of a boolean parameter in Result format.
CtBoolToState	Returns a string showing the status of a boolean parameter in State format.
CtBoolToString	Returns a string showing the status of a boolean parameter in String format.
CtCase	Performs quote-string aware case conversions on a string.
CtCharCount	Returns the number of instances of a specified character in a string.
CtCharPos	Returns the index position of the first instance a character in a string.
CtCharPosNext	Searches for the next occurrence of a character in a string.
CtCharReplace	Replaces all instances of a character with a replacement.
CtCJustify	Center justifies a string.
CtCount	Returns the number of instances of one string in another string.
CtDetab	Converts tab characters in a string to spaces.
CtEntab	Converts spaces in a string to tab characters.
CtFill	Fills a string with a specified number characters.
CtFirstCapital	Returns the character of the first capital letter in a string.
CtFirstCapitalPos	Returns the position of the first capital letter in a string.
CtForceRightChar	Forces a specified trailing character on to the right of a string.
CtForceRightBackslash	Forces a backslash character on to the right of a string.
CtInsituPasToStr	Converts a Pascal string to a null terminated string, using the original Pascal string memory.
CtInsituStrToPas	Converts a null terminated string to a Pascal string, using the original PChar memory.
CtIsCharInString	Checks whether a specified character is in a string.
CtIsNumeric	Checks a string to see if it contains only numerals.
CtLeft	Returns characters from the left of a string argument.
CtLJustify	Returns a left hand justified string.
CtLowerCase	Returns a string with all uppercase case characters converted to lowercase.
CtLRotate	Rotates all characters within a string one position to the left.
CtLShift	Shifts a character from the left side of a string.
CtLStrip	Strips all instances of a character from the left side of a string.
CtLStripSet	Strips all instances of a character(s), specified in a set, from the left side of a string.
CtOverlay	Overlays a string onto another string starting at a specified index position.
CtParse	Parses a string into separate text tokens.
CtParseClean	Cleans up a string making it ready to be efficiently parsed.
CtPercentToText	Converts a TPercent value to a text string.
CtPosSet	Searches a string for the first character contained in a set.
CtRCharPos	Searches for a character in a string, starting the search from the right hand side.
CtRemove	Removes all instances of a sub string from a string.
CtReplace	Replaces all instances of a sub string with a replacement sub string.
CtReverse	Reverses and returns a string argument.
CtRight	Returns characters from the right of a string argument.
CtRJustify	Returns a right hand justified string.
CtRPos	Search for a sub string, starting from the right side of the string.
CtRRotate	Rotates all characters within a string one position to the right.
CtRShift	Shifts a character from the right side of a character.

CtRStrip	Strips a specified character from the right hand side of a string.
CtRStripSet	Strips all instances of a character(s), specified in a set, from the right side of a string.
CtSplit	Splits a string on the first occurrence of a specified character.
CtSqueeze	Condenses repeated occurrences of a character in a string into a single character.
CtStrip	Procedure to remove a specified character from a string. The character can be stripped from left, from right, both left and right, or from all of the string.
CtStripAll	Removes every instance of a character from a string.
CtStripSetAll	Removes all instances of a set of characters from a string.
CtSwapString	Swaps the contents of two string variables.
CtTrim	Removes tabs, nulls and spaces from the right hand side of a string.
CtUpperCase	Returns a string with all lowercase case characters converted to uppercase.
CtWordCount	Returns the count of the number of words in a string.
CtWordExtract	Returns a number of specified words from a string.
CtWordNext	Returns the next word from a string.
CtWordPos	Returns the starting index position of a specified word number.
CtWordProperCase	Converts a string to proper case, i.e. the first character of each word is capitalized.

See also

PZStr handling routines
String conversion routines
Swapping routines

CtInsituStrToPas Function

Unit

CtlStrng

Declaration

function CtInsituStrToPas (P: PChar): PString;

Description

The CtInsituPCharToString function converts a null terminated string to a Pascal string, using the original memory pointed to by the PChar parameter.

If the length of *P* > 255, then it is truncated to a length of 255 characters.

This function is useful when making calls to API functions which only accept and return null terminated strings.

Parameters

P	A pointer to a null terminated string. The contents of the memory pointed to are converted to a Pascal type string.

Returns

The function returns a pointer to a Pascal type string.

Exceptions

ECtError: Message = **CtlStrng: CtInsituStrToPas**

See also

CtInsituPasToStr

CoreTools string conversion routines

CoreTools string handling routines

CtLeft Function

Unit

CtlStrng

Declaration

function CtLeft (S: string; N: byte): string;

Description

The CtLeft function returns the leftmost *N* characters of the string argument *S*.

Parameters

S	string to copy from
N	number of characters to copy

Returns

The CtLeft function returns the left *N* characters of *S*. If *N* > length of *S*, then *S* is returned.

Exceptions

ECtError	**CtlStrng: CtLeft**

See also

CtLRotate
CtLShift
CtRight
CtRRotate
CtRShift
CoreTools string handling routines

CtRight Function

Unit

CtlStrng

Declaration

function CtRight (S: string;   N: byte): string;

Description

The CtRight function returns the rightmost *N* characters of the string argument *S*.

Parameters

S	the string from which to extract the sub string
N	the number of characters to extract

Returns

The function returns the rightmost *N* characters of *S*. If *N* > length of *S*, then *S* is returned.

Exceptions

ECtError	**CtlStrng: CtRight**

See also

CtLeft

CoreTools string handling routines

CtRPos Function

Unit

CtlStrng

Declaration

function CtRPos (Sub, S: string): byte;

Description

Searches for the sub string *Sub*, starting from the right side of the string *S*.

Parameters

Sub	The sub string to search for.
S	The string to search.

Returns

The function returns the index position of the first match.
Returns 0 if no match found.

Exceptions

ECtError	**CtlStrng: CtRPos**

See also

CtCharPos
CtCharPosNext
CtPosSet
CtRCharPos
CtWordPos
CoreTools string handling routines

CtReverse Function

Unit

CtlStrng

Declaration

function CtReverse (S: string): string;

Description

Reverses and returns the string argument *S*.

Parameters

S	The string to reverse.

Returns

The reversed string.

Exceptions

ECtError	**CtlStrng: CtReverse**

See also

CoreTools string handling routines

CtFirstCapitalPos Function

Unit

CtlStrng

Declaration

function CtFirstCapitalPos(const S : string): byte;

Description

This function returns the index position of the first capital letter in the string *S*.

Parameters

S	The string to search for the capital letter.

Returns

The function returns the index position of the first capital letter in *S.*
Returns 0 if no capital letter is found.

Exceptions

ECtError	**CtlStrng: CtFirstCapitalPos**

See also

CtFirstCapital
CoreTools string handling routines

CtWordProperCase Function

Unit

CtlStrng

Declaration

function CtWordProperCase (S: string): string;

Description

The CtWordProperCase function converts *S* to proper case, i.e. the first character of each word is capitalized, and the remainder of the string is unaltered.

Parameters

S	The string to capitalize.

Returns

The capitalized string.

Exceptions

ECtError	**CtlStrng: CtWordProperCase**

See also

CtWordCount
CtWordExtract
CtWordNext
CtWordPos
CoreTools string handling routines

CtFirstCapital Function

Unit

CtlStrng

Declaration

function CtFirstCapital (const S : string): char;

Description

Returns the first capital letter in the string *S*.

Parameters

S	The string to search for the capital letter.

Returns

The function returns the first capital letter in *S*.

The function returns a null #0 if the string does not contain any capital letters.

Exceptions

ECtError	Message = **CtFirstCapital**

See also

CtFirstCapitalPos

CoreTools string handling routines

CtWordCount Function

Unit

CtlStrng

Declaration

function CtWordCount (S: string): byte;

Description

Returns the count of the number of words in the string *S*.
The function only recognizes spaces as word delimiters, therefore any tabs should firstly be converted to spaces, perhaps by using CtParseClean.

Parameters

S	The string to search for words.

Returns

The number of words in the string.

Exceptions

ECtError	**CtlStrng: CtWordCount**

See also

CtWordExtract
CtWordNext
CtWordPos
CtWordProperCase
CoreTools string handling routines

CtStrip Procedure

Unit

CtlStrng

Declaration

procedure CtStrip (StripCode: TCtStrip; C: char; S: string): string;

Description

Removes a specified character from a string. The character can be removed from left, right, both left and right or from all of the string.

Parameters

StripCode	The type of stripping required
C	The character to strip
S	The string to strip.

Returns

The stripped string.

Exceptions

ECtError	**CtlStrng: CtStrip**

See also

CtLStrip
CtLStripSet
CtRemove
CtRStrip
CtRStripSet
CtSqueeze
CtStripAll
CtStripSetAll
CtTrim
CoreTools string handling routines

CtBoolToState Function

Unit
CtlStrng

Declaration
function CtBoolToState(B: boolean): string;

Description
Returns a string showing the state of the boolean *B* in status format, either **On** or **Off**.

Parameters
B	The subject of the conversion operation.

Returns
The function returns a textual status description for the state of *B*.

Exceptions
ECtError	**CtlStrng: CtBoolToState**

See also
CtBoolToResult
CtBoolToString

TCtStrip Type

Unit

CtlStrng

Declaration

TCtStrip = (stripLeft, stripRight, stripBoth, stripAll);

Description

The TCtStrip enumeration is used with the CtStrip function to specify the form of stripping required.

CoreTools swapping routines

Description

Routines to swap values between two variables of the same type.

Routines

CtSwapByte	Swaps the contents of two byte variables.
CtSwapCardinal	Swaps the contents of two cardinal variables.
CtSwapChar	Swaps the contents of two character variables.
CtSwapComp	Swaps the contents of two complex variables.
CtSwapDouble	Swaps the contents of two double variables.
CtSwapExtended	Swaps the contents of two extended variables.
CtSwapInteger	Swaps the contents of two integer variables.
CtSwapLongInt	Swaps the contents of two longint variables.
CtSwapPChar	Swaps the contents of two PChar variables, also applies to PZStr types.
CtSwapReal	Swaps the contents of two real variables.
CtSwapShortInt	Swaps the contents of two shortint variables.
CtSwapSingle	Swaps the contents of two single variables.
CtSwapString	Swaps the contents of two string variables.
CtSwapWord	Swaps the contents of two word variables.

See also

Pascal string handling routines
PZStr handling routines
String conversion routines

CtSwapString Procedure

Unit

CtlSwap

Declaration

procedure CtSwapString (var x, y: string);

Description

Swaps the contents of two strings.

Parameters

x and *y* are the strings to be swapped.

Exceptions

ECtError	Message = **CtlSwap: CtSwapString**

See also

CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapWord
CoreTools swapping routines
CoreTools string-handling routines

CtlSwap Unit

Description

The CtlSwap unit contains routines that swap variable values.

Routines

CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord

See also

CoreTools swapping routines

CtSwapCardinal Procedure

Unit

CtlSwap

Declaration

procedure CtSwapCardinal (var x, y: cardinal);

Description

Swaps the contents of two cardinal values.

Parameters

x and *y* are the values to be swapped.

Exceptions

ECtError	Message = **CtlSwap: CtSwapCardinal**

See also

CtSwapByte
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines

ListFilesItems Property

Applies to

TSearchTool

Declaration

property ListFileItems : integer;

Description

The ListFileItems property stores the number of items in the ListFiles property.

CtSwapInteger Procedure

Unit

CtlSwap

Declaration

procedure CtSwapInteger (var x, y: longint);

Description

Swaps the contents of two integer values.

Parameters

x and *y* are the values to be swapped.

Exceptions

EctError   Message = **CtlSwap: CtSwapInteger**

See also

CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines

CtSwapComp Procedure

Unit
CtlSwap

Declaration
procedure CtSwapComp (var x, y: comp);

Description
Swaps the contents of two complex variables.

Parameters
x and *y* are the values to be swapped.

Exceptions
ECtError Message = **CtlSwap: CtSwapComp**

See also
CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines

CtSwapChar Procedure

Unit
CtlSwap

Declaration
procedure CtSwapChar (var x, y: char);

Description
Swaps the contents of two char variables.

Parameters
x and *y* are the values to be swapped.

Exceptions
ECtError Message = **CtlSwap: CtSwapChar**

See also
CtSwapByte
CtSwapCardinal
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines

CtCount Function

Unit

CtlStrng

Declaration

function CtCount (Pattern, S : string): byte;

Description

The CtCount function returns the number of instances of *Pattern* in the source string, *S*.

Parameters

Pattern	The pattern to search for.
S	The string on which perform the search.

Returns

The total of pattern matches.

Exceptions

ECtError	**CtlStrng: CtCount**

See also

CtCharCount

CoreTools string handling routines

CtSplit Function

Unit

CtlStrng

Declaration

function CtSplit (S: string; C: char; var Before, After: string): byte;

Description

Splits *S* on the first occurrence of the character *C*.
The character *C* is not included in either new sub string.
If *C* is not found, then *S* is copied to *Before* and *After* is a zero length string.

Parameters

S	the string to split.
C	the character to search for.
Before	the sub string before the split character.
After	the sub string after the split character.

Returns

The function returns the index position in *S* at which the split occurred. Returns 0 if no split.

Exceptions

ECtError	**CtlStrng: CtSplit**

See also

CoreTools string handling routines

CtIsNumeric Function

Unit

CtlStrng

Declaration

function CtIsNumeric (S: string): boolean;

Description

The CtIsNumeric function checks a string, *S,* to test if it contains only numerals.

Parameters

S	The string that is the subject of the check operation.

Returns

True if only numerals in string, otherwise returns False.

Exceptions

ECtError	**CtlStrng: CtIsNumeric**

See also

CoreTools string handling routines

Inventory of files

The distribution file, CORE1A.ZIP, contains the following files:

CTREG.PAS	Palette registration unit
CTREG.DCR	Component bitmaps
CTCBASE.DCU	The superclass for all components
CTCFBASE.DCU	The superclass for file based components
CTCFILE.DCU	TFileTool component
CTCPARSE.DCU	TParseTool component
CTCSCAN.DCU	TScanTool component
CTCSERCH.DCU	TSearchTool component
CTCSYSIN.DCU	TSysInfoTool component
CTEABOUT.DCU	TAboutInfoPropertyEditor code
CTEABOUT.DFM	TAboutInfoPropertyEditor form
CTEDATE.DCU	TDateTextPropertyEditor code
CTEDATE.DFM	TDateTextPropertyEditor form
CTEFILEN.DCU	TFileNamePropertyEditor code
CTESELCT.DCU	TSelectListPropertyEditor code
CTESELCT.DFM	TSelectListPropertyEditor code
CTETIME.DCU	TTimeTextPropertyEditor code
CTETIME.DFM	TTimeTextPropertyEditor form
CTLCHAR.DCU	Character classification library
CTLCRC16.DCU	CRC-16 library
CTLDATE.DCU	Date and time routines
CTLDPMI.DCU	DPMI library
CTLERROR.DCU	Error handling unit
CTLFILE.DCU	File handling routines
CTLSTRNG.DCU	String library
CTLSWAP.DCU	Swap library
CTLSYS.DCU	System routines
CTLZSTRN.DCU	PZStr library
CTSWITCH.INC	The file containing switch settings used when building the toolset.
CORETOOL.HLP	CoreTools for Delphi help file
CORETOOL.KWF	CoreTools for Delphi help keyword file

Technical notes

Naming conventions

Routine prefixes

Following the accepted practise amongst vendors of software libraries, all **CoreTools for Delphi** routines start with a unique identifier, in our case **Ct**, e.g. *CtReadDirectory*.

This convention has arisen due to the problems of conflicting names between language standard library functions and other vendors supplying routines that perform the same or similar functionality.

The convention has the advantage of informing you immediately of the source of any routine you meet in your code.

Component names

All our component names comply with the Borland practise of starting all types with the letter **T,** and end with the Suffix **Tool**, e.g. *TScanTool*.

Type names

All type declarations use the prefix **TCt**, e.g. *TCtFullFileName*.

Constant names

All constant declarations use the prefix **Ct**, e.g. *CtMaxDriveNumber*.

Exception names

Following the Borland example, all exceptions have the prefix **E**, again followed by the **Ct** identifier, e.g. *ECtCRCError*.

Performance

Whenever performance is desirable, our routines are coded in assembler. Future upgrades will feature more assembler routines and resultant speed gains.

The spelling used in this manual

US spelling is the international standard for computing documentation, and this document has complied with that standard, albiet not always succesfully!

See also

Compiler switch settings

TPercentText Type

Unit

CtcBase

Declaration

TPercentText = string[4];

Description

The TPercentText type stores a cardinal percentage value in 0..100 range.

Windows Version Constants

Unit

CtlSys

Declaration

CtWinMajor : byte;

CtWinMinor : byte;

Description

These typed constants are initialized at bootup to the current Windows version parameters.

Example

For Windows .3.2:

```
        CtWinMajor = 3
        CtWinMinor = 20
```

See also

CtDOSMajor

CtDOSMinor

CoreTools system routines

CtGetWinDir Function

Unit

CtlSys

Declaration

function CtGetWinDir: string;

Description

The CtGetWinDir function retrieves the path of the Windows directory as a Pascal string.

The path that this function retrieves does not end with a backslash unless the Windows directory is the root directory. For example, if the system directory is named **WINDOWS** on drive **C**, the path of the system directory retrieved by this function is **C:\WINDOWS**. If Windows is installed in the root directory of drive **C**, the path retrieved is **C:\**.

The CtForceRightBackslash function will add the backslash character to the returned string if such a character is required.

Comments

The Windows directory contains such files as Windows applications, initialization files, and help files.

The Windows directory is the only directory where an application should create files. If the user is running a shared version of Windows, the Windows directory is the only directory guaranteed private to the user.

Target

Windows

Returns

The function returns the Windows directory as a Pascal string.

See also

CtGetSysDir

CoreTools system routines

CtGetWinSettings Function

Unit

CtlSys

Declaration

function CtGetWinSettings: TCtWinSettings;

Description

The CtGetWinSettings function retrieves the current Windows system and memory configuration.

Target

Windows, DOS Protected Mode (WinAPI unit).

Returns

The function returns, as a set, the current system and memory configuration.   The configuration returned by CtGetWinSettings can be a combination of the TCtWinSettings values.

Exceptions

None

See also

CoreTools system routines

CoreTools CRC-16 library

Description

The library routines calculate a cyclic redundancy check value, known as **CRC-16**, using the polynomial **X^16 + X^15 + X^2 + 1.**

Routines

CtCRC16FileCalculate	Calculates the CRC-16 value for a specified file.
CtCRC16FileCopy	Calculates the CRC-16 value and copies a specified file
CtCRC16String	Calculates the CRC-16 value for a Pascal string.
CtCRC16Update	Calculates the CRC-16 value for a buffer.

Exceptions

All routines generate the ECtCRC16 exception.

CtlCRC16 Unit

Description

Contains the CoreTools CRC-16 routines.

Routines

CtCRC16Update

CtCRC16FileCalculate

CtCRC16FileCopy

Exceptions

ECtCRC16

CtCRC16FileCopy Function

Unit

CtlCRC16

Declaration

function CtCRC16FileCopy (InFileName, OutFileName: TFileName): TCtCRC16**;**

Description

Calculates the CRC-16 value for *InFileName*. If *OutFileName* is specified then *InFileName* is copied to *OutFileName*. In both cases, the CRC-16 value is returned.

The DateTime stamp of *OutFileName* is set to the same as *InFileName*.

This function can be used to verify the integrity of a file's contents, and is especially useful for detecting viruses, defective media or unauthorized modifications. If you use this function to copy a file you can then store the CRC-16 value separately from the original file, and can later verify the file's integrity by calculating the CRC-16 on *OutFileName* and comparing that value with the stored CRC-16 value. If the values do not match, the file has been modified or corrupted.

The routine uses CoreBuffer in which to read the file's contents, and you should ensure that you are not using that buffer for your own purposes when calling this routine.

Parameters

InFileName	The file to be CRC-16 checked and the source for the CRC-16 copy.
OutFileName	The target file for the CRC-16 copy.

Returns

The CRC-16 value of *In Filename*.

Exceptions raised

ECtCRC16	**CTLCRC16: Same source and destination**
	CTLCRC16: Cannot open input file
	CTLCRC16: Cannot open output file
	CTLCRC16: I/O error
	CTLCRC16: Calculation failed

See also

CtCRC16FileCalculate

CtCrc16String

CtCRC16Update

CtCRC16Update Function

Unit

CtlCRC16

Declaration

function CtCRC16Update (var Buffer; Value, Count: integer) : TCtCRC16**;**

Description

The CtCRC16Update function is used to calculate the CRC-16 value of data that is too large to fit into memory in one pass, the function is called consecutively, producing an accumulated CRC-16 value.

The first iteration should be with *Value* initialized to zero and passing *Buffer,* containing the data to be checked, along with the buffer size in *Count*. Continue calling CtCRC16Update for subsequent blocks, passing the previous value of *Value* until the data requiring the CRC-16 check has all passed through the function.

Parameters

Buffer	the buffer to be added to the calculation.
Value	the accumulated CRC-16 value.
Count	the size, in bytes, of the buffer.

Returns

The function returns the CRC-16 value of the buffer.

Exceptions

ECtCRC16 Message = **CTLCRC16: Calculation failed**

ERangeError

See also

CtCRC16FileCalculate

CtCRC16FileCopy

CtCRC16String

CtCRC16FileCalculate Function

Unit
CtlCRC16

Declaration
function CtCRC16FileCalculate (F: TFileName): TCtCRC16**;**

Description
The CtCRC16FileCalculate function computes and returns the CRC-16 value for the contents of *F*. This function can be used to verify the integrity of a file's contents, and is especially useful for detecting viruses, defective media or unauthorized modifications. If you calculate a file's CRC-16 value and store the value separately from the original file, you can later verify the file's integrity by calculating the CRC-16 on the current version and comparing that value with the stored CRC-16 value. If the values do not match, the file has been modified or corrupted.

The routine uses CoreBuffer in which to read the file's contents, and you should ensure that you are not using that buffer for your own purposes when calling this routine.

Parameters

F	The name of the file requiring CRC-16 check. If the file is not in the current directory, then *F* must contain the drive and/or directory if either is different from the default drive or current directory.

Returns
The function returns the CRC-16 value for *F*.

Exceptions

ECtCRC16	**CTLCRC16: Cannot open file** **CTLCRC16: file read error** **CTLCRC16: Calculation failed**

See also
CtCRC16FileCopy
CtCRC16String
CtCRC16Update

ECtCRC16 Exception

Unit
CtlCRC16

Declaration
ECtCRC16 = class (Exception);

Description
This exception is raised for all CTLCRC16 library error conditions. The Message property defines the exact error condition that occurred.

Messages
CTLCRC16: Cannot open input file <filename>
CTLCRC16: Cannot open output file <filename>
CTLCRC16: Input error
CTLCRC16: I/O error
CTLCRC16: Calculation failed

ECtError Exception

Unit

CtlError

Declaration

ECtError = class(Exception);

Description

Raised for all CoreTools for Delphi components error conditions. The Message property defines the exact error condition that occurred, see also TCtError type.

Messages for each error code

feOpenFile	**Open file error**
feCloseFile	**Close file error**
feInvalidDirectory	**Invalid directory**
feReadError	**Read error**
feInvalidFile	**Invalid file**
feSettingFileAttr	**Fail setting file attributes**
feCreateFile	**Create file error**
feDeleteError	**Delete file error**
feErrorDuringCopy	**Copy file error**
feRenameError	**Rename file error**
feLineTooLong	**Line too long error**
feFieldTooLong	**Field too long error**
feConversionError	**Time Conversion error**
feDateConversion	**Date conversion error**
feTimeConversion	**Time conversion error**
feSameSourceAndTarget	**Same source and destination**
feCRCError	**CRC-16 error**
feInvalidFieldIndexSet	**Invalid field index set**
feInvalidFieldIndexGet	**Invalid field index get**

TSysInfoTool Component

Unit

CtcSysIn

Description

The TSysInfoTool component provides system information from DOS and Windows.

Properties

About	Product and component version information and facilities to register your product.
Environment	A list containing the applications environment variables.
FreeGDI	Returns the percentage of free space for GDI resources.
FreeMEM	Returns the amount of available memory, in bytes.
FreeSYS	Returns the percentage of free space for system resources.
FreeUSR	Returns the percentage of free space for USER resources.
MaxMEM	Returns the size of the largest contiguous memory block available.
NameCompany	Returns the Company Name registered with Windows.
NameUser	Returns the User Name registered with Windows.
PathSystem	Returns the path of the Windows system directory.
PathWindows	Returns the path of the Windows directory.
SaverDelay	Returns the screen-saver delay time-out in minutes.
VerDOS	Returns the installed DOS version number.
VerWin	Returns the installed Windows version number.
WinSettings	Returns a set containing the windows processor and memory configuration.

Methods

None

Events

OnError

FreeGDI Property

Applies to
TSysInfoTool

Declaration
property FreeGDI : TPercentText

Description
Returns as text the percentage of free space for GDI resources. GDI resources include device-context handles, brushes, pens, regions, fonts, and bitmaps.

See also
FreeMEM
FreeSYS
FreeUSR

FreeMEM Property

Applies to

TSysInfoTool

Declaration

property FreeMEM : longint

Description

Returns the amount of available memory, in bytes.

Note that a contiguous block of storage the size of the returned value is unlikely to be available due to fragmentation of the heap. To find the largest free block, use the MaxMEM property.

In standard mode, the value returned represents the number of bytes in the global heap that are not used and that are not reserved for code.

In 386-enhanced mode, the return value is an estimate of the amount of memory available to an application. It does not account for memory held in reserve for non-Windows applications.

See also

FreeGDI
FreeSYS
FreeUSR
MaxMEM

FreeSYS Property

Applies to

TSysInfoTool

Declaration

property FreeSYS : TPercentText

Description

Returns as text the percentage of free space for system resources.

See also

FreeGDI
FreeMEM
FreeUSR

FreeUSR Property

Applies to

TSysInfoTool

Declaration

property FreeUSR : TPercentText

Description

Returns as text the percentage of free space for USER resources. These resources include window and menu handles.

See also

FreeGDI
FreeMEM
FreeSys

NameCompany Property

Applies to

TSysInfoTool

Declaration

property NameCompany : string;

Description

The NameCompany property returns the company name registered with Windows API.

See also

NameUser

NameUser Property

Applies to

TSysInfoTool

Declaration

property NameUser : string;

Description

The NameUser property returns the user name registered with Windows API.

See also

NameCompany

PathSystem Property

Applies to

TSysInfoTool

Declaration

property PathSystem : string;

Description

Returns the path of the Windows system directory. The system directory contains such files as Windows libraries, drivers, and font files.

Applications should not create files in the system directory. If the user is running a shared version of Windows, the application will not have write access to the system directory.

Applications should create files only in the directory returned by the PathWindows property.

The path that this property returns ends with a backslash.

See also

PathWindows

PathWindows Property

Applies to

TSysInfoTool

Declaration

property PathWindows : string;

Description

Returns the path of the Windows directory. The Windows directory contains such files as Windows applications, initialization files, and help files.

The Windows directory is the only directory where an application should create files. If the user is running a shared version of Windows, the Windows directory is the only directory guaranteed private to the user.

The path this property returns ends with a backslash

See also

PathSystem

SaverDelay Property

Applies to
TSysInfoTool

Declaration
property SaverDelay : word;

Description
The SaverDelay property returns the screen-saver delay time-out as minutes.

VerDOS Property

Applies to

TSysInfoTool

Declaration

property VerDOS : string;

Description

Returns the version number of the installed DOS operating system.

See also

VerWin

VerWin Property

Applies to

TSysInfoTool

Declaration

property VerWin : string;

Description

Returns the version number of the installed Windows operating system.

See also

VerDOS

CtcSysIn Unit

Description

This unit contains the TSysInfoTool component.

Types

TSysInfoTool

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

Complier switch settings

The CoreTools for Delphi toolset has been constructed using the following compiler switches:

```
{$A-}       {Align Data}
{$B-}       {Complete Boolean Evaluation}
{$C MOVEABLE DEMANDLOAD DISCARDABLE} { Code Segment Attribute }
{$D-}       {Debug Information}
{$F-}       {Force FAR Calls}
{$G-}       {Generate 286 Instructions}
{$I+}       {Input/Output Checking}
{$L-}       {Local Symbol Information}
{$N+}       {8087 Math Coprocessor Support}
{$P+}       {Open String Parameters}
{$Q+}       {Arithmetic Overflow Checking}
{$R+}       {Range Checking}
{$S+}       {Stack-Overflow Checking}
{$T+}       {Typed @ Operator}
{$U-}       {Pentium-Safe FDIV}
{$V+}       {Var-String Checking, overriden by $P}
{$W-}       {Windows Stack Frame}
{$X+}       {Extended Syntax}
{$Y-}       {Symbol Reference Information}
{$Z-}       {Word Size Enumerated Types}
```

CtForceRightChar Function

Unit

CtlStrng

Declaration

function CtForceRightChar (var S: string; const C:char): string;

Description

Forces a trailing character, *C*, on to the string *S*.

If the right hand side character is already set to *C* then no additional action is taken, otherwise *C* is appended to the right hand side of *S*.

Parameters

S	The string that is to be the subject of the operation.
C	The character to force append.

Returns

The function returns a copy of *S* with the specified character forced onto the RIGHT HAND SIDE of the string.

Exceptions

ECtError **CtlStrng: CtForceRightChar**

See also

CtForceRightBackslash

CoreTools string handling routines

CtPercentToText Function

Unit

CtlStrng

Declaration

function PercentToText (P: TPercent): TPercentText;

Description

Converts *P* to a text string of the type TPercentText. The % character is appended to the right side of the string.

Parameters

P	The value to convert.

Returns

A string representation of the % value.

Exceptions

ECtError	**CtlStrng: CtPercentToText**

See also

CoreTools string handling routines

CtForceRightBackslash Function

Unit

CtlStrng

Declaration

function CtForceRightBackslash (S:string): string;

Description

Forces a trailing backslash character on to the string *S*.

If the right hand character is already set to backslash then no additional action is taken, otherwise a backslash is appended to the right of *S*.

Parameters

S	The string on which to append the backslash (\) character.

Returns

A copy of the parameter *S*, with a backslash character (\) forced onto the RIGHT HAND SIDE if one does not already exist.

Exceptions

ECtError	**CtlStrng: CtForceRightBackslash**

See also

CtForceRightChar

CoreTools string handling routines

CtPZStrToString Function

Unit

CtIZStrn

Declaration

function CtPZStrToString (Source: PZStr; const DeleteSource: boolean): string;

Description

Creates a new Pascal string and copies to it the contents of *Source*. *Source* is deallocated if *DeleteSource* is set true..

Parameters

Source	the string to be copied
DeleteSource	if true, then *Source* is deallocated

Returns

A string containing the contents of *Source*.

Exceptions

ECtError   Message = **CtIZStrn: CtPZStrToString**

See also

CtFreePZStr
CtNewPZStr
CtPCharToPZStr
CtPZStrToPString
CtStringToPZStr
CoreTools PZStr handling routines

CtStringToPZStr Function

Unit
CtlZStrn

Declaration
function CtStringToPZStr (const S: string): PZStr;

Description
Constructs a new PZStr on the heap, able to contain 255 characters and the null terminator, and copies to it the contents of *S*.
Strings created using this function can be freely used with Pascal type strings as they are long enough to avoid heap corruption, the Windows API routines are the best example where the PZStr type is useful..

Caveat
The returned string should always be deallocated using CtFreePZStr, as that ensures that all 256 characters allocated are correctly deallocated.

Parameters
S	the Pascal string to convert.

Returns
A PZStr value, pointing to a PZStr containing the text copied from *S*.

Exceptions
ECtError	**CtlZStrn: CtStringToPZStr**

See also
CtFreePZStr
CtNewPZStr
CtPCharToPZStr
CtPZStrToPString
CtPZStrToString
CoreTools PZStr handling routines

CtPZStrToPString Function

Unit

CtlZStrn

Declaration

function CtPZStrToPString (Source: PZStr; const DeleteSource: boolean): PString;

Description

Creates a new Pascal string on the heap and copies to it the contents of *Source*. Optionally *Source* can be deallocated at the same time.

The resulting PString dynamic string should be deallocated with DisposeStr.

Caveat

Do not change the length of the resultant string. Increasing the length of the string overwrites other variables on the heap. Decreasing the length of the string prevents some of the memory from being deallocated.

Parameters

Source	The null terminated string from which the contents are to be copied.
DeleteSource	If true, *Source* is deallocated.

Returns

The function returns a new PString containing the contents of *Source*.

Exceptions

ECtError Message = **CtlZStrn: CtPZStrToPString**

See also

CtFreePZStr
CtNewPZStr
CtPCharToPZStr
CtPZStrToString
CtStringToPZStr
CoreTools PZStr handling routines

CtPCharToPZStr Function

Unit

CtlZStrn

Declaration

function CtPCharToPZStr (Source: PChar; const DeleteSource:boolean): PZStr;

Description

Converts *Source*, a null terminated string, to a PZChar string, which is a 256 byte null terminated string. Optionally, *Source* can be deallocated. If *Source* is longer than 255 bytes, then only the first 255 bytes are copied.

Caveat

The resulting PZStr string must be deallocated using CtFreePZStr, and not the StrDispose procedure.

Parameters

Source	The null terminated string from which the contents are to be copied.
DeleteSource	If true, *Source* is deallocated.

Returns

The function returns a PZStr pointer to a TZStr null terminated string containing the contents of *Source*.

Exceptions

ECtError Message = **CtlZStrn: CtPCharToPZStr**

See also

CtFreePZStr
CtNewPZStr
CtPZStrToPString
CtPZStrToString
CtStringToPZStr
CoreTools PZStr handling routines

CtFreePZStr Procedure

Unit

CtlZStrn

Declaration

procedure CtFreePZStr (Str: PZStr);

Description

Deallocates a PZStr created using CtNewPZStr, CtPCharToPZStr or CtStringToZStr functions. This ensures that the full 256 characters originally allocated are deallocated correctly.

Parameters

Str	The string pointer to be deallocated.

Exceptions

EInvalidPointer	
ECtError	**CtlZStrn: CtFreePZStr**

See also

CtNewPZStr
CtPCharToPZStr
CtPZStrToPString
CtPZStrToString
CtStringToPZStr
CoreTools PZStr handling routines

CoreTools PZStr handling routines

Description

The PZStr routines are a library of routines to support the PZStr type, which is a dynamic, null terminated string of 256 bytes in length, the same length as a Pascal type string. Using the PZStr type avoids the risk of memory heap corruption when copying strings between the two types.

The standard Delphi routines for conversion between String, PString and PChar can be freely used on the PZStr type. Routines have only been provided to supplement those already existing in Delphi.

Routines

CtFreePZStr	Deallocates a PZStr type string.
CtNewPZStr	Allocates heap memory for a PZStr string.
CtPCharToPZStr	Converts a null terminated string to a PZStr.
CtPZStrToPString	Converts a PZStr value to a PString, optionally allows deallocation of the original value.
CtPZStrToString	Converts a PZStr value to a Pascal string.
CtStringToPZStr	Converts a Pascal string value to a PZStr value.

See also

CoreTools string handling routines
CoreTools string conversion routines
CoreTools swapping routines

CtNewPZStr Function

Unit

CtlZStrn

Declaration

function CtNewPZStr: PZStr;

Description

Allocates heap memory for a PZStr type. Deallocation should always be performed with the CtFreePZStr function.

Returns

A pointer to a PZStr string, allocated on the heap. If memory is not available, then the EOutOfMemory exception is raised.

Exceptions

ECtError	Message = **CtlZStrn: CtNewPZStr**
EOutOfMemory	

See also

CtFreePZStr
CtPCharToPZStr
CtPZStrToPString
CtPZStrToString
CtStringToPZStr
CoreTools PZStr handling routines

CtSwapPChar Procedure

Unit

CtlSwap

Declaration

procedure CtSwapPChar (var x, y: PChar);

Description

Swaps two PChar values.

Parameters

x,y	values to swap.

Exceptions

ECtError	Message = **CtlSwap: CtSwapPChar**

See also

CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines
CoreTools Pascal string handling routines

PZStr Type

Unit

CtlZStrn

Declaration

TZStr = array[0..255] of char;

PZStr = ^TZStr;

Description

The PZStr type is comparable to the PChar type, except for the fact that it has its own memory allocation and deallocation functions, which ensure that a fixed length of 256 bytes, the same as a Pascal string, is allocated and correctly deallocated.

The type can be used with Pascal type strings without fear of heap corruption, as may occur if a PChar of less than 255 characters capacity is used and the Pascal string contains more characters than the memory allocated to the PChar.

See also

CtFreePZStr
CtNewPZStr
CtPCharToPZStr
CtPZStrToPString
CtPZStrToString
CtStringToPZStr
CoreTools PZStr handling routines

Technical Support

Who is eligible?

Only registered license holders of CoreTools for Delphi are eligible for techincal support.

Before you contact us

In order to save everyone some time, we'd like you to do a few things before you contact us. We know the urge to call is great, but
we'd appreciate it if you double check this list:

- Have you read the whole manual? All of it?
- Have you used the search feature of the manual?
- Have you looked at the demo source?

If you got this far, it must be a tough one. Maybe it's time to tell us about your problem.

How to contact us

The preferred method of contacting us for technical support is on CompuServe, including through services such as MCI Mail, BIX, the Internet, etc. You can send any questions or problems to our CompuServe address. Do not leave questions on CompuServe forums as they probably won't be anwsered.

We currently offer free telephone support on an "as available" basis, Monday through Friday usually from 9 a.m. to 6 p.m UK time. Please note the time before you call. Sorry, we cannot return international calls.

Although there is currently no expiry date on free support, we reserve the right to restrict it to the first 90 days from the time of purchase, without any prior notice.

MaxMEM Property

Applies to

TSysInfoTool

Declaration

property MaxMEM : longint

Description

The MaxMEM property returns the size of the largest contiguous free block in the heap.   MaxMEM returns the larger of:

- The largest free blocks within the heap manager's sub-allocation space
- The Windows global heap

The value corresponds to the size of the largest dynamic variable that can be allocated at that time. To find the total amount of free memory in the heap, check FreeMEM.

See also

FreeGDI
FreeMEM
FreeSYS
FreeUSR

Address, Phone, Email, etc.

You can contact us through any of the following:

Post: Core Software Limited, 3 Tearne Street, St Johns, WORCESTER, WR2 6BL, UK

Phone: +44 1905 420 784

CompuServe: 100041,3143

Internet: 100041.3143@compuserve.com

Others: For other online services, such as America Online, Prodigy, MCI Mail, etc., please see your manuals about contacting CompuServe or the Internet, and use the addresses listed above.

Due to time differences we cannot return international calls.

Preferred

Send e-mail.

CtlZStrn Unit

Description

This unit contains routines supporting the PZStr type.

Types

PZStr
TZStr

Routines

CtFreePZStr
CtNewPZStr
CtPCharToPZStr
CtStringToPZStr
CtPZStrToString
CtPZStrToPString

See also

CtlStrng Unit

TransferName Property

Applies to

TFileTool

Declaration

property TransferName: TFileName

Description

Specifies the new name for copy and move actions.

See also

faCopy
faCopyVerify
faMove
faMoveVerify

IsRegistered Property

Applies to

All CoreTools components

Declaration

property IsRegistered: boolean;

Description

Returns whether the file CORE.LIC is present in the Windows directory and contains a valid serial number for CoreTools for Delphi.

See also

InDesignMode

InDesignMode Property

Applies to

All CoreTools components

Declaration

property InDesignMode: boolean;

Description

Returns whether Delphi is currently running.

See also

IsRegistered

CtIsAlpha Function

Unit

CtlChar

Declaration

function CtIsAlpha (C: char): boolean;

Description

Tests *C* to check if it is an alphabetical character. Characters checked for are: A-Z or a-z.

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlNum
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsLower
CtIsPrint
CtIsPunct
CtIsReal
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

CoreTools character classification routines

Description

This library contains assembler coded boolean functions that classify characters within a given group in the lower ASCII table (<#128).

The routines are comparable in speed to using character sets.

Routines

CtIsAlNum	Tests for alpha-numeric character
CtIsAlpha	Tests for alphabetic character
CtIsASCII	Tests for lower ASCII table character
CtIsCntrl	Tests for printer control character
CtIsDigit	Tests for numbers zero through nine
CtIsGraph	Tests for a "black" printable character
CtIsLower	Tests for lower case character
CtIsPrint	Tests for printable character
CtIsPunct	Tests for punctuation character
CtIsReal	Tests for real number character
CtIsSigned	Tests for signed number character
CtIsSpace	Tests for non-printing paper-movement character
CtIsUpper	Tests for upper case character
CtIsXDigit	Tests for hex digit character

See also

Character classification sets

CtlDPMI Unit

Description

This unit contains the DPMI library routines.

Routines

CtRealIntr

Types

TCtRegs

Constants

dpmiFreeDesc	dpmiGetDesc	dpmiGetExcept
dpmiGetInt	dpmiGetRealInt	dpmiGetRMCB
dpmiGetSelInc	dpmiGetSegBase	dpmiGetVersion
dpmiSegToDesc	dpmiSetExcept	dpmiSetInt
dpmiSetRealInt	dpmiSetSegBase	dpmiSetSegSize
flagAuxiliary	flagCarry	flagDirection
flagInterrupt	flagOverflow	flagParity
flagSign	flagTrap	flagZero

See also

CoreTools DPMI routines
DPMI Constants

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

TFileBaseTool Component

Unit

CtcFBase

Description

TFileBaseTool is the abstract super-class from which all file based CoreTools for Delphi components descend.

See also

Component logical model

CtcFBase Unit

Description

CtcFBase is the unit containing the abstract super-class, TFileBaseTool, from which all file based CoreTools for Delphi components descend.

Types

TCtDateText
TCtFileExt
TCtFileNameOnly
TCtFullFileName
TCtTimeText
TFileBaseTool

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

CtIsAlNum Function

Unit

CtlChar

Declaration

function CtIsAlNum (C: char): boolean;

Description

Tests *C* to check if it is a alphanumeric character. Alphanumeric characters are: A-Z or a-z or a digit 0-9.

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsLower
CtIsPrint
CtIsPunct
CtIsReal
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

CtIsDigit Function

Unit

CtlChar

Declaration

function CtIsDigit (C: char): boolean;

Description

Tests *C* to check if it is a decimal-digit character. The check performed is: 0-9.

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsGraph
CtIsLower
CtIsPrint
CtIsPunct
CtIsReal
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

CtIsGraph Function

Unit

CtlChar

Declaration

function CtIsGraph (C: char): boolean;

Description

Tests *C* to check if it is a printing character, excluding blank space (' '). The check performed is: .#33-#126

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsLower
CtIsPrint
CtIsPunct
CtIsReal
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

CtIsLower Function

Unit

CtlChar

Declaration

function CtIsLower (C: char): boolean;

Description

Tests *C* to check if it is a lowercase character. The check performed is: .a-z.

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsPrint
CtIsPunct
CtIsReal
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

CtIsPrint Function

Unit

CtlChar

Declaration

function CtIsPrint (C: char): boolean;

Description

Tests *C* to check if it is a printing character, including the blank space (' '). The check performed is: .#32 - #126.

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsLower
CtIsPunct
CtIsReal
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

CtIsPunct Function

Unit

CtlChar

Declaration

function CtIsPunct (C: char): boolean;

Description

Tests *C* to check if it is a punctuation character. The check performed is: .CtIsGraph but not CtIsAlNum.

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsLower
CtIsPrint
CtIsReal
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

CtIsReal Function

Unit

CtlChar

Declaration

function CtIsReal (C: char): boolean;

Description

Tests *C* to check if it is a real number character. The check performed is: 0-9 + Minus . e E

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsLower
CtIsPrint
CtIsPunct
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

CtIsSigned Function

Unit

CtlChar

Declaration

function CtIsSigned (C: char): boolean;

Description

Tests *C* to check if it is a signed number character. The check performed is: 0-9 + Minus

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsLower
CtIsPrint
CtIsPunct
CtIsReal
CtIsSpace
CtIsUpper
CtIsXDigit

CtIsSpace Function

Unit
CtlChar

Declaration
function CtIsSpace (C: char): boolean;

Description
Tests *C* to check if it is a space character. The check performed is: #09-#13

Parameters
C	the character on which to perform the test.

Returns
True if the test is successful, otherwise returns False.

Exceptions
None

See also
CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsLower
CtIsPrint
CtIsPunct
CtIsReal
CtIsSigned
CtIsUpper
CtIsXDigit

CtIsUpper Function

Unit

CtlChar

Declaration

function CtIsUpper (C: char): boolean;

Description

Tests *C* to check if it is an uppercase character. The check performed is: A-Z

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsLower
CtIsPrint
CtIsPunct
CtIsReal
CtIsSigned
CtIsSpace
CtIsXDigit

CtlsXDigit Function

Unit

CtlChar

Declaration

function CtlsXDigit (C: char): boolean;

Description

Tests *C* to check if it is a hexadecimal character. The check performed is: 0-9 A-F a-f

Parameters

C	the character on which to perform the test.

Returns

True if the test is successful, otherwise returns False.

Exceptions

None

See also

CtlsAlNum
CtlsAlpha
CtlsASCII
CtlsCntrl
CtlsDigit
CtlsGraph
CtlsLower
CtlsPrint
CtlsPunct
CtlsReal
CtlsSigned
CtlsSpace
CtlsUpper

CtlChar Unit

Description

This unit contains character classification routines.

Routines

CtIsAlNum
CtIsAlpha
CtIsASCII
CtIsCntrl
CtIsDigit
CtIsGraph
CtIsLower
CtIsPrint
CtIsPunct
CtIsReal
CtIsSigned
CtIsSpace
CtIsUpper
CtIsXDigit

Constants

CtAlphaSet
CtAlphaNumSet
CtASCIISet
CtControlSet
CtDelimSet
CtDigitSet
CtGraphSet
CtLowerSet
CtPrintSet
CtPunctSet
CtQuoteSet
CtRealSet
CtSignedSet
CtSpaceSet
CtUpperSet
CtXDigitSet

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

CtCRC16String Function

Unit

CtlCRC16

Declaration

function CtCRC16String (S: string): TCtCRC16**;**

Description

The CtCRC16String function returns the CRC-16 value for the string argument.

Parameters

S	The string that is the subject of this operation.

Returns

The function returns the CRC-16 value of the string.

Exceptions

ECtCRC16 Message = **CTLCRC16: Calculation failed**

See also

CtCRC16FileCalculate
CtCRC16FileCopy
CtCRC16Update

CtRealIntr Function

Unit

CtlDPMI

Declaration

function CtRealIntr (InterruptNumber: byte; var SimRegisters: TCtRegs): boolean;

Description

The CtRealIntr function simulates an interrupt in real mode using DPMI function $0300. When the interrupt is simulated, the registers will contain the values input in *SimRegisters*. When the interrupt returns, *SimRegisters* will contain the values returned by the real mode interrupt.

It is recommended that when the user creates an instance of TCtRegs that the Delphi **FillChar** procedure is used first to set all registers to zero before use, as the memory allocated is not cleared by the Windows API before allocation.

Parameters

InterruptNumber	The interrupt to be simulated.
SimRegisters	The instance of TCtRegs that is used to pass register values to the DPMI API and to return register values to the caller.

Returns

The function returns true if the DPMI call was successful, otherwise returns false. In addition, if the interrupt that you invoked modified the contents of any of the registers, the new values are retuned in *SimRegisters*.

Exceptions

None

Remarks

When passing buffer addresses to the BIOS or DOS it is essential that these should have been allocated using the Windows API function **GlobalDOSAlloc**, which allocates memory addressable by DOS in 'real mode' and Windows applications in protected mode.

Memory allocated by **GlobalDOSAlloc** should only be released using **GlobalDOSFree**. It is important that the buffer should be released as soon as possible, because the memory pool from which the object is allocated is a scarce system resource.

The **GlobalDosAlloc** function allocates global memory that can be accessed by MS-DOS running in real mode. The memory is guaranteed to exist in the first megabyte of linear address space.The return value contains a paragraph-segment value in its high-order word and a selector in its low-order word. An application can use the paragraph-segment value to access memory in real mode and the selector to access memory in protected mode. If Windows cannot allocate a block of memory of the requested size, it returns zero.

When copied to the pseudo register, the segment/selector address obtained from the Windows API function **GlobalDOSAlloc** should be handled thus:

```
    globalDosBuffer := GlobalDOSAlloc( size );
    ...
    realModeReg.DS := HiWord(globalDosBuffer);
```

When used in the application:

```
    P^ := Pointer(MakeLong(0, LoWord(globalDosBuffer)));
    ...
    GlobalDOSFree(globalDOSBuffer);
```

In the example above, the pseudo DS register and P^ are addressing the same memory by different routes. The pseudo register is using real mode segment:offset addressing, whilst the P^ is using Protected modes selector:offset addressing.

Memory allocated by using the GlobalDosAlloc function does not need to be locked by using the Windows API **GlobalLock** function.

There is no guarantee that an interrupt is supported under Windows - many are not. You are entirely on your own (like all of us) at the mercy of Windows.

This routine was written to handle those services or interrupts that use any memory addresses or 'undocumented' DOS calls that require 'real mode', or lower DOS addresses.

Most normal INT $21 functions are supported under Windows, and those that are will accept the protected-mode memory addresses available from the Windows **GlobalDOSAlloc** function. For those interrupts and functions, you can write asm code directly in Object Pascal, using the Windows DOS3Call, as its a great deal quicker than using the DPMI API to switch from protected mode to real mode and back.

Caveat

You should NOT use this routine with

- INTERRUPT $20, INTERRUPT $21 function 0, INTERRUPT $21 function $4C (all of which terminate programs)
- INTERRUPT $27 and INTERRUPT $21 function $31, which both make programs memory-resident
- INTERRUPT $25 and $26 (read/write absolute).

CoreTools System routines

Description
This library contains DOS and Windows system routines.

Routines

CtGetCompany	Returns the registered company name.
CtGetEnv	Parses the environment and inserts all environment variables into a list.
CtGetEnvUsed	Returns the number of bytes actually used in the DOS environment.
CtGetEnvVar	Returns the environment variable parameters for a specified variable
CtGetSysDir	Returns the path of the Windows system directory as a Pascal string.
CtGetUser	Returns the registered user name.
CtGetUserAndCompany	Returns the registered user name and company name.
CtGetWinDir	Returns the path of the Windows directory as a Pascal string.
CtGetWinSettings	Returns the current Windows system and memory configuration.

See also
CtDOSMajor
CtDOSMinor
CtWinMajor
CtWinMinor

CoreTools DPMI support

Description

The DOS Protected Mode Interface (DPMI) was defined to allow DOS programs to access the extended memory of PC architecture computers whilst maintaining system protection.

DPMI defines a specific subset of DOS and BIOS calls that can be made by protected mode DOS programs. It also defines a new interface via software interrupt $31 that protected mode programs use to allocate memory, modify descriptors, call real mode software, etc.

Windows claims to be capable of supporting DPMI without affecting system security.

Routines

CtRealIntr	Simulates an interrupt in real mode using DPMI function $0300.

See also

DPMI Constants

TCtRegs Type

Unit

CtlDPMI

Declaration

PCtRegs = ^TCtRegs;

TCtRegs = record		Offset	Register
EDI:	longint;	**$00**	**EDI**
ESI:	longint;	**$04**	**ESI**
EBP:	longint;	**$08**	**EBP**
reserved:	longint;	**$0C**	**Reserved**
EBX:	longint;	**$10**	**EBX**
EDX:	longint;	**$14**	**EDX**
ECX:	longint;	**$18**	**ECX**
EAX:	longint;	**$1C**	**EAX**
FLAGS:	word;	**$20**	**Flags**
ES:	word;	**$22**	**ES**
DS:	word;	**$24**	**DS**
FS:	word;	**$26**	**FS**
GS:	word;	**$28**	**GS**
IP:	word;	**$2A**	**IP**
CS:	word;	**$2C**	**CS**
SP:	word;	**$2E**	**SP**
SS:	word;	**$30**	**SS**

end;

Description

The TCtRegs record is a pseudo register set for real mode DPMI services.

See also

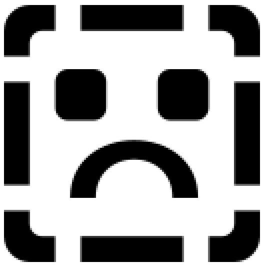
CtRealIntr

TAboutInfo Type

Used internally by CoreTools for Delphi.

Component logical model

The logical model of the CoreTool for Delphi components, in Booch notation, is shown below:



CtlFile Unit

Description
This unit contains the file handling library.

Types

TCtFileExt	TCtFileNameOnly	TCtFullFileName
TCtFoundFunction	TCtFileInfo	

Routines

CtExtractFileDrive	CtExtractFileNameOnly	CtExtractPathNameOnly
CtFileAgeCmp	CtFileAttrFull	CtFileAttrShort
CtFileAttrToSet	CtFileAttrToWord	CtFileCompare
CtFileCopy	CtFileDateStr	CtFileDateTime
CtFileInformation	CtFileLineCount	CtFileSetAttr
CtFileSetStamp	CtFileSize	CtIsRootDir
CtParseFileSpec	CtReadDirectory	CtRemakeDirRoot
CtVerifyOFF	CtVerifyON	CtVerifyState

See also
CoreTools file-handling routines

CoreBuffer Variable

Unit
CtcBase

Declaration
CoreBuffer : array[1..16384] of byte;

Description
The CoreBuffer variable is the shared file buffer used internally by CoreTools for Delphi libraries and components. There is little performance gain with a larger buffer and experimentation has shown this to be the best compromise between memory usage and performance.

Caveat
If the buffer is employed by users for their own use, care must be taken to ensure that there is no conflict with any CoreTools usage.

CoreTools file handling routines

Description

This library contains routines that process files.

Routines

CtExtractFileDrive	Extracts the drive information from a file specification.
CtExtractFileNameOnly	Extracts the file name, without the extension, from a file specification.
CtExtractPathNameOnly	Extracts the path, without the drive information, from a file specification.
CtFileAgeCmp	Compares the date and time stamps of two files
CtFileAttrFull	Returns a full textual description of a file attribute set.
CtFileAttrShort	Returns a short description of a file attribute set.
CtFileAttrToSet	Returns an attribute set corresponding to a DOS attribute word.
CtFileAttrToWord	Converts a file attribute set to a DOS file attribute word.
CtFileCompare	Performs byte-by-byte comparison of two files.
CtFileCopy	Copies a file.
CtFileDateStr	Returns the date/time stamp of the specified file as a string.
CtFileDateTime	Returns the date/time stamp of the specified file as a TDateTime value.
CtFileInformation	Returns file information, including the file date/time, size, and attributes.
CtFileLineCount	Returns the number of lines in a text file.
CtFileSetAttr	Sets a files attributes to those contained in an attribute set.
CtFileSetStamp	Sets the date/time stamp for the specified file.
CtFileSize	Returns the size of a file in bytes.
CtIsRootDir	Tests if a path specification is a root directory.
CtParseFileSpec	Fills a TStringList object with the separate parts of a file specification string.
CtReadDirectory	Searches a specified directory for a file mask specification.
CtRemakeDirRoot	Extracts a specified root path and replaces it with another.
CtVerifyOFF	Turns the DOS verify flag off.
CtVerifyON	Turns on the DOS verify flag.
CtVerifyState	Returns the state of the DOS verify flag.

See also

CoreTools string handling routines

TCtFoundFunction Type

Unit

CtlFile

Declaration

TCtFoundFunction = function (const F:TFileName; const searchRecord:TSearchRec): boolean of object;

Description

The TCtFoundFunction type defines a callback function used in the CtReadDirectory function parameters.

See also

CtReadDirectory

CtBoolToResult Function

Unit

CtlStrng

Declaration

function CtBoolToResult (B: boolean): string;

Description

Returns a string showing the state of the boolean *B* in text format, either **Passed** or **Failed**.

Parameters

B	The subject of the conversion operation.

Returns

The function returns a textual result description for the state of *B*.

Exceptions

ECtError	**CtlStrng: CtBoolToResult**

See also

CtBoolToState
CtBoolToString

CtCase Function

Unit

CtlStrng

Declaration

function CtCase (const Options:TCtCaseConvertSet; const S:string) : string;

Description

Performs quote-string aware case conversions on a string.

By default the function does not perform case conversion on portions of a string embedded between quote characters.

The default can be overriden by setting the caseIgnoreQuotes attribute in *Options*, however the CtUpperCase and CtLowerCase functions will perform that simple conversion much faster than this specialist function.

Parameters

Options	a set of conversion options
S	the string that is the subject of this operation.

Returns

Returns a copy of *S* with character conversion performed in accordance with the *Options* settings.

Exceptions

ECtError	**CtlStrng: CtCase**

See also

CtLowerCase
Ctwordpropercase
CtUpperCase

CtBoolToString Function

Unit

CtlStrng

Declaration

function CtBoolToString (B: boolean): string;

Description

Returns a string showing the state of the boolean *B* in text format, either **True** or **False**.

Parameters

B	The subject of the conversion operation.

Returns

The function returns a textual result description for the state of *B*.

Exceptions

ECtError	**CtlStrng: CtBoolToString**

See also

CtBoolToResult
CtBoolToState

CtUpperCase Function

Unit

CtlStrng

Declaration

function CtUpperCase (const S:string):string;

Description

This function converts all lowercase characters in *S* to uppercase. *S* remains unaltered.

Parameters

S	The string that is the subject of the convert operation.

Returns

The function returns a copy of S with all lowercase characters converted converted to uppercase.

Exceptions

None

See also

CtCase

CtLowerCase

Ctwordpropercase

CoreTools string handling routines

TCtCaseConvertSet Type

Unit

CtlStrng

Declaration

```
type
  TCtCaseConvertSet = set of (caseIgnoreQuotes, caseUpper, caseLower);
```

Description

This set dictates the case conversion operations performed by the CtCase routine.

caseIgnoreQuotes	When set overrides the default characteristic, which is to ignore quoted strings when performing case conversion operations. Quotes recognized are **'** and **"**.
caseUpper	When set, all characters in the string are converted to upper case, other than the default of those in quote strings.
caseLower	When set, all characters in the string are converted to lower case, other than the default of those in quote strings.

CtLowerCase Function

Unit

CtlStrng

Declaration

function CtLowerCase (const S:string):string;

Description

This function converts all uppercase characters in S to lower case. S remains unaltered.

Parameters

S	The string that is the subject of the convert operation.

Returns

The function returns a copy of S with all uppercase characters converted converted to lowercase.

Exceptions

None

See also

CtCase
Ctwordpropercase
CtUpperCase
CoreTools string handling routines

CtCharCount Function

Unit

CtlStrng

Declaration

function CtCharCount (C:char; const S:string): byte;

Description

The CtCharCount function scans a string counting occurrences of a specified character.

Parameters

C	the character to be searched for.
S	the string that is the subject of this operation.

Returns

The function returns the number of occurrences of *C* in *S*.

Exceptions

None

CtDetab Function

Unit

CtlStrng

Declaration

procedure CtDetab(var D, S: string; TabStops: byte);

Description

The function CtDetab returns string *S* in string *D* with tab character(s) (ASCII $09) converted to space(s). The number of space(s) is from the last non-space character to the next tab stop.

Parameters

D	The destination string in which the converted copy of S will be returned.
S	The source string on which the operation will be performed.
TabStops	The tab stop interval. *TabStops* must be >= 2.

Returns

None

Exceptions

ECtError:	**CtlStrng: CtDetab**

See also

CtEntab

CtCharPos Function

Unit

CtlStrng

Declaration

function CtCharPos (C:char; S:string): byte;

Description

The CtCharPos function returns the index position of the first instance a character in a string.

This function performs the same operation as the Pascal POS function except that it works only with a single character.

Parameters

C	the character to search for.
S	the string that is to be the subject of the operation.

Returns

Returns the first index position of *C* in *S*, if not found returns 0.

Exceptions

None

See also

CtCharPosNext
CtPosSet
CtRCharPos
CtRPos
CtWordPos

CtCJustify Function

Unit

CtlStrng

Declaration

function CtCJustify (S:string; C:char; L:byte): string;

Description

Places the source string *S* in the center of a string *L* characters long. Any additional characters that have to be added are implemented using *C* category characters.

Parameters

S	the source string.
L	the new string length.
C	the character to expand the string with.

Returns

The function returns a copy of *S* center justified in a string of *L* length, using *C* characters to pad the additional characters necessary to achieve the specified length.

Exceptions

ECtError:	**CtlStrng: CtCJustify**

See also

CtLJustify
CtRJustify

CtCharPosNext Function

Unit

CtlStrng

Declaration

function CtCharPosNext (C:char; S:string; StartPos:byte):byte;

Description

This function searches for the next occurrence of *C* in *S* after position *StartPos*.
Note: the character at *StartPos* is not tested.

Parameters

C	character to search for.
S	string to search.
StartPos	starting index position.

Returns

The function returns the offset from the beginning of the string, NOT the offset from *StartPos*.

Exceptions

None

See also

CtCharPos
CtPosSet
CtRCharPos
CtRPos
CtWordPos

CtWordPos Function

Unit

CtlStrng

Declaration

function CtWordPos (const N:byte; const S:string): byte;

Description

Determines the starting index position of a word.

Parameters

N	The number of the word to check.
S	The source string on which the operation will be performed.

Returns

The function returns the starting index position of the specified word number. If there are too few words in the source string, then 0 is returned.

Exceptions

ECtError **CtlStrng: CtWordPos**

See also

CtWordCount
CtWordExtract
CtWordNext
CtWordProperCase
CoreTools string handling routines

CtRJustify Function

Unit

CtlStrng

Declaration

function CtRJustify (S:string; C:char; L:byte): string;

Description

The CtRJustify function expands and right justifies a string.

Parameters

S	The source String.
C	The character to expand the string with.
L	The new string length.

Returns

Returns a right hand justified version of *S*, padded with character *C*, and of length *L*.

Exceptions

ECtError	**CtlStrng: CtRJustify**

See also

CtCJustify

CtLJustify

CoreTools string handling routines

CtEntab Function

Unit

CtlStrng

Declaration

procedure CtEntab (var D, S: string; TabStops: byte);

Description

The function returns a copy of string *S* in string *D* with space character(s) converted to tab(s) (ASCII $9).
A tab replaces space characters from the last non-space character to the next tab stop.

Parameters

S	The source string on which the operation will be performed.
D	The destination string for the entabbing operation.
TabStops	The tab stop interval. *TabStops* must be >= 2.

Returns

None

Exceptions

ECtError: **CtlStrng: CtEntab**

See also

CtDetab

CtFill Function

Unit

CtlStrng

Declaration

function CtFill (C:char; N:byte): string;

Description

This function returns a string containing *N* instances of *C*.

Parameters

C	The character to use to fill the string.
N	The number of instances of *C* required.

Returns

The function returns a string filled with the specified characters.

Exceptions

None

See also

CtOverlay

CtReplace

CtParse Procedure

Unit
CtlStrng

Declaration
procedure CtParse(const S: string; const Separators: TCtCharSet**; const Quotes:** TCtCharset**;**
const FilterQuotes: boolean; List: TStringList**);**

Description
The procedure CtParse parses *S* into separate text tokens, placing the results into a TStringList object.
The function parses the passed string using the separator characters defined by the *Separators* parameter. These usually include space, tab, null, comma, period and slashes.

Text strings delimited by *Quotes*, such as **'** or **"**, can be treated as a single field if *Quotes* are specified, otherwise text in the delimited strings is parsed into separate tokens. This allows, for instance, complete text strings in the parsed tokens.

If *FilterQuotes* is true and *Quotes* are specified, then the quote characters will be filtered (removed) from the resultant text tokens.

The index position in *S* of each text token is added to the Objects property of the TStringList object. The property is a pointer and the value has to be cast to a word to be used correctly.

The text tokens can be accessed later using the normal indexing of the TStringList class and its Count property to show the number of test tokens parsed.

Parameters

S	The string that is to be the subject of the parsing operation.
Separators	Characters in this set are used as delimiters to separate the text tokens extracted from the string.
Quotes	If any characters are set, then strings in quote characters are treated as a single token.
FilterQuotes	If true, and *Quotes* are specified, then the quote characters are removed from the text tokens.
List	The TStringList object in which to insert the text tokens.

Exceptions
ECtError **CtlStrng: CtParse**

See also
CtParseClean

CtIsCharInString Function

Unit

CtlStrng

Declaration

function CtIsCharInString (C:char; S:string): boolean;

Description

The CtIsCharInString function checks whether a specified character is in a string.

Parameters

C	The character to search for.
S	The string to search.

Returns

This function returns True if *C* is encountered, otherwise returns False.

Exceptions

None

See also

CoreTools string handling routines

CtLRotate Function

Unit

CtlStrng

Declaration

function CtLRotate (var S:string): integer;

Description

The CtLRotate function rotates all characters within a string one position to the left and returns the ASCII value of the original string.

Parameters

S	The string that is the subject of the operation.

Returns

The function returns the ASCII value of the first character of the original string.

If the string is null on entry, the function returns -1.

Exceptions

ECtError	**CtlStrng: CtLRotate**

See also

CtLShift

CtRRotate

CtRShift

CoreTools string handling routines

CtLShift Function

Unit

CtlStrng

Declaration

function CtLShift (var S:string): integer;

Description

The CtLShift function shifts a character of the left end of the string (shortening its length) and returns the ASCII code of the original string.

Parameters

S	The string that is to be the subject of the operation.

Returns

The function returns the ASCII code of the character removed from the original string.

If the string is null on entry, then the function returns -1.

Exceptions

ECtError	**CtlStrng: CtLShift**

See also

CtLRotate

CtRRotate

CtRShift

CoreTools string handling routines

CtRRotate Function

Unit

CtlStrng

Declaration

function CtRRotate (var S:string): integer;

Description

The CtRRotate function rotates all characters within a string one position to the right and returns the ASCII value of the first character of the resultant string.

Parameters

S	The string that is the subject of the operation.

Returns

The function returns the ASCII value of the last character of the original string.
If the string is null on entry, the function returns -1.

Exceptions

ECtError	**CtlStrng: CtRRotate**

See also

CtLRotate
CtLShift
CtRShift
CoreTools string handling routines

CtRShift Function

Unit

CtlStrng

Declaration

function CtRShift (var S:string): integer;

Description

The CtRShift function shifts a character from the right end of the string (shortening its length) and returns the ASCII code of the deleted character.

Parameters

S	The string that is to be the subject of the operation.

Returns

The function returns the ASCII code of the character removed from the right hand side of the original string.

If the string is null on entry, then the function returns -1.

Exceptions

ECtError	**CtlStrng: CtRShift**

See also

CtLRotate

CtLShift

CtRRotate

CoreTools string handling routines

CtLJustify Function

Unit

CtlStrng

Declaration

function CtLJustify (const S:string; C:char; L:byte): string;

Description

Returns a left hand justified string padded on the right to the specified length, using the specified pad character.

Parameters

S	The string to extend to the specified length, using the pad character, *C*.
C	The character to use to pad the the string to the specified length.
L	The required length to pad to.

Returns

Returns a modified version of the string *S*, padded to the the specified length with the pad character, *C*.

Exceptions

ECtError	**CtlStrng: CtLJustify**

See also

CtCJustify

CtRJustify

CtOverlay Function

Unit

CtlStrng

Declaration

function CtOverlay (const Overlay, Underlay:string; const P:byte):string;

Description

The function CtOverlay overlays *Overlay* onto *Underlay* starting at index P, i.e. the function combines the two overlapping strings.

Parameters

Overlay	The string to be added to the right hand side of the resultant string, starting at index position *P*.
Underlay	The string forming the left side of the combined string, up to index position *P*-1.
P	The character position that string *Overlay* will be overlaid on string *Underlay*. Any characters after the *P*th position in string *Underlay* will be overwritten by the characters in string *Overlay*.

Returns

Returns an amalgamation of *Underlay* and *Overlay*, with *Overlay* starting at index *P*.

Exceptions

ECtError	Message = **CtOverlay**

See also

CtFill

CtReplace

CoreTools string-handling routines

CtLStrip Function

Unit

CtlStrng

Declaration

function CtLStrip (C:char; S:string): string;

Description

Strips all instances of the character C from the left side of the string S.

Parameters

C	the character to search and strip.
S	the string that is the subject of this operation.

Returns

Returns the stripped string, with all instances of the specified character removed from the left of the string.

Exceptions

ECtError	**CtlStrng: CtLStrip**

See also

CtLStripSet
CtRemove
CtRStrip
CtRStripSet
CtSqueeze
CtStrip
CtStripAll
CtStripSetAll
CtTrim
CoreTools string handling routines

CtLStripSet Function

Unit

CtlStrng

Declaration

function CtLStripSet(CS: TCtCharSet**; S: string): string;**

Description

Strips the left hand side characters from *S* that are specified in the *CS* character set.

Parameters

S	the string that is to be the subject of the operation.
CS	Character set containing the characters to be searched for during the operation.

Returns

The function returns a copy of the string S, without the left hand side characters specified in CS.

Exceptions

ECtError	**CtlStrng: CtLStripSet**

See also

CtLStrip
CtRemove
CtRStrip
CtRStripSet
CtSqueeze
CtStrip
CtStripAll
CtStripSetAll
CtTrim
CoreTools string handling routines

CtRemove Function

Unit

CtlStrng

Declaration

function CtRemove(const S, Unwanted : string; var Count: byte) : string;

Description

Removes all instances of *Unwanted* from the string *S*. The number of instances removed is reported back in the *Count* reference.

Parameters

S	The string that is to be the subject of the operation.
Unwanted	The string that is searched for in the search and delete operation.
Count	A reference that facilitates the reporting back of the number of instances removed from the string.

Returns

The function returns a copy of *S* with all instances of *Unwanted* removed.

Exceptions

ECtError	**CtlStrng: CtRemove**

See also

CtStripAll

CtRStripSet Function

Unit

CtlStrng

Declaration

function CtRStripSet(CS: TCtCharSet; S: string): string;

Description

Strips the right hand side characters from *S* that are specified in the *CS* character set.

Parameters

S	the string that is to be the subject of the operation.
CS	Character set containing the characters to be searched for during the operation.

Returns

The function returns a copy of the string S, without the right hand side characters specified in CS.

Exceptions

ECtError	**CtlStrng: CtRStripSet**

See also

CtLStrip
CtLStripSet
CtRemove
CtRStrip
CtSqueeze
CtStrip
CtStripAll
CtStripSetAll
CtTrim
CoreTools string handling routines

CtSqueeze Function

Unit

CtlStrng

Declaration

function CtSqueeze (const C: char; const S: string): string;

Description

Condenses repeated occurrences of a character in a string into a single character.

Parameters

C	The character to condense (replace) in the string with a single occurrence of this character.
S	The string to condense.

Returns

The function returns a string containing the modified version of *S*, where repeated occurrences of *C* have been condensed into a single character.

Exceptions

None

See also

CtLStrip
CtLStripSet
CtRemove
CtRStrip
CtRStripSet
CtStrip
CtStripAll
CtStripSetAll
CtTrim
CoreTools string handling routines

CtRStrip Function

Unit

CtlStrng

Declaration

function CtRStrip (C:char; S:string): string;

Description

Strips all instances of the character *C* from the right side of the string *S*.

Parameters

C	the character to search and strip.
S	the string that is the subject of this operation.

Returns

Returns the stripped string, with all instances of the specified character removed from the right of the string.

Exceptions

ECtError	**CtlStrng: CtRStrip**

See also

CtLStrip
CtLStripSet
CtRemove
CtRStripSet
CtSqueeze
CtStrip
CtStripAll
CtStripSetAll
CtTrim
CoreTools string handling routines

CtStripAll Function

Unit

CtlStrng

Declaration

function CtStripAll (C:char; S:string): string;

Description

Strips all instances of the character C from the string *S*.

Parameters

C	The character to search and strip.
S	The string that is the subject of this operation.

Returns

Returns a copy of *S*, with all instances of the specified character removed from the string.

Exceptions

None

See also

CtLStrip
CtLStripSet
CtRemove
CtRStrip
CtRStripSet
CtSqueeze
CtStrip
CtStripSetAll
CtTrim
CoreTools string handling routines

CtStripSetAll Function

Unit

CtlStrng

Declaration

function CtStripSetAll (CS:TCtCharSet; S:string): string;

Description

Strips all characters from *S* that are specified in the *CS* set.

Parameters

CS	The character specification for stripping from *S*.
S	The string that is the subject of the operation.

Returns

The function returns a copy of *S* with all characters that are set in *CS* stripped from the string.

Exceptions

ECtError	**Ctlstrng: CtStripAll**

See also

CtLStrip
CtLStripSet
CtRemove
CtRStrip
CtRStripSet
CtSqueeze
CtStrip
CtStripAll
CtTrim
CoreTools string handling routines

CtTrim Function

Unit

CtlStrng

Declaration

function CtTrim (S:string): string;

Description

The CtTrim function removes tabs, nulls and spaces from the right hand side of a string.

Parameters

S	The string that is the subject of this operation.

Returns

The function returns a copy of *S* with all tabs, nulls and spaces removed from the right hand side.

Exceptions

ECtError	**CtlStrng: CtTrim**

See also

CtLStrip
CtLStripSet
CtRemove
CtRStrip
CtRStripSet
CtSqueeze
CtStrip
CtStripAll
CtStripSetAll
CoreTools string handling routines

TCtCharSet Type

Unit

CtlChar

Declaration

TCtCharSet = set of char;

Description

This character set type is used throughout CoreTools for Delphi whenever a character set is required.

CtReplace Function

Unit

CtlStrng

Declaration

function CtReplace (S, Unwanted, Replacement : string; var Changes: byte): string;

Description

Replaces all occurrences of *Unwanted* with *Replacement* in string *S*.

Parameters

S	The primary string on which you wish to perform the search and replace operation.
Unwanted	The string for which you are searching in *S*.
Replacement	The string with which to replace *Unwanted*.
Changes	A reference that facilitates the reporting back of the number of instances replaced during the operation.

Returns

The function returns a string resulting from the search and replace operation.

Changes shows the number of replacements that occurred during the search and replace operation.

Exceptions

ECtError	**CtlStrng: CtReplace**

See also

CtCharReplace

CoreTools string handling routines

CtParseClean Procedure

Unit

CtlStrng

Declaration

procedure CtParseClean (var S: string);

Description

Cleans-up a string ready to be parsed.

1. Converts tabs to spaces.
2. Strips leading and trailing spaces.
3. Removes duplicate spaces.

Parameters

S	The string to clean-up.

Exceptions

ECtError	**CtlStrng: CtParseClean**

See also

CtParse

CtPosSet Function

Unit

CtlStrng

Declaration

function CtPosSet (CS:TCtCharSet; S:string): byte;

Description

The CtPosSet function searches *S* for the first instance of one of the characters contained in *CS*.

Parameters

CS	The character set for which to search.
S	The string that is the subject of the search operation.

Returns

The function returns the index position in *S* where the first instance of a *CS* character is located.

Exceptions

ECtError	**CtlStrng: CtPosSet**

See also

CtCharPos
CtCharPosNext
CtRCharPos
CtRPos
CtWordPos

CtRCharPos Function

Unit

CtlStrng

Declaration

function CtRCharPos (C:char; S:string): byte;

Description

The CtRCharPos function searches for a specified character in a string, starting the search from the right hand side of the string, so that it returns the last occurrence of the character in the string..

Parameters

C	The character to search for.
S	The string that is to be the subject of the search operation.

Returns

The function returns the last index position of *C* in *S*. If no instance of the character is found, then the function returns 0.

Exceptions

None

See also

CtCharPos
CtCharPosNext
CtPosSet
CtRPos
CtWordPos
CoreTools string-handling routines

CtCharReplace Function

Unit

CtlStrng

Declaration

function CtCharReplace (S: string; Unwanted, Replacement: char; var Changes: byte): string;

Description

Replaces all instances of *Unwanted* in *S* with *Replacement*.

Parameters

S	The source string that is the subject of the operation.
Unwanted	The character to be replaced.
Replacement	The replacement character.
Changes	The reference that allows the reporting back of the number of changes made during the operation.

Returns

The function returns a copy of *S* with all instances of *Unwanted* replaced by *Replacement* characters. Changes shows the number of replacements that occurred during the search and replace operation.

Exceptions

None

See also

CtReplace

CoreTools string handling routines

CtWordNext Function

Unit

CtlStrng

Declaration

function CtWordNext (var S: string): string;

Description

Extracts and returns the next space-delimited string from *S*. *S* is returned with the sub string stripped off. If *S* is empty on entry, both *S* and the return value will be empty on return.

The function only recognizes spaces as word delimiters, therefore any tabs should firstly be converted to spaces, perhaps by using CtParseClean.

Parameters

S	The string to be the subject of the operation.

Returns

The function returns the next word extracted from the left hand side of *S*.

Exceptions

ECtError	**CtlStrng: CtWordNext**

See also

CtWordCount
CtWordExtract
CtWordPos
CtWordProperCase
CoreTools string handling routines

CtWordExtract Function

Unit

CtlStrng

Declaration

function CtWordExtract(const StartWord, NumWords:byte; const S:string): string;

Description

The CtWordExtract function returns a number of specified words from a string.

The function only recognizes spaces as word delimiters, therefore any tabs should firstly be converted to spaces, perhaps by using CtParseClean.

Parameters

StartWord	The number of the first word to extract.
NumWords	The number of words to extract.
S	The string that is to be the subject of the operation.

Returns

The function returns a string containing the specified number of words.

Exceptions

ECtError	**CtlStrng: CtWordExtract**

See also

CtWordCount
CtWordNext
CtWordPos
CtWordProperCase
CoreTools string handling routines

Character classification sets

Unit

CtlChar

Declarations

```
Const
  CtAlphaSet    : TCtCharSet = ['A'..'Z', 'a'..'z'];
  CtAlphaNumSet        : TCtCharSet = ['0'..'9', 'A'..'Z', 'a'..'z'];
  CtASCIISet    : TCtCharSet = [#0..#127];
  CtControlSet  : TCtCharSet = [#0..#31, #127];
  CtDelimSet    : TCtCharSet = [#0..#32];
  CtDigitSet    : TCtCharSet = ['0'..'9'];
  CtGraphSet    : TCtCharSet = [#33..#126];
  CtLowerSet    : TCtCharSet = ['a'..'z'];
  CtPrintSet    : TCtCharSet = [#32..#126];
  CtPunctSet    : TCtCharSet = [#33..#126]-['0'..'9', 'A'..'Z', 'a'..'z'];
  CtQuoteSet    : TCtCharSet = ['''',''''];
  CtRealSet     : TCtCharSet = ['0'..'9', '+', '-', '.', 'E', 'e'];
  CtSignedSet   : TCtCharSet = ['0'..'9', '+', '-'];
  CtSpaceSet    : TCtCharSet = [#9..#13, #32];
  CtUpperSet    : TCtCharSet = ['A' .. 'Z'];
  CtXDigitSet   : TCtCharSet = ['0'..'9','A'..'F','a'..'f'];
```

CtSwapDouble Procedure

Unit

CtlSwap

Declaration

procedure CtSwapDouble (var x, y: double);

Description

Swaps the contents of two double variables.

Parameters

x and *y* are the values to be swapped.

Exceptions

ECtError	Message = **CtlSwap: CtSwapDouble**

See also

CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines

CtSwapExtended Procedure

Unit

CtlSwap

Declaration

procedure CtSwapExtended (var x, y: extended);

Description

Swaps the contents of two extended variables.

Parameters

x and *y* are the values to be swapped.

Exceptions

ECtError	Message = **CtlSwap: CtSwapExtended**

See also

CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines

CtSwapLongInt Procedure

Unit

CtlSwap

Declaration

procedure CtSwapLongInt (var x, y: longint);

Description

Swaps the contents of two longint variables.

Parameters

x and *y* are the values to be swapped.

Exceptions

ECtError	Message = **CtlSwap: CtSwapLongint**

See also

CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines

CtSwapShortInt Procedure

Unit

CtlSwap

Declaration

procedure CtSwapShortInt (var x, y: shortint);

Description

Swaps the contents of two shortint variables.

Parameters

x and *y* are the values to be swapped.

Exceptions

EctError   Message = **CtlSwap: CtSwapShortInt**

See also

CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines

CtSwapSingle Procedure

Unit
CtlSwap

Declaration
procedure CtSwapSingle (var x, y: single);

Description
Swaps the contents of two single precision variables.

Parameters
x and *y* are the values to be swapped.

Exceptions
ECtError Message = **CtlSwap: CtSwapSingle**

See also
CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapString
CtSwapWord
CoreTools swapping routines

CtSwapWord Procedure

Unit
CtlSwap

Declaration
procedure CtSwapWord (var x, y: word);

Description
Swaps the contents of two word variables.

Parameters
x and *y* are the values to be swapped.

Exceptions

ECtError	Message = **CtlSwap: CtSwapWord**

See also
CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapReal
CtSwapShortInt
CtSwapSingle
CtSwapString
CoreTools swapping routines

CtSwapReal Procedure

Unit

CtlSwap

Declaration

procedure CtSwapReal (var x, y: real);

Description

Swaps the contents of two real variables.

Parameters

x and *y* are the values to be swapped.

Exceptions

ECtError   Message = **CtlSwap: CtSwapReal**

See also

CtSwapByte
CtSwapCardinal
CtSwapChar
CtSwapComp
CtSwapDouble
CtSwapExtended
CtSwapInteger
CtSwapLongInt
CtSwapPChar
CtSwapShortInt
CtSwapSingle
CtSwapString
CtSwapWord
CoreTools swapping routines

CtlDate Unit

Description

This unit contains Date and Time based routines and their supporting types and constants.

Types

TCtDateText

TCtTimeText

Routines

None

CtExtractFileDrive Function

Unit

CtlFile

Declaration

function CtExtractFileDrive (const F:TFileName):string;

Description

The CtExtractFileDrive function returns the drive specification extracted from *F*.

Parameters

F	The file specification that is the subject of this operation.

Returns

The function returns the drive definition, extracted from the file path.

Exceptions

None

See also

CtExtractFileNameOnly
CtExtractPathNameOnly
CoreTools file handling routines

CtExtractFileNameOnly Function

Unit

CtlFile

Declaration

function CtExtractFileNameOnly(const F:TFileName): TCtFileNameOnly;

Description

Extracts the file name, without the extension, from *F*.

Parameters

F	The file specification to be parsed.

Returns

The function returns the file name, without the extension, extracted from the file specification.

Exceptions

None

See also

CtExtractFileDrive
CtExtractPathNameOnly
CoreTools file handling routines

CtExtractPathNameOnly Function

Unit

CtlFile

Declaration

function CtExtractPathNameOnly (const F:TFileName): TFileName;

Description

The function parses the path name, without the drive letter, from *F*.

Parameters

F	The file specification to be parsed.

Returns

The function returns the path name, without the drive information, extracted from a file specification.

Exceptions

None

See also

CtExtractFileDrive
CtExtractFileNameOnly
CoreTools file handling routines

CtFileAgeCmp Function

Unit

CtlFile

Declaration

function CtFileAgeCmp (const F1,F2: TFileName): shortint;

Description

Compares the file date and time stamps for the two files.

Parameters

F1,F2	files for comparison.

Returns

-1, if F1 older than F2
 0, if same time stamp
 1, if F1 younger than F2

Exceptions

ECtError	Message = **CtlFile: CtFileAgeCmp**

See also

CtFileSetStamp
CoreTools file handling routines

CtFileAttrFull Function

Unit

CtlFile

Declaration

function CtFileAttrFull (const A:TFileType): string;

Description

The function returns a string containing a textual description of the attributes in the *A* set.

Parameters

A	The file attribute set to be deciphered.

Returns

A string containing the verbose description for the attribute(s) passed in *A*. The attributes are processed in the following order:

- Normal
- Read only
- Hidden
- System
- Volume
- Directory
- Archive

Exceptions

None

See also

CtFileAttrShort
CtFileAttrToSet
CtFileAttrToWord
CtFileSetAttr
CoreTools file handling routines

CtFileAttrShort Function

Unit

CtlFile

Declaration

function CtFileAttrShort (const A:TFileType): string;

Description

The CtFileAttrShort function returns a string containing a letter or dot in each position corresponding to one of the allowed file attributes.

Parameters

A	The file attribute set to be deciphered.

Returns

A string of length 7 bytes, containing either the first letter of each attribute (in lower case) to indicate the attribute is set, or a dot "." to indicate it is not set. The order of the attributes is 'nrhsvda', where:

n = normal
r = read only
h = hidden
s = system
v = volume
d = directory
a = archive

Exceptions

None

See also

CtFileAttrFull
CtFileAttrToSet
CtFileAttrToWord
CtFileSetAttr
CoreTools file handling routines

CtFileAttrToSet Function

Unit

CtlFile

Declaration

function CtFileAttrToSet (const W:word): TFileType;

Description

The Ctfileattrtoset conversion routine returns a set containing members representing each of the attribute bits set in *W*. The set members are:

Bit	Constant	Value	Set member
0	faReadOnly	$01	ftReadOnly
1	faHidden	$02	ftHidden
2	faSysFile	$04	ftSystem
3	faVolumeID	$08	ftVolumeID
4	faDirectory	$10	ftDirectory
5	faArchive	$20	ftArchive

Parameters

W	The word to be decipher.

Returns

The function returns a set containing the members with corresponding bits set in *W*.

Exceptions

None

See also

CtFileAttrFull
CtFileAttrShort
CtFileAttrToWord
CtFileSetAttr
CoreTools file handling routines

CtFileAttrToWord Function

Unit
CtlFile

Declaration
function CtFileAttrToWord (const A:TFileType): word;

Description
The CtFileAttrToWord function returns a word where each bit set represents the following DOS file attributes:

Bit	Constant	Value	Description
0	faReadOnly	$01	Read-only files
1	faHidden	$02	Hidden files
2	faSysFile	$04	System files
3	faVolumeID	$08	Volume ID files
4	faDirectory	$10	Directory files
5	faArchive	$20	Archive files
6	faAnyFile	$3F	All attributes

Parameters
A	The attribute set to decipher.

Returns
The function returns a word containing bits set to represent each of the set members present in *A*.

Exceptions
None

See also
CtFileAttrFull
CtFileAttrShort
CtFileAttrToSet
CtFileSetAttr
CoreTools file handling routines

CtFileCompare Function

Unit

CtlFile

Declaration

function CtFileCompare (const F1, F2:TFileName; const TimeStamp: boolean): longint;

Description

The CtFileCompare function compares the two specified files byte-by-byte and reports the offset where a difference was detected. In addition, additional date/time and size comparisons are performed if *TimeStamp* is set true.

Parameters

F1, F2	The files to compare.
TimeStamp	If set true, then date/time comparisons are performed.

Returns

If the function result is zero, the files are identical. A positive result indicates that the files are not identical and contains the offset (1-based) in *F2* where a byte did not match *F1*.

If the result is negative, then the file dates or sizes did not match.

Exceptions

ECtError	Message = **CtlFile: CtFileCompare**

See also

CoreTools file handling routines

CtFileCopy Procedure

Unit

CtlFile

Declaration

procedure CtFileCopy (S, D: TFileName);

Description

Copies file from *S* to *D*, without verification. If *D* is null then an ECtError exception is raised.

Parameters

S	The source file to be copied to *D*.
D	The destination file name for the copy.

Exceptions

ECtError	Message = **CtlFile: CtFileCopy**
EFOpenError	
EFCreateError	
EWriteError	

See also

CtVerifyOFF
CtVerifyON
CtVerifyState
CoreTools file handling routines

CtFileDateStr Function

Unit
CtlFile

Declaration
function CtFileDateStr (const F:TFileName):string;

Description
The CtFileDateStr function returns the date stamp of the specified file as a string. The date is returned in the ShortDateFormat.

Parameters
F	The file that is to be the subject of the operation.

Returns
The function returns a string containing the converted file stamp.

Exceptions
ECtError	Message = **CtlFile: CtFileDateStr**

See also
CtFileDateTime

CtFileInformation

CtFileSetStamp

CoreTools file handling routines

CtFileDateTime Function

Unit

CtlFile

Declaration

function CtFileDateTime (const F: T FILENAME): TDateTime;

Description

Returns the date/time stamp of the specified file as a TDateTime value.

Parameters

F	The file that is to be the subject of the operation.

Returns

Returns the file date/time stamp converted to a Delphi TDateTime type.

Exceptions

ECtError	Message = **CtlFile: CtFileDateTime**

See also

CtFileDateStr

CtFileInformation

CtFileSetStamp

CoreTools file handling routines

CtFileInformation Procedure

Unit

CtlFile

Declaration

procedure CtFileInformation (var I: TCtFileInfo; const F: TFileName);

Description

The CtFileInformation procedure returns file information for a specified file, including the file date/time, size (in bytes), and attributes.

Parameters

I	The record to take the file information.
F	The name of the file for which the information is required.

Exceptions

None

See also

CtFileDateStr
CtFileDateTime
CtFileSetStamp
CoreTools file handling routines

CtFileLineCount Function

Unit

CtlFile

Declaration

function CtFileLineCount (const F: TFileName): longint;

Description

The CtFileLineCount function returns the number of carriage return characters, Chr(13), in the specified text file.

CtFileLineCount can determine the number of lines in a standard ASCII file as long as each line in the file is terminated with a carriage return/line feed pair.

Parameters

F	The file to check. If the file is not in the current directory, F must contain the directory and/or drive if either is different from the default drive or current directory.

Returns

The function returns the number of Chr(13) characters found.

Exceptions

ECtError	Message = **CtlFile: CtFileLineCount**

See also

CoreTools file handling routines

CtFileSetAttr Function

Unit

CtlFile

Declaration

function CtFileSetAttr (const F: TFileName; const A: TFileType): boolean;

Description

The CtFileSetAttr function sets the file attributes of the file specified by *F* to the attribute set values represented by *A*.

Parameters

F	The filename to set.
A	The file attribute set containing the required file attributes.

Returns

The function returns true on success, otherwise returns false.

Exceptions

None

See also

CtFileAttrFull

CtFileAttrShort

CtFileAttrToSet

CtFileAttrToWord

CoreTools file handling routines

CtFileSetStamp Function

Unit

CtlFile

Declaration

function CtFileSetStamp (const F: TFileName; const T: TDateTime): boolean;

Description

The CtFileSetStamp function sets the date/time stamp for the specified file.

Parameters

F	The file for which to set the date and time stamp.
T	The time and date value to be set.

Returns

The function returns true on success, otherwise returns false.

Exceptions

None

See also

CoreTools file handling routines

CtFileSize Function

Unit

CtlFile

Declaration

function CtFileSize (const F: TFileName): longint;

Description

The CtFileSize function returns the size of the specified file in bytes.

Parameters

F	The file that is to be the subject of the operation.

Returns

The function returns the size of the file in bytes, if the file does not exist then it returns -1.

Exceptions

None

See also

CoreTools file handling routines

CtIsRootDir Function

Unit

CtlFile

Declaration

function CtIsRootDir (S:TFileName): boolean;

Description

The CtIsRootDir function tests if a file specification is that for a root directory, e.g. C:\ is a root directory specification.

Parameters

S	The file specification to test.

Returns

The function returns true if the specification is for a root directory, otherwise it returns false.

Exceptions

None

See also

CtRemakeDirRoot

CoreTools file handling routines

CtParseFileSpec Procedure

Unit

CtlFile

Declaration

procedure CtParseFileSpec (const F: TFileName; var S:TStringList);

Description

The CtParseFileSpec procedure fills a TStringList object with the individual components of a file specification string.

Parameters

F	The file specification to be parsed.
S	The target object in which to insert the parsed parts of the file specification.

Exceptions

None

See also

CoreTools file handling routines

CtReadDirectory Procedure

Unit

CtlFile

Declaration

procedure CtReadDirectory (const SearchPath:TFileName; const SearchMask:TFileName; const SubDirs:boolean; const FileFound:TCtFoundFunction; const DirFound:TCtFoundFunction);

Description

The procedure starts at the directory given in *SearchPath* and searches it for files matching the mask. *FileFound* will be called for each file found within the current directory. The path will always end in a backslash,   so the parameters *F* + *SearchRecord.Name* yields the full name of the found file to the callback function. If the callback function returns False, the recursion will stop and CtReadDirectory returns immediately.

After the directory has been scanned for files it is again scanned for directories and each found directory is in turn scanned in the same manner, and *DirFound* is called for every directory found. If the callback function returns False then CtReadDirectory returns immediately.

If *FileFound* = nil or *DirFound* = nil then the procedure returns immediately.

Parameters

SearchPath	The directory at which to start the search
SearchMask	The file search mask, allows wild cards. If this is an empty string, ***.*** is used.
FileFound	The callback function to call when every file is found.
DirFound	The callback function to call when every directory is found.

Exceptions

ECtError	Message = **CtlFile: CtReadDirectory**

See also

CoreTools file handling routines

CtRemakeDirRoot Function

Unit

CtlFile

Declaration

function CtRemakeDirRoot (const NewRoot, OldRoot, F:TFileName): TFileName;

Description

Extracts *OldRoot* from the start of *F* and replaces it with *NewRoot*.

Parameters

NewRoot	The new root path to prefix F
OldRoot	The old root path to extract from F
F	The file specification that is the subject of this operation

Returns

The function returns: <*NewRoot*> + <right hand side of *F*>

Exceptions

None

See also

CtIsRootDir

CoreTools file handling routines

CtVerifyOFF Procedure

Unit

CtlFile

Declaration

procedure CtVerifyOFF;

Description

The CtVerifyOFF procedure turns the DOS verify flag off.

Exceptions

None

See also

CtVerifyON

CtVerifyState

CoreTools file handling routines

CtVerifyON Procedure

Unit
CtlFile

Declaration
procedure CtVerifyON;

Description
The CtVerifyON procedure turns on the DOS verify flag.

Exceptions
None

See also
CtVerifyOFF
CtVerifyState
CoreTools file handling routines

CtVerifyState Function

Unit

CtlFile

Declaration

function CtVerifyState: boolean;

Description

Returns the current state of the DOS verify flag.

Returns

The function returns true if the DOS verify flag is true, otherwise returns false.

Exceptions

None

See also

CtVerifyOFF
CtVerifyON
CoreTools file handling routines

TCtFileInfo Type

Unit

CtlFile

Declaration

```
TCtFileInfo = record
    Atts:        TFileType;
    DateTime:    TDateTime;
    Size:        longint;
    end;
```

Description

This record type stores file information. The data has been converted from the low-level DOS format to Delphi high-level types.

See also

CtFileInformation
CoreTools file-handling routines