Common pascal units documentation

Pasdoc

May 5, 2004

Contents

Chapter 1

Unit crc

1.1 Description

CRC generation unit

CRC generation routines, compatible with ISO 3309 and ITU-T-V42.

1.2 Overview

`UpdateCrc32`

1.3 Functions and Procedures

UpdateCrc32 function

Declaration `function UpdateCrc32(InitCrc:longword; b:  byte):longword;`

Description Routine to get the CRC-32 value. Normally to be compatible with the ISO 3309 standard, the first call to this routine should set `InitCRC` to `$FFFFFFFF`, and the final result of the CRC-32 should be `XOR`'ed with $FFFFFFFF.

Parameters **InitCRC** The value of the previous CRC

b The data byte to get the CRC-32 of

Returns The updated CRC-32 value

1.4 Author

Carl Eric Codere

Chapter 2

Unit dpautils

2.1 Description

Delphi/Kylix compatbility unit

This unit includes common definitions so that common code can be compiled under the Delphicompilers. It supports Delphi 6 and higher that are targeted for Win32 as well as WDOSX/DOS.

2.2 Author

Carl Eric Codere

Chapter 3

Unit fpautils

3.1 Description

Free Pascal compatibility unit

This unit includes common definitions so that common code can be compiled under the Free pascal compilers. It supports Freepascal 1.0.6 and higher (all targets).

3.2 Author

Carl Eric Codere

Chapter 4

Unit locale

4.1 Description

Localisation unit

This unit is used to convert different locale information. ISO Standards are used where appropriate. Credits where credits are due, information on the ISO and date formats where taken from

4.2 Overview

`GetISODateString`

`GetISOTimeString`

`UNIXToDateTime`

4.3 Functions and Procedures

GetISODateString function

Declaration `function GetISODateString(Year, Month, Day: Word): shortstring;`

Description Returns the preferred date string as recommended by ISO 8601 (Gregorian Calendar). Returns an empty string if there is an error.

Parameters

year Year of the date - valid values are from 0000 to 9999

month Month of the date - valid values are from 0 to 12

day Day of the month - valid values are from 1 to 31

GetISOTimeString function

Declaration `function GetISOTimeString(Hour, Minute, Second: Word; UTC: Boolean): shortstring;`

Description Returns the preferred time string as recommended by ISO 8601 (Gregorian Calendar). .

Returns Empty string if there is an error

UNIXToDateTime procedure

Declaration `procedure UNIXToDateTime(epoch: longword; var year, month, day, hour, minute, second: Word);`

Description Converts a UNIX styled time (the number of seconds since 1970) to a standard date and time representation.

4.4 Author

Carl Eric Codere

Chapter 5

Unit tpautils

5.1 Description

Turbo Pascal 7 Compatibility unit

This unit includes common definitions so that common code can be compiled under the Turbo/Borland pascal compilers. It supports both Turbo Pascal 7.0 and Borland Pascal 7.0 and higher.

5.2 Author

Carl Eric Codere

Chapter 6

Unit unicode

6.1 Description

unicode support unit

This unit contains routines to convert between the different unicode encoding schemes. The code was converted to Pascal from the C code located at:

6.2 Overview

`convertUTF16toUTF8` Convert an UTF-16 string to an UTF-8 string

`convertUTF8toASCII` Convert an UTF-8 string to an ASCII string

`lengthUTF16` Returns the current length of an UTF-16 string

`lengthutf8` Returns the current length of an UTF-8 string

`setlengthUTF16` Set the length of an UTF-16 string

`setlengthUTF8` Set the length of an UTF-8 string

6.3 Functions and Procedures

convertUTF16toUTF8 function

Declaration `function convertUTF16toUTF8(s: array of utf16; var outstr: utf8string): integer;`

Description Convert an UTF-16 string to an UTF-8 string

convertUTF8toASCII function

Declaration `function convertUTF8toASCII(s: array of utf8): shortstring;`

Description Convert an UTF-8 string to an ASCII string

lengthUTF16 function

Declaration `function lengthUTF16(s: array of utf16): integer;`

Description Returns the current length of an UTF-16 string

lengthutf8 function

Declaration `function lengthutf8(s: array of utf8): integer;`

Description Returns the current length of an UTF-8 string

setlengthUTF16 procedure

Declaration `procedure setlengthUTF16(var s: array of utf16; l: integer);`

Description Set the length of an UTF-16 string

setlengthUTF8 procedure

Declaration `procedure setlengthUTF8(var s: array of utf8; l: integer);`

Description Set the length of an UTF-8 string

6.4 Types

utf8

Declaration `utf8 = char;`

Description UTF-8 base data type

utf16

Declaration `utf16 = word;`

Description UTF-16 base data type

utf32

Declaration `utf32 = longword;`

Description UTF-32 base data type

utf8string

Declaration `utf8string = array[0..1024] of utf8;`

Description UTF-8 string declaration

utf32string

Declaration `utf32string = array[0..255] of utf32;`

Description UTF-32 string declaration

utf16string

Declaration `utf16string = array[0..255] of utf16;`

Description UTF-16 string declaration

6.5 Constants

UNICODE_ERR_OK

Declaration `UNICODE_ERR_OK = 0;`

Description Return status: conversion successful

UNICODE_ERR_SOURCEILLEGAL

Declaration `UNICODE_ERR_SOURCEILLEGAL = -1;`

Description Return status: source sequence is illegal/malformed

6.6 Author

Carl Eric Codere

Chapter 7

Unit utils

7.1 Description

General utilities common to all platforms.

7.2 Overview

`EscapeToPascal`

`FileExists` Verifies the existence of a filename

`hexstr` Convert a value to an ASCII hexadecimal representation

`Printf` Format a string and print it out to the console

`StreamErrorProcedure` Generic stream error procedure

`SwapLong` Change the endian of a 32-bit value

`SwapWord` Change the endian of a 16-bit value

`TrimLeft` Remove all whitespace from the start of a string

`TrimRight` Remove all whitespace from the end of a string

`UpString` Convert a string to uppercase ASCII

`ValBinary`

`ValDecimal`

`ValHexadecimal`

`ValOctal`

7.3 Functions and Procedures

EscapeToPascal function

Declaration `function EscapeToPascal(const s:string; var code: integer): string;`

Description Converts a C style string (containing escape characters), to a pascal style string. Returns the converted string. If there is no error in the conversion, `code` will be equal to zero.

Parameters **s** String to convert

code Result of operation, 0 when there is no error

FileExists function

Declaration `Function FileExists(FName : string): Boolean;`

Description Verifies the existence of a filename

This routine verifies if the file named can be opened or if it actually exists. FName Name of the file to check Returns FALSE if the file cannot be opened or if it does not exist.

hexstr function

Declaration `function hexstr(val : longint;cnt : byte) : string;`

Description Convert a value to an ASCII hexadecimal representation

Printf function

Declaration `function Printf(const s : string; var Buf; size : word): string;`

Description Format a string and print it out to the console

This routine formats the string specified in s to the format specified and returns the resulting string. The following specifiers are allowed: %d : The buffer contents contains an integer %s : The buffer contents contains a string, terminated by a null character. %bh : The buffer contents contains a byte coded in BCD format, only the high byte will be kept. %bl : The buffer contents contains a byte coded in BCD format, only the low byte will be kept. s The string to format, with format specifiers buf The buffer containing the data size The size of the data in the buffer Returns The resulting formatted string

StreamErrorProcedure procedure

Declaration `procedure StreamErrorProcedure(Var S: TStream);`

Description Generic stream error procedure

Generic stream error procedure that can be used to set `streamerror`

SwapLong procedure

Declaration `Procedure SwapLong(var x : longword);`

Description Change the endian of a 32-bit value

SwapWord procedure

Declaration `Procedure SwapWord(var x : word);`

Description Change the endian of a 16-bit value

TrimLeft function

Declaration `function TrimLeft(const S: string): string;`

Description Remove all whitespace from the start of a string

TrimRight function

Declaration `function TrimRight(const S: string): string;`

Description Remove all whitespace from the end of a string

UpString function

Declaration `function UpString(s : string): string;`

Description Convert a string to uppercase ASCII

ValBinary function

Declaration `function ValBinary(const S:String; var code: integer):longint;`

Description Convert a binary value represented by a string to its numerical value. If there is no error, `code` will be equal to zero.

ValDecimal function

Declaration `function ValDecimal(const S:String; var code: integer):longint;`

Description Convert a decimal value represented by a string to its numerical value. If there is no error, `code` will be equal to zero.

ValHexadecimal function

Declaration `function ValHexadecimal(const S:String; var code: integer):longint;`

Description Convert an hexadecimal value represented by a string to its numerical value. If there is no error, `code` will be equal to zero.

ValOctal function

Declaration `function ValOctal(const S:String;var code:  integer):longint;`

Description Convert an octal value represented by a string to its numerical value. If there is no error, `code` will be equal to zero.

7.4 Author

Carl Eric Codere

Chapter 8

Unit vpautils

8.1 Description

Virtual Pascal Compatibility unit

This unit includes common definitions so that common code can be compiled under the Virtual pascal compiler. It supports Virtual Pascal 2.1 and higher for the Win32, DOS and OS/2 targets.

8.2 Author

Carl Eric Codere